Maintaining Tree-Structured P2P Overlay Being Resilient to Simultaneous Leave of Several Peers*

Tatsuya KOUCHI†, Nonmember and Satoshi FUJITA(a), Member

SUMMARY A key issue in Peer-to-Peer (P2P) live streaming systems is that several participant peers tend to leave within a short time period. For example, such a phenomenon is common at the half time of football games and at the end of the performance of famous artists. Such selfish behavior of the participants causes several problems in P2P networks such as the disconnection of the overlay, the departure of backup peers and the occurrence of cyclic reference to backup peers. In this paper, we propose several techniques for tree-structured P2P live streaming systems to enhance their resilience to the simultaneous departure of some participants. As the baseline of the discussion, we will focus on mTreebone which is a typical churn-resilient P2P live streaming system based on the notion of peer stability. The performance of the proposed techniques is evaluated by simulation. The simulation result indicates that even under high churn rates, the proposed techniques significantly reduce the number of attempts needed to connect to backup peers and the recovery time after a fail.

key words: Peer-to-Peer live streaming, churn, resilience to simultaneous leave, cyclic reference

1. Introduction

During the past decade, Peer-to-Peer (P2P) technology has emerged as an attractive way to distribute large contents to many users. Many P2P applications are actually used in several domains, such as file sharing, VoIP phone, video-on-demand, live streaming and others. Among them, P2P live streaming is expected to improve the performance of classical server-based live streaming systems with respect to scalability, reliability and extendibility [9].

Conventional P2P live streaming systems can be classified into three types by the structure of the overlay [9], namely tree-structured, mesh-structured and their hybrid. In tree-structured systems [2], [15], [16], a stream which is given to the tree root is delivered to the other peers along tree edges. Although such a simple forwarding scheme attains low communication overhead, it faces a serious problem in that the departure of a non-leaf peer immediately stops the feed to down-stream peers. This motivates the study of tree-structured P2P systems that are resilient to churn, which include those with backup peers [18] and those with a multiple-tree structure [3], [5], [6], [9]. On the other hand, in mesh-structured systems such as CoolStreaming [20], the delivery of a stream is realized by repeating data exchange between adjacent peers [4], [8], [11]–[13]. Since the departure of a peer can easily be recovered by adjacent peers, these mesh-structured systems are more resilient to churn than tree-structured ones, but as a side effect, it causes a high communication overhead and a relatively long delivery delay. Recently hybrid systems such as HON [21] and ChunkySpread [17] have also been studied to overcome the drawback of the above two types.

In this paper, we propose a method to enhance the resilience of tree-structured live streaming systems against churn. More concretely, we focus on mTreebone [18] as the baseline of the discussion, and propose several techniques to tolerate simultaneous leave of several peers. The reader should note that such a simultaneous leave of the participants frequently occurs in live streaming services, e.g., at the end of football games and at the end of the performance of famous artists. mTreebone uses the following techniques to raise its robustness to churn:

1. it introduces the notion of stability representing the tendency of staying in the system and organizes a tree-structured overlay consisting merely of such stable peers (a formal definition of “stability” will be given in Sect. 2);
2. for each peer in the tree-structured overlay, it selects a candidate for the reconnection (i.e., backup peer) beforehand so that it can immediately contact the candidate when the current parent leaves.

Although these techniques are effective against the departure of a single peer, when several peers simultaneously leave, mTreebone faces the following problems:

1. the scarcity of the upload capacity of the candidate, due to the exhaustion of the capacity by several peers losing their parent;
2. the occurrence of cyclic reference to the parents which can happen when a peer becomes a child of a descendant which was not a descendant of the peer at the time of being selected as a candidate; and
3. the departure of candidate peer.

In the following, we propose several techniques to overcome these issues.

