Internet of Things and Cloud Computing
2013; 1(2): 15-22
Published online October 20, 2013 (http://www.sciencepublishinggroup.com/j/iotcc)
doi: 10.11648/j.iotcc.20130102.11

Improving streaming capacity in P2P live streaming systems via resource sharing

Yifeng He*, Shujjat Ahmed Khan

Electrical and Computer Engineering, Ryerson University, Toronto, Canada

Email address:
yhe@ee.ryerson.ca (Yifeng He)

To cite this article:
Yifeng He, Shujjat Ahmed Khan. Improving Streaming Capacity in P2P Live Streaming Systems via Resource Sharing. *Internet of Things and Cloud Computing.* Vol. 1, No. 2, 2013, pp. 15-22. doi: 10.11648/j.iotcc.20130102.11

Abstract: Peer-to-Peer (P2P) streaming systems provide a large number of channels to users. The *streaming capacity* for a channel is defined as the maximum streaming rate that can be received by every user in the channel. In the this paper, we study the *streaming capacity problem* in both tree-based and mesh-based P2P live streaming systems. In tree-based multi-channel P2P live streaming systems, we propose a *cross-channel resource sharing* approach to improve the streaming capacity. We employ *cross-channel helpers* to establish the cross-channel overlay links, with which the unused upload bandwidths in a channel can be utilized to help the bandwidth-deficient peers in another channel, thus improving the streaming capacity. In meshed-based P2P live streaming systems, we formulate the *streaming capacity problem* into a resource optimization problem. By solving the resource optimization problem, we can optimally allocate the link rates for each peer to improve the streaming capacity. Through simulations, we demonstrate that the proposed resource sharing approaches can significantly improve the *streaming capacity* compared to the scheme without resource sharing.

Keywords: P2P Live Streaming, Streaming Capacity, Resource Sharing, Resource Optimization, Cross-Channel Helpers

1. Introduction

In recent years, video streaming services have become very popular. Video streaming services originally worked in the client/server architecture. However, this centralized architecture cannot provide streaming to a large number of users due to the limited and expensive upload bandwidth from the server. Peer-to-Peer (P2P) technology has recently become a capable approach to provide live and on-demand video streaming services over the Internet at a low cost. Commercial P2P live streaming and Video-On-Demand (VoD) systems, such as PPLive [1], PPStream [2], UUSee [3], have been successfully supporting tens of thousands of users. Apart from the large number of users, these systems have a common feature of providing a large number of channels for users to watch, and hence are referred to as multi-channel P2P streaming systems. P2P streaming systems can be categorized into P2P live streaming systems, in which the users in the same channel watch almost the same position of the video, and P2P VoD systems, in which the users in the same channel may watch different positions of the video at any time. Based on the overlay structure, P2P live streaming systems can be classified into tree-based P2P live streaming systems and mesh-based P2P live streaming systems. In a tree-Based P2P live streaming systems, a single application-layer tree or multiple application-layer trees are constructed to deliver the video streams. In mesh-based P2P live streaming systems, each peer exchanges the data with a set of neighbors.

The maximum streaming rate that can be received by every user in a channel is defined as the streaming capacity of the channel in a multi-channel P2P live streaming system [4]. The streaming capacity can be served as the indicator of video quality for a channel. A higher streaming capacity for a channel means that the users in the channel can receive a higher video quality. Therefore, the objective of the paper is to achieve a high streaming capacity for a channel in P2P live streaming systems.

It is challenging to achieve a high streaming capacity in multi-channel P2P live streaming systems. The streaming capacity for a channel is dependent on the bandwidth capacity of each peer in the channel and the overlay structure of the channel. A higher streaming capacity can be achieved by optimizing the resources among the peers within the same channel, which is, however, a challenging task. Peers have heterogeneous bandwidths. Some peers may have deficient upload bandwidths, which limit the streaming capacity, while some other peers may have

abundant upload bandwidths which have not been fully utilized. Resource sharing in a multi-channel P2P live streaming system is expected to improve the streaming capacity. However, resource sharing in multi-channel P2P live streaming systems is difficult due to the following reasons. 1) In a P2P live streaming system, each peer has different upload and download capacity, and each peer may join or leave the channel at any time. The heterogeneous characteristics and dynamic behaviors of the peers cause dynamic resource distribution among peers and among channels. 2) It is difficult to shift the unused resources in a channel to improve the streaming capacity for another channel, since there is originally no overlay connection between the two channels.

