Flash Crowd in P2P Live Streaming Systems: Fundamental Characteristics and Design Implications

Fangming Liu, *Member*, *IEEE*, Bo Li, *Fellow*, *IEEE*, Lili Zhong, Baochun Li, *Senior Member*, *IEEE*, Hai Jin, *Senior Member*, *IEEE*, and Xiaofei Liao, *Member*, *IEEE*

Abstract—Peer-to-peer (P2P) live video streaming systems have recently received substantial attention, with commercial deployment gaining increased popularity in the internet. It is evident from our practical experiences with real-world systems that, it is not uncommon for hundreds of thousands of users to choose to join a program in the first few minutes of a live broadcast. Such a severe flash crowd phenomenon in live streaming poses significant challenges in the system design. In this paper, for the first time, we develop a mathematical model to: 1) capture the fundamental relationship between time and scale in P2P live streaming systems under a flash crowd, and 2) explore the design principle of population control to alleviate the impact of the flash crowd. We carry out rigorous analysis that brings forth an in-depth understanding on effects of the gossip protocol and peer dynamics. In particular, we demonstrate that there exists an upper bound on the system scale with respect to a time constraint. By trading peer startup delays in the initial stage of a flash crowd for system scale, we design a simple and flexible population control framework that can alleviate the flash crowd without the requirement of otherwise costly server deployment.

Index Terms—Live video streaming, peer-to-peer, flash crowds, population control.

1 Introduction

Recently, the internet has witnessed a significant increase in the popularity of peer-to-peer (P2P) live video streaming applications, that deliver real-time and sustained media content to potentially millions of users [1], [2]. Measurement studies [1], [3] on real-world P2P live streaming systems reveal that video streaming performance can be typically maintained at a high level once systems have reached a reasonable scale. However, this is challenged by a severe phenomenon called the *flash crowd*, in which there could be a large number of peers arriving at the system during the initial few minutes of a live broadcast. This is evident in our empirical experiences from Coolstreaming+ [4] that, it is considerably more challenging for a P2P live streaming system to accommodate such an abrupt surge of newly arrived peers, requiring reasonable streaming qualities and low initial startup delays. As a result, we also observe a considerable portion of peers undergoing a flash crowd could opt to leave the system due to impatience, which leads to more dynamic system scaling behavior and more constrained limits on the system scale.

In real-world systems, the flash crowd is usually handled by the deployment of a potentially large number of servers (e.g., 60 dedicated servers in Coolstreaming+ [2]) or by leveraging content delivery networks [5]. This is apparently not cost effective. On the other hand, it is widely known and was confirmed in earlier experiments with Coolstreaming [6], in which one or two servers were sufficient to support tens of thousands of peers once the system reached an adequate scale. What this implies is that the servers are only necessary during the initial ramp up process as an accelerator. In other words, the video streaming quality in a P2P live streaming system can be self-sustained without the need for expensive server deployment once the system reaches a reasonable scale. Therefore, our focus in this paper is to examine the fundamental characteristics of a flash crowd. Based on these characteristics, we seek to design a simple and flexible population control framework that helps to alleviate the impact of a flash crowd, which offers an alternative and complementary solution to the use of dedicated servers. Our original contributions in this paper are two-fold:

First, understanding flash crowds. We first analyze the inherent relationship between scale and time in P2P live streaming systems in the flash crowd scenario (henceforth referred to as *scale-time*). We derive an upper bound of the system scale over time. The key insight from the analysis is that having enough upload bandwidth alone is insufficient to sustain a flash crowd, as it takes time for peers to locate available resources and content. We further quantitatively characterize how the system scale is constrained by the timing constraint under typical gossip protocols, in which

- *F. Liu, H. Jin, and X. Liao are with the Services Computing Technology and System Lab, Cluster and Grid Computing Lab in the School of Computer Science and Technology, Huazhong University of Science and Technology, No. 5 Eastern Building, No. 1037 Luoyu Road, Hongshan District, Wuhan 430074, China. E-mail: {fmliu, hjin, xfliao}@mail.hust.edu.cn.*
- *B. Li and L. Zhong are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon 999077, Hong Kong. E-mail: {bli, lilizh}@cse.ust.hk.*
- *B. Li is with the Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, ON M5S 3G4, Canada. E-mail: bli@eecg.toronto.edu.*

Manuscript received 23 Mar. 2011; revised 23 Oct. 2011; accepted 26 Oct. 2011; published online 29 Nov. 2011.
Recommended for acceptance by E. Leonardi.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-03-0168.
Digital Object Identifier no. 10.1109/TPDS.2011.283.

only partial knowledge of peers and their competition for limited upload bandwidth resources in the system are taken into account. Motivated by our empirical experiences [4], our model also demonstrates the effects of peer churns with general peer arrival and departure patterns on the system scale.

