Peer-to-Peer Networking and Applications (2020) 13:190–206
https://doi.org/10.1007/s12083-019-00755-x

An overlay management strategy to improve QoS in CDN-P2P live streaming systems

Shilpa Budhkar[1] · **Venkatesh Tamarapalli**[1]

Received: 17 August 2018 / Accepted: 4 April 2019 / Published online: 25 April 2019

Abstract

With the tremendous growth in demand of live video streaming services over Internet, content delivery networks (CDNs) get overloaded. The CDN-P2P live streaming systems are being deployed to reduce the stream delivery load of CDNs. The inherent peer churn and heterogeneity in upload contribution of peers make it challenging to maintain the quality of service (QoS) in these systems. To deal with this challenge, we propose an overlay management strategy that organizes peers in the overlay based on their *serviceability*, which we define based on stability, stream chunk availability, upload and download capacities of peers. Peers are arranged in a hybrid tree-mesh overlay at the edges of CDN. The peers with higher upload capacity are part of an extended CDN tree to facilitate stable seeders. A part of the upload capacity of the existing peers is reserved to form *virtual sources* that provide startup chunks for quick playback. The peers joining the mesh overlay select partners with highest *serviceability* to ensure better streaming quality. They also adapt the topology by replacing partners based on *serviceability* and upload capacity utilization to maintain QoS during churn. Overall, the proposed serviceability-aware overlay management strategy (SOMS) enhances the QoS and upload capacity utilization of peers while dealing with heterogeneity in upload contribution and peer churn. Comparison with the existing CDN-P2P systems shows that the upload capacity utilization of peers is improved by 30%, while the startup delay and streaming quality are improved by 20% and 25%, respectively.

Keywords CDN-P2P system · Live streaming · Overlay management · QoS · Serviceability

1 Introduction

In an Internet traffic forecast report given by Cisco, it is reported that the video traffic is expected to increase from 73% to 82% of the total traffic by 2021 [1]. Of this, 13% is expected to be live video traffic by 2021. To manage such an enormous volume of video traffic in the Internet, design of efficient content delivery networks (CDNs) is essential. It is estimated that by the year 2021 CDNs are expected to support 77% of all Internet video traffic [1]. However, the CDNs suffer from high infrastructure cost and limited scalability [2]. They get overloaded and cannot meet the QoS requirements during peak demand [3, 4]. In a recent article on live streaming of FIFA World Cup and Wimbledon 2018, it is reported that CDNs were able to provide ultra high definition quality video to upto 60,000 users, with more than one million users waiting for similar quality [5]. It is also noticed that there is a trade-off between minimizing the latency and improving the streaming quality. Hence, CDN-P2P or peer-assisted CDNs are being deployed, in which the upload capacity of peers is utilized to increase the scalability without increasing the infrastructure cost [6, 7].

Overview of CDN-P2P live streaming systems The CDN-P2P live streaming systems consist of an overlay of a source server, multiple CDN servers, and peers. The CDN servers act as intermediate relay nodes between the source server and peers. Such an integration of P2P overlay with the CDN servers results in delivery of a significant amount of video content using resources of peers [8]. The studies in [9, 10] showed that with the help of P2P overlays

✉ Shilpa Budhkar
b.shilpa@iitg.ac.in

Venkatesh Tamarapalli
t.venkat@iitg.ac.in

[1] Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Guwahati, India

50% to 88% traffic can be offloaded from the CDN servers. Considering this benefit, various commercial CDNs also modified their infrastructure to include peer-assisted streaming. For example, Akamai acquired RedSwoosh P2P system in 2007 and started its hybrid CDN-P2P service named NetSession in 2010 [11]. It is reported that Akamai offloaded traffic of CDN servers by up to 66% using peer-assistance [12]. This reduces the cost of content delivery and increases the profit margin of both CDN operators as well as content providers [13].

Limitations of CDN-P2P live streaming systems Some studies showed that the upload capacity of peers can be utilized better in live streaming compared to video on-demand streaming, due to the synchronized streaming and longer peer lifetime in the system [9, 10]. Despite this CDN-P2P live streaming systems suffer from inefficient utilization of the upload capacity of the peers. The primary reasons for this are reported to be the heterogeneity in session duration (lifetime) and upload capacity of peers [14, 15]. It is observed that the session duration and upload contribution of peers depend on various parameters such as startup delay, playback delay, streaming quality, peer bandwidth, peer arrival rate and departure rate [16–18]. The peers are more likely to leave the system if they notice unacceptable startup delay or streaming quality [4, 19].

Considering these issues, CDN-P2P systems should organize peers in the overlay to enhance their upload capacity utilization. Typically, existing systems organize peers in the overlay based on their stability and upload capacity. Authors of [20, 21] proposed to predict stability of peers using their elapsed session duration and organize the most stable peers to form a backbone tree in the overlay. However, it is difficult to predict the stability of peers in the beginning of a streaming session using these mechanisms, because all the peers have almost equal and small elapsed session duration. In [22, 23], authors proposed to organize peers with higher upload contribution closer to the CDN servers to improve QoS. They neglect utilizing the upload capacity of peers with short session duration and/or low bandwidth. The authors of [24] proposed a mechanism for the CDN servers to service emergency chunks requested and save server upload bandwidth, but do not focus on maximizing the upload capacity utilization of peers. Overall, the existing systems do not consider the capability of peers to provide the desired streaming quality while organizing them in the overlay. They also do not consider stability, chunk availability, upload and download capacities of peers in overlay management. The stability prediction is also not precise because of not considering the instantaneous streaming quality experienced by peers. We argue that it is required to consider the streaming capability of peers and then strategically exploit the capacity of peers with heterogeneous session duration and/or upload capacity.

Overview of the proposed overlay management strategy In this paper, we propose a *serviceability-aware overlay management strategy* (SOMS) for CDN-P2P live streaming systems, to enhance the upload capacity utilization of peers and improve the QoS. SOMS organizes the peers considering their *serviceability*, which we define in terms of stability, streaming quality, upload and download capacities of peers. The overlay is built as a set of connected sub-overlays, which are built and managed by corresponding CDN servers. The CDN load manager picks the CDN server(s) for a new peer considering the geographical proximity, fraction of peers that receive inadequate quality and experience larger than average joining delay. The CDN server selection strategy balances stream delivery load of CDN servers and distributes the peers among the sub-overlays to give a lower startup delay.

Within a sub-overlay, the peers are organized in a hybrid tree-mesh topology. The peers with higher upload capacity are part of an extended CDN tree with the server at the root. Other peers are part of the mesh topology. The CDN servers create and maintain *virtual sources* by reserving some upload capacity of existing peers to provide startup chunks to new peers. The mesh peers predict serviceability of partners and select the ones with highest serviceability value to ensure better streaming quality. The mesh peers also adapt the topology by replacing partners based on serviceability and upload capacity utilization to maintain QoS during churn. The performance of SOMS is compared with PROSE [22], LiveSky [23], and AERO [24] to show that the upload capacity utilization of peers is enhanced by 30%, while the startup delay and streaming quality of peers are improved by 20% and 25%, respectively.

The major contribution of this work is the design of *serviceability-aware overlay management strategy (SOMS)* to improve QoS in CDN-P2P live streaming systems with the following features:

- A CDN server selection strategy for new peers to balance the stream delivery load of CDN servers and peers.
- Creating a hybrid tree-mesh overlay topology with a resilient tree to generate stable and high upload capacity seeders.
- A bandwidth allocation mechanism for peers to distribute their upload and download capacities among multiple sub-overlays.
- Creating virtual sources using peers to ensure quick startup with better streaming quality to new peers.
- A peer selection strategy to select partners for mesh peers that ensure better streaming quality.

– A topology adaptation strategy to replace partners during churn to maintain streaming quality.

The rest of this paper is organized as follows. We present a survey of the literature in Section 2. Section 3 presents the details of the proposed SOMS. The results from simulation and the discussion are presented in Section 4. Section 5 concludes the paper.

2 Related work

In most of the existing literature, peers in CDN-P2P systems are organized based on their stability and upload capacity. In [22], authors proposed that CDN servers serve only those peers labeled as choke point expansion nodes and super nodes. The choke point expansion nodes are those that supply data to a large number of peers, but the demand is more than the supply. The super nodes are the ones with higher upload capacity, are highly stable and have sufficient data in buffers. In [25], it is suggested that the CDN servers serve the peers with highest stability and longer elapsed session duration.

The Bayesian network model of user behavior proposed in [26] and Cox proportional hazards model in [21] show that session duration of peers depends on parameters such as elapsed session duration, initial streaming quality, average streaming quality, content type, popularity, peer arrival rate, and departure rate. The peers getting better QoS are likely to stay in the system for a longer duration. Hence, the authors of [21] proposed that peers keep only superior peers (or stable peers) as their partners. A superior peer is identified using a *superior index*, calculated as the product of predicted session duration and average upload bandwidth. In [20], the stability of peers was predicted using elapsed session duration and a backbone tree of stable peers was created to ensure better QoS.

Overall, the existing mechanisms identify the stable peers and enhance the utilization of their upload capacity to improve QoS. Contrary to this, we propose to utilize the upload capacity of all the peers considering their serviceability measured with the help of upload contribution of peers, streaming quality, and elapsed session duration. By this, we strategically utilize the upload capacity of the peers with heterogeneous session duration and upload bandwidth.