In this paper, we propose resource sharing approach to improve the streaming capacity in P2P live streaming systems. The contributions of the paper is two-fold:

1) We propose a cross-channel resource sharing approach in tree-based multi-channel P2P live streaming systems to improve the streaming capacity. The proposed approach employs cross-channel helpers to establish cross-channel overlay links, via which the unused resources in a channel can be utilized to help the bandwidth-deficient peers in another channel. The proposed scheme can significantly improve the streaming capacity compared to the case without cross-channel resource sharing.

2) We formulate the streaming capacity in mesh-based P2P streaming systems into a resource optimization problem. By solving the optimization problem, we can optimally allocate the link rates for each peer to improve the streaming capacity.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 presents the proposed cross-channel resource sharing approach to improve the streaming capacity in tree-based multi-channel P2P live streaming systems. Section 4 presents the proposed resource allocation scheme to improve the streaming capacity in mesh-based P2P streaming systems. Finally, the conclusion is drawn in Section 5.

2. Related Work

Streaming capacity in P2P live systems has been observed in the recent literature [4-7]. Most recent work was done to improve the efficiency and performance of P2P streaming systems. Lui *et al*. [8] presented algorithms that find near-optimal streaming rates when nodes can only support a bounded number of children. Picconi *et al.* [9] demonstrated that P2P live streaming systems can incorporate locality-awareness and thus be ISP-friendly. Other work [10] uses network coding for improving download speeds and reducing the insufficiency of data.

Sengupta *et al.* [4] provided a taxonomy of sixteen problem formulations on streaming capacity, depending on whether there is a single P2P session or there are multiple simultaneous sessions, whether the given topology is a full mesh graph or an arbitrary graph, whether the number of peers a node can have is bounded or not, and whether there are non-receiver relay nodes or not. Liu *et al.* investigated the performance bounds for minimum server load, maximum streaming rate, and minimum tree depth in tree-based P2P live systems [6]. The streaming capacity under node degree bound is inspected in [7].

Resource allocations in multi-channel P2P live streaming systems have been investigated in the literature. A View-Upload Decoupling (VUD) scheme was proposed in [11] to decouple what a peer uploads from what it views, bringing stability to multi-channel P2P streaming systems and enabling cross-channel resource sharing. In other papers [12, 13], the performance and the efficiency of multi-channel P2P live or VoD streaming systems were investigated using a verity of techniques.

In order to maintain the smooth streaming, peers must download enough chunks before they can start playback. Such delay before the playback is called startup delay or initial buffering delay. In [13-15], this startup delay was investigated and various solutions were proposed to overcome this issue. It was argued in [14, 16] that it is worth scaling the system to support more peers. When the average streaming rate exceeds the inherent playback rate of a movie, the extra bandwidth of the system can be used to support additional peers to make the system scalable.

Sustainable streaming rate is a key factor for a P2P streaming system. It is defined as the rate that video can be played without skips or pauses. Zhou *et al.* [13] and Zhao *et al.* [17] developed probabilistic models to characterize the playback continuity. A promising technique that may facilitate P2P video streaming is network coding [18]. Network coding was proposed to improve the throughput by making the optimal use of bandwidth resources in a network for content distribution [19]. Its effects have been studied in [20, 21], which demonstrate that random linear network coding is feasible for both P2P live streaming and P2P VoD systems.

3. Improving Streaming Capacity in Tree-Based Multi-Channel P2P Live Streaming Systems

In this section, we will present a cross-channel resource sharing approach to improve the streaming capacity in tree-based multi-channel P2P live streaming systems.

3.1. Overview of Tree-Based P2P Live Streaming System

In a tree-based multi-channel P2P live streaming system, the peers in the same channel are organized into a tree for delivering media streaming, with the media source (e.g., the streaming server) as the root of the tree. P2P streaming has become an increasingly popular approach for one-to-many multimedia streaming applications, mostly as it does not involve any particular support (e.g. IP multicast or any content distribution infrastructure) from the network. P2P live streaming is a typical media streaming service

designed for all peers receiving stream at the same time. A common idea in P2P streaming systems is that participating peers form an overlay where each peer receives content from the parent node in a session.

Any P2P streaming system consists of two diverse but correlated components that are *overlay construction* and *content delivery*. Overlay construction is a method that organizes participating peers into an overlay, and content delivery is another method that delivers the multimedia content to each participating peer through the overlay. In a tree-based multi-channel P2P live streaming system, the peers watching the same channel form a tree-based overlay, and share the resources based on parent-child relationship.