Second, controlling the population of flash crowds. From the perspective of system design, our analysis indicates that we can trade peer startup delays during a flash crowd to achieve better system scale. This inspires us to explore the design and potential benefits of *population control* to alleviate negative effects of a flash crowd. Specifically, we identify critical criteria to make decisions of controlling the population of a flash crowd, by analyzing an inherent relationship between how system scale increases and how peers arrive in P2P live streaming systems. Based on this, we first design an ideal population control procedure with its fundamental limit in improving the system scale, which provides a benchmark for a more practical design and performance evaluation. Furthermore, by investigating the robustness and sensitivity of population control, we propose a simple and flexible framework with practical guidelines.

To our knowledge, this paper represents the first attempt to provide an analytical characterization and understanding of the inherent scale-time relationship in P2P live streaming systems, with a particular focus on the effects of the flash crowd and how it can be controlled. In addition, our analytical framework offers the flexibility to examine the effects of various critical factors including the initial system scale, the scale of the flash crowd, peer upload capacities, and the number of partners each peer has. Our simulations have validated the model and analytical results in a wide range of settings. We believe that this work provides substantial understanding complementary to the existing literature, and offers insightful guidelines toward the future system design and optimization.

The remainder of this paper is organized as follows: Section 2 discusses our contributions in the context of related work. Section 3 presents our theoretical model for P2P live streaming systems under a flash crowd, along with its scale-time relationship and roles of various critical factors. Section 4 carries out a series of sensitivity analysis to demonstrate the effects of various critical factors on system scalability and peer startup delays, verified by simulations. Section 5 proceeds to explore the design and potential benefits of population control for alleviating flash crowds. Finally, Section 6 concludes the paper.

2 Related Work

With respect to analytical studies on P2P live streaming systems, Kumar et al. [7] have derived the maximum streaming rate for churnless systems and developed a stochastic fluid model with peer churn to examine its performance. There were also a number of analyses on the performance bounds of tree-based or mesh-based systems in terms of streaming rate, delay, and server load (e.g., [8], [9], [10], [11]), particularly through the perspective of chunk dissemination to participating peers. Zhou et al. [12] have compared, through a stochastic model, different chunk scheduling strategies based on the performance metrics of continuity and startup latency. While recognizing the significance of these prior works, our study is different from and complementary to them in that: we analyze the asymptotic scaling behavior of P2P live streaming systems during a flash crowd, which provides an in-depth understanding on the inherent scale-time relationship and the effects of partial knowledge of peers and peer churn. This further leads to useful design guidelines of population control for alleviating flash crowds.

Flash crowd issues were examined in other P2P applications. Yang and Veciana [13] used a branching process model to examine the service capacity of BitTorrent-like file sharing systems during flash crowds. Another modeling work [14] evaluated the scalability of a distributed randomized P2P search protocol that provides transmission of objects from servers currently suffering with flash crowds. Hoßfeld and Leibnitz [15] compared the file download performance of client/server and P2P systems while considering flash crowd arrivals and user impatience. A recent study [16] investigated the transient behavior and performance of a file sharing system, which shifts from a congested state where all servers available are saturated by file download demands to a state where a growing number of servers are idle. Differing from these works, we examine the flash crowd problem in P2P live video streaming systems, which is subject to the requirements for an adequate bandwidth to sustain the streaming rate and stringent startup delays with respect to peer impatience. This is more critical than other types of P2P applications such as file sharing, in which either the peer joining process can be stretched over a relatively longer period of time or the application itself can tolerate relatively longer delays.