Some earlier studies also recommended prioritizing the delivery of stream to peers based on their playback deadline [27–29]. In [27] it was proposed to consider three regions of the buffer i.e.: startup region, common region and emergency region. The chunks of the startup region are prioritized over common region. The chunks of the emergency region are delivered by the CDN server only when other partners fail to deliver them and the playback deadline is close. In [28], the CDN servers maintain three queues to handle chunk requests. The first queue stores the requests of all new joining peers in the first-come-first-serve (FCFS) order. The second queue stores chunks which are not served by peers and chunks closer to the playback deadline, in the order of playback deadline. The third queue is for those chunks which are available only at the server. The authors of [24] proposed an adaptive emergency request optimization (AERO) mechanism with the goal of reducing number of streams seeded by the CDN servers and saving their upload bandwidth. In this mechanism, the CDN server dynamically updates the number of streams it seeds considering the *seeding ratio*, number of emergency chunk requests received and chunk distribution efficiency of P2P overlay. *Seeding Ratio* is the ratio between the aggregate upload bandwidth of seeded peers and the aggregate upload bandwidth of all peers. This mechanism is complementary to our work that aims to improve QoS and upload capacity utilization of peers through better organization in the overlay.

In [29], a peer first fetches chunks from the other peers that belong to same cluster considering the availability of chunks in their buffers. If chunks are not delivered within a threshold, the peer requests the CDN server for the chunks. In LiveSky [23] peers are initially connected to edge nodes to receive startup chunks and reduce startup delay. The peers that contribute upload capacity by becoming a LiveSky client are ensured better QoS. However, the peers that contribute more upload capacity are not prioritized. In [30], it was proposed to select LiveSky clients based on their upload contribution and regular stream delivery performance.

In summary, the proposed techniques utilize upload capacity of CDN servers to provide startup chunks and missing chunks. As a result, the number of peers that receive assured QoS is limited by the capacity of servers. Moreover, the CDN servers may also get overloaded due to the geographically skewed distribution of peer population [23]. To overcome these issues, the overlay management strategy proposed in this paper delegates the responsibility of assuring quick startup and desired streaming quality to peers by creating virtual sources, by using serviceability-based peer selection and by topology adaptation during churn. The upload capacity of CDN servers is also utilized better to generate more number of stable and high upload capacity seeders.

A preliminary version of our work in [31], proposed a stability-based partner selection strategy for only a mesh overlay. We significantly extend the proposed overlay management strategy to a CDN-P2P live streaming system. The following are the major modifications in this work: i) A CDN server selection strategy, ii) a mechanism to distribute upload and download capacity of a peer among multiple

CDN servers and sub-overlays, iii) organizing peers to create a hybrid tree-mesh overlay topology, iv) creation of virtual sources considering the download capacity of peers instead of streaming rate, and v) a topology adaptation strategy to maintain QoS during churn.

3 SOMS: serviceability-aware overlay management strategy for CDN-P2P live streaming systems

In this section, we first state the system model and then present the proposed overlay management strategy in detail along with its various modules.

3.1 System model

The CDN-P2P live streaming system considered in this work is broadly similar to LiveSky [23]. Figure 1 illustrates the system, consisting of a source server, a number of CDN servers, a CDN load manager, and peers. The source generates video stream in the form of equal and small-sized blocks/chunks distributed as sub-streams to CDN servers. The sub-streams are created so that a peer does not need all the sub-streams to play the video. But, downloading each additional sub-stream improves the quality of video playback. The peers advertise the aggregate upload capacity while joining the system (could also be zero). The download/upload capacity is defined in terms of the number of sub-streams a peer can download/upload.

The overlay consists of multiple connected sub-overlays such that each sub-overlay is built and maintained by a CDN server. A peer may be part of many sub-overlays by sharing its upload and download capacity. The CDN load manager selects the CDN servers for a new peer joining the system. The selected CDN server then helps the new peer with the stream data as well as information about the other peers in the sub-overlay. The topology of a sub-overlay is a hybrid tree-mesh topology. Peers which are part of the tree topology are termed tree peers and those that are part of the mesh topology are termed mesh peers. Considering the role of peers in the stream delivery, we use the terms parent peer (or parent partner) and child peer (or child partner) in this paper.

Next, we describe the various modules of SOMS, which mainly include CDN server selection and maintaining the hybrid tree-mesh topology at the sub-overlay.

3.2 CDN server selection strategy

The proposed CDN server selection strategy aims to distribute the the stream delivery load of servers and peers among sub-overlays and provide better QoS to peers. The CDN load manager selects most suitable servers for the new peers to help them in joining corresponding sub-overlays. It calculates the suitability values of all servers as the normalized sum of geographical distance of servers from the new peer, fraction of peers experiencing larger than average joining delay and inadequate chunk rate in the respective sub-overlays. The closest servers with peers experiencing

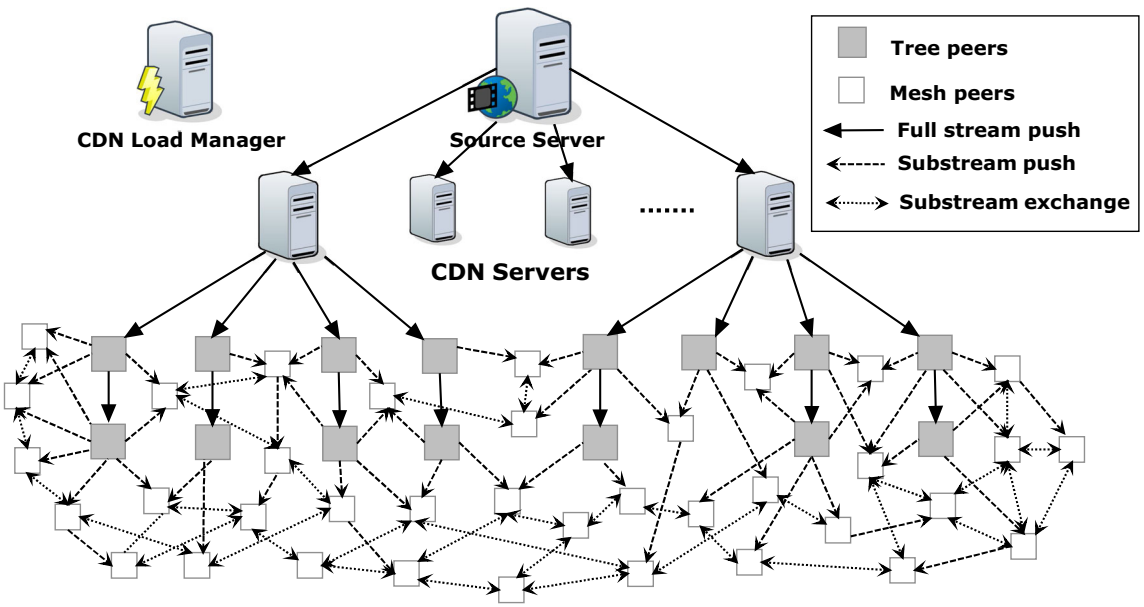


Fig. 1 A CDN-P2P live streaming system

least joining delay and receiving desired chunk rate in their corresponding sub-overlays are considered the most suitable ones. The number of servers a peer selects to join depends on the population of peers, number of servers, and supply-demand ratio of upload capacity in the overlay.

To keep it simple, let us assume that a peer can join two servers. The load manager finds out two most suitable servers A and B for the new peer, in that order. For this selection, the load manager maintains a record of the fraction of peers experiencing larger than average joining delay and receiving inadequate chunk rate in the sub-overlays. This information is obtained from the servers of the corresponding sub-overlays regularly. The joining delay of a peer is the time it takes to join the sub-overlay either as a tree peer or as a mesh peer and starts receiving the stream. When a new peer starts its video playback it sends a *start playback message* to the server of the joined sub-overlay. This helps the servers in calculating the joining delay experienced by peers in the sub-overlay. Each peer of the sub-overlay also informs its server whenever it receives chunk rate lower than its desired rate to calculate the fraction of peers receiving inadequate chunk rate.

CDN server selection considering the fraction of peers receiving inadequate chunk rate helps a new peer in joining the sub-overlays where peers are satisfied with the streaming quality. It also indicates that the sub-overlay is relatively less loaded with the streaming traffic and the new peer can receive desired streaming streaming quality. The fraction of peers experiencing larger than average joining delay indicates the load of servers to provide startup chunks and information of existing peers to the new peer. Selecting a server with smaller value of this fraction ensures that it is lightly loaded and can support new peers in joining the overlay faster. These two parameters also help in distributing stream delivery load among the sub-overlays. Considering geographical distance along with these parameters helps in selecting the closest lightly loaded servers and sub-overlays for the new peers.

3.3 Building and maintaining hybrid tree-mesh topology

In our work, the objective of building the hybrid tree-mesh sub-overlay is to enhance the utilization of upload capacity of peers and improve QoS during churn. Peers join either the extended CDN tree of the overlay (called tree peers) or the mesh overlay (called mesh peers). Next, we discuss how new peers are added either as tree peers or as mesh peers in the sub-overlays.

Adding a new peer With the help of the load manager a new peer sends a *join request message* to the selected CDN servers (A or B) to join the corresponding sub-overlay(s) (denoted as $subA$ and $subB$). The message includes the aggregate upload and download capacity of the new peer. Algorithm 1 shows the method used by the servers to decide whether the new peer should be added to extended CDN tree or mesh considering their instantaneous residual upload capacity, the aggregate download and upload capacities of the new peer.

Algorithm 1 Adding a new peer as Tree or Mesh peer.