Figure 1 illustrates a tree-based multi-channel P2P live streaming system with two channels. The peers watching the same channel form a tree-based overlay, respectively, with the common root, the streaming server. The outgoing degree of the tree is 3, which means that each peer can have at most three child nodes. The video content is delivered from the root to each peer along the tree.

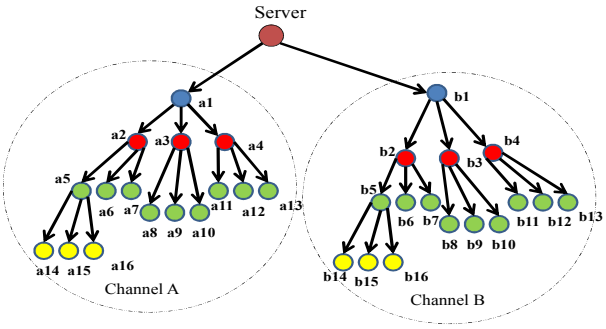

Figure 1. *Illustration of a tree-based multi-channel P2P live streaming system.*

3.2. The Proposed Cross-Channel Resource Sharing Approach

The streaming capacity for a channel in the tree-based P2P live streaming system is dependent on the upload bandwidths of the peers. For example, suppose that peer a4 in Figure 1 has an upload bandwidth of 900 Kbps and it supplies the stream to three child nodes. In this case, each child node of peer a4 can receive a streaming rate of 300 Kbps. In other words, the streaming capacity for a channel is limited by the internal peer (e.g., the peer who has child node) who has a low upload bandwidth. On the other hand, many other peers have unused upload bandwidth. The leaf nodes (e.g., the peer who has no child) do not use their upload bandwidth. Moreover, some internal nodes may have large remaining upload bandwidths which have not been used.

The streaming capacity for a channel is expected to be improved if the resources in each channel can be re-organized. Therefore, we propose a *cross-channel resource sharing approach* to improve the streaming capacity in the tree-based multi-channel P2P live streaming system. The principle idea of the proposed approach is to utilize the unused upload bandwidths of the peers in a channel to help the bandwidth-deficient peers in another channel.

The proposed approach is illustrated in Figure 2. Channels A and B are a pair of channels which help with each other. Suppose peer a4 is a bandwidth-deficient peer in channel A, peer b14 is bandwidth-abundant peer in channel B. We define a *cross-channel helper* as the peer who will contribute its remaining upload bandwidth to help the bandwidth-deficient peer in the partner channel. For example, peer b14 is a *cross-channel helper*, who downloads a segment of the stream from the server and then serves the segment to the child nodes of peer a4 in channel A. The advantage of cross-channel helper is the amplification of upload bandwidth. For example, peer b14 can download a segment of channel-A video at a rate of 50 Kbps, and serve the video to peers a11, a12, and a13 at a rate of 50 Kbps, respectively. In the same way, the bandwidth-abundant peer in channel A (e.g., peer a16) can help the bandwidth-deficient peer (e.g., peer b3) in channel B. The streaming capacity of channels A and B can both be improved by such cross-channel resource sharing.

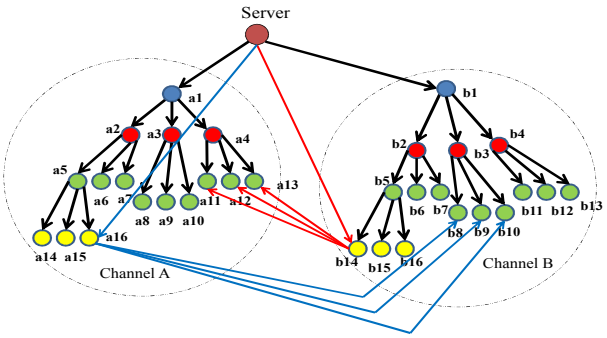

Figure 2. *Illustration of proposed cross-channel resource sharing approach*

The proposed *cross-channel resource sharing* approach consists of five steps as follows.

1) *Determine channel pair*: Each channel selects a partner channel to help with each other. We first order the channels based on the system upload bandwidth. We then group the channel pair based on a resource-balancing criterion, with which the channel with a larger system upload bandwidth is assigned to help the one with a smaller system upload bandwidth.

2) *Determine cross-channel helpers in each channel*: The peers with an unused upload bandwidth larger than a pre-set threshold are chosen as *cross-channel helpers*.

3) *Determine bandwidth-deficient peers in each channel*: The *average upload bandwidth per child* is defined as the ratio between the upload bandwidth of the internal peer and the number of its child nodes. The internal peers with an *average upload bandwidth per child* smaller than a pre-set threshold are chosen as bandwidth-deficient peers.