The performance of the proposed method is evaluated by simulation. The simulation result indicates that even under a high churn rate, the proposed scheme significantly re-
duces the number of attempts connecting to backup peers and the recovery time after a fail. More concretely, 1) it reduces the number of fails due to the lack of the upload capacity to almost zero regardless of the churn model and the churn rate and 2) it reduces the recovery time of mTreebone to almost half, while 3) it slightly increases the number of transmitted messages. This paper is an extended version of our conference paper [7]. The main difference from the conference version is as follows: 1) the proposed scheme is extended to recover the departure of candidate and the number of reconnections is optimized, 2) new experimental results are added to reflect above changes, and 3) the notion of cyclic reference is explicitly introduced to clarify the exposition, as well as the detailed explanation of key techniques.

The remainder of this paper is organized as follows. Section 2 overviews techniques used in mTreebone. Section 3 describes proposed techniques. Section 4 describes simulation model and Sect. 5 describes simulation results. Finally, Sect. 6 concludes the paper with future work.

2. Preliminaries

This section overviews key techniques used in mTreebone [18]. In this system, several selected peers organize a tree-structured overlay called tree-bone to deliver a given live stream along tree-bone edges. The other unselected peers can receive the stream from any tree-bone peer, and a mesh-structure is used to compensate the weakness of the peers. In this system, several selected peers organize a tree-structured overlay called tree-bone to deliver a given live stream along tree-bone edges. The other unselected peers can receive the stream from any tree-bone peer, and a mesh-structure is used to compensate the weakness of the peers.

2.1 Tree-Bone

In the following, a peer is said to be stable if it continuously stays in the system for more than $T(t)$ time, where $t$ is the elapsed time after starting the current session of the live stream and function $T(\cdot)$ is determined so that the expected service time (EST) is maximized (see Sect. 2.2 for the details). Basically, mTreebone is designed so that stable peers are promoted as tree-bone peers. However, such a simple promotion rule causes the lack of upload capacity when the number of tree-bone peers is not large (e.g., at the beginning of the session). To overcome such an issue, Wang et al. add another rule so that any peer can be promoted as a tree-bone peer in a probabilistic manner. More concretely, after passing $s$ time after the join, each peer is promoted as a tree-bone peer with probability

$$p(s) = \frac{1}{T(t) - s + 1}. \quad (1)$$

With such a probabilistic promotion rule, the probability of being a tree-bone peer is given as $s/T(t)$, which reaches one when $s = T(t)$.

To enhance the resilience to churn, each tree-bone peer keeps a candidate for reconnection which is randomly selected from adjacent peers in the mesh-structured overlay to have sufficient upload capacity. With the notion of candidate, each tree-bone peer detecting the departure of the parent can immediately contact an appropriate peer to become a new child of the peer.

2.2 Calculation of Function $T$

Consider a session of length $L$. Under the simple promotion rule, each peer arrived at time $t = \tau$ can serve as an uploader for at most $L - \tau - T(\tau)$ time (it can start serving at time $t = \tau + T(\tau)$ and the session ends at time $t = L$). As such, since $EST$ is obtained by subtracting $T(\tau)$ from the expected service time of peers, it can be represented as follows:

$$EST(t) = \int_{T(t)}^{L-t} x f(x) dx + \int_{L-t}^{\infty} (L - t) f(x) dx - T(t), \quad (2)$$

where $f(\cdot)$ denotes the probability density function of the life time of peers. In Eq. (2), the integral in the numerator reflects the fact that the upper bound on the life time is $L - \tau$, and the denominator normalizes it so that the integral from $T(\tau)$ to the infinity becomes one. Wang et al. assume that function $f$ follows a Pareto distribution with shape parameter $k$ according to the analyses in the literature [1], [14]. For such a specific $f$, function $EST(t)$ can be given as follows [18]:

$$EST(t) = \frac{T(t)}{k - 1} \left[ 1 - \left( \frac{T(t)}{L - t} \right)^{k-1} \right]. \quad (3)$$

This function is maximized when

$$T(t) = (L - t) \left( \frac{1}{k} \right)^{\frac{1}{k-1}}, \quad (4)$$

which implies that “optimal $T(\cdot)$ maximizing $EST$” is proportional to $L - t$, i.e., proportional to the remaining time of the session at the time of join. In addition, since it is known that the coefficient $\left(1/k\right)^{1/(k-1)}$ converges to 0.3 as $k \to 1$, if function $f$ follows a Pareto distribution with shape parameter $k \approx 1$, the optimal $T(\cdot)$ is about 30% of the remaining time of the session.