More recently, Wang et al. [17] considered a class of algorithms and policies that provide statistically guaranteed average streaming quality in the steady state, by ensuring a certain probability for a video channel to have sufficient overall upload bandwidth. In contrast, in this paper, we both qualitatively and quantitatively demonstrate that, solely relying on the condition of overall upload bandwidth resources is insufficient to guarantee the scalability and performance of P2P live streaming systems undergoing a flash crowd. We reveal that this is due to the scale-time constraint, the incomplete knowledge of peers and their resource competition. In practice, most commercial P2P live streaming systems (e.g., PPLive [1], UUSee [18]) Livesky [5] do not impose any explicit population control for dealing with flash crowds; instead, they overprovision dedicated server bandwidth capacities. While our framework is general and flexible to allow the use of such approaches, we further propose the design of population control mechanisms as an alternative and complementary approach, especially when costly server bandwidth capacities are insufficient to meet exploding user demand.

3 System Model and Fundamental Principles

3.1 System Model

In this section, we present our basic model for P2P live video streaming systems under a flash crowd, with the

TABLE 1
Key Parameters in the System Model

Notation	Definition
M	Initial system scale.
N	Flash crowd scale.
R	Video streaming rate ($= xr$).
u_i	Upload capacity of peer i.
h_i	Relative surplus upload capacity of peer i ($= (u_i - R)/r$).
u	Average peer upload capacity.
h	Relative average peer surplus capacity ($= (u - R)/r$).
k	Number of partners of a new peer ($\geq x$).
$S(t)$	System scale (number of existing peers) in the t-th time slot.
U_s	Server capacity provisioning.
u_s	Relative server capacity provisioning ($= U_s/R$).

assumptions and notations summarized in Table 1. We consider a video with rate $R = xr$ to be streamed to all participating peers, where r is the bit rate corresponding to one unit of bandwidth, and R corresponds to the bandwidth requirement of x units. This can be related to the concept of *substreams* in *Coolstreaming+* [3], a large-scale real-world P2P live streaming system, in which a media stream is divided into multiple substreams and peers could subscribe to different substreams from different partners.

For a peer i, let u_i denote its upload capacity of the peer. The peer download capacity is assumed not to be the bottleneck, which is in accordance with most of the recent internet access technologies and measurement studies of existing P2P systems [19]. Given a streaming rate R, we define the *relative surplus upload capacity* h_i of a peer i as the ratio of $(u_i - R)$ to r. Let u be the average peer upload capacity and $h = (u - R)/r$ be the relative average peer surplus capacity. The intuition of h is consistent with that of the *Resource Index* [20] in reflecting the bandwidth resource capacity of the system with respect to the streaming rate R. Specifically, the Resource Index is defined as the ratio of the *entire* supply of bandwidth U (consisting of the bandwidth supply from servers U_s and peers $\sum u_i$) to the demand for bandwidth D in the system, while h takes a particular focus on the *residual* (surplus) bandwidth from peers in addition to satisfying D. Suppose the current number of participating peers is $S(t)$, $D = S(t)R$ and $U = \sum u_i + U_s = S(t)R + (\sum u_i - S(t)R) + U_s$. Then, the Resource Index can be formally described as

$$\frac{U}{D} = \frac{\sum u_i + U_s}{S(t)R} = \frac{S(t)R + (\sum u_i - S(t)R) + U_s}{S(t)R} = 1 + \frac{u - R}{R} + \frac{U_s}{S(t)R} = 1 + \frac{h}{x} + \frac{U_s}{S(t)R},$$

where $R = xr$ and $h = (u - R)/r$ as defined previously. This clearly shows that the higher (lower) the relative average surplus capacity h is, the higher (lower) the Resource Index is, indicating a less (more) constrained system capacity.

To capture essential aspects of practical systems, yet be still simple enough for analytical tractability, our model mainly considers the following aspects:

First, initial system capacity. We assume initially there are M existing peers that have already joined the system. That is, they have obtained sufficient upload bandwidth resources to satisfy the streaming rate, and are able to contribute their upload capacities to the system. We assume that there exists one or multiple servers in the system with aggregate upload capacity U_s. Given a streaming rate R, the relative server capacity u_s is defined as the ratio of U_s/R.

Second, flash crowd. We start by focusing on an extreme flash crowd scenario where $N(\gg M)$ peers arrive at approximately the same time [11], just after a new live event has been released. Each new peer that has yet to join the system needs to gather at least x units of upload bandwidth resource from the existing peers to meet the streaming rate requirement. Our model strives to capture the difficulty for peers to gather sufficient upload bandwidth resources at startup, which we believe is the most critical problem in a flash crowd. Furthermore, we also extend our model to more general and realistic peer arrival patterns [15], [21], [22] during a flash crowd, in order to make our analysis more representative of real-world systems.