4) *Establish cross-channel overlay links*: The cross-channel overlay links are established by enabling the

cross-channel helper to download a segment of the stream from the server or other peer and forward the segment to the child nodes of the bandwidth-deficient peer in the partner channel.

5) *Determine the streaming capacity for each channel*: We can find the streaming capacity for each channel with cross-channel resource sharing.

Peer dynamics have an impact on the streaming capacity in tree-based multi-channel P2P live streaming systems. First, the peers may leave or join a channel dynamically. Second, the *cross-channel helpers* may leave the channel, which causes the disconnection of the cross-channel overlay links. To handle the dynamic conditions, the proposed *cross-channel resource sharing* approach needs to be performed in a discrete-time manner, considering the time-varying status of the peers.

3.3. Simulation Results for Tree-Based Live Streaming

We perform simulations for a tree-based multi-channel P2P live streaming system with two channels (channel A and channel B). The numbers of peers in channels A and B are 100 and 80, respectively. The peers have heterogeneous bandwidths. The download bandwidths of the peers are uniformly distributed between 2.5 Mbps and 4.0 Mbps, and the upload bandwidths of the peers are uniformly distributed between 1.5 Mbps and 3.0 Mbps. The tree overlay for each channel is based on the order of the arrival time of the peer. The newly coming peer chooses an existing peer as the parent and connects itself with the parent node. The outgoing degree is defined as the maximum number of the child nodes that an internal peer can have. The outgoing degree is set to 5 in the default setting.

Figure 3 compares the streaming capacity between the case without any resource sharing and the case with the proposed cross-channel resource sharing. The proposed cross-channel resource sharing better utilizes the upload bandwidths, thus significantly improving the streaming capacity compared to the case without any resource sharing. The average improvements of streaming capacity are 135% for channel A and 143% for channel B, respectively.

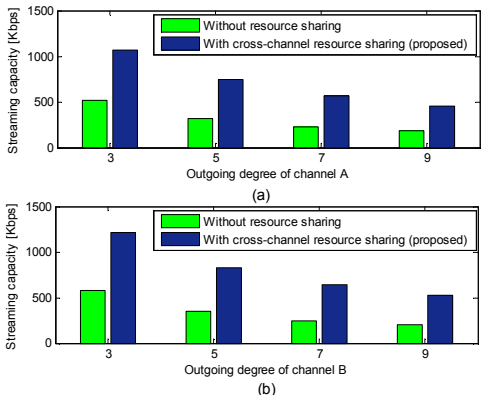

Figure 3. *Comparison of streaming capacity with different outgoing degrees: (a) channel A, and (b) channel B.*

Figure 4 compares the streaming capacity with different *total helping rate*, which is defined as the sum of the outgoing rates of all *cross-channel helpers*. We can see in Figure 4 that the streaming capacity is almost linearly increased when the *total helping rate* is increased from 13.7 Mbps to 137.2 Mbps for channel A and from 11.3 Mbps to 113.4 Mbps for channel B.

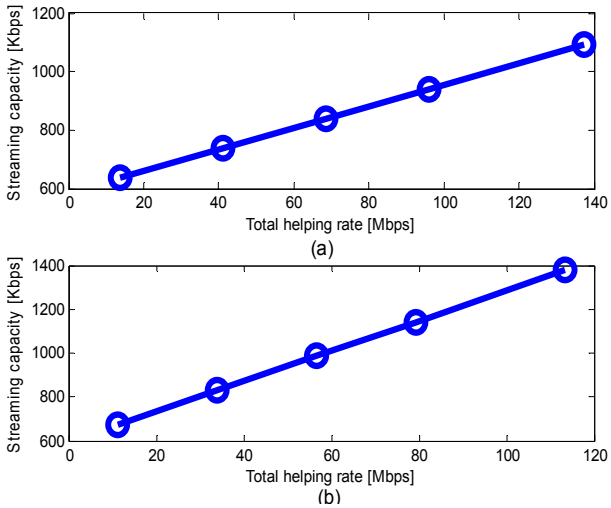

Figure 4. *Comparison of streaming capacity with different total helping rate: (a) channel A, and (b) channel B.*

