PAPER

Flash Crowd Absorber for P2P Video Streaming*

Satoshi FUJITA^{†a)}, Member

SUMMARY This paper proposes a method to *absorb* flash crowd in P2P video streaming systems. The idea of the proposed method is to reduce the time before a newly arrived node becoming an uploader by explicitly constructing a group of newly arrived nodes called flash crowd absorber (FCA). FCA grows continuously while serving a video stream to the members of the group, and it is explicitly controlled so that the upload capacity of the nodes is fully utilized and it attains a *nearly optimal* latency of the stream during a flash crowd. A numerical comparison with a naive tree-based scheme is also given.

key words: Peer-to-Peer video streaming, flash crowd, scheduling, optimal broadcast time

1. Introduction

According to the popularization of video sharing services such as YouTube and Hulu, and the performance limitation of content delivery networks (CDNs) due to the intrinsic bottleneck at the edge servers, Peer-to-Peer (P2P) video streaming has attracted considerable attention in recent years [4], [5], [11], [16], [30]. In those systems, video stream is given as a *sequence of chunks* of constant size (e.g., 256 KB) and those chunks are delivered to all subscribers by repeating the relay of chunks among participant nodes. To reduce the latency of a video streaming, those chunks should be delivered to all subscribers as quickly as possible. In addition, since those chunks are given from an external media server in a *consecutive* manner and each node has a limited upload capacity, we need to carefully design a chunk delivery schedule in such a way that the delivery of a chunk is not disturbed by the delivery of other chunks, while keeping the low latency of each chunk**.

Unfortunately, even if we could design an optimal chunk delivery schedule for a fixed node set, in P2P environment, such an optimality of the delivery schedule can be easily disturbed by the join/leave of nodes. In particular, the quality of video streaming significantly degrades if many nodes issue a request to join the system almost at the same time. Such a phenomenon of the concentration of many join requests is referred to as the **flash crowd**, which

has been recognized as a crucial issue for general distributed systems including P2P systems [3], [6], [7], [12], [15], [20], [22], [23], [26]. A reason for the occurrence of a flash crowd in P2P systems is the existence of a non-negligible delay before newly joined downloader becoming an uploader. Recall that the key idea of P2P systems is to encourage each participant to serve as a service provider (uploader) in addition to the role of a consumer (downloader). This implies that the power of uploaders increases as the number of downloaders increases, but when the speed of joining new downloaders is much higher than the speed of deploying new uploaders, the system could not keep the upload capacity required by the downloaders, which causes a significant performance degradation.

This paper proposes a method to absorb flash crowd in P2P video streaming systems. The idea of the proposed method is to explicitly construct a group of newly arrived nodes called **flash crowd absorber** (FCA, for short), and to dynamically grow it while delivering chunks to the nodes in an FCA. Note that in general P2P streaming systems, a newly arrived node firstly contacts to a bootstrapping node called tracker and starts to receive a video stream after connecting to a “rich” node with a sufficient upload capacity. However, if too many nodes scramble for the upload capacity of the existing nodes, it takes long time before connecting to a rich node, since rich nodes are easily exhausted during a flash crowd. FCA is designed to overcome this issue; namely unlike conventional systems, nodes in an FCA are rigidly controlled so that the upload capacity of each node could be fully utilized. In addition, it is designed in such a way to achieve *nearly optimal* latency even during a flash crowd.

The remainder of this paper is organized as follows. Section 2 overviews related work. Section 3 describes an optimal broadcast scheme for static networks. Section 4 describes the details of FCA. Section 5 summarizes the results of performance evaluations. Finally, Sect. 6 concludes the paper with future work.

2. Related Work

There are several techniques to resolve flash crowd in P2P video streaming systems [1], [27]–[29], [31], which can be classified into two types. The first type is to apply an explicit

Manuscript received July 7, 2018.

Manuscript revised October 15, 2018.

Manuscript publicized October 26, 2018.

[†]The author is with the Department of Information Engineering, Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima-shi, 739–8527 Japan.

*This work was partially supported by KAKENHI, the Grant-in-Aid for Scientific Research (B), Grant Number 16H02807.

a) E-mail: fujita@se.hiroshima-u.ac.jp

DOI: 10.1587/transinf.2018EDP7238

**If we do not take care of the latency, one simplest solution is to flow chunks over a linear chain of nodes.

admission control to incoming join requests at the tracker. Recall that the system can accept a limited number of join requests, since it has a limited upload capacity at any point in time. Thus a scramble for the upload capacity of existing nodes could be relaxed by allowing a limited number of selected nodes to send a join request to the tracker. Although it could adaptively adjust the number of join requests to the current network capacity, this naive approach has several drawbacks such as: 1) many peers should wait for their turn in a waiting queue for a long time; and 2) it is difficult to precisely calculate the network capacity in a distributed environment.