Input: *Join*(**Req**)← Join request sent by new peer 'n'
U_A^r (or U_B^r) ← Residual upload capacities of server A (or B)
D_n^a ← Aggregate download capacity of the new peer 'n'
U_n^a ← Aggregate upload capacity of the new peer 'n'
$Child$ ← Set of all the tree peers which are directly connected to the CDN server
Output: Response for the join request (*Join*(**Req**)).
Join(**CDNTree**)← Server's response to new peer to join as tree peer
Join(**Mesh**)← Server's response to new peer to join as mesh peer

```
if D_n^a ≥ S & U_n^a ≥ S then
    if U_A^r > S OR U_n^a ≥ U_i^a, ∀i ∈ Child then
        Send Join(CDNTree)
    end
else
    Send Join(Mesh)
end
```

A server finds the new peer eligible to be a tree peer, if aggregate upload and download capacity of the new peer are greater than or equal to streaming rate. But, it sends a response to join the extended CDN tree to the eligible new peer only if one of the following two conditions is satisfied: i) residual upload capacity of the server is greater than the streaming rate, or ii) the aggregate upload capacity of the new peer is greater than the aggregate upload capacity of atleast one of the current children of the server. Otherwise, the server responds the new peer with an offer to join as a mesh peer. If a new peer receives an offer to join the extended CDN tree from at least one of the servers then it joins as a tree peer, otherwise it joins sub-overlays of both the servers as a mesh peer. A new peer may also receive offers to join the extended CDN tree from both the servers. In this case, the new peer joins the server with higher suitability value.

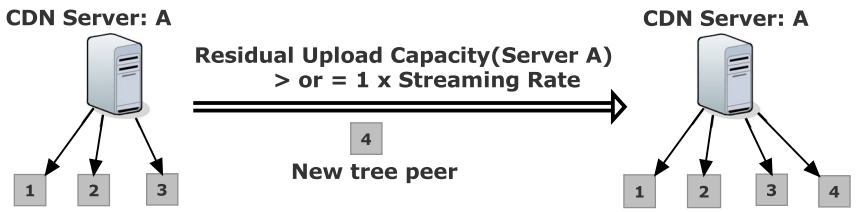
The upload and download capacity of a tree peer are completely dedicated to the joining sub-overlay. For a mesh peer, it is distributed among both the joining sub-overlays. This is because the tree peers receive full stream either from

the CDN server or other tree peers whereas, mesh peers can receive sub-streams either from the tree or mesh peers. The detailed procedure of upload and download capacity allocation of mesh peers is discussed later in Section 3.3.2.

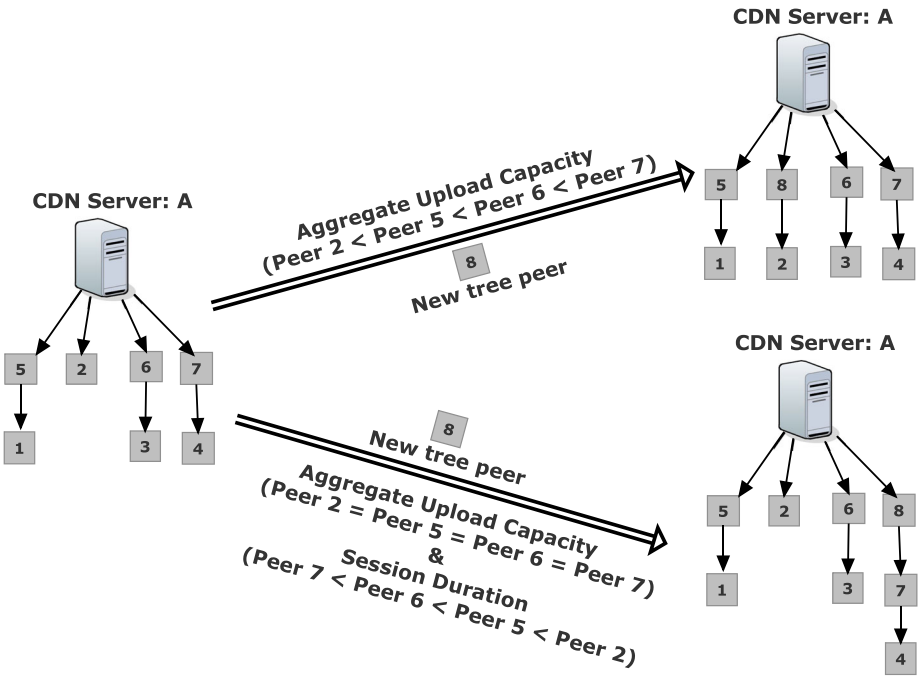
3.3.1 Building and maintaining extended CDN tree

The extended CDN tree is formed to ensure QoS to high upload capacity peers, which in turn generates stable seeders. In the tree, each peer has only one parent which provides the full stream to the peer and one child to which the peer provides the full stream. This helps in handling peer churn and maintaining stream continuity among tree peers. The residual upload capacity of intermediate nodes and the aggregate upload capacity of leaf nodes are utilized to deliver substreams to mesh peers.

Adding a new tree peer A new tree peer always joins the server as a direct child as shown in Fig. 2. If the server does not have enough residual upload capacity to serve full stream to the new peer, one of the existing children is replaced by the new peer. The replaced peer along with its descendants joins the new peer. The peer for replacement is selected based on upload capacity and elapsed session duration, as illustrated in Fig. 2b. First, the peer with least upload capacity is selected. If all of them have same upload capacity, then the peer with smallest elapsed session duration is selected. Here, the elapsed session duration of a peer represents its stability. We do not consider other parameters such as streaming quality received by the peer and its download capacity to predict the stability, because the peer to be replaced is directly connected to the CDN server and therefore, it always receives good quality stream.



(a) Case 1: Residual upload capacity of CDN server is greater than streaming rate



(b) Case 2: Replace one of the existing children

Fig. 2 Adding a new tree peer

This peer joining process takes care that the highest upload capacity peers are closer to server in the extended CDN tree. The peers are added to the tree only in the initial half of the streaming session to avoid increasing the height of the tree in an unlikely scenario of peers joining in the increasing order of their upload capacity. This also increases the possibility of adding only stable peers to the tree because, peers who join later cannot have total lifetime greater than those who joined in the initial half of the streaming session and stay till the end.

Handling tree peer departure Since the tree peers receive prioritized service by either connecting to the server or other peers with high upload capacity, the probability of their departure is small. The tree peer may leave the overlay only due to lack of interest in the content. To preserve stream continuity, each tree peer maintains information of its ancestors as well as its descendants. The CDN server also maintains a record of all the tree peers who join its sub-overlay along with their parent and child information. When a new tree peer joins the sub-overlay, it sends its address to its descendants upto two hops, which is also recorded at the CDN server. In response, the descendant peers also send the address along with their parent's address. If a tree peer leaves the overlay then the orphaned child along with its descendants join its grandparent. In a very unlikely scenario, when a chain of connected tree peers leave the overlay, the orphaned peers contact the CDN server for alternative parent. Therefore, stream recovery is quick in case of tree peer departure. The tree peers can also temporarily fetch chunks from the server if they run out of chunks buffered.

Utilizing residual upload capacity of tree peers The residual upload capacity of a tree peer after serving one stream to its child is utilized to serve sub-streams to mesh peers. A tree peer is considered stable when the elapsed session duration is greater than half of the average value in the same sub-overlay. Until a tree peer becomes stable, its residual upload capacity is used by the server to create virtual sources to serve startup chunks to newly joined mesh peers (details are discussed in the next section).

Overall, the construction of extended CDN tree helps in utilizing upload capacity of peers to ensure QoS. It also utilizes the upload capacity of newly joined, not yet stable, tree peers for providing startup chunks to the mesh peers.

3.3.2 Building and maintaining mesh topology

First, each new peer distributes its upload and download capacities among different sub-overlays of the selected CDN servers. Each server creates a virtual source that provides startup chunks to the new peer. Servers also suggest a list of prospective partners to the new peer. Next, the new peer selects its partners considering their serviceability. Each mesh peer also use a topology adaptation strategy to replace partners that do not provide the desired quality of stream. With these operations, quick startup and the desired streaming quality are ensured to the mesh peers. Next, we explain these operations in detail.

Upload and download capacity allocation of peers When a new peer receives offers to join as a mesh peer, it distributes its upload and download capacity among sub-overlays of those servers. The upload and download capacities of a peer are allocated in terms of the number of sub-streams it uploads and downloads from sub-overlays.

For this, the selected servers (A and B) first calculate the aggregate upload and download capacities of their sub-overlays ($subA$ and $subB$) an then send this information to the new peer. The aggregate upload capacity of a sub-overlay is calculated as the sum of total upload capacities allocated by the peers in that sub-overlay and the corresponding server. The aggregate download capacity of a sub-overlay is the sum of total download capacities of the peers in that sub-overlay. Let U^a_{subA} and U^a_{subB} be the aggregate upload capacities and let D^a_{subA} and D^a_{subB} be the aggregate download capacities of the sub-overlays. Each new peer first calculates the fraction by which its upload and download capacities are divided among sub-overlays of selected servers (denoted as β). β is calculated considering the supply and demand ratio of upload capacities in the sub-overlays, as

$$\beta = \frac{\left(U^a_{subA}/D^a_{subA}\right) - \left(U^a_{subB}/D^a_{subB}\right)}{\left(U^a_{subA}/D^a_{subA}\right) + \left(U^a_{subB}/D^a_{subB}\right)} \quad (1)$$

Let the aggregate upload and download capacities of a new peer n be denoted by U^a_n and D^a_n, respectively. If D^a_n is greater than the streaming rate then it is set equal to the streaming rate. Let U^A_n and U^B_n be the fractions of upload capacity of the new mesh peer allocated to servers A and B, respectively with $U^a_n = U^A_n + U^B_n$. Similarly, let D^A_n and D^B_n be the fractions of download capacity of the new mesh peer allocated to servers A and B, respectively with $D^a_n = D^A_n + D^B_n$. The rules for calculation of these parameters are given in Table 1.

