How to Tolerate Simultaneous Leave of Peers in Tree-Structured P2P Live Streaming Systems

Tatsuya KOUCHI and Satoshi FUJITA

1Department of Information Engineering, Hiroshima University
Higashi-Hiroshima, 739-8527, Japan

Abstract—A characteristic of Peer-to-Peer (P2P) systems is that several peers tend to leave the system in a short time. Such a simultaneous leave of peers causes serious problems such as the leave of backup peers and the occurrence of cyclic reference to the backup peers. In this paper, we propose several techniques to enhance the resilience of tree-structured P2P live streaming systems to such simultaneous peer leaves. The effect of the proposed techniques is evaluated by simulation. The result of simulation indicates that the proposed scheme reduces the number of fails to connect to a backup peer and the time required for the reconnection after a fail, even under a high churn rate.

Keywords: Peer-to-Peer live streaming, peer churn, resilience to simultaneous leave, cyclic reference.

1. Introduction

Recently, Peer-to-Peer (P2P) technology has attracted considerable attentions as a way of distributing large contents to many users without causing a bottleneck at specific nodes. Many network applications based on the P2P technology are widely used, which include file sharing, VoIP phone, video-on-demand, and live streaming. Among those applications, P2P live streaming is one of the most promising approaches to improve the performance of existing systems with respect to the scalability, the reliability and the extendibility.

The architecture for P2P live streaming systems can be classified into three types by the structure of the overlay, i.e., tree-type, mesh-type and their hybrid. In tree-structured P2P live streaming systems [2, 5], a live stream is fed to the root of the tree and is delivered to the other peers by repeating the forwarding of the received stream toward down-streams. The overhead of such a simple forwarding scheme is small, while it has a serious drawback such that the leave of a non-leaf peer immediately stops the feeding to down-streams. This motivates the proposal of techniques to tolerate such a “peer churn” in tree-structured systems, such as the preparation of backup peers and the use of multiple-trees. On the other hand, in mesh-structured systems such as CoolStreaming [8], the delivery of a live stream is realized by repeating the data exchange between adjacent peers. Thus in contrast to tree-structured systems, mesh-structured systems are more resilient to peer churns, since even if one adjacent peer leaves, it still has a chance to receive a copy of the stream from other peers. However, such a redundancy significantly increases the overhead to maintain the link to all adjacent peers, which frequently causes a long delay in the delivery of live streams. Hybrid architecture was proposed to overcome the drawback of the above two types, which includes HON [9] and ChunkySpread [6] as the representatives.

In this paper, we propose several techniques to enhance the resilience of tree-structured P2P live streaming systems to peer churns. More concretely, we focus on a hybrid architecture proposed by Wang et al. called mTreebone [7], and propose a way to tolerate simultaneous leave of several peers. The mTreebone realizes a resilience to peer churns in the following manner: 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 Section 2); 2) for each peer in the tree-structured overlay, it selects a candidate for the reconnection so that it can immediately start the communication with the candidate when the current parent leaves the system. Although such techniques used in the mTreebone are effective to tolerate the leave of a single peer, in actual systems, several peers tend to leave the system in a short time period. Such a simultaneous leave of several peers would cause the following serious problems: 1) scarcity of the upload slot of candidate peers due to the exhaustion by several peers; 2) the occurrence of a cycle by connecting to a descendant peer which was not being a descendant at the time of selecting candidate; and 3) the leave of the candidate peer which does not occur when merely a single peer leaves. In this paper, we propose several techniques to overcome such issues.

The effect of the proposed techniques is evaluated by simulation. The result of simulation indicates that the proposed scheme reduces the number of fails to connect to a candidate peer, which indicates that candidates are appropriately selected so that it causes no concentration of the selections, and it reduces the time required for the reconnection after a fail even under a high churn rate such that several peers simultaneously leaves.

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

In the mTreebone, several peers satisfying a certain stability condition organize a tree-structured overlay called treebone and a live stream is given to the root of the tree-bone and is delivered to all peers along tree-bone edges. Peers which do not participate in the tree-bone can receive the stream from any peer participating in the tree, and a mesh-structure is used to compensate the weakness of the tree-structure against peer churns. In this section, we describe key techniques used in the mTreebone to be resilient to peer churns.