The second approach is to construct another overlay called **batch** consisting of newly arrived nodes and to allow few nodes in the batch to connect to the existing overlay. A batch is constructed and controlled by the tracker [8] or by an appropriate cloud service [13]. Although it overcomes a drawback of the first approach since all new nodes could receive chunks through the batch, it is not clear how to design batch to optimize the latency and/or the continuity of received chunks. Let μ denote the upload capacity of nodes indicating that each node can upload μ chunks to other nodes in a step. When $\mu = 1$, a single chunk can be broadcast to n nodes in $\lceil \lg n \rceil$ steps[†] by using a binomial tree structure. However, it is not trivial whether we could achieve such a short latency for each of successively incoming chunks since in the broadcast scheme of $\lceil \lg n \rceil$ steps, all nodes receiving a chunk should become busy to forward the chunk to other nodes until the overall broadcast completes; namely conflict of chunks could easily occur if we want to broadcast a stream of chunks with a short latency.

The problem of minimizing the latency of a stream of chunks was firstly studied by Liu [17]. Liu proved that when $\mu = 1$ and $|V| = 2^m$ for some integer m , the minimum latency of m steps is achieved by a scheme called snow ball algorithm. An extension of the scheme to the cases in which each node has at most k children in the overlay was investigated by Bianchi *et al.* [2] (note that in the model of [2], the capacity of each node is still bounded by one). It is widely known that when $k = 2$, a chunk given to the source in the τ^{th} step is received by at most F_i nodes in the $(\tau + i)^{\text{th}}$ step, where F_i is the i^{th} Fibonacci number^{††}. Bianchi *et al.* extend this observation to general k 's, and derived optimal broadcast schemes for such cases. More recently, Fujita proposed a scheme which attains optimal latency of $\lceil \lg |V| \rceil$ steps by using an overlay network with degree $O(\lg^2 |V|)$ [9], which is an explicit implementation of the snow ball scheme suggested by Liu [17]. This scheme is extended in such a way that each broadcast completes in $\lg |V| + o(\lg |V|)$ steps when the number of children is bounded by $O((f(|V|))^2)$ for any function f satisfying $f(x) = \omega(1)$ and $f(x) = o(\lg x)$ [9]. In addition to the above theoretical results, many researches have pointed out that the structure of the overlay plays an

important role in realizing a short latency in P2P streaming systems [4], [5], [11], [14], [16], [18], [19], [21], [24], [25], [30]. In this paper, we will propose a method to organize a batch during a flash crowd with a nearly optimal chunk stream delivery scheme. The reader should note that the structure and the size of such an optimal batch should *dynamically change* as the number of new nodes increases. The proposed overlay consists of a collection of amplifiers and the terminator, where the first part is equivalent to the multi-tree-structure proposed in the literature [18] and the second part plays a crucial role in absorbing the increase of the number of participants during a flash crowd.

3. Scheme for Static Node Set

3.1 A Naive Scheme

Let V be a set of nodes subscribing to a stream of chunks which is issued by an external media server called the **source** in such a way that the i^{th} chunk in the stream is issued in the i^{th} step for each $i \geq 1$. In the following, we assume that set V is statically given and each node in V can upload a chunk to $\mu \geq 2$ nodes in a step^{†††}. The former assumption will be relaxed in Sect. 4, and the latter assumption will be relaxed in Sect. 3.4.

A naive scheme to deliver a stream of chunks to all nodes in V is to construct a μ -ary tree^{††††} T with node set V and to flow chunks over T in such a way that each node forwards a chunk received from the parent in a step to all children in the next step. The latency realized by such a naive scheme is evaluated as follows. Let us define the level of T as the distance from the root; e.g., the root of T is at level 0, children of the root are at level 1, and so on. The **depth** of T is defined to be the maximum level of T , which corresponds to the latency of the naive scheme. There are exponential number of μ -ary trees constructed over V . Since the i^{th} level of T contains at most μ^i nodes, the minimum depth of tree with node set V (where the minimum is taken over exponential number of μ -ary trees constructed over V) is given by the minimum integer ℓ satisfying the following inequality:

$$\sum_{j=0}^{\ell} \mu^j = \frac{\mu^{\ell+1} - 1}{\mu - 1} \geq |V|.$$

By solving it, we have

$$\ell = \lceil \log_{\mu} (|V|(\mu - 1) + 1) \rceil - 1.$$

If the root of the tree directly receives chunks from the source, the minimum latency achieved by the scheme is given by ℓ , while it would be slightly larger than ℓ when the root is connected to a node besides the source.

Unfortunately however, it is difficult to build such an

[†]In this paper $\lg n$ denotes the logarithm of n to base two.

^{††}We can prove the claim by mathematical induction, i.e., in the i^{th} step, nodes which received the chunk in the $(i-1)^{\text{st}}$ and $(i-2)^{\text{nd}}$ steps can send the chunk to the next nodes.

^{†††}If $\mu = 1$, we can broadcast each chunk to all subscribers in $\lceil \lg |V| \rceil$ steps [9], [17].

^{††††}A tree is called k -ary if each node has at most k children.