The new peer allocates a higher fraction of its upload capacity to the sub-overlay where the ratio of aggregate upload and download capacities is smaller. Contrarily, a higher fraction of its download capacity is allocated to the sub-overlay where the ratio of aggregate upload and download capacities of the sub-overlay is larger. This allocation balances the stream delivery load of peers and

Table 1 Calculating U_n^A, U_n^B, D_n^A and D_n^B

Condition	Value of U_n^A	Value of U_n^B	Value of D_n^A	Value of D_n^B
If $\beta > 0$ & $\beta < 1-\beta$	$\|\| \beta \times U_n^a \|\|$	$\|\| (1-\beta) \times U_n^a \|\|$	$\|\| (1-\beta) \times D_n^a \|\|$	$\|\| \beta \times D_n^a \|\|$
If $\beta > 0$ & $\beta > 1-\beta$	$\|\| (1-\beta) \times U_n^a \|\|$	$\|\| \beta \times U_n^a \|\|$	$\|\| \beta \times D_n^a \|\|$	$\|\| (1-\beta) \times D_n^a \|\|$
If $\beta < 0$ & $\|\beta\| < 1-\|\beta\|$	$\|\| (1-\|\beta\|) \times U_n^a \|\|$	$\|\|\| \beta \| \times U_n^a \|\|$	$\|\|\| \beta \| \times D_n^a \|\|$	$\|\| (1-\|\beta\|) \times D_n^a \|\|$
If $\beta < 0$ & $\|\beta\| > 1-\|\beta\|$	$\|\|\| \beta \| \times U_n^a \|\|$	$\|\| (1-\|\beta\|) \times U_n^a \|\|$	$\|\| (1-\|\beta\|) \times D_n^a \|\|$	$\|\|\| \beta \| \times D_n^a \|\|$
If $\beta = 0$	$\|\| U_n^a/2 \|\|$	$\|\| U_n^a/2 \|\|$	$\|\| D_n^a/2 \|\|$	$\|\| D_n^a/2 \|\|$

servers in the sub-overlays and also ensures better QoS, because more sub-streams are downloaded from lightly loaded sub-overlays. Next, we discuss the procedure used by a CDN server to create, maintain and utilize virtual sources to give startup chunks to the mesh peers.

Virtual sources A virtual source consists of a group of peers providing a sub-stream to the new mesh peer. Servers designate virtual sources by reserving a portion of upload capacity of a few peers in the sub-overlay. The virtual sources ensure quick startup with desired streaming quality to the mesh peers and prevent the initial departure of peers. A server creates a virtual source when a new peer joins the mesh sub-overlay. The number of peers in a virtual source is equal to the number of sub-streams the new peer can download. To find these peers, the server maintains a reserved peers list that consist of peers who joined sub-overlay before the concerned new peer. The peers are arranged in the list in the decreasing order of their joining time from top to bottom. The server selects a larger fraction of peers from the top of the list to include both newly joined and stable peers. The stable peers ensure consistent delivery of stream to the new peer. The selection of larger fraction of new peers improves their upload capacity utilization.

Maintaining virtual sources The CDN servers continue adding newly joined peers to the reserved peers list as soon as they receive the stream. A newly joined mesh peer is added to the list after reserving its upload capacity worth one sub-stream whereas, a newly joined tree peer is added after reserving its residual upload capacity. As explained earlier (see Section 3.3.1), the residual upload capacity of a tree peer is reserved and utilized to create virtual sources until it becomes stable. This ensures better streaming quality to mesh peers during startup because tree peers are connected closer to the CDN servers.

The peers in the list are maintained considering peer arrival rate, average streaming rate and reserved upload capacity of peers. Let the arrival rate be λ peers/unit time and the desired average streaming rate be R chunks/unit time. Let X be the set of peers available in the reserved peers list and X_i be the amount of reserved upload capacity of a peer i, $i \in X$. The size of the reserved peers list is maintained such that the inequality in Eq. 2 is satisfied throughout the streaming session.

$$\lambda \times R < \sum_{i=1}^{|X|} |X_i| \tag{2}$$

The above condition reserves upload capacity for serving startup chunks to mesh peers until the peer arrival rate either stabilizes or decreases. Once the peer arrival rate reduces, peers from the bottom of the list are released to prioritize the release of upload capacity of peers with longer elapsed session duration. This ensures that upload capacity of stable peers is utilized more to serve sub-streams to mesh peers by becoming their partners.

Stream delivery using virtual sources Each peer who is a part of the virtual source (group) delivers its best quality sub-stream to the new peer until the new peer selects partners and receives the stream from them. It finds out the best quality sub-stream using the number of chunks received before the playback deadline. Let the chunks be generated at the rate of C chunks/unit time. $q_{i,j}$ be the quality of the i^{th} sub-stream received by a peer j. $N_{i,j}$ be the number of chunks of i^{th} sub-stream received by a peer j before playback deadline. l_j be the elapsed session duration of j^{th} peer and M be the total number of sub-streams created by the source. Equation 3 defines the quality of a sub-stream as the ratio of number of chunks received by a peer before the playback deadline to the total number of chunks generated at the source for the same sub-stream.

$$q_{i,j} = \frac{N_{i,j}}{(C/M) \times l_j}, 1 \le i \le M \tag{3}$$

Each peer j delivers the i^{th} sub-stream to maximize $q_{i,j}$, $i \in \{1 \ldots M\}$. If more than one peer of the designated virtual source delivers the same sub-stream to the new peer, then the new peer requests the peer with lower quality sub-stream to deliver an another required sub-stream. When a virtual source is not able to provide desired quality to the new peer, it fetches missing chunks from the server.

Overall, virtual sources ensure quick startup with the desired streaming quality to new mesh peers by utilizing upload capacity of the existing peers. It reduces the stream

delivery load of servers and utilizes the upload capacity of peers with shorter session duration and lower upload capacity.

Serviceability based peer selection The serviceability of a peer is measured considering its stability, streaming quality, upload and download capacities. It represents the ability of a peer to deliver a desired quality of stream to other peers consistently.

First, each new mesh peer receives the list of prospective partners from the selected servers and the virtual source. The servers suggest tree peers with free upload capacity and peers of the virtual source. Each peer of the virtual source also suggests its parents or partners as prospective partners to the new peer. The suggested list of prospective partners altogether consists of tree peers, new peers and stable peers. This helps a new peer to select partners that have chunks in the buffers, free upload capacity and are consistent in providing good quality stream.

The new peer use this list to find the partners with residual upload capacity worth one sub-stream. Then, it calculates their serviceability using the elapsed session duration, download capacity, received streaming quality, and total stream duration. The new peer select prospective partners with highest serviceability value as partners. These partners are further categorized as sub-stream partners and backup partners. A sub-stream partner continuously pushes the selected sub-stream(s) to the peer, whereas backup partners are used to recover the missing chunks due to packet loss or partner departure.

Let P be the subset of prospective partners with residual upload capacity worth one sub-stream, l_i be the elapsed session duration of the i^{th} peer, l_e be the elapsed stream duration and l be the total stream duration. Let N_i be the number of chunks received by the i^{th} peer before playback deadline, Q_i be the quality of stream received by the i^{th} peer and D_i be its download capacity. A peer first calculates the streaming quality of i^{th} peer, $i \in P$ as the ratio of the total number of chunks received by the peer before the playback deadline to the total number of chunks generated. The following equation defines this computation.

$$Q_i = \frac{N_i}{C \times l_i}, \forall i \in P \tag{4}$$

Next, the serviceability value of i^{th} peer, $i \in P$ is calculated as

$$S_i = \frac{l_e}{l}\left(\frac{l_i}{\max_{\forall j \in P}(l_j)}\right) + \left(1 - \frac{l_e}{l}\right)\left(\frac{Q_i}{\max_{\forall j \in P}(Q_j)} \times \frac{D_i}{\min\left(\max_{\forall j \in P}(D_j), C\right)}\right) \tag{5}$$

The serviceability is calculated as the weighted sum of the normalized values of the elapsed session duration, the received stream quality and the download capacity. The weights consider the elapsed and the total stream durations.

The rationale behind the selection of the parameters to calculate serviceability of a peer is as follows. The stability of a peer depends on the interest of user in the content and its quality, which can be predicted using the elapsed session duration, received quality of stream and the download capacity. It is assumed that the peers who stay for a longer duration are satisfied with the streaming quality. The streaming quality and download capacity also affect the probability of peer departure. The peer with higher download capacity has higher degree of connectivity and receives better quality, thereby remains unaffected by peer departure. The received quality also indicates the availability of chunks in the buffer, which predicts the ability to deliver better quality. The ratio of elapsed stream duration to the total duration gives higher weightage to the quality of stream and download capacity of the peer at the beginning of the session. This is due to the fact that all the peers have almost equal (and short) elapsed session duration initially, which cannot be used to predict satisfaction of the peers. With the progress in streaming session, the elapsed session duration of peers is diversified and can be used to predict the interest and satisfaction of peer with the content and quality of stream.

Selecting sub-stream and backup partners After computing the serviceability, the new mesh peer selects its sub-stream partners and backup partners. A partner that provides a sub-stream with the highest quality becomes a sub-stream partner. The number of sub-stream partners selected is equal to the number of sub-streams a mesh peer can download, because each partner provides only one sub-stream. The mesh peer also maintains a list of all the partners providing next best quality of the required sub-streams as backup partners.

When a sub-stream partner leaves, the mesh peer temporarily pulls chunks from the backup partners and also searches for another sub-stream partner based on the serviceability. Each mesh peer also maintains backup partners list because they play a significant role in maintaining quality and stability during churn. This strategy prevents peers from experiencing stream quality degradation for longer duration and provides faster recovery.