2.1 Stability of Peers

In the mTreebone, a peer is said to be stable if it stays in the system for a time exceeding a threshold $T(t)$, where $t$ is the elapsed time after starting the current session of live stream (i.e., the threshold used in the stability condition is a function of the elapsed time). Threshold $T(t)$ is determined so that the expected service time (EST) of each peer is maximized. Assume that a session starts at time $t = 0$ and ends at time $t = L$. Let $f(x)$ be the probability density function of the life time of a peer. The reader should note that the maximum service time of a peer which joins the system at time $t = \tau$ is at most $L - \tau - T(\tau)$ since the session ends at time $L$ and it can start the service to other peers after passing $T(\tau)$ time after the join (recall that the mTreebone does not allow “unstable” peers to participate in the tree-bone as an uploader). Hence, since $EST$ is obtained by subtracting $T(\tau)$ from the expected life time, it can be represented as follows:

$$EST(t) = \frac{\int_{(t)}^{L-t} xf(x)dx + \int_{L-t}^{\infty} (L-t)f(x)dx}{\int_{T(\tau)}^{\infty} f(x)dx} - T(t) \quad (1)$$

Here 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.

According to the observation shown in [1], [4], the mTreebone assumes that the probability density function follows a Pareto distribution with shape parameter $k$ [7]. For such a specific function, $EST(t)$ is calculated as follows:

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

which takes the maximum value when

$$T(t) = (L-t) \left( \frac{1}{k} \right)^{\frac{1}{k-1}} . \quad (2)$$

From Equation (2), we can observe that an optimal value of the threshold is proportional to the remaining time of the session at the time of join. In addition, the coefficient $(1/k)^{1/(k-1)}$ converges to 0.3 as the value of $k$ converges to 1. Thus, if the life time follows a Pareto distribution and the shape parameter $k$ is close to 1, the optimal value of the threshold can be approximated by the 30% of the remaining time of the session at the time of join.

2.2 Probabilistic Promotion

The simplest rule for the participation in the tree-bone as an uploader is that each peer which joins the system at time $t$ can participate in the tree-bone at time $t + T(t)$ or later. However, if we strictly apply this rule, it causes the lack of upload bandwidth particularly in an early stage of the live streaming session. To overcome such an issue, in the mTreebone, it takes an approach such that it randomly promotes unstable peers as a tree-bone peer. More concretely, when the staying time of a peer is $s$, it is promoted as a tree-bone peer with the following probability:

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

Using such a probabilistic mechanism, we can realize a situation such that the probability of participating in the tree-bone is $s/T(t)$, which reaches one when $s = T(t)$ [7].

2.3 Candidate for the Reconnection

To watch live streams in a continuous manner, each peer which detects the leave of the parent should immediately find another tree-bone peer to have enough upload slot, and establish a connection to the peer to receive the suspended stream. In addition, to reduce the suspension time as much as possible, such a candidate for the reconnection should be selected before actually detecting the leave of the parent. In the mTreebone, each tree-bone peer randomly selects candidates to have enough upload slots from adjacent peers in the mesh-structured overlay, and the other peers, which do not have a child in the overlay, select no candidate in advance to reduce the cost for the maintenance.

3. Proposed Scheme

The mTreebone described in the last section tolerates the leave of peers by preparing a candidate for reconnection for each tree-bone peer. Such a simple approach works well if the frequency of the leave of peers is not high. However, in actual systems, several peers tend to leave the system in a short time period as in the case of the halftime of football game and the end of the performance of specific musicians. Such a simultaneous leave of several peers would cause the following serious problems: 1) scarcity of the upload slot of candidate peers due to the exhaustion by several peers; 2) the occurrence of a cycle by connecting to a descendant peer which was not being a descendant at the time of selecting candidate; and 3) the leave of the candidate peer which does not occur when merely a single peer leaves. In this section, we propose three techniques to resolve those issues.
3.1 Fractional Reservation of Upload Slot

Upload slots of a candidate peer will be exhausted if it is selected by many peers as a candidate and its upload slots are consumed by those peers due to the leave of their parent. As was described above, in the mTreebone, such a concentration of the selection is relaxed by using a randomized approach. In contrast to that, in the proposed scheme, we take an approach such that each selection reserves a small fraction of the bandwidth at the time of selecting the candidate (hereafter, we will call this technique Proposal 1).