Table 1 The distribution of the tree depth obtained through random sampling, where the minimum tree depth is given by $\lceil \lg n \rceil$.

	10	11	12	13	14	15	16	17	18
$n = 1024$	0	211	8417	1349	23	0	0	0	0
$n = 2048$	0	0	5	7418	2513	64	0	0	0
$n = 4096$	0	0	0	0	5465	4420	114	1	0

ideal tree in actual P2P environment. In distributed systems such as P2P, each node should decide the set of adjacent nodes (including parent and child) in a distributed manner. To conduct such a decision making as quickly as possible, before conducting the decision making, each node must be aware of: 1) the set of candidate nodes for each level of the current tree, or 2) the depth of subtree rooted at each node in the tree. In fact, without such global information, each node should rely on a random sampling during a depth-first search starting from the root, for example. The tree depth obtained through such a random sampling is worse than optimal as is shown in Table 1, while the cost for calculating such a global information increases as the number of nodes in the tree increases.

3.2 Proposed Scheme

The key idea of the proposed scheme is described as follows (note that although we are considering a static node set in this section, we can extend it to dynamic node set as will be described in Sect. 4)

1. Chunks are disseminated through different paths unlike the naive scheme.
2. The delivery of a chunk to nodes in V consists of two phases. In Phase 1, the number of nodes receiving the chunk exponentially increases, in such a way that $(\mu + 1)^i$ nodes receive the chunk during the first i steps for each $1 \leq i \leq \alpha$, where we assume that exactly one node has the chunk before starting the first step, and α denotes the number of steps in Phase 1. On the other hand, the role of Phase 2 is to complete the broadcast of chunks to all nodes in V by spending β more steps (thus the latency of the resulting scheme is $\alpha + \beta$).
3. Phase 1 uses a tree-structured overlay called **amplifier** which is devoted to the delivery of a chunk; namely each chunk *exclusively* uses an amplifier during α successive steps. This means that in order to deliver several chunks in a concurrent manner, we need to prepare at least α disjoint amplifiers for Phase 1. On the other hand, Phase 2 is realized by another tree-structured overlay called **terminator** which is shared by all amplifiers.

Figure 1 illustrates the proposed overlay network. In the following two subsections, we explain the details of each component.

3.2.1 Amplifier

In this paper, we often call a static subset of nodes with a

Fig. 1 Overview of the proposed overlay.

designated cardinality a block. In particular a block with $(\mu + 1)^i$ nodes is simply called an **i -block**. With the notion of blocks, an **i -amplifier** is formally defined as follows:

- 0-amplifier is a tree consisting of one 0-block; i.e., it is a trivial tree consisting of a single node.
- for each $i \geq 1$, an i -amplifier consists of one $(i - 1)$ -amplifier and $\mu (i - 1)$ -blocks. As is shown in Lemma 1 described below, the number of nodes in an i -amplifier is $(\mu + 1)^i$ for any $i \geq 0$. An $(i - 1)$ -amplifier and $\mu (i - 1)$ -blocks are statically connected by a set of edges in such a way that: 1) each node in the $(i - 1)$ -amplifier is adjacent with μ child nodes in the collection of $\mu (i - 1)$ -blocks and 2) each node in $\mu (i - 1)$ -blocks is adjacent with exactly one parent node in the $(i - 1)$ -amplifier.

Given an $(\alpha - 1)$ -amplifier denoted X , the chunk delivery in Phase 1 proceeds in α steps as follows:

1. Since an i -amplifier is decomposed into one $(i - 1)$ -amplifier and $\mu (i - 1)$ -blocks, by recursively applying such a decomposition, we can have j -amplifier for each $0 \leq j \leq \alpha - 2$. Let $\mathcal{A}$ denote the resulting set of amplifiers generated from X .
2. In the j^{th} step of Phase 1, where $1 \leq j \leq \alpha - 1$, nodes in a $(j - 1)$ -amplifier in $\mathcal{A}$ forward the chunk received during the first $j - 1$ steps to μ children in the overlay to organize a j -amplifier in $\mathcal{A}$.
3. In the α^{th} step, each node in X forwards the received chunk to (at most) μ nodes in the terminator, and then X is released so that it can be used for the delivery of

other chunk.

Lemma 1: For any $i \geq 0$, an i -amplifier contains $(\mu + 1)^i$ nodes.

Proof. We prove the claim by the induction on i . The claim clearly holds for $i = 0$ since 0-amplifier consists of exactly one node. Assume that the claim holds for $i = j$ and consider the case of $i = j + 1$. Since $(j + 1)$ -amplifier consists of one j -amplifier and μ j -blocks, by the induction hypothesis, the number of nodes in a $(j + 1)$ -amplifier is given as

$$(\mu + 1)^j + \mu(\mu + 1)^j = (\mu + 1)^{j+1}.$$

Hence the lemma follows. Q.E.D.