3. Proposed Scheme

In mTreebone, each tree-bone peer selects another peer as a candidate for the reconnection to enhance the churn resilience. Such a simple approach works well if the churn rate is not high. However, in actual streaming systems, several peers tend to leave in a short time period, and such a simultaneous leave causes the following serious problems: 1) the scarcity of the upload capacity of candidates due to the exhaustion by several peers; 2) the occurrence of a cyclic reference to the candidates; and 3) the departure of the candidate. In this section, we propose several techniques to resolve these issues.

3.1 Fractional Reservation of Upload Capacity

The upload capacity of a candidate can be exhausted if it is
selected by many peers as a candidate and its upload capacity is consumed by these peers, even if the original capacity was large. In mTreebone, such a concentration of selections is relaxed by using a randomization [18]. In contrast to that, in the proposed scheme, we take an approach such that each selection reserves a small fraction of the capacity at the time of selection (hereafter, we call this technique RESV).

Assume that one stream consumes one unit of capacity. Let \( c[i] \) denote the residual capacity of peer \( i \), i.e., \( c[i] \) is the capacity originally given to \( i \) minus the number of peers currently downloading a stream from \( i \). In RESV, when peer \( j \) selects \( i \) as a candidate, \( j \) reserves the capacity of \( i \) of amount \( \alpha \in (0, 1] \), where the number of candidates is determined by the underlying streaming scheme. Accordingly, the selection rule (in the original scheme) is modified as follows:

Peer \( i \) can be selected as a candidate for a peer \( j \) if \( c[i] \) minus the amount of reserved capacity is at least \( \alpha \).

The concrete behavior of peer \( j \) after detecting the departure of the parent is as follows. Assume that \( j \) selected \( i \) as a candidate and that \( i \) knows all peers which are selecting \( i \) as a candidate. Let \( n_i \) be a variable representing the number of peers selecting \( i \) as a candidate. If \( c[i] < 1 \), peer \( j \) should find a new candidate \( i’ \) satisfying \( c[i’] \geq 1 \) by conducting a search over the overlay (after detecting the departure). Otherwise, peer \( j \) simply becomes a new child of \( i \) and reduces variables \( c[i] \) and \( n_i \) by one. Then \( j \) requests children of \( i \) to change their candidate from \( i \) in an appropriate order until \( c[i] \geq n_i \alpha \) holds.

Note that in this method, parameter \( \alpha \) is used to control the number of peers which can select peer \( i \) as a candidate for each \( i \). This number decreases as the amount of residual capacity decreases, which reflects an intuition such that the residual capacity of a peer becomes small as the percentage of tree-bone peers becomes small, and in such a case, we need to reduce the number of peers competing at a candidate to bound the influence of the change of candidates as much as possible. The departure of a candidate can be overcome by allowing each peer to have several candidates (note that we do not need to distinguish the departure of candidate \( i \) and the case of \( c[i] < 1 \)) but even in such a case, parameter \( \alpha \) effectively controls the number of competing peers depending on the residual capacity of each peer.

3.2 Prevention of Cyclic Reference

This section proposes two techniques to avoid the occurrence of a cycle in the tree-bone. Recall that when peer \( j \) selects peer \( i \) as a candidate, \( i \) must not be a descendant of \( j \) in the tree-bone since the reconnection of \( j \) to \( i \) as a child immediately causes a cycle in the overlay. However, even if \( i \) was not a descendant of \( j \) at the time of selection, the simultaneous leave of several peers would cause such a cycle if there is a cyclic reference of the candidates. For example, if the candidate for \( i \) is a descendant of \( j \), the departure of parent of \( i \) followed by the departure of parent of \( j \) causes such a cycle. To avoid such a situation, we need to prevent from organizing such a cyclic reference of the candidates.