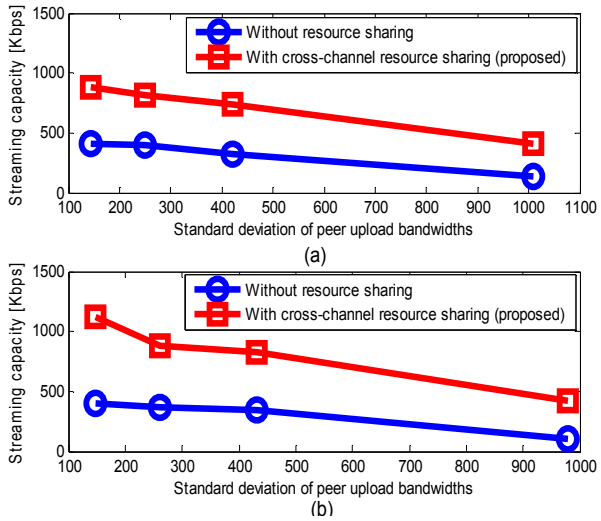

Figure 5. *Comparison of streaming capacity with different standard deviation of peer upload bandwidths: (a) channel A, and (b) channel B.*

Figure 5 evaluates the impact of the heterogeneity of peer upload bandwidths to the streaming capacity. The heterogeneity of peer upload bandwidths can be measure by the *standard deviation* of peer upload bandwidths. A higher *standard deviation* indicates a higher heterogeneity of peer upload bandwidths. From Figure 5, we can see that a higher heterogeneity of peer upload bandwidths leads to a lower streaming capacity. As shown in Figure 5, the proposed *cross-channel resource sharing* approach obtains a significantly improved streaming capacity compared to the case without resource sharing.

4. Improving Streaming Capacity in Mesh-Based P2P Live Streaming Systems

In this section, we will first formulate the streaming capacity problem in mesh-based P2P live streaming systems, and then propose an optimal resource allocation approach to increase the streaming capacity.

4.1. Overview of Mesh-Based P2P Live Streaming System

Due to bandwidth constraint, most of the current P2P live systems provide the video at a low bit rate. For example, the source video rate in PPLive system is usually between 381 to 450 Kbps [22]. How to obtain the maximum streaming rate becomes an attractive topic. *Streaming capacity* is defined as the maximum supported streaming rate that can be received by every receiver [4]. This section focuses on the following problems: 1) What is the streaming capacity in the mesh-based P2P live streaming systems? and 2) How to improve such streaming capacity?

In a mesh-based P2P live system, each peer obtains a set of neighbors from the server. The peer periodically exchanges data availability information with the neighbors, and then retrieves unavailable data from its neighbors, and supplies available data to its neighbors. Figure 6 illustrates a mesh-based P2P live system, in which Peer 2 gets the unavailable video blocks from the server (e.g., peer 1) and its neighbors (e.g., Peers 4 and 6).

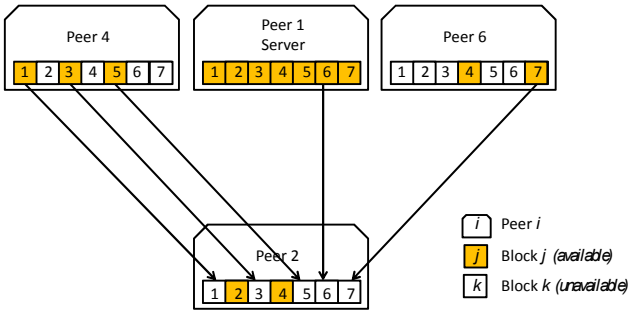

Figure 6. *Illustration of block request at a peer in the mesh-based P2P live streaming system*

4.2. Streaming Capacity Problem

The overlay of the mesh-based P2P live streaming system can be modeled as a directed graph G = (N, L), where N is the set of nodes and L is the set of directed overlay links. Peer 1 is defined as the server. The neighbor set of peer i is denoted by B_i. Each peer can reach the server and request any block from it. The streaming rate that can be received by every peer is denoted by r.

The relationship between a node and its outgoing links is represented with a matrix A^+, whose elements are given by

$$a_{il}^{+} = \begin{cases} 1, & \text{if link } l \text{ is an outgoing link from node } i, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The relationship between a node and its incoming links is represented with a matrix A^-, whose elements are given by

$$a_{il}^{-} = \begin{cases} 1, & \text{if link } l \text{ is an incoming link into node } i, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

In order to distinguish the server from the other peers, we define a *server-filtering element* f_i as follows.

$$f_i = \begin{cases} 0, & \text{if } i = 1, \\ 1, & \text{otherwise.} \end{cases} \quad (3)$$