Since we should prepare α $(\alpha - 1)$ -amplifiers to broadcast a stream of chunks, the following claim holds.

Remark 1: Given μ and α , Phase 1 uses at least $\alpha(\mu + 1)^{\alpha-1}$ nodes.

In the following, to clarify the exposition, we often denote the number of nodes in an $(\alpha - 1)$ -amplifier as σ ; i.e., $\sigma \stackrel{\text{def}}{=} (\mu + 1)^{\alpha-1}$.

3.2.2 Terminator

The terminator used in Phase 2 is a collection of μ -ary trees of depth at most $\beta - 1$. Let γ denote the number of trees in the terminator, where the root of each tree receives a chunk from a node in an $(\alpha - 1)$ -amplifier in the α^{th} step of Phase 1. Thus $\gamma \leq (\mu + 1)^{\alpha-1} (= \sigma)$ must hold.

Lemma 2: Given μ and V , the terminator consisting of γ ($\leq \sigma$) trees contains at least $\frac{|V| - \gamma}{\mu}$ nodes.

Proof. Let n_2 be the number of nodes in the terminator. Since each node has upload capacity μ , the terminator consisting of γ trees has an ability of forwarding $(\mu - 1)n_2 + \gamma$ chunks in a step. Since any node in V should be a member of the terminator or should receive a chunk from the terminator, it must satisfy

$$(\mu - 1)n_2 + \gamma + n_2 \geq |V|,$$

which implies

$$n_2 \geq \frac{|V| - \gamma}{\mu}.$$

Hence the lemma follows. Q.E.D.

The above lemma implies that the lower bound on n_2 is proportional to $|V|$; namely the terminator should grow as the number of nodes in V increases. The delivery of chunks in Phase 2 proceeds as follows. Let $U \subset V$ be a subset of nodes which are *not* used in the amplifiers prepared for Phase 1. At first, we statically partition U into γ subsets as evenly as possible, and construct a μ -ary tree for each subset independently. Let $\mathcal{T}$ be the set of resulting trees. Then

for each tree in $\mathcal{T}$, we associate at most $(\mu - 1)n' + 1$ nodes in $V - U$ with the tree as leaves in a static manner, where n' denotes the number of nodes in the tree. As will be described later, the height of trees in $\mathcal{T}$ can be bounded by two for practical number of nodes if $\mu \geq 3$.

3.3 Latency

This subsection evaluates the latency of the proposed scheme. Recall that the latency of the scheme is given by $\alpha + \beta$, and if $\alpha = 0$, the scheme is equivalent to the naive scheme described in Sect. 3.1. The following lemma gives a tight upper bound on α to enable the construction of the proposed overlay:

Lemma 3: Given μ and V , we can construct the proposed overlay if and only if

$$\alpha \leq \left(1 - \frac{1}{\mu}\right) \frac{|V|}{(\mu + 1)^{\alpha-1}}.$$

Proof. The proposed scheme correctly works if and only if the number of nodes used in two phases does not exceed $|V|$. Thus by Remark 1 and Lemma 2, this condition is represented as

$$\alpha\sigma + \frac{|V| - \gamma}{\mu} \leq |V|. \quad (1)$$

Since $1 \leq \gamma \leq \sigma$, Eq. (1) holds if $\alpha \leq \left(1 - \frac{1}{\mu}\right) \frac{|V|}{\sigma}$ and does not hold if $\alpha > \left(1 - \frac{1}{\mu}\right) \frac{|V|}{\sigma} + \frac{1}{\mu}$. Since $\mu \geq 2$, the lemma follows. Q.E.D.

Lemma 4: Let $m = \lceil \log_{\mu+1} |V| \rceil$. If $\alpha = m - \delta$ for some integer $\delta \geq 1$, then

$$(\mu + 1)^\delta < \frac{|V|}{\sigma} \leq (\mu + 1)^{\delta+1}.$$

Proof. Immediate from $(\mu + 1)^{m-1} < |V| \leq (\mu + 1)^m$ and $\sigma = (\mu + 1)^{\alpha-1} = (\mu + 1)^{m-\delta-1}$. Q.E.D.

Corollary 1: If $\alpha = m - \delta$ for some $\delta \geq 1$, then we can construct the proposed overlay if

$$\alpha < \left(1 - \frac{1}{\mu}\right) (\mu + 1)^\delta.$$

By the corollary, we can increase the number of steps taken by Phase 1 to $m - 2$ if $m \leq 6$ for $\mu = 2$; $m \leq 12$ for $\mu = 3$; or $m \leq 20$ for $\mu = 4$ (the last two cases corresponds to V with more than 16 million nodes). More precisely, when $\mu = 2$, $m = 7$, and $\alpha = m - 2 (= 5)$, since $\sigma = (\mu + 1)^{\alpha-1} = 3^4 = 81$, the condition in Lemma 3 is satisfied if $810 \leq |V| \leq 2187 (= 3^7)$, but is not satisfied if $730 (= 3^6 + 1) \leq |V| \leq 809$, while it is satisfied again if $|V| \leq 729 (= 3^6)$. On the other hand, when $\mu = 2$ and $\alpha = m - 3$, the sufficient condition is

satisfied for $m \leq 13$ which corresponds to V with less than $3^{13} \approx 1.59 \times 10^6$ nodes.