Topology adaptation To maintain streaming quality during churn, peers relocate themselves in the overlay by replacing their partners based on *utility value*. The partners with lowest utility value are replaced first. Let N_i^t be the number of sub-streams delivered by the i^{th} peer at time t. The utility

T_i of the i^{th} peer is measured as the sum of its serviceability value and upload capacity utilization, as defined by

$$T_i = \frac{\sum_{t=0}^{l_i} N_i^t}{l_i \times U_i^a} + S_i \tag{6}$$

In Eq. 6, the ratio of number of sub-streams delivered by a peer to the total number it could have delivered (using its aggregate upload capacity) indicates the upload capacity utilization. A peer replaces its partners to connect with those who have better stability, streaming quality and upload capacity utilization.

The adaptation process is triggered either when a peer retrieves more than fifty percent of chunks from its backup partners instead of sub-stream partners or the servers trigger adaptation based on the peer arrival rate. In the former case, the peer initiates the adaptation process whereas, in the latter case, the server sends a trigger message to few of the newly joined and geographically diverse peers each time after ten percent peers join its sub-overlay. The newly joined peers forward the trigger message to their parent partners and the receiving partners initiate the adaptation process. The steps of adaptation process are described as follows:

- **Step-1:** Each peer initiating the adaptation first calculates its own utility value (using Eq. 6), which is advertised to its parents up to two hops. The utility advertisement has sender address, utility value and a hopcount flag set to two.
- **Step-2:** When a peer receives a utility advertisement, it first decrements the hopcount flag by one and then forwards it to parents, unless the flag becomes zero. Every time a peer receives a utility advertisement, it compares its own utility with the utility of sender. If its utility is smaller, then it starts adaptation process and advertises its utility value using the same procedure. Otherwise if the sender is not its child, it compares the utility values of its children with that of sender's. If it finds a child with smaller utility value, then it sends a *JoinMe* message to the sender with its utility value. Else, it discards the received utility advertisement.
- **Step-3:** When a peer receives *JoinMe* message(s), it compares the received utility values with that of its parents. It selects peers with highest utility value as its parent and replaces the existing parents if they have smaller utility value. The peer then sends a *JoinedYou* message to the new parents and retrieves best quality sub-streams from them. The *JoinedYou* message also consists of a list of peers rejected by the peer as parents.
- **Step-4:** When a peer receives a *JoinedYou* message in response to a *JoinMe* message, it abandons its child with least utility value and adds the sender of the *JoinedYou* message as a new child. It also sends the list of its current children and the list of peers available in the *JoinedYou* message to the abandoned child to help it find another parent.
- **Step-5:** Every time a peer replaces a parent, it starts a new adaptation and continues to do so until it receives no *JoinMe* messages in response.

The topology adaptation strategy maintains the streaming quality throughout the session by updating the partners. It also relocates peers with higher serviceability and upload capacity utilization closer to the server to ensure better QoS to all the peers. The message passing overhead of this strategy is also lower compared to the schemes where each peer triggers adaptation periodically irrespective of churn rate and peer population. This strategy frequently updates the overlay when the peer arrival rate is high. As the peer population increases and the peer arrival rate decreases, the frequency of adaptation is reduced to lower the message overhead.

3.4 Overhead and message complexity

When a new peer joins it receives a message from the joined CDN server(s) with information on streaming rate in terms of number of chunks per unit time (C), number of sub-streams generated by the source (M), total stream duration (l) and elapsed stream duration (l_e). The new peer in turn informs the server about its aggregate or allocated upload and download capacity. Suppose there are N peers joining the system. Then the number of messages exchanged between peers and CDN servers is in $O(N)$.

Each peer also exchange messages with its prospective partners to compute their serviceability. The new peer requests its prospective partners to provide information on session duration (l_i), number of chunks received before playback deadline (N_i) and download capacity (D_i). Later, peers also calculate and exchange their own utility value with their partners to maintain the streaming quality. The number of messages exchanged among peers is also in $O(N)$. Overall, the overhead of collecting all the information needed to implement proposed overlay management strategy is in $O(N)$ when N peers join the system.

4 Performance evaluation

We evaluate SOMS through simulations and compare its performance with three existing CDN-P2P live streaming systems i.e., PROSE [22], LiveSky [23], and AERO [24]. We compare AERO [24] only for upload capacity utilization of peers and stream delivery load of servers, but omit comparison for other parameters because AERO does not focus on improving the QoS of peers.

4.1 Evaluation scenario

SOMS is implemented in OMNeT++ [32]. We use OverSim [33] framework to simulate P2P overlay and INET [34] framework to simulate underlay TCP/IP network architecture with UDP protocol at the transport layer. The simulated network consists of a source server, a CDN load manager, six CDN servers and several peers. 20,000 peers join the system during a 100-minute streaming session. Each CDN server has an upload capacity of 200 Mbps [23]. The upload capacity distribution of peers is shown in Table 2, where approximately thirty percent of the peers are free riders and the total upload contribution of peers is approximately forty percent of the upload capacity of the system [23]. The source server uses Scalable Video Codec (SVC) standard [35] to encode the video stream and supports streaming rate upto 1 Mbps. The source generates five sub-streams. The peers download different number of sub-streams according to the desired streaming quality. The peers are categorized into three groups based on the desired streaming rate and the download capacity, as reported in Table 3 [36].

We simulate flash crowd during the initial ten percent duration of the session and peer departure throughout [37]. Peers arrive following Poisson distribution with mean inter-arrival time between 0.5 and 5 seconds [36, 38]. The session duration of peers follows Pareto distribution [20]. Peers depart either due to unacceptable QoS or after staying for its specified session duration. A peer waits for one minute after QoS deterioration and leaves the system afterwards if it cannot recover QoS. The peers are set to randomly tolerate 10 to 20 percent degradation in desired streaming rate and 15 to 45 seconds of startup delay [37, 39, 40].

4.2 Evaluation metrics

The performance of SOMS is evaluated using metrics primarily related to QoS and upload capacity utilization. We also assess the stream delivery load of servers since it affects the QoS and affected by the upload capacity utilization of peers. The main QoS parameters for evaluation are startup delay and streaming quality. In addition, the stability of peers, upload contribution of peers, and early departure rate are used to understand the impact of QoS. We define the metrics used for evaluation below.

Table 2 Upload capacity distribution of peers

Peers %	Upload capacity
30	0 Kbps
50	512 Kbps
8	1.4 Mbps
12	1.6 Mbps

Table 3 Download capacity and desired streaming rate of peers

Peers %	Download capacity	Desired streaming rate
56	512 Kbps	503 Kbps
30	1 Mbps	705.3 Kbps
14	3 Mbps	905.8 Kbps

- **Upload capacity utilization:** It is the ratio of total number of chunks uploaded by peers to the maximum number that could have been uploaded.
- **Startup delay:** It is the duration for which peers wait to start playback after joining the system.
- **Streaming quality:** It is the percentage of chunks delivered to a peer before the playback deadline compared to the total number that should have been received for a desired streaming rate.
- **Server stream delivery load:** It is the percentage of peers, not directly connected to the server, that retrieve chunks from the CDN server either for quick startup or for better quality.
- **Stream recovery time:** It is the duration for which peers experience degradation in streaming quality before recovering the desired quality.
- **Peer stability:** It is the ratio of actual session duration to the maximum session duration of the peer. The maximum session duration is the time span between the peer joining time and the end of streaming session.
- **Early departure rate:** It is the percentage of peers leaving before the end of the session due to unacceptable startup delay and/or streaming quality.
- **Peer upload contribution:** It is the ratio of total upload capacity contributed by peers to the total upload capacity of the system.

4.3 Results and discussions

First, we examine the upload capacity utilization of peers and then discuss its impact on QoS as well as the stream delivery load of servers. Next, we examine the impact of QoS on upload contribution and stability of peers.

Figure 3 shows the upload capacity utilization of peers with the SOMS, PROSE, LiveSky and AERO. SOMS exhibits better upload capacity utilization as compared to others, because it organizes peers in the overlay considering their serviceability. The peers with higher upload capacities and longer session duration are used for extended CDN tree and peer partners, whereas others are used to create virtual sources. On the other hand, PROSE and LiveSky do not arrange peers in the overlay so that they utilize