A video with a length of L_v is evenly divided into a set of blocks, denoted by M. Each block has the equal duration, denoted by d. Each block is labeled with a playback time stamp T_j for $j \in$ M. Users in P2P live applications watch almost the same position of the video. We ignore the variation of the playback times of the users, and assume that each user has the same playback time t_t^p at time t where $0 \leq t \leq L_v$. Each peer maintains a buffer, called the *sliding window*, which contains the emergent blocks after and close to the playback time. The start time of the *sliding window* at time t is the playback point of the video, denoted by t_t^p. The length of the sliding window is denoted by L_{sw}. The end time of the sliding window is given by $t_t^p + L_{sw}$. The sliding window moves forward at the same speed of the playback progress. The blocks following within the current sliding window are denoted by a set S_t at time t. Each peer maintains a *block-availability matrix* H whose elements are given by

$$h_{ij} = \begin{cases} 1, & \text{if block } j \text{ in } S_t \text{ is available at peer } i, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

The block request is performed in a discrete-time manner with an increment of τ ($0 < \tau < L_{sw}$). At time t, peer i requests the unavailable block from the neighbors who are owning it by checking the *block-availability matrix* H. The block request at a peer is illustrated in Figure 6.

Each unavailable block is requested in a prioritized way. For a requesting peer (peer i), block j ($j \in S_t$) is assigned with a priority weight p_{ij}, which is determined by the scheduling policy. For example, the scheduling algorithm that places a higher priority to the dissemination of the blocks in the P2P network will request the rarest block in the neighborhood first, while the scheduling algorithm that cares more about the playback continuity of the requesting peer will request the block closest to the playback position first.

The status of the peers in the P2P live streaming system is time varying. At the current time t, peer i performs the following steps to request an unavailable block.

1) Peer i exchanges the information of block availability in the sliding window with the peers in the neighbor set B_i;

2) Peer i requests the unavailable block j ($j \in S_t$), which has the highest priority weight p_{ij} in the current sliding window, from the neighbor peers;

3) Peer i establishes an incoming link from peer k ($k \in$

B_i) to peer i if peer k owns the requested block in its current sliding window;

4) If all neighbor peers of peer i do not own the requested block, peer i establishes an incoming link from the server and requests the block from it.

The *streaming capacity* at the current time t in the mesh-based P2P live system can be described as to maximize the streaming rate r by optimizing the streaming rate r and the link rate x_l ($\forall l \in$ L), subject to the equality constraint that each receiving peer has to receive the same streaming rate, the upload bandwidth constraint and the download bandwidth constraint. Mathematically, the *streaming capacity* problem can be formulated mathematically as follows.

$$\begin{aligned} &\text{maximize} && r \\ &\text{subject to} && \Sigma_{l\in \mathrm{L}}\, a_{il}^{-}\, x_l = f_i r, \quad \forall i \in \mathrm{N}, \\ &&& \Sigma_{l\in \mathrm{L}}\, a_{il}^{+} x_l \le O_i, \quad \forall i \in \mathrm{N}, \\ &&& 0 \le r \le \min_{i\in \mathrm{N}} \{I_i\}, \\ &&& x_l \ge 0, \qquad \forall l \in \mathrm{L}. \end{aligned} \tag{5}$$

In the optimization problem (5), the objective function is the streaming rate, the first constrain, $\Sigma_{l\in L}\, a_{il}^{-}\, x_l = f_i r$, represents that each receiving peer has to receive the same streaming rate r, the second constraint, $\Sigma_{l\in L}\, a_{il}^{+} x_l \le O_i$, represents that the outgoing rate from each peer needs to be no larger than the upload capacity O_i of the peer, and the third constraint, $0 \le r \le \min_{i\in \mathrm{N}} \{I_i\}$, represents that the received streaming rate at each peer needs to be nonnegative and no larger than the download capacity I_i of the peer.

The optimization problem (5) is a Linear Programming (LP) problem, which can be solved efficiently using the simplex method or the interior point method [23]. The optimal solution for the optimization problem (5) provides the maximal streaming capacity supported by the mesh-based P2P live streaming system.

4.3. Simulation Results for Mesh-Based P2P Live Streaming

In the numerical simulations, there are two classes of peers: cable/DSL peers and Ethernet peers. Cable/DSL peers take 85% of the total population with download capacity uniformly distributed between 0.6 Mbps and 1.0 Mbps and upload capacity uniformly distributed between 0.2 Mbps and 0.4 Mbps. Ethernet peers take the remaining 15% of the total population with both upload and download capacities uniformly distributed between 1Mbps and 2 Mbps. The length of the video is 120 minutes, which is evenly divided into 120 blocks.