Third, system scale and initial startup delays. Without loss of generality, we assume that time t is slotted. If a *new peer*—one that has not yet joined the system—has obtained sufficient upload bandwidth resource (i.e., x units) at the tth time slot, it is regarded as "joined the system" and counted toward the system scale $S(t)$ of existing peers. In other words, during $[0, t]$, the number of peers that have obtained sufficient upload bandwidth resource to satisfy the streaming rate are regarded as the system scale $S(t)$. Otherwise, the peer will continue to seek upload bandwidth resources in subsequent time slots, until it joins the system. From the perspective of user experience, the time t in our model represents the initial startup delays for peers. In addition, motivated by our recent empirical study [4] that peers in a flash crowd may opt to leave the system due to excessive startup delays, we further refine our model by taking into account *peer impatience* and its impact on the system scale over time.

Finally, we consider the case that each peer has global knowledge on the content availability and there is a central control in the system. This leads to an upper bound of the system scale over time. We then proceed to demonstrate the effects of partial or incomplete knowledge, by assuming a *mesh-based* system using a simple random partner selection strategy. This is different from a tree-based system enforcing a preconstructed topology, in which each node retrieves the video content from its predetermined parent. Specifically, each new peer randomly selects k partners from the current set of existing peers to ask for their surplus upload capacities in each time slot.[1] Since an existing peer can be requested by more than one new peer, it may supply its upload bandwidth resources to satisfy subset of such requests, depending on its current surplus capacity. Such a random partner selection strategy with parameter k essentially represents decentralized gossiping among peers to gather upload bandwidth resource. This

1. Essentially, a time slot in our model represents the time that a new peer takes to contact k partners to locate available bandwidth resource and content, so that the streaming rate is satisfied.

strategy is typically adopted in practical *mesh-based* P2P live streaming systems such as Coolstreaming [3], mainly due to its simplicity.

Differing from the perspective of chunk dissemination that takes peer streaming buffer states and chunk scheduling as their main consideration (e.g., [8], [9], [12]), in this paper, we focus on the asymptotic scaling behavior of the entire system and the effects of various influential factors, rather than individual peer behavior. Given that chunk scheduling policies [12] have been extensively studied and optimal schemes have been proposed [8], [9], [10], [11], our model assumes an existing peer who accepts to serve another new peer always has chunks to provide, i.e., content availability is not the bottleneck. Nevertheless, this assumption will be relaxed by our simulation in Section 4.3, so as to obtain complementary insights into the impact of content availability on system scaling.

3.2 Scale-Time Relationship with Critical Factors

We first derive the fundamental constraint of the scale-time relationship in a P2P live streaming system, with global knowledge and centralized control: *While "the average peer uploading capacity should be no less than the average peer downloading rates" is a necessary condition for P2P live streaming systems to scale, it is insufficient to capture the system scale, as the upload bandwidth resource from newly arrived peers cannot be utilized immediately*. This leads to the following upper bound of system scale over time.

Theorem 1. *For a P2P live streaming system with a given streaming rate R and average peer upload capacity u, the system scale after the tth time slot, $S(t)$, has the following upper bound:*

$$S(t) \leq \min\left\{\left(\frac{u}{R}\right)^t (M+C) - C, N+M\right\}, \tag{1}$$

where $C = U_s/(u-R)$, M is the initial system scale at time $t=0$, U_s is the total server capacity, and N is a flash crowd or the number of newly arrived peers.

Remark. Interested readers are referred to Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2011.283, for a complete proof. It is interesting to observe that the upper bound neither depends on specific flash crowd arrival patterns, nor the bandwidth unit. However, it intuitively would be too optimistic as it assumes all current surplus bandwidth resources from existing peers can be fully utilized. *Since the system scale is further constrained by the partial knowledge of peers and their competition for limited resources, how can we characterize and quantify such effects?* To this end, we proceed to analyze the scale-time relationship with a random partner selection strategy as follows:

Since it has already been proved in [7], [11] that the average peer upload capacity u satisfies $u > R$ in large-scale live streaming systems, we first focus on the general homogeneous case where $u_i = u > R$(i.e., $h_i = h > 0$) for all peers. This is reasonable as we are more interested in the asymptotic collective behavior of the system rather than the individual peer behavior. As we focus on such a homogeneous case, we first ignore the server capacity, and will introduce it as a parameter later. Heterogeneous cases are further investigated by simulations in Section 4.3.