Assume that each stream consumes one unit of the upload bandwidth. Let $c[i]$ denote the residual bandwidth of peer $i$ including the bandwidth which was reserved but is not being used by the other peers. In the proposed scheme, when peer $j$ selects peer $i$ as a candidate, peer $j$ reserves the upload bandwidth of peer $i$ of amount $\alpha \in (0,1]$. According to this modification, the rule for the selection of candidates in the mTreebone is modified as follows:

Peer $j$ can select peer $i$ as a candidate only when $c[i]$ minus reserved bandwidth is at least $\alpha$, and $j$ can select candidate $i$ as the parent only when $c[i] \geq 1$.

The concrete behavior of peer $j$ after detecting the leave of the parent is as follows. Assume that peer $j$ selected peer $i$ as a candidate; i.e., it has successfully reserved the bandwidth of $i$ of amount $\alpha$. If $c[i] \geq 1$, peer $j$ simply connects with peer $i$ after the leaving of its (former) parent. If there are $k$ peers which select $i$ as a candidate and the use of the bandwidth by peer $j$ reduces $c[i]$ to be less than $1+(k-1)\alpha$, $i$ requests those $k$ peers to change the candidate from $i$, in an increasing order of receiving such a request, until the condition of $c[i] \geq 1+(k-1)\alpha$ holds (note that the value of $k$ decreases if a peer changes its candidate from $i$).

3.2 Prevention from the Occurrence of a Loop

Our second technique is to prevent from the occurrence of a cyclic reference of the candidate peers. We propose two ideas in this subsection. The first idea, which will be called Proposal 2a hereafter, is to prepare a list of peers at each peer to keep the set of ancestors in the tree-bone. For example, if peer $c$ receives the stream from the source through tree-bone peers $a$ and $b$, then the list held by $c$ is determined as $[a,b,c]$. This list is updated whenever the parent changes, by receiving the list held by the new parent and by adding itself to the end of the received list. If the updated list contains a peer $j$ which selects $i$ as a candidate, since this implies that $j$ is selecting a descendant peer as a candidate, peer $i$ sends a message to $j$ to change the candidate from $i$. For each of the other children $j'$, $i$ forwards the updated list to $j'$ to keep their list up-to-date.

The second idea, which will be referred to as Proposal 2b hereafter, is to prepare a local variable representing the depth in the tree-bone for each peer\(^1\), and to select a peer to have the depth smaller than the current peer as the candidate. With such a variable, as long as the value of the variable correctly reflects the actual depth of the peer, it can prevent from the occurrence of a cyclic reference to the candidates. The power of this technique is heavily dependent on the accuracy of the variables, which causes a trade-off between the risk of cyclic reference and the cost for the maintenance.

3.3 Recovery from Cyclic Reference

The third technique, which is called Proposal 3 hereafter, is to realize a quick recovery from cyclic reference to the candidates. The basic idea is to transmit a “probe message” toward the root of the tree-bone whenever a peer changes its parent. Each peer can proactively detect the occurrence of a cycle by receiving the message transmitted by itself, which can shorten the time required for the recovery from cyclic reference. Upon detecting a cycle, which is conducted merely by the peer which recently changes its parent, the peer disconnects the link to the parent, and tries to find (and connect to) another parent. In addition to the above simple idea, to reduce the number of reconnections as much as possible, we design the scheme in such a way that if a peer which submitted a probe message receives another probe message, it compares its ID with the ID of the originator of the message, and it forwards the message to the parent only when its ID is smaller than the ID of the originator. With such a mechanism, we can disconnect only one link contained in a cyclic reference even if several peers cause a cyclic reference by changing their parent.

4. Evaluation

4.1 Setup

We evaluate the performance of the proposed scheme by simulation using PeerSim simulator [3]. One step of the simulation is set to 0.02 sec and we assume that in each step, each peer conducts the following operations:

1) Receive messages from the input buffer.
2) Calculation according to the received messages.
3) Send 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 do not consider the variance of the delay of the message delivery.