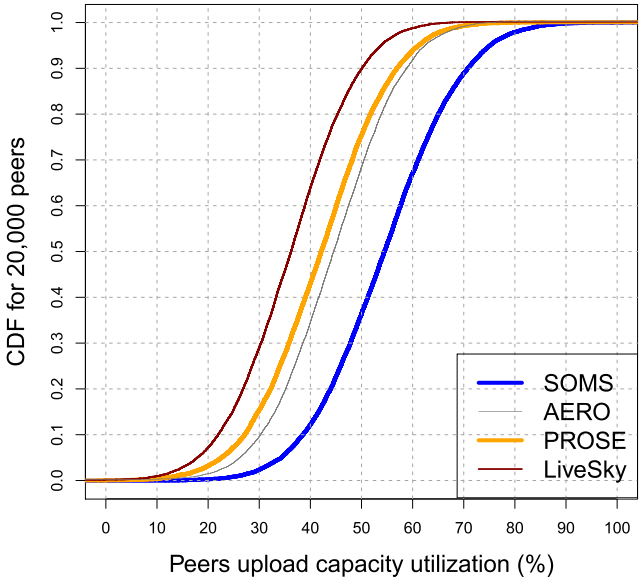


Fig. 3 Upload capacity utilization of peers

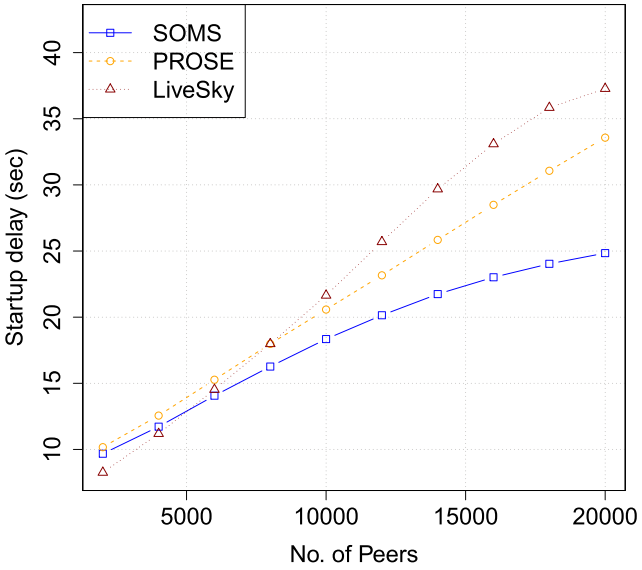


Fig. 4 Startup delay

upload capacity of all the peers. AERO only focuses on optimizing number of streams seeded through CDN servers adaptively as per the emergency chunk requests received to increase upload capacity utilization of peers. The CDN servers proactively increase the number of seeds when they receive more emergency chunk requests. However, this does not force peers to retrieve more chunks from other peers of the overlay and does not ensure upload capacity utilization of all the peers in the overlay. The distribution of peers among sub-overlays also results in better utilization in SOMS. No such mechanism exists in PROSE, LiveSky and AERO. Overall, we see that there is 28-30% improvement in the upload capacity utilization of peers.

Figures 4 and 5 show the startup delay and streaming quality perceived by the peers, respectively. The startup delay increases with the number of peers due to increase in stream delivery load of CDN servers and existing peers. It is also observed that when the number of peers is smaller, the startup delay of LiveSky is better because, the CDN servers initially provide startup chunks to new peers. In PROSE and SOMS new peers receive startup chunks from other existing peers of the system. SOMS still exhibits lower startup delay than PROSE because, it creates a dedicated virtual source for each new peer to provide startup chunks to the new peer.

Figure 4 also shows SOMS exhibits lower startup delay when a large number of peers join the system. LiveSky exhibits higher startup delay because all the new peers get startup chunks from the CDN server that causes overload. In PROSE, the existing peers do not provide the desired streaming quality to new peers during startup, because the CDN server suggests peers randomly without considering upload capacity and data availability in the buffer. In SOMS, the startup chunks are served by the virtual sources to avoid the CDN server overloading. The selection of CDN servers considering the fraction of peers experiencing higher joining delay and poorer streaming quality also ensured quick startup with desired streaming quality. With SOMS, startup delay reduces by about 20% compared to PROSE and LiveSky.

Figure 5 compares the streaming quality perceived by the peers with SOMS, PROSE and LiveSky. It is observed

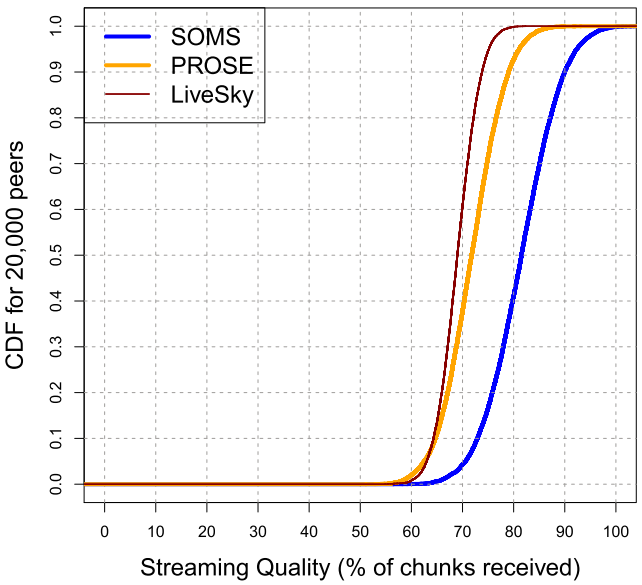


Fig. 5 Streaming quality

that PROSE gives better streaming quality than LiveSky because, CDN servers pro-actively forward the stream to peers with high upload capacity. SOMS exhibits better streaming quality than both because it generates more high capacity seeders in the overlay (with extended CDN tree) in addition to what is done in PROSE. It also selects peers considering their serviceability whereas, partners are selected randomly in LiveSky and PROSE. SOMS also prevents quality degradation by getting missing chunks from the backup partners and topology adaptation to re-select partners that give better streaming quality. In LiveSky and PROSE, the peers depend on only CDN servers that increases the stream delivery load on the CDN servers further. It is found that SOMS improves perceived streaming quality of peers by 25% compared to PROSE and LiveSky by efficiently utilizing the upload capacities of peers.

To study the improvement in QoS due to better upload capacity utilization of peers, we measured the stream delivery load of CDN servers, which is reported in Fig. 6. It is evident from the results that stream delivery load of CDN servers is approximately reduced by 18-20% using SOMS as compared to LiveSky, PROSE and AERO. This is because responsibility of providing startup chunks is delegated to peers by creating virtual sources, whereas peers in PROSE, LiveSky and AERO receive startup from the CDN server. Retrieval of missing chunks from CDN servers due to stream quality degradation is reduced with serviceability-based peer selection, backup partners and topology adaptation. This also results in significantly lower number of emergency chunk requests received by CDN servers in SOMS as compared to AERO. Additionally, there is an even distribution of the stream delivery load on servers based on supply-demand ratio of upload capacities in the overlay.

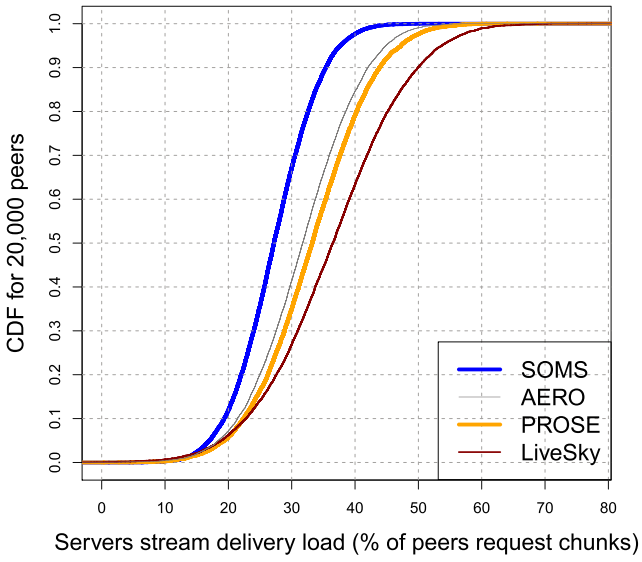


Fig. 6 Stream delivery load of CDN servers

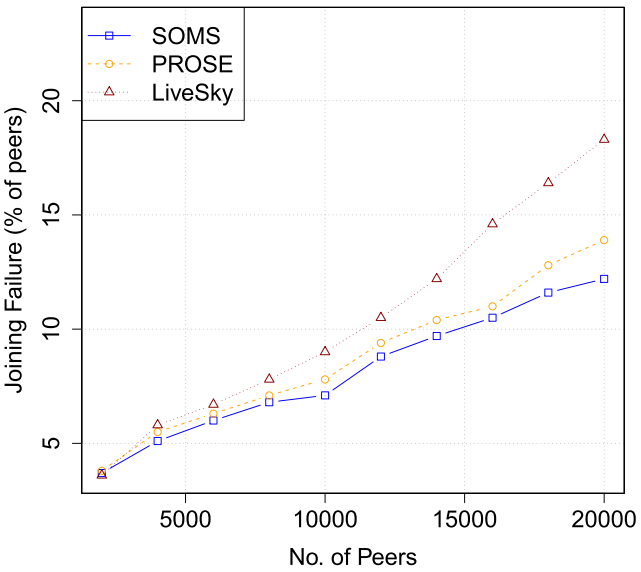


Fig. 7 Percentage of peers facing joining failure

The departure of peers due to unacceptable startup delay is termed joining failure. Figure 7 shows percentage of peers departing (due to joining failure). It is observed that this follows similar trend as startup delay because lower the startup delay experienced by peers, lower is the departure of peers. Figure 8 compares the percentage of peers leaving due to stream quality degradation in all the three strategies.

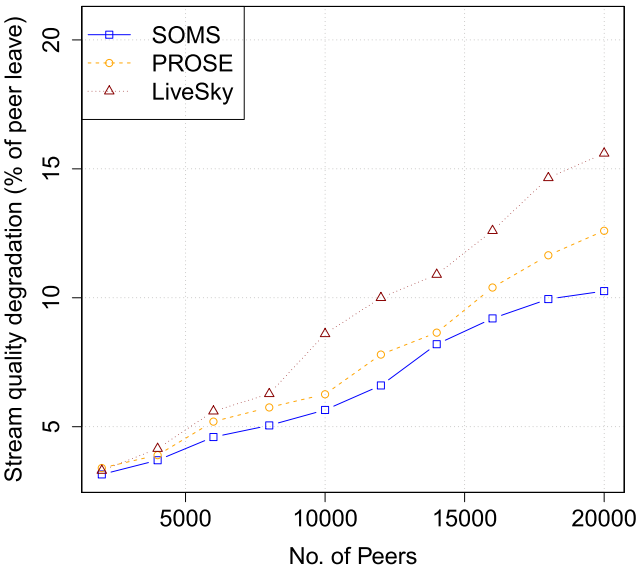


Fig. 8 Peer departure due to stream quality degradation

It is observed that SOMS results in lower departure rate of peers due to stream quality degradation, following the exactly opposite trend as streaming quality experienced by peers.