Thus we have the following remark.

Remark 2: If the number of nodes in V is smaller than 1.5 million, we can set α to $m - 2$ for $\mu \geq 3$ and set α to $m - 3$ for $\mu = 2$, where $m = \lceil \log_{\mu+1} |V| \rceil$.

Lemma 5: Suppose that the terminator contains $\frac{|V|}{\mu}$ nodes and they are partitioned into σ subsets as evenly as possible, where $\sigma = (\mu + 1)^{\alpha-1}$. If $\alpha = m - \delta$ for some $\delta \geq 1$, then the size of each subset is at most $\frac{(\mu+1)^\delta}{\mu}$, where $m = \lceil \log_{\mu+1} |V| \rceil$.

Proof. Immediate from Lemma 4.

Q.E.D.

If we construct a complete μ -ary tree with depth $\beta - 1$ for each subset, to accommodate all nodes shown in the above lemma, β should satisfy the following inequality:

$$(\mu + 1)^{\delta-1} \leq \sum_{j=0}^{\beta-1} \mu^j = \frac{\mu^\beta - 1}{\mu - 1} \quad \text{i.e.,}$$

$$\mu^\beta \geq (\mu + 1)^\delta \left(1 - \frac{1}{\mu}\right) + 1$$

With numerical calculations, we can verify that for each $2 \leq \mu \leq 7$ and $1 \leq \delta \leq 3$, the above inequality holds when $\beta = \delta + 1$.

Remark 3: If $\alpha = m - \delta$, then $\beta \leq \delta + 1$ for any $2 \leq \mu \leq 7$ and $1 \leq \delta \leq 3$.

By Remarks 2 and 3, we have the following claim.

Remark 4: If the number of nodes in V is smaller than 1.5 million, then we can bound the latency of the proposed scheme by $m + 1$ for any $\mu \geq 2$, and can bound the depth of each tree in the terminator by two for $\mu \geq 3$ and by three for $\mu = 2$, where $m = \lceil \log_{\mu+1} |V| \rceil$. Since any broadcast takes at least m steps, it takes at most one more step than the trivial lower bound.

3.4 Non-Uniform Case

In Phase 1, each node is expected to forward a received chunk to μ child nodes in a step. However, since such a forwarding is simultaneously done by all nodes contained in the same block, Phase 1 correctly works as long as the average capacity of the nodes contained in the same block is (at least) μ , even if there is a node with a capacity less than μ . Similarly, since Phase 2 expects the terminator to forward a received chunk to a specific number of nodes determined by α and β , it still works correctly if the average capacity of the nodes in the terminator is (at least) μ .

4. Flash Crowd Absorber

This section proposes flash crowd absorber (FCA) which is an application of the chunk delivery scheme proposed in Sect. 3 to the cases in which node set dynamically grows.

Let V denote the current node set and assume that Phase 1 of the scheme is realized by a collection of $(\alpha - 1)$ -amplifiers. Recall that the number of nodes in an $(\alpha - 1)$ -amplifier and the maximum number of trees in the terminator are both given by $\sigma = (\mu + 1)^{\alpha-1}$. Suppose that the number of nodes in the system now increases from $|V|$ to $|V| + \Delta$.

In Sect. 3, we clarify the number of nodes used in the proposed overlay. For example, Lemma 2 derives a lower bound on the number of nodes in the terminator. The terminator is designed so that it can contain more nodes than the lower bound, as long as the total number of nodes used in the overlays does not exceed $|V|$. Thus if we do not mind the tree depth in the terminator, we can *absorb* the increase Δ of the number of nodes by increasing the number of nodes in the terminator to $|V| + \Delta - \alpha\sigma$. However, if we want to bound the depth of trees by a constant to keep the low latency of the chunk streaming, we must increase the *number* of trees in the terminator.

Since the number of trees in the terminator cannot exceed σ , to increase the number of trees, we must increase the *size* of amplifiers from $\alpha - 1$ to α , and before doing so, we must increase the *number* of $(\alpha - 1)$ -amplifiers from α to $\alpha + 1$. Thus the way of absorbing the increase Δ of the number of nodes while keeping the low latency of the chunk delivery is described as follows.