Lemma 1. *For a P2P live streaming system with each peer having partial knowledge of the system and a random partner selection strategy (i.e., each new peer independently and randomly selects k partners from the set of existing peers), the number of new partners of an existing peer during the tth time slot, $q(t,k)$, is a random variable that follows a binomial distribution with parameters $(N+M-S(t-1), k/S(t-1))$, and an expected value of*

$$E[q(t,k)] = \frac{k(N+M-S(t-1))}{S(t-1)}, \tag{2}$$

where $S(t-1)$ is the current number of existing peers in the system.

The proof of Lemma 1 can be found in Appendix B, available in the online supplemental material. Based on Lemma 1, we can derive an approximation of the expected system scale as follows:

Theorem 2. *For a P2P live streaming system with each peer having partial knowledge of the system and a random partner selection strategy, assume that each existing peer could randomly provide each of its new partner with 1 unit of upload bandwidth resource with a probability of $h/q(t,k)$. If we use the expected value $E[q(t,k)]$ given by (2) as an approximation of $q(t,k)$, then the expected system scale after the tth time slot, $E[S(t)]$, can be approximated by*

$$\begin{aligned} E[S(t)] \approx S(t-1) + (N+M-S(t-1)) \\ \times \sum_{i=x}^{k} C_k^i p(t,k,h)^i (1-p(t,k,h))^{k-i}, \end{aligned} \tag{3}$$

where $p(t,k,h) \approx h\alpha(t)/k$ is the probability for a new peer to obtain 1 unit of upload bandwidth resource from an existing peer; and $\alpha(t) = S(t-1)/(N+M-S(t-1))$ is the ratio of the number of existing peers to the number of new peers in the system at the beginning of the tth time slot.

The proof of Theorem 2 is given in Appendix C, available in the online supplemental material, and its accuracy is verified by simulations in Section 4.3. Theorem 2 with (3) qualitatively indicates that, $p(t,k,h)$ plays an important role for the system scale, which depends on $\alpha(t)$, h, and k. The effects of these factors will be thoroughly examined in Section 4.

Furthermore, as demonstrated by both the real-world experience [4] and the numerical results (Section 4) derived from our model, P2P live streaming systems by nature do not react well to a flash crowd. Specifically, the system scale grows relatively slower during the initial time slots. This motivates a natural question: *How can a certain amount of server capacity provisioning help improve the system scale?* Based on Theorem 2, we can approximately derive the improved system scale with a given amount of server capacity provisioning as follows:

Corollary 1. *For a P2P live streaming system with a streaming rate of R and an aggregate server upload capacity U_s, assume*

that server(s) support a number of $u_s = U_s/R$ randomly selected new peers at the beginning of each time slot. The remaining $N + M - S(t-1) - u_s$ new peers still rely on the $S(t-1)$ existing peers through a random partner selection strategy. Then, the expected system scale $E[S(t)]$ given by Theorem 2 can be potentially improved as

$$E[S(t)] \approx S(t-1) + u_s + (N + M - S(t-1) - u_s) \times \sum_{i=x}^{k} C_k^i p'(t,k,h,u_s)^i (1 - p'(t,k,h,u_s))^{k-i}, \quad (4)$$

where $p'(t,k,h,u_s) = h\alpha'(t,u_s)/k$, $\alpha'(t,u_s) = S(t-1)/(N + M - S(t-1) - u_s)$, and $u_s = U_s/R$ is the relative server capacity.

The proof of Corollary 1 is similar to the proof of Theorem 2. The effects of the parameter u_s will be demonstrated through both analytical and simulation results in Section 4.