In the simulation, we used two churn models described below. In the first model, peers arrive at the system according to a Poisson distribution with mean $\lambda$ and leave the system according to a Pareto distribution. The probability density function of Pareto distribution is given as

$$f(t) = \frac{kt_{m}^{k}}{t^{k+1}}, \quad \text{(3)}$$

\(^1\)The depth of a peer is defined to be the length of the path from the root to the peer.
where k is called a shape parameter and \( t_m \) is called a scale parameter which is associated with the participation time of the peers, i.e., under this distribution, peers which participate in the system earlier will leave the system with a higher probability. Since those two parameters do not directly control the longevity of peers, in the simulation, we vary parameter \( \lambda \) from 4 to 12 [peers/sec] to virtually control the longevity of peers, since with a larger \( \lambda \), many peers have an earlier participation time which causes a higher leaving probability within a short time. On the other hand, in the second model, all peers participate in the system at the beginning of the session, and a half of peers randomly leave the system during \( T_d \) time units, where parameter \( T_d \) is varied from 100 to 600 sec, i.e., a smaller \( T_d \) implies a shorter life time. In both models, once a peer leaves the system, it will not join the system again.

In the succeeding two subsections, we evaluate the impact of three proposed techniques to the churn tolerance of the underlying P2P live streaming systems. More concretely, as for the first technique, we evaluate the impact of parameter \( \alpha \) and as for the second and the third techniques, we compare the performance with a scheme without such a loop avoidance/recovery mechanism. Metrics used in the evaluation are: 1) the number of fails to connect to the new parent, 2) the time spent for the reconnection after failing the connection to the new parent, and 3) the number of messages. The number of fails is itemized to the reason so that: 1a) fails due to the lack of upload bandwidth, 1b) fails due to the leave of the candidate, and 1c) fails due to the occurrence of a cyclic reference. Similarly, the time for reconnection is itemized to each of the above three reasons. Each value shown in the figures is an average over 100 trials.

Finally, parameters used in the simulation are as follows:

- The length of a live stream \( L \) is 600 sec.
- The total number of peers is 2000.
- The upload bandwidth of the source is 16 times of the full stream rate and the upload bandwidth of the other peers is varied from the fourth to the eighth of the full stream rate.

### 4.2 Proposal 1

This subsection evaluates the impact of parameter \( \alpha \in (0, 1) \) to the performance of the P2P live streaming systems with respect to the number of fails and the time required for the reconnection. We will also observe the impact of the churn rate to the performance.

Figure 1 illustrates the number of fails under the first churn model, where the horizontal axis indicates \( \alpha \) and (a), (b) and (c) correspond to the case of \( \lambda = 4 \), 8 and 12, respectively. The number of fails due to the leave of

---

**Fig. 1:** Impact of parameters \( \alpha \) and \( \lambda \) to the number of fails under the first churn model.

**Fig. 2:** Impact of parameters \( \alpha \) and \( T_d \) to the number of fails under the second churn model.
candidates (indicated by red bars) rapidly decreases as $\alpha$ increases from 0.1 to 0.3, which is apparently because of the effect of the relaxation of the concentration of the selection of a specific peer as a candidate. When $\alpha = 0.1$, the number of fails due to the lack of upload bandwidth and the number of fails due to the leave of candidates gradually increase as the value of $\lambda$ increases from 4 to 8, whereas the former decreases as $\lambda$ further increases from 8 to 12, which is because under such a high churn rate, the fail due to the leave of peers dominates the fail due to the lack of the bandwidth. The number of fails due to the lack of upload bandwidth (indicated by green bars) decreases as $\alpha$ increases and becomes (almost) zero for all values of $\lambda$ by selecting $\alpha$ to be larger than 0.6. Thus we can conclude that Proposal 1 is effective even under a high churn rate.