1. Increase the number of nodes in the terminator up to the bound determined by the number of trees σ and the depth of trees $\beta - 1$, which is given as

$$\sigma \times \sum_{j=0}^{\beta-1} \mu^j.$$

Note that the resulting terminator can forward the chunk stream to $\sigma\mu^\beta$ nodes, which implies that the maximum number of nodes covered by the resulting terminator is

$$\sigma \times \sum_{j=0}^{\beta} \mu^j = (\mu + 1)^{\alpha-1} \frac{\mu^{\beta+1} - 1}{\mu - 1}.$$

2. In the above (critical) situation, at most $\sigma\mu^\beta$ nodes are *unused* as a member of the overlays. During the following steps, all unused nodes receive chunks from the terminator, while they might be used as a member of an amplifier.
3. If the number of unused nodes reaches σ , those nodes are organized as an $(\alpha - 1)$ -amplifier and are removed from the set of unused nodes. After completing the organization, it becomes an $(\alpha - 1)$ -amplifier which increases the number of $(\alpha - 1)$ -amplifiers used in Phase 1 from α to $\alpha + 1$.
4. If the number of unused nodes reaches $(\alpha + 1)\mu\sigma$, those nodes are used for growing $(\alpha - 1)$ -amplifiers to α -amplifiers. Recall the the growth of an $(\alpha - 1)$ -amplifier to an α -amplifier needs $\mu\sigma$ nodes (i.e., $\mu(\alpha - 1)$ -blocks) and there are $\alpha + 1$ amplifiers at this time.

5. After completing the growth of $(\alpha - 1)$ -amplifiers, it switches the route of chunks in each amplifier so that nodes in $(\alpha - 1)$ -blocks forwards received chunk to the terminal. Such a switch can be realized in a distributed manner by embedding a control signal to the head of transmitted chunk. Note that the switch of the routes increases the latency by exactly one.
6. Now the size of amplifiers used in Phase 1 increases by one, which increases the maximum number of trees in the terminator by $\mu + 1$ times.

The above procedure correctly works without increasing the tree depth in the terminator from $\beta - 1$ if it holds $\alpha\sigma + (\alpha + 1)\mu\sigma + \sigma \leq \sigma\mu^\beta$, namely if

$$(\alpha + 1)(\mu + 1) \leq \mu^\beta.$$

The above inequality holds if

$$\alpha \leq \frac{\mu^\beta}{\mu + 1} - 1.$$

Numerical calculations indicate that when $\mu = 2$, β must be larger than three to apply the procedure to marginal number of nodes such as 100 or 1000. However, it correctly works for $\beta = 3$ if $\mu \geq 3$ and $|V| \leq 1.6 \times 10^4$.

Remark 5: If $\mu \geq 3$ and $|V| \leq 1.6 \times 10^4$, FCA absorbs the increase Δ of the number of nodes and converges to the overlay for $|V| + \Delta$ nodes, by fixing the maximum tree depth in the terminator to two.

Finally, to implement FCA in real-world P2P environments, we need to design the way of cooperation among peers and the low overhead synchronization among them. In particular, a long communication delay between distant peers (e.g., more than 200 ms) would become a big issue in an implementation paper. However, since FCA has a modular structure, it is not difficult to organize the whole overlay as a collection of individual components such as amplifiers and the terminator. For example, a distributed implementation of amplifier has been addressed in our recent paper [10].

5. Evaluation

This section evaluates the performance of FCA during flash crowd. More specifically, we trace the behavior of FCA under a scenario in which the set of nodes successively grows from an initial overlay, and evaluate the transition of the latency of the resulting scheme.

5.1 Running Example

In this subsection, we consider the case of $\mu = 2$. Since $\sigma = (\mu + 1)^{\alpha-1}$, a network with nine nodes illustrated in Fig. 2 is the smallest overlay for $\mu = 2$ and $\alpha = 2$. Table 2 summarizes the number of nodes used in Phase 1 and the upload capacity of the terminator for each combination of α and β .

Scenario 1: Suppose that the maximum depth of trees

Fig. 2 The minimum overlay for $\mu = 2$ and $\alpha = 2$. Nodes a and b alternately receive chunks from the source.

Table 2 Critical values in the proposed overlay ($\mu = 2$).

(a) The number of nodes used in Phase 1 for each α .

	α					
	2	3	4	5	6	7
Phase 1	6	27	108	405	1458	5103

(b) The capacity of terminator for each α and β .

		α					
		2	3	4	5	6	7
β	3	24	72	216	648	1944	5832
	4	48	144	432	1296	3888	11664
	5	96	288	864	2592	7776	23328

in the terminator is fixed to two ($= \beta - 1$). When $\sigma = 3$ and $\beta = 3$, the terminator can contain at most $21(= \sigma(1 + \mu + \mu^2))$ nodes, and when the terminator is saturated with 21 nodes, it can forward chunks to $24(= \sigma\mu^3)$ nodes; i.e., it can accommodate at most $45(= 21 + 24)$ nodes including six nodes used in Phase 1. However, it is not enough to increment α to three, since Phase 1 needs $27(> 24)$ nodes for $\alpha = 3$, while the latency of the scheme increases from 3 to $5(= \alpha + \beta)$ during the scenario.