It is also found that percentage of peers leaving due to joining failure is higher than that due to quality degradation with SOMS. This is because when peers do not provide desired streaming quality, the backup partners are used to retrieve chunks and topology adaptation replaces the partners. Moreover, lower stream delivery load of CDN servers helps in reducing early departure rate of peers. We see that SOMS results in approximately 30% reduction in early departure rate of peers, which is reported in Fig. 9.

To substantiate these claims, we show the percentage of peers that recovered from unacceptable streaming quality and the time taken for recovery in Figs. 10 and 11, respectively. We see in Fig. 10 that an additional 12% peers recover from unacceptable streaming quality with SOMS compared to PROSE and LiveSky. Figure 11 shows that the time taken by peers to recover the desired streaming quality is also smallest with our strategy. These improvements are due to lower stream delivery load of CDN servers, quick service from backup partners and topology adaptation.

We also show the impact of QoS perceived on the stability and upload contribution of peers in Fig. 12. Figure 12a shows that the stability of peers is approximately 20% higher with SOMS because, peers stay longer due to improved QoS. This also improves the total upload contribution of peers in the system. Figure 12b compares the upload capacity contribution of peers with the ideal contribution of peers, which is the total upload capacity contributed if the peers stay till the end.

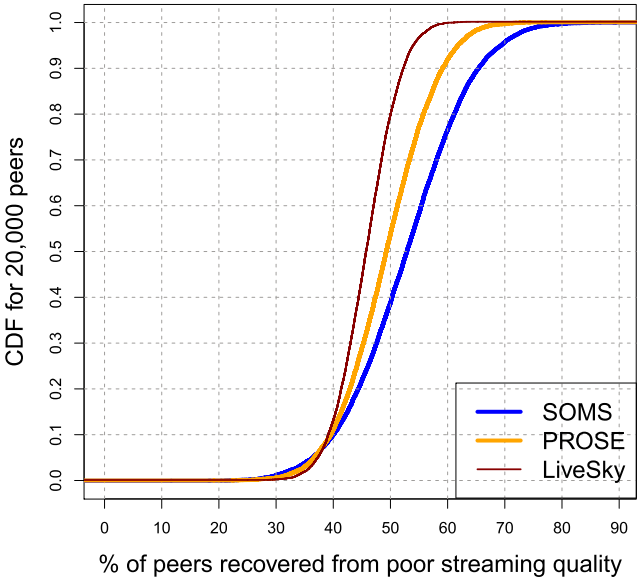


Fig. 10 Percentage of peers recovered from poor streaming quality

It is clear that the upload contribution of peers is higher as well as closer to the ideal contribution. We also found that upload capacity contributed by the tree peers is higher than that by the mesh peers because, tree peers get better QoS being closer to the CDN servers and they also stay longer. Figure 12c compares the loss in upload contribution of peers due to poor QoS and departure of peers. It is observed that SOMS shows a loss of approximately 22%, whereas PROSE and LiveSky show a loss of approximately 32% and 38%, respectively. Therefore, both the stability of peers and upload contribution of peers are improved using SOMS.

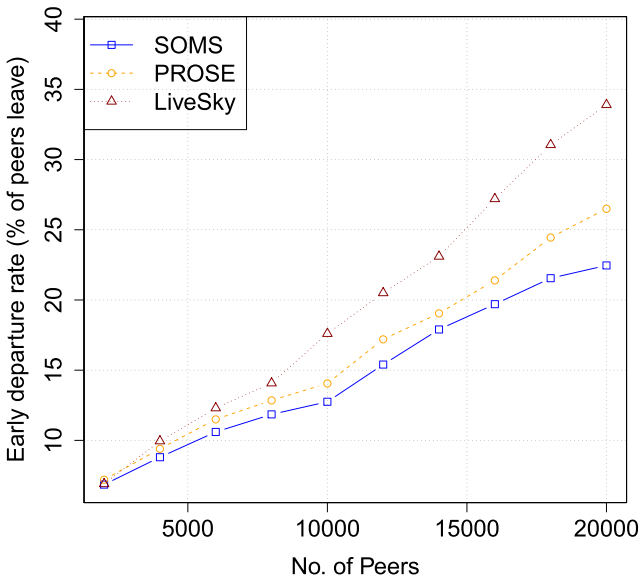


Fig. 9 Early departure rate of peers due to poor QoS

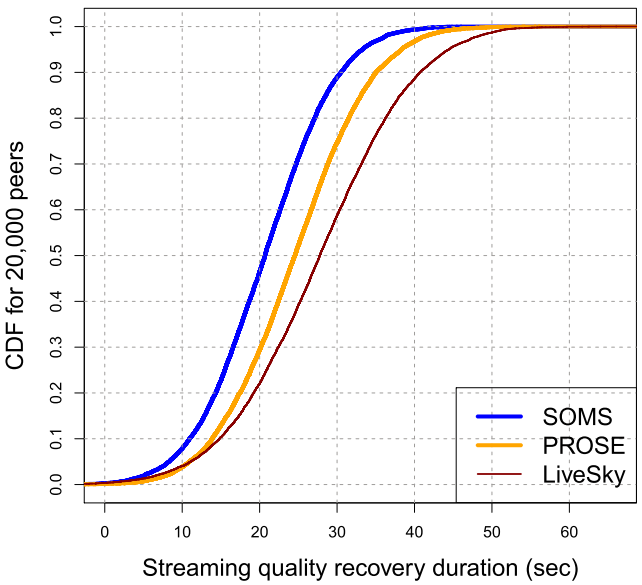
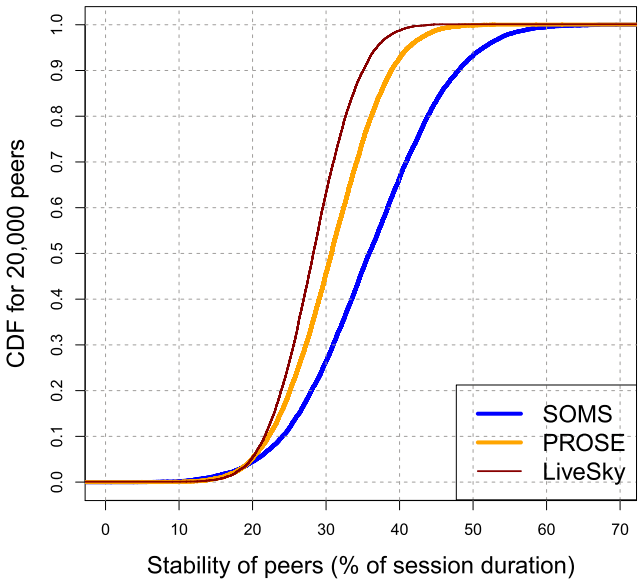
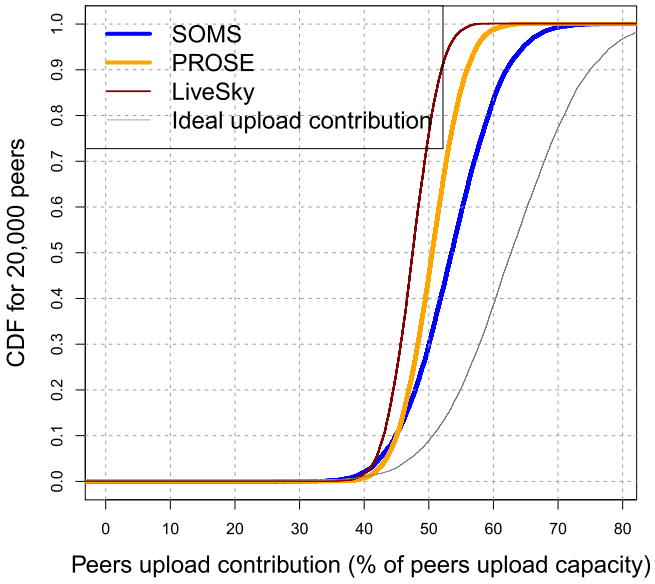


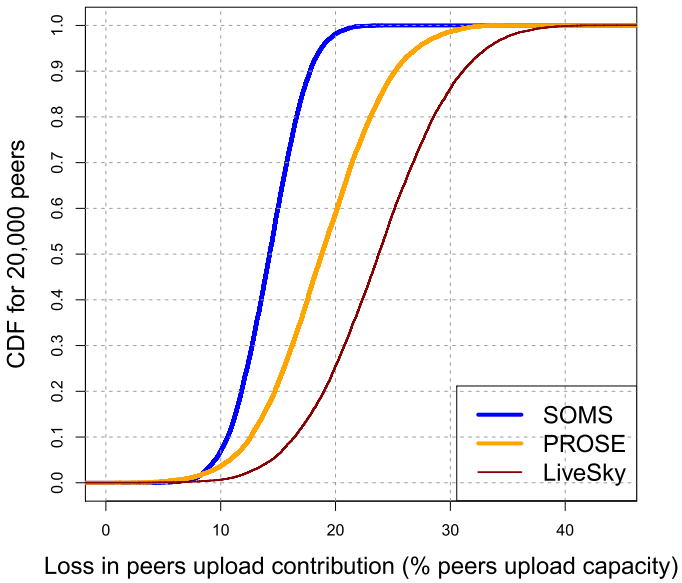
Fig. 11 Time taken for recovery from degradation of streaming quality



(a) CDF of stability of peers



(b) CDF of upload contribution of peers



(c) CDF of loss in upload contribution

Fig. 12 Comparing stability, upload contribution and loss in upload contribution of peers due to QoS and peer departure

5 Conclusion

In this work, we proposed a serviceability-aware overlay management strategy (SOMS) for CDN-P2P live streaming systems which exploited the serviceability of peers to improve QoS and offload CDN servers. In SOMS, the peers are arranged to form a hybrid tree-mesh overlay topology based on their serviceability. It utilizes peers with heterogeneous upload capacity as well as session duration to provide streaming services. The peers with higher upload capacity are added to the extended CDN tree which generated stable and high capacity seeders. The peers with lower upload capacity and/or shorter session duration used to create virtual sources, that provide startup chunks to new peers. This reduced startup delay experienced by peers and offloaded CDN servers from providing startup chunks to

new peers. The mesh peers selected peers with long session duration and high upload capacity as their parent partners to improve streaming quality. The topology adaptation strategy used by mesh peers also helped in maintaining QoS during churn and offloading CDN servers from providing missing chunks. The CDN server selection strategy and bandwidth allocation mechanism of peers also balanced the stream delivery load of CDN servers and peers in the overlay and improved QoS.