The first technique LIST tries to prevent such a cyclic reference by keeping the list of ancestors at each peer in the tree-bone. For example, if peer \( c \) is receiving a stream through tree-bone peers \( a \) and \( b \), then the list held by \( c \) is given as \([a, b, c]\). This list is updated whenever the set of ancestors changes, and using the updated list, each peer detects the occurrence of cyclic reference in the following manner:

- The update is triggered by a peer connecting to a new parent; it receives an ancestor list from the new parent and forwards it to all children after adding itself at the end of the list;
- Each peer \( i \) which receives an ancestor list from the parent forwards it to all children after adding itself to the end of the list. If the received list contains peer \( j \) which selects \( i \) as a candidate, then \( i \) requests \( j \) to change the candidate from \( i \) and refuses the request for reconnection from \( j \) (such a situation could occur if the parent of \( j \) leaves before receiving a message from \( i \)).

The second technique DEPTH tries to prevent the occurrence of cyclic reference by using a variable representing the depth in the tree-bone, where the depth of a peer is defined to be the path length from the root to the peer. Let \( d_i \) denote the local variable representing the depth of peer \( i \). In DEPTH, peer \( j \) can select peer \( i \) as a candidate if \( d_j < d_i \). The change of the parent of a peer is reflected to the local variables using a similar way to LIST. Although we could not guarantee the accuracy of local variables under high churn rate, as long as the value of variables correctly reflects the actual depth of the peer or the actual depth of \( i \) is smaller than \( d_i \) and the actual depth of \( j \) is at least \( d_j \), it can effectively prevent from the occurrence of cyclic reference.

3.3 Recovery from Cyclic Configuration

This section describes a technique to quickly recover from a cyclic configuration in the overlay (note that this “cycle” is different from the “cyclic reference of candidates” discussed in the last section). We call it RECV. The basic idea of RECV is that each peer transmits a probe message toward the root of the tree-bone whenever it changes its parent, so that it can proactively detect the occurrence of a cycle by receiving the message transmitted by itself through the cycle, if any. Upon detecting a cycle, which can be done merely by a peer which recently changes its parent, it disconnects the link to the parent, and tries to find (and connect to) another parent. In addition to such a simple recovery procedure, to reduce the number of reconnections as much as possible, RECV is designed so that if a peer \( i \) which has issued a probe message receives another probe message issued by \( j \) (\( \neq i \)), then it forwards the received message only when \( i < j \), where \( i \) and \( j \) are unique identifier of peers drawn from an ordered set. With such a refinement, we can reduce the number of disconnections concerned with a cycle forwarding many probe messages to exactly one, which sig-
significantly reduce the number of reconnections under a high churn rate (note that in RECV, every peer which changed its parent issues a probe message).

4. Simulation Model

4.1 Overview

The performance of the proposed techniques is evaluated by simulation. More concretely, we simulate the behavior of peers against churn using PeerSim [10], where the forwarding of a steam is omitted here. One step of the simulator corresponds to 20 ms and each peer conducts the following operations in a step:

1. Receive messages from the input buffer of the peer,
2. Conduct necessary calculation, and
3. Send out one message to an adjacent peer,

where a message sent in the $i$th step is received by the receiver at the beginning of the $(i+3)$rd step; i.e., we assume that peers are completely connected by routing paths and the message delay on each path is fixed to 60 ms.

Other parameters are determined as follows:

- The length of live stream is 600 sec.
- The total number of peers is 2000.
- There is a specific peer called root which always exists in the tree-bone as the root during the simulation, and the remaining 1999 peers join/leave according to the churn model described later. The upload capacity of the root is 16 and the upload capacity of the other peers is varied from 4 to 8 (recall that one child consumes unit capacity).
- Parameter $\alpha$ used in RESV is varied from 0 to 1.
- The number of candidates to be selected by a peer is fixed to one.