Figure 2 illustrates the number of fails under the second churn model, where (a), (b) and (c) correspond to the case of $T_d = 600$, 300 and 100, respectively. As for the number of fails due to the lack of the upload bandwidth and the occurrence of a cyclic reference is the same with the case of the first churn model. However, the number of fails due to the leave of candidates (indicated by red bars) significantly increases for large $\alpha$'s, which implies a side-effect of selecting a larger $\alpha$. As a result, the number of fails takes the smallest value when $\alpha$ is around 0.4 to 0.5. When $\alpha = 0.1$, in contrast to the first churn model, the number of fails due to the lack of bandwidth takes almost the same value for all $T_d$'s, while it becomes almost zero for $\alpha \geq 0.6$ as in the case of the first churn model. The analysis of such an interesting phenomenon is left as a future work.

Figure 3 illustrates the impact of $\alpha$ to the time required for the reconnection. Although it takes long time when $\alpha$ is small and the churn rate is high, the reconnection time converges to 1.5 sec as $\alpha$ increases under each churn model. In addition, the value of $\alpha$ which minimizes the reconnection time gradually increases as the churn rate increases; e.g., it changes as $\alpha = 0.2, 0.3$ and 0.4 as the value of parameter $\lambda$ increases as 4, 8 and 12. By letting $\alpha_{\text{min}}$ be the value of $\alpha$ minimizing the reconnection time and $\alpha_{\text{max}}$ be the minimum value of $\alpha$ which stabilizes the reconnection time, the value of $\alpha$ should be determined to satisfy $\alpha_{\text{min}} \leq \alpha \leq \alpha_{\text{max}}$, since a too large $\alpha$ causes frequent fails as was described above. In summary, the result of simulation indicates that an appropriate value of $\alpha$ is around 0.5.

4.3 Proposals 2 and 3

In this subsection, we evaluate the impact of Proposals 2 and 3 to the performance by comparing it with the basic scheme which detects and resolves a cyclic reference in the following manner:

**Basic scheme:** Upon detecting the delay of the stream exceeding 1 sec, a 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.

In the following, we fix the value of parameter $\alpha$ used in Proposal 1 to 0.6.

Figure 4 illustrates the impact of the churn rate to the number of fails to connect to the candidates, where four bars for each churn rate indicate the basic scheme, a scheme with Proposals 2a, 2b and 3, respectively. For each bar, red indicates the number of fails due to the leave of candidates and blue indicates the number of fails due to the occurrence of a cycle. 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, Proposal 2a reduces it to almost zero. However, the number of fails due to the leave of candidates does not decrease by the proposed techniques, and the effect of those techniques slightly differs for each churn model; i.e., in the first model based on the Pareto distribution, the increase under Proposal 3 is the largest and in the second
model, the increase under Proposal 2b is the largest. As for the time required for the reconnection, we found that Proposal 3 significantly reduces the reconnection time for each churn model. The reconnection time slightly increases by Proposal 2a under the first churn model, whereas there are no difference between Proposals 2a and 2b under the second churn model, regardless of the churn rate.

The above results indicate that the combination of Proposals 2a and 3 is the most effective. Thus finally, we evaluate the performance of the combined scheme of Proposals 2a and 3 in detail. Tables 1, 2 and Figure 5 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 high. 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 Proposal 2a is worse than the basic scheme under the first churn model, we can improve the basic scheme by combining it with Proposal 3. Figure 5 compares the number of messages issued in the combined scheme with the basic scheme. From the figure, we can see that the increase of the number of messages caused by the combined scheme is bounded by 20% of the basic scheme.

In summary, the combination of Proposals 2a and 3 could effectively bound the occurrence of cyclic references and reduces the time required for the reconnection by more than 0.4 sec compared with the basic scheme. However, as the churn rate becomes too high, it causes a significant number of fails due to the leave of the candidates, which causes the performance degradation such as the increase of the total number of fails and the number of messages.

5. Concluding Remarks

This paper proposes techniques to increase the resilience of P2P live streaming systems to the simultaneous leave of several peers. The result of simulation indicates that the proposed techniques reduce the number of fails to connect to a candidate (backup) peer and the time required for the reconnection after a fail even under a high churn rate. The refinement of the proposed techniques is left an important future work. The implementation in actual P2P live streaming systems is another crucial issue.

References

Fig. 5: Impact of the combined scheme to the number of messages.