Scenario 2: Suppose that the maximum depth of trees in the terminator is fixed to three ($= \beta - 1$). When $\sigma = 3$ and $\beta = 4$, the terminator can contain at most $45(= \sigma(1 + \mu + \mu^2 + \mu^3))$ nodes, and when the terminator is saturated with 45 nodes, it can forward chunks to $48(= \sigma\mu^4)$ nodes, which is enough to increment α to three, since by using $21(= 27 - 6) < 48$ unused nodes, we can increment the number of amplifiers by one and increment the size of each amplifier by one.

When $\alpha = 3$, the terminator can contain at most 135 nodes, and when the terminator is saturated with 135 nodes, it can forward chunks to 144 nodes, which is enough to increment α to four, since Phase 1 needs $108(< 144)$ nodes for $\alpha = 4$. When $\alpha = 4$, the terminator contains at most 405 nodes, and when the terminator is saturated with 405 nodes, it can forward chunks to 432 nodes, which is enough to increment α to five, since Phase 1 needs $405(< 432)$ nodes for $\alpha = 5$. When $\alpha = 5$, the terminator can contain at most 1215 nodes, and when the terminator is saturated with 1215 nodes, it can forward chunks to 1296 nodes. However, it is *not enough* to increment α to six, since Phase 1 needs $1458 > 1296$ nodes for $\alpha = 6$. Note that the latency of the

Fig. 3 Latency of schemes for given number of nodes (the curve of scenario 1 is “hidden” by the curve of scenario 2).

scheme increases from 3 to 9 during the scenario.

5.2 Comparison to Naive Scheme

Figure 3 illustrates the latency of the resulting scheme for $\mu = 2$, where the horizontal axis is the number of nodes in a logarithmic scale. For example, if the number of nodes added to the initial network (consisting of 9 nodes) is $\Delta = 100$, it absorbs all of them and attains the latency of 7 steps by conducting the saturation of the terminator and the increment of α . On the other hand, if $\Delta = 2000$, it absorbs all of them and attains the latency of 8 steps by repeating the saturation of the terminator and the increment of α twice. Note that as is shown in the figure, the resulting latency beats the latency of the naive scheme if $\Delta \geq 245$.

6. Concluding Remarks

This paper proposes a method to resolve flash crowd in P2P video streaming systems. In the proposed method, upon detecting the occurrence of a flash crowd, it generates a group of new nodes called flash crowd absorber and grows its size while continuing the delivery of chunks to the members of the group. If the average upload capacity of nodes is at least three (i.e., if it can forward a received chunk to three nodes in a step, on average) and the number of nodes does not exceed 1.6×10^4 , the proposed method correctly absorbs the increase of the number of nodes while keeping a low latency of the chunk delivery, which is larger than a trivial lower bound by *at most one*; namely it attains a nearly optimal latency. A future work is to conduct simulations to evaluate the time required for the dynamic growth of the overlay.

References

[1] N. Bayat and H. Lutfiyya, “Network Coding for Coping with Flash Crowd in P2P Multi-Channel Live Video Streaming,” In Proc. the 11th International Conference on the Design of Reliable Communication Networks (DRCN), pp.243–246, 2015.

[2] G. Bianchi, N.B. Melazzi, L. Bracciale, F. Lo Piccolo, and S. Salsano, “Streamline: An Optimal Distribution Algorithm for Peer-to-Peer Real-Time Streaming,” IEEE Trans. Parallel Distrib. Syst., vol.21, no.6, pp.857–871, June 2010.

[3] C. Carbutaru, Y.M. Teo, B. Leong, and T. Ho, “Modeling Flash Crowd Performance in Peer-to-Peer File Distribution,” IEEE Trans. Parallel Distrib. Syst., vol.25, no.10, pp.2617–2626, 2014.

[4] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “SplitStream: High-Bandwidth Content Distribution in Cooperative Environments,” In Proc. of ACM SOSP’03, pp.298–313, 2003.

[5] M. Chen, M. Chiang, P. Chou, J. Li, S. Liu, and S. Sengupta, “P2P Streaming Capacity: Survey and Recent Results,” In Proc. of 47th Annual Allerton Conference on Communication, Control, and Computing, pp.378–387, 2009.

[6] Y. Chen, B. Zhang, and C. Chen, “Modeling and Performance Analysis of P2P Live Streaming Systems under Flash Crowds,” In Proc. the International Conference on Communications (ICC), pp.1–5, 2011.

[7] Y. Chen, B. Zhang, C. Chen, and D.M. Chiu, “Performance Modeling and Evaluation of Peer-to-Peer Live Streaming Systems Under Flash Crowds,” IEEE/ACM Trans. Netw., vol.22, no.4, pp.1106–1120, 2014.

[8] T.Y. Chung and O. Lin, “A Batch Join Scheme for Flash Crowd Reduction in IPTV Systems,” In Proc. the 17th ICPADS, pp.823–828, 2011.

[9] S. Fujita, “Optimal Serial Broadcast of Successive Chunks,” Theoretical Computer Science, vol.575, pp.3–9, 2015.