4.2 Churn Model

We consider the following two churn models in the simulation. The first model is the same as the model used in the literature [18], i.e., 1) peers arrive at the system according to a Poisson distribution with mean $\lambda$ and 2) they leave the system according to a Pareto distribution. The probability density function of Pareto distribution is given as

$$f(t) = \frac{kt^m}{t^{m+1}}$$

where $k$ is a parameter called the shape parameter and $t_m$ is a parameter called the scale parameter which is associated with the participation time of the peers, i.e., under this distribution, peers arrived earlier will leave with a higher probability. In our setting, we associate $t$ to the elapsed time from the beginning of the current session and $f(t)$ to the surviving probability as an active peer at time $t \geq t_m$. In the simulation, we vary parameter $\lambda$ from 4 to 12 [peers/sec] to control the longevity of peers since parameters $k$ and $t_m$ do not directly control the longevity. More precisely, for large $\lambda$, many peers have an earlier participation time which causes a higher leaving rate within a short time.

The second model is more straightforward, i.e., 1) all peers simultaneously arrive at the system at time $t = 0$, and 2) a half of peers randomly leave during succeeding $T_d$ time, where parameter $T_d$ is varied from 100 to 600 sec. In both models, once a peer leaves, it will not join again.

4.3 Metrics

We use the following three metrics to evaluate the performance of the proposed techniques: 1) the number of fails before becoming a child of the new parent, 2) the time spent for the reconnection after failing the connection to the first candidate, and 3) the number of messages issued by the peers. The number of fails is itemized to the reason of fails so that:

- 1a) fails due to the lack of upload capacity,
- 1b) fails due to the departure of the candidate, and
- 1c) fails due to the occurrence of a cyclic configuration.

Note that although (1a) and (1b) are not distinguished by RESV, the ratio of these items is expected to vary depending on the churn rate and parameter $\alpha$. The cycle in the overlay is resolved by the basic scheme described in Sect. 5.2, unless
otherwise designated. We compare the performance of the proposed method with the original mTreebone with respect to the above metrics. Each value shown in figures and tables is an average over 100 runs.

5. Result

5.1 RESV

At first we evaluate the effect of the first technique by applying RESV to mTreebone. Figure 1 illustrates the number of fails under the first churn model, where the horizontal axis indicates parameter $\alpha$ and (a) and (b) correspond to the case of $\lambda = 4$ and 12, respectively. The number of fails due to the departure of candidates (indicated by red bars) rapidly decreases as $\alpha$ increases from 0.1 to 0.3, which implies that the concentration of selections is effectively relaxed by $\alpha$. The number of fails due to the lack of upload capacity (indicated by green bars) also decreases as $\alpha$ increases and becomes (almost) zero for $\alpha \geq 0.6$. Although the number of fails due to the cyclic reference (indicated by blue bars) gradually increases as $\alpha$ increases, the above results indicate that RESV is effective to reduce the total number of fails even under a high churn rate. Figure 2 shows the result for the second churn model. A crucial observation we could learn from the figure is that the number of fails due to the departure of candidates (red bars) increases for large $\alpha$’s due to the side-effect of selecting large $\alpha$, which indicates that to reduce the number fails as much as possible we should set $\alpha$ to around 0.4 to 0.5 under the second churn model.

Figure 3 shows the impact of $\alpha$ to the reconnection time. Although it takes long time when $\alpha$ is small and the churn rate is high, as $\alpha$ becomes large, it converges to 1.5 sec for each churn model. In addition, the value of $\alpha$ minimizing the reconnection time gradually increases as the churn rate increases. For example, under the first churn model, it changes as $\alpha = 0.2, 0.3$ and 0.4 as $\lambda$ increases as 4, 8 and 12. By letting $\alpha_{\text{min}}$ be the value minimizing the reconnection time and $\alpha_{\text{max}}$ be the minimum value stabilizing the reconnection time, $\alpha$ should be fixed to satisfy $\alpha_{\text{min}} \leq \alpha \leq \alpha_{\text{max}}$, since too large $\alpha$ causes frequent fails as was described above. In the next subsection, we assume that mTreebone has been extended by technique RESV with parameter $\alpha = 0.6$.