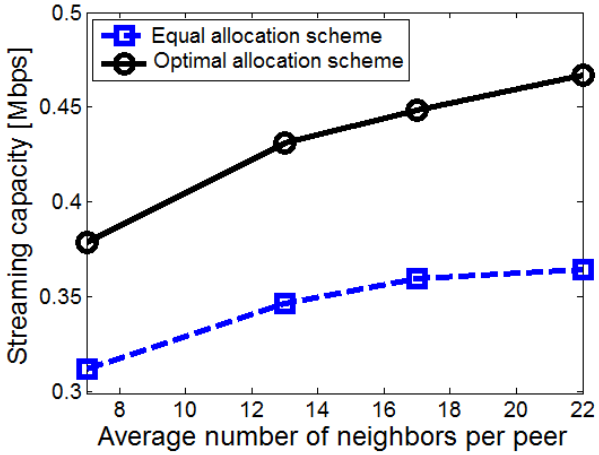

Figure 7. *Comparison of streaming capacity with different number of neighbors per peer.*

In Figure 7, we compare the streaming capacity between the two schemes: 1) optimal allocation scheme (the proposed scheme), in which the link rates are optimized by solving the streaming capacity problem (5); and 2) the equal allocation scheme, in which the link rates from each peer are equally allocated. As shown in Figure 7, when the number of neighbors per peer is increased, each peer can download the unavailable block from more neighbors, thus increasing the streaming capacity. By optimally utilizing the peer upload bandwidths, the proposed optimal allocation scheme improves the streaming capacity by 24.7% in average, compared to the equal allocation scheme.

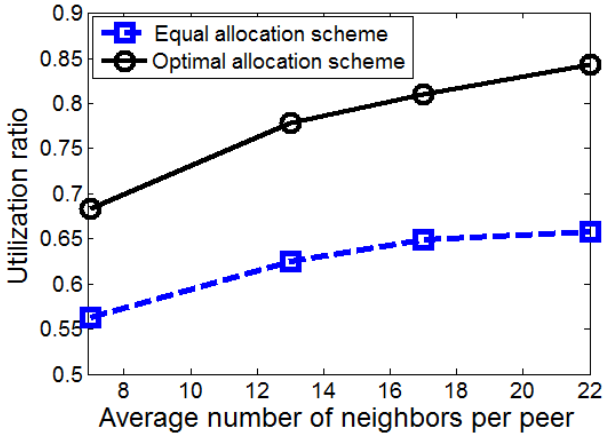

Figure 8. *Comparison of utilization ratio with different number of neighbors per peer.*

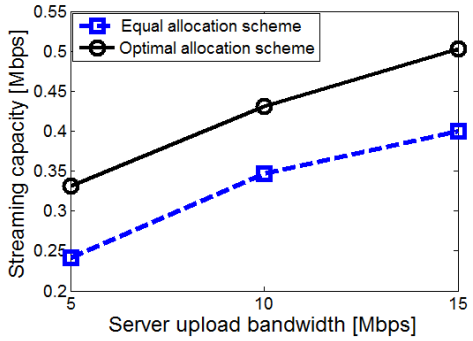

Figure 9. *Comparison of streaming capacity with different server upload bandwidth.*

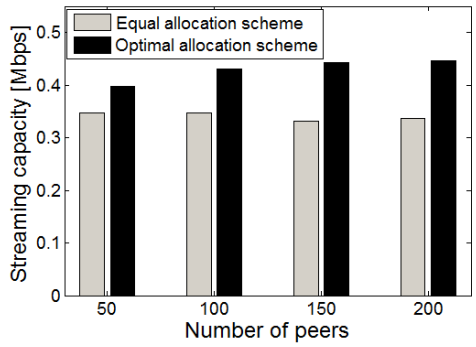

Figure 10. *Comparison of streaming capacity with different number of peers.*

In Figure 8, we compare the *utilization ratio* of peer upload bandwidth, which is defined as: *utilization ratio = streaming capacity/(total system upload bandwidth/number of peers)*. As shown in Figure 8, the *utilization ratio* is increased as the number of neighbors is increased. The proposed optimal allocation scheme achieves a much higher *utilization ratio* than the equal allocation scheme.

In Figure 9, we compare the streaming capacity with different server upload bandwidth. When the server upload bandwidth is increased from 5 Mbps to 15 Mbps, the streaming capacity in the proposed optimal allocation scheme is increased from 0.331 Mbps to 0.503 Mbps. The proposed optimal allocation scheme improves the streaming capacity by 29.1% in average, compared to the equal allocation scheme.