The performance of SOMS is evaluated by comparing it with PROSE, LiveSky, and AERO. Results show that SOMS improved startup delay and streaming quality by 20% and 25%, respectively. It also improved the stability of peers by 20% leading to an improvement in their upload contribution by approximately 15%. The utilization of upload capacity of peers is enhanced by 30%, which also reduced the stream delivery load of CDN servers by approximately 20%.

References

1. Cisco Visual Networking Index (2017) Forecast and methodology, 2016 - 2021. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html
2. Kim Y, Kim Y, Yoon H, Yeom I (2015) Peer-assisted multimedia delivery using periodic multicast. Inform Sci 298(C):425–446
3. Wendell P, Freedman MJ (2011) Going viral: going flash crowds in an open CDN. In: ACM SIGCOMM conference on internet measurement conference, pp 549–558
4. Xi L, Dobrian F, Milner H, Jiang J, Sekar V, Stoica I, Zhang H (2012) A case for a coordinated internet video control plane. In: ACM SIGCOMM conference on applications, technologies, architectures, and protocols for computer communication, pp 359–370
5. Simmons J (2018) Broadcasting the World Cup and Wimbledon in UHD - the full story. http://www.bbc.co.uk/blogs/internet/entries/5c5b8f80-891d-4b51-babd-8814c1511b4e
6. Liu Z, Ding Y, Liu Y, Ross K (2012) Peer-assisted distribution of user generated content. In: IEEE international conference on peer-to-peer computing (P2P), pp 261–272
7. Karamshuk D, Sastry N, Secker A, Chandaria J (2015) ISP-friendly peer-assisted on-demand streaming of long duration content in BBC iPlayer. In: IEEE conference on computer communications (INFOCOM), pp 289–297
8. Zhang G, Liu W, Hei X, Cheng W (2015) Unreeling Xunlei Kankan: understanding hybrid CDN-P2P video-on-demand streaming. IEEE Trans Multimed 17(2):229–242
9. Anjum N, Karamshuk D, Shikh-Bahaei M, Sastry N (2017) Survey on peer-assisted content delivery networks. Comput Netw 116(C):79–95
10. Balachandran A, Sekar V, Akella A, Seshan S (2013) Analyzing the potential benefits of CDN augmentation strategies for internet video workloads. In: ACM SIGCOMM internet measurement conference, pp 43–56
11. Zhao M, Aditya P, Chen A, Lin Y, Haeberlen A, Druschel P, Maggs B, Wishon B, Ponec M (2013) Peer-assisted content distribution in Akamai Netsession. In: ACM SIGCOMM internet measurement conference, pp 31–42
12. Huang C, Wang A, Li J, Ross KW (2008) Understanding hybrid CDN-P2P: why limelight needs its own red swoosh. In: International workshop on network and operating systems support for digital audio and video, pp 75–80
13. Garmehi M, Analoui M (2016) Envy-free resource allocation and request routing in hybrid CDN-P2P networks. J Netw Syst Manag 24(4):884–915
14. Shen H, Li Z, Lin Y, Li J (2014) SocialTube: P2P-assisted video sharing in online social networks. IEEE Trans Parallel Distrib Syst 25(9):2428–2440
15. Passarella A (2012) A survey on content-centric technologies for the current internet: CDN and P2P solutions. Comput Commun 35(1):1–32
16. Dobrian F, Awan A, Joseph D, Ganjam A, Zhan J, Berkeley UC (2011) Understanding the impact of video quality on user engagement. In: ACM SIGCOMM, pp 362–373
17. Ullah I, Doyen G, Bonnet G, Gaiti D (2012) A survey and synthesis of user behavior measurements in P2P streaming systems. IEEE Commun Surv Tutor 14(3):734–749
18. Zhang B, Kreitz G, Isaksson M, Ubillos J, Urdaneta G, Pouwelse JA, Epema D, Trace A (2013) Understanding user behavior in Spotify. In: IEEE INFOCOM, pp 220–224
19. Hei X, Liang C, Liang J, Liu Y, Ross KW (2007) A measurement study of a large-scale P2P IPTV system. IEEE Trans Multimed 9(8):1672–1687
20. Wang F, Liu J, Xiong Y (2011) On node stability and organization in peer-to-peer video streaming systems. IEEE Syst J 5(4):440–450
21. Liu Z, Wu C, Li B, Zhao S (2009) Distilling superior peers in large-scale P2P streaming systems. In: IEEE INFOCOM, pp 82–90
22. Lv Z-H, Chen L-J, Wu J, Da D, Huang S-J, Huang Y (2013) PROSE proactive, selective CDN participation for P2P streaming, vol 28
23. Yin H, Liu X, Zhan T, Sekar V, Qiu F, Lin C, Zhang H, Li B (2010) LiveSky enhancing CDN with P2P. ACM Trans Multimed Comput Commun Appl (TOMM) 6(3):16,1–16,19
24. Oliveira JFAe, Cunha I, Miguel E, Campos SVA (2017) AERO: adaptive emergency request optimization in CDN-P2P live streaming. In: IEEE GLOBECOM, pp 1–7
25. Deng H, Xu J (2013) CorePeer: a P2P mechanism for hybridCDN-P2P architecture. In: International workshops: web-age information management, pp 278–286
26. Ullah I, Doyen G, Bonnet G, Gaiti D (2012) A Bayesian approach for user aware peer-to-peer video streaming systems. Signal Process Image Commun 27(5):438–456
27. Lu ZH, Gao XH, Huang SJ, Huang Y (2011) Scalable and reliable live streaming service through coordinating CDN and P2P. In: IEEE International conference on parallel and distributed systems, pp 581–588
28. Hu C, Chen M, Xing C, Xu B (2012) EUE principle of resource scheduling for live streaming systems underlying CDN-P2P hybrid architecture. Peer-to-Peer Network Appl 5(4):312–322
29. Ha TT, Kim J, Nam J (2017) Design and deployment of low-delay hybrid CDN-P2P architecture for live video streaming over the web. Wirel Pers Commun 94(3):513–525
30. Okada S, Fujita S (2014) P2P overlay for CDN-P2P being aware of the upload capacity of participants. In: IEEE International conference on parallel and distributed systems (ICPADS), pp 823–828
31. Budhkar S, Tamarapalli V (2016) An overlay management strategy to improve peer stability in P2P live streaming systems. In: IEEE International conference on advanced networks and telecommunications systems (ANTS), pp 1–6

32. Varga A, Hornig R (2008) An overview of the OMNeT++ simulation environment. In: Simutools, pp 1–10
33. Baumgart I, Heep B, Krause S (2007) OverSim: a flexible overlay network simulation framework. In: IEEE Global internet symposium, pp 79–84
34. Steinbach T, Kenfack HD, Korf F, Schmidt TC (2011) An extension of the OMNeT++ INET framework for simulating real-time ethernet with high accuracy. In: International conference on simulation tools and techniques, pp 375–382
35. Schwarz H, Wien M (2008) The scalable video coding extension of the H. 264/AVC standard. IEEE Signal Process Mag 25(2):135–141
36. Hu H, Guo Y, Liu Y (2011) Peer-to-peer streaming of layered video efficiency, fairness and incentive, vol 21
37. Agarwal S, Singh JP, Mavlankar A, Baccichet P, Girod B (2008) Performance and quality-of-service analysis of a live P2P video multicast session on the internet. In: Interntional workshop on quality of service, pp 11–19
38. Wichtlhuber M, Richerzhagen B, Rückert J, Hausheer D (2014) TRANSIT: supporting transitions in peer-to-peer live video streaming. In: IFIP Networking conference, pp 1–9
39. Xie S, Keung GY, Li B (2007) A measurement of a large-scale peer-to-peer live video streaming system. In: International conference on parallel processing workshops (ICPPW), pp 57–57
40. Park K, Chang D, Kim J, Yoon W, Kwon T (2010) An analysis of user dynamics in P2P live streaming services. In: IEEE International conference on communications (ICC), pp 1–6

Shilpa Budhkar is currently a Ph.D. student in the Department of Computer Science and Engineering at Indian Institute of Technology Guwahati. She received her Bachelor of Engineering (B.E.) degree from Rajiv Gandhi Technological University, Bhopal, India in 2010. Her research interests include P2P overlay networks, content delivery networks and multimedia streaming over Internet.

Venkatesh Tamarapalli received the Ph.D. degree from the Indian Institute of Technology Madras in 2009. He is an Associate Professor with the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati. He is currently serving as an Associate Editor for the Journal of Network and Systems Management (Springer). He has co-authored a book entitled An Analytical Approach to Optical Burst Switched Networks (Springer, 2009). His areas of research include wireless sensor networks, data center networks, and multimedia content delivery. He was a recipient of the Microsoft Ph.D. Fellowship and the Microsoft Young Outstanding Faculty Award.