5.2 Impact of Occurrence of Cycles to the Performance

This subsection evaluates the impact of LIST, DEPTH and RECV to the performance of mTreebone with respect to the occurrence of cyclic configuration, by comparing it with
mTreebone combined with the following basic recovery scheme (note that the original mTreebone does not provide a mechanism to recover from the occurrence of cycles):

**Basic scheme:** If the stream is suspended by more than one second, each peer transmits a probe message to the parent to verify the existence of a path to the root. If the peer receives the message transmitted by itself, it identifies the existence of a cycle, and initiates the reconnection procedure.

Figure 4 shows the number of fails in each scheme where four bars indicate the results for the basic scheme, LIST, DEPTH and RECV, respectively. For each bar, red indicates the number of fails due to the departure of candidates and blue indicates the number of fails due to the occurrence of a cycle (note that fails due to the lack of capacity did not occur thanks to RESV). Each technique certainly reduces the number of fails due to cycles compared with the basic scheme regardless of the churn model and the churn rate. In particular, LIST reduces the number of fails to almost zero. However, each technique does not sufficiently reduce the number of fails due to the departure of candidates, e.g., among examined four schemes, RECV exhibits the worst performance under the first churn model and DEPTH exhibits the worst performance under the second churn model (as will be described later, this problem can be overcome by combining two techniques). As for the reconnection time, we find that RECV significantly reduces it for each churn model. See Fig. 5 for illustration. Although LIST slightly increases the reconnection time under the first churn model, there is no difference between LIST and DEPTH under the second churn model, regardless of the churn rate.

Finally, we evaluate the performance of the combination of LIST and RECV in detail. Tables 1, 2 and Fig. 6 summarize the results. From Table 1, we can observe that although the combined scheme reduces the number of fails as the churn rate increases from low to moderate, it becomes worse than the basic scheme when the churn rate is very...
As for the time required for the reconnection due to the occurrence of a cycle, the combined scheme significantly improves the basic scheme regardless of the churn model and the churn rate, as is shown in Table 2. In particular, although the reconnection time of LIST is worse than the basic scheme under the first churn model, we can improve the basic scheme by combining it with RECV. Figure 6 compares the number of messages issued in the combined scheme. From the figure, we can see that the increase of the number of messages is bounded by 20% of the basic scheme.

In summary, the combination of LIST and RECV effectively bounds the occurrence of cyclic configurations and reduces the reconnection time of the original mTreebone scheme. However, the performance of the combined scheme gradually degrades as the churn rate increases, although it is still better than the original mTreebone. The improvement of the performance under the high churn rate, as well as the combination with other backup-based schemes, is left as an important future work. The theoretical analysis of the performance of the proposed scheme is another crucial issue.

6. Concluding Remarks

This paper proposes techniques to increase the resilience of P2P live streaming systems to the simultaneous leave of several peers. The results of simulation indicate that under a marginal churn rate, the combination of LIST and RECV effectively bounds the occurrence of cyclic configurations and reduces the reconnection time of the original mTreebone scheme. However, the performance of the combined scheme gradually degrades as the churn rate increases, although it is still better than the original mTreebone. The improvement of the performance under the high churn rate, as well as the combination with other backup-based schemes, is left as an important future work. The theoretical analysis of the performance of the proposed scheme is another crucial issue.

References

[1] K.C. Almeroth and M.H. Ammar, “Collecting and modeling the join/leave behavior of multicast group members in the MBone,” In Proc. IEEE Int’l Symp. on High Performance Distributed Computing, pp.209–216, 1996.

[2] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “SplitStream: High-bandwidth multicast in cooperative environments,” In Proc. 19th ACM Symp. on Operating Systems Principles (SOSP ’03), pp.298–313, 2003.

[3] G. Dan, V. Fodor, and I. Chatzidrossos. “On the Performance of Multiple-Tree-Based Peer-to-Peer Live Streaming,” In Proc. 26th IEEE Int’l Conf. on Computer Communications (INFOCOM 2007), pp.2556–2560, 2007.

[4] S. Fujita, “Optimal Serial Broadcast of Successive Chunks,” In Proc. 8th Int’l Workshop on Algorithms and Computation (WALCOM 2014), LNCS, vol.8344, pp.102–113, 2014.