In Figure 10, we compare the streaming capacity with different number of the peers. We vary the number of peers from 50 to 200. The server upload bandwidth is set to 0.1* (number of peers) Mbps. The proposed optimal allocation scheme improves the streaming capacity by 26.5% in average, compared to the equal allocation scheme.

5. Conclusion

In this paper, we proposed resource sharing approaches to improve the streaming capacity in P2P live streaming systems. In tree-based P2P live streaming systems, we propose a cross-channel resource sharing approach to improve the streaming capacity. The proposed approach employs cross-channel helpers to establish the cross-channel overlay links, which enable the unused upload bandwidth in a channel to be utilized in the partner channel, thus improving the streaming capacity of the partner channel. In mesh-based P2P live streaming systems, we formulate the streaming capacity problem into a LP problem. By solving the optimization problem, we can obtain the streaming capacity supported by the P2P live streaming system. The simulation results demonstrate that the proposed resource sharing approaches can significantly improve the streaming capacity for P2P live streaming systems.

References

[1] PPLive, http://www.pplive.com

[2] PPStream, http://www.pps.tv

[3] UUSee, http://www.uusee.com

[4] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, "Streaming Capacity in Peer-to-Peer Networks with Topology Constraints," *Microsoft Research Technical Report*, 2008.

[5] S. Liu, R. Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," in *Proc. of ACM SIGMETRICS*, pp. 313–324. 2008.

[6] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, "Peer-to-Peer Streaming Capacity," *IEEE Transactions on Information Theory*, vol. 57, no. 8, pp. 5072–5087, Aug. 2011.

[7] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. A. Chou, "P2P Streaming Capacity under Node Degree Bound," in *Proc. of IEEE ICDCS*, pp. 587–598, 2011.

[8] T. Bonald, L. Massoulié, F. Mathieu, D. Perino, and A. Twigg, "Epidemic live streaming: optimal performance trade-offs," in *Proc. of ACM SIGMETRICS*, pp. 325–336, 2008.

[9] X. Zhang, J. Liu, B. Li, and T. S. P. Yum, "DONet/CoolStreaming: A data-driven overlay network for live media streaming," in *Proc. of IEEE INFOCOM*, 2005.

[10] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Randomized decentralized broadcasting algorithms," in *Proc. of IEEE INFOCOM*, 2007.

[11] D. Wu, C. Liang, Y. Liu and K. W. Ross, "View-upload decoupling: a redesign of multi-channel P2P video systems," in *Proc. of IEEE INFOCOM*, pp. 2726–2730, 2009.

[12] R. Kumar, Y. Liu, and K.W. Ross, "Stochastic fluid theory for P2P streaming systems," in *Proc. of IEEE INFOCOM*, 2007.

[13] Y. Zhou, D.M. Chiu, and J.C.S. Lui, "A simple model for analyzing P2P streaming protocols," in *Proc. of IEEE ICNP*, 2007.

[14] C. Feng and B. Li, "On large-scale peer-to-peer streaming systems with network coding," in *Proc. of ACM MM*, pp. 269–278, 2008.

[15] C. Feng, B. Li, and B. Li, "Understanding the Performance Gap between Pull-based Mesh Streaming Protocols and Fundamental Limits," in *Proc. of IEEE INFOCOM*, 2009.

[16] F. Liu, B. Li, L. Zhong, B. Li, and D. Niu, "How P2P Streaming Systems Scale Over Time Under a Flash Crowd? " in *Proc. of IPTPS*, 2009.

[17] B.Q. Zhao, J. Lui, and D. M. Chiu, "Exploring the Optimal Chunk Selection Policy for Data-Driven P2P Streaming Systems," in *Proc. of IEEE International Conference on Peer-to-Peer Computing*, pp. 271–280, 2009.

[18] R. Ahlswede, N. Cai, S. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory,* vol. 46, no. 4, pp. 1204–1216, 2000.

[19] C. Gkantsidis and PR Rodriguez, "Network coding for large scale content distribution," in *Proc. of IEEE INFOCOM,* 2005.

[20] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P.R. Rodriguez, "Is high-quality VoD feasible using P2P swarming?" in *Proc. of ACM international conference on World Wide Web*, pp. 903–912, 2007.

[21] M. Wang and B. Li, "R2: Random Push with Random Network Coding in Live Peer-to-Peer Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 9, pp. 1655–1666, 2007.

[22] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in *Proc. of ACM SIGCOMM*, vol. 38, no. 4, pp. 375–388, 2008.

[23] R. J. Vanderbei, *Linear programming: foundations and extensions*, 2nd Edition, Springer Press, 2001.