[10] S. Fujita, “Broadcasting a Stream of Chunks in Heterogeneous Networks with a Short Maximum Broadcast Time,” Journal of Interconnection Networks, vol.18, no.2n03, 2018.

[11] Y. Guo, C. Liang, and Y. Liu, “AQCS: Adaptive Queue-Based Chunk Scheduling for P2P Live Streaming,” In Proc. of IFIP networking, vol.4982, pp.433–444, 2008.

[12] S. Jia, R. Zhang, Y. Ma, L. Zhong, and C. Xu, “Modeling and Optimization of Bandwidth Supply Performance for Cloud-Assisted Video Systems under Flash Crowd,” IEEE Journals and Magazines, vol.13, no.9, pp.151–162, 2016.

[13] S. Jinbo, B.A. Alghazawy, and S. Fujita, “Batch-based flash crowd relaxation in cloud-assisted P2P live streaming,” In Proc. the 18th IEEE/ACIS Int’l Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp.95–100, 2017.

[14] R. Kumar, Y. Liu, and K. Ross, “Stochastic Fluid Theory for P2P Streaming Systems,” In Proc. of INFOCOM, pp.919–927, 2007.

[15] B. Li, G.Y. Keung, S. Xie, F. Liu, Y. Sun, and H. Yin, “An Empirical Study of Flash Crowd Dynamics in a P2P-Based Live Video Streaming System,” In Proc. GLOBECOM, pp.1–5, 2008.

[16] C. Liang and Y. Guo, and Y. Liu, “Is Random Scheduling Sufficient in P2P Video Streaming?,” In Proc. of 28th ICDCS, pp.53–60, 2008.

[17] Y. Liu, “On the Minimum Delay Peer-to-Peer Video Streaming: How Realtime Can It Be?,” In Proc. of MULTIMEDIA ’07, pp.127–136, 2007.

[18] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, “Performance Bounds for Peer-Assisted Live Streaming,” In Proc. of SIGMETRICS, pp.313–324, 2008.

[19] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P.A. Chou, “P2P Streaming Capacity under Node Degree Bound,” In Proc. of ICDCS, pp.587–598, 2010.

[20] F. Liu, B. Li, L. Zhong, B. Li, H. Jin, and X. Liao, “Flash Crowd in P2P Live Streaming Systems: Fundamental Characteristics and Design Implications,” IEEE Trans. Parallel Distrib. Syst., vol.23, no.7, pp.1227–1239, 2012.

[21] L. Massoulie, A. Twig, C. Gkantsidis, and P. Rodriguez, “Randomized Decentralized Broadcasting Algorithms,” In Proc. of INFOCOM, pp.1073–1081, 2007.

[22] Y. Niu, B. Luo, F. Liu, J. Liu, and B. Li, “When Hybrid Cloud

- Meets Flash Crowd: Towards Cost-Effective Service Provisioning," In Proc. of INFOCOM, pp.1044–1052, 2015.
- [23] Y. Niu, F. Liu, X. Fei, and B. Li, "Handling Flash Deals with Soft Guarantee in Hybrid Cloud," In Proc. of INFOCOM, pp.1–9, 2017.
 - [24] T. Nguyen, K. Kolazhi, R. Kamath, S. Cheung, and D.A. Tran, "Efficient Multimedia Distribution in Source Constraint Networks," IEEE Trans. Multimedia, vol.10, no.3, pp.523–537, 2008.
 - [25] 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.
 - [26] Y.M. Teo, "Modeling Flash Crowd Performance in Peer-to-Peer Systems: Challenges and Opportunities," In Proc. the 5th International Conference on Intelligent Systems, Modeling and Simulation, pp.3–4, 2014.
 - [27] H. Wu, H. Jiang, J. Liu, Y. Sun, J. Li, and Z. Li, "How P2P live streaming systems scale quickly under a flash crowd?," In Proc. the 30th International Performance Computing and Communications Conference, pp.1–8, 2011.
 - [28] H. Wu, J. Liu, H. Jiang, Y. Sun, J. Li, and Z. Li, "Bandwidth-Aware peer selection for P2P live streaming systems under flash crowds," In Proc. the 31st International Performance Computing and Communications Conference (IPCCC), pp.360–367, 2012.
 - [29] H. Wu, K. Xu, M. Zhou, A.K. Wong, J. Li, and Z. Li, "Multiple-tree topology construction scheme for P2P live streaming systems under flash crowds," In Proc. Wireless Commun. and Networking Conference (WCNC), pp.4576–4581, 2013.
 - [30] S. Xie, B. Li, Y. Keung, and X. Zhang, "Coolstreaming: Design, Theory, and Practice," IEEE Trans. Multimedia, vol.9, no.8, pp.1661–1671, 2007.
 - [31] G. Xu, Y. Hu, Y. Shen, and M. Guo, "DirectedPush - A High Performance Peer-to-Peer Live Streaming System Using Network Coding," In Proc. the 18th ICPADS, pp.292–298, 2012.

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, and

IEEE Computer Society.