[5] M. Fesci-Sayit, E.T. Tunali, and A.M. Tekalp, “Bandwidth-aware multiple multicast tree formation for P2P scalable video streaming using hierarchical clusters;” In Proc. 16th IEEE Int’l Conf. on Image Processing (ICIP), pp.945–948, 2009.

[6] S. Grau, M. Fischer, M. Brinkmeier, and G. Schäfer, “On Complexity and Approximability of Optimal DoS Attacks on Multiple-Tree P2P Streaming Topologies,” IEEE Trans. Dependable and Secure Computing, vol.8, no.2, pp.270–281, 2011.

[7] T. Kouchi and S. Fujita, “How to Tolerate Simultaneous Leave of Peers in Tree-Structured P2P Live Streaming Systems,” In Proc. Int’l Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA), pp.149–155, 2014.

[8] Y. Liu, “Delay Bounds of Chunk-Based Peer-to-Peer Video Streaming,” IEEE/ACM Trans. Networking, vol.18, no.4, pp.1195–1206, 2010.

[9] N. Magharei, R. Rejaie, and G. Yang, “Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches,” In Proc. 26th IEEE Int’l Conf. on Computer Communications (INFOCOM 2007), pp.1424–1432, 2007.

[10] URL: http://peersim.sourceforge.net

[11] D. Ren, Y.-T.H. Li, and S.-H.G. Chan, “Fast-Mesh: A Low-Delay High-Bandwidth Mesh for Peer-to-Peer Live Streaming,” IEEE Trans. Multimedia, vol.11, no.8, pp.1446–1456, 2009.

[12] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P.A. Chou, “Peer-to-Peer Streaming Capacity,” IEEE Trans. Inf. Theory, vol.57, no.8, pp.5072–5087, 2011.

[13] H. Shen, Z. Li, and J. Li, “A DHT-Aided Chunk-Driven Overlay for Scalable and Efficient Peer-to-Peer Live Streaming,” IEEE Trans. Parallel Distrib. Syst, vol.24, no.11, pp.2125–2137, 2013.

[14] K. Sripanidkulchai, B. Maggs, and H. Zhang, “An analysis of live streaming workloads on the Internet,” In Proc. 4th ACM SIGCOMM Conf. on Internet Measurement (IMC ’04), pp.41–54, 2004.

[15] G. Tan and S.A. Jarvis, “Improving the Fault Resilience of Overlay Multicast for Media Streaming,” IEEE Trans. Parallel Distrib. Syst, vol.18, no.6, pp.721–734, 2007.
[16] D.A. Tran, K.A. Hua, and D. Tai, “ZIGZAG: An efficient peer-to-peer scheme for media streaming,” In Proc. 22nd Annual Joint Conf. of the IEEE Computer and Communications (INFOCOM 2003), pp.1283–1292, 2003.

[17] V. Venkataraman, K. Yoshida, and P. Francis, “Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast,” In Proc. 14th IEEE Int’l Conf. on Network Protocols (ICNP ’06), pp.2–11, 2006.

[18] F. Wang, Y.-Q. Xiong, and J.-C. Liu, “mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast,” In Proc. 27th Int’l Conf. on Distributed Computing Systems (ICDCS ’07), p.49, 2007.

[19] F. Wang, J.-C. Liu, and Y.-Q. Xiong, “On Node Stability and Organization in Peer-to-Peer Video Streaming Systems,” IEEE Syst. J., vol.5, no.4, pp.440–450, 2011.

[20] X.-Y. Zhang, J.-C. Liu, B. Li, and T.-S. Peter Yum, “CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming,” In Proc. 24th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2005), pp.2102–2111, 2005.

[21] M. Zhou and J.-C. Liu, “A Hybrid Overlay Network for Video-on-Demand,” In Proc. IEEE Int’l Conf. on Communications (ICC 2005), pp.1309–1313, 2005.

Tatsuya Kouchi received the B.E. degree and M.E. degree in information engineering, from Hiroshima University in 2012 and 2014, respectively. His research interests include computer networks and P2P systems.

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