3.3 Effects of Peer Arrival Patterns and Peer Departures

In addition to the extreme flash crowd scenario, our model can easily be extended to more general peer arrival patterns as follows:

Corollary 2. *For a P2P live streaming system with each peer having partial knowledge of the system and a random partner selection strategy as assumed in Theorem 2, then given any specific peer arrival pattern $\lambda(t)$, the expected system scale $E[S(t)]$ given by Theorem 2 can be extended as*

$$E[S(t)] \approx S(t-1) + \left(\int_0^t \lambda(\tau) d\tau + M - S(t-1) \right) \times \sum_{i=x}^{k} C_k^i p(t,k,h)^i (1 - p(t,k,h))^{k-i}, \quad (5)$$

where $p(t,k,h) \approx h\alpha(t)/k$ is the probability for a new peer to obtain 1 unit of upload bandwidth resource from an existing peer; and $\alpha(t) = S(t-1)/(\int_0^t \lambda(\tau)d\tau + M - S(t-1))$ is the ratio of the number of existing peers to the number of new peers in the system at the beginning of the tth time slot.

The proof of Corollary 2 is similar to the proof of Theorem 2. We will quantitatively examine the scale-time relationship in P2P live streaming systems under both the extreme flash crowd scenario and another two typical peer arrival patterns in Section 4 plus Appendix E, available in the online supplemental material.

It was observed in [4] that a considerable portion of users undergoing a flash crowd could opt to leave the system due to excessive startup delays. To capture this behavior and its impact on the system scale, we define a random variable θ, as a *peer impatience time threshold* specifying the time after which a new peer aborts its joining attempt and leaves the system.[2] Let $E[\theta]$ be the expected value; then, the scale-time relationship given in Corollary 2 can be further extended as follows:

2. Note that the retrying peers [4] can be viewed as part of the new peers in our model, who try to join the streaming system in subsequent time slots.

Corollary 3. *For a P2P live streaming system with each peer having partial knowledge of the system and a random partner selection strategy as assumed in Theorem 2, if we use an expected threshold $E[\theta]$ as an approximation of impatience time for all the new peers, after waiting for which new peers would leave the system; then, given any specific peer arrival pattern $\lambda(t)$, the expected system scale $E[S(t)]$ given by Corollary 2 can be extended as*

$$E[S(t)] \approx S(t-1) + \left(\int_0^t \lambda(\tau) d\tau + M - S(t-1) - \int_0^{t-1} D(\tau) d\tau \right) \sum_{i=x}^{k} C_k^i p'(t,k,h,\theta)^i (1 - p'(t,k,h,\theta))^{k-i}, \quad (6)$$

where $D(\tau) \approx \lambda(\tau - E[\theta])l(\tau)$ for $\tau \geq E[\theta]$ (otherwise, $D(\tau) = 0$) is the number of new peers that leave the system over time slots due to impatience, $l(\tau) = 1 - ((S(\tau) - S(\tau - E[\theta]))/(\int_0^\tau \lambda(t)dt + M - S(\tau - E[\theta]) - \int_0^{\tau - E[\theta]} D(t)dt))$ is a probability for new peers that arrived at the $(\tau - E[\theta] + 1)$th time slot yet still have not obtained sufficient upload bandwidth resources for startup at the end of τth time slot. $p'(t,k,h,\theta) \approx h\alpha'(t,\theta)/k$, and

$$\alpha'(t,\theta) = S(t-1) \Big/ \left(\int_0^t \lambda(\tau) d\tau + M - S(t-1) - \int_0^{t-1} D(\tau) d\tau \right).$$

Remark. Interested readers are referred to Appendix D, available in the online supplemental material, for a complete proof. Corollary 3 with (6) qualitatively indicates that the departures of impatient peers during a flash crowd could indirectly alleviate the heavy competition among new peers for the limited pool of upload bandwidth resources, thus the system could scale up more quickly in a transient period. This, however, is at the expense of the reduced system scale in steady state, as demonstrated in Section 4 quantitatively.

4 Results and Discussions

In this section, we present both analytical and simulation results to demonstrate the fundamental scale-time relationship in P2P live streaming systems under a flash crowd, as well as the effects of various critical factors.

4.1 Scale-Time Relationship and Join Time Distribution

Fig. 1 compares the approximated system scale over time obtained by Theorem 1, 2 and Corollary 1, under the same flash crowd scenario setting. We observe that:

First, the system scale grows relatively slower during the initial time, as a surge of newly arrived peers compete for the limited surplus capacities from a relatively smaller number of existing peers. This results in considerable difficulty for new peers to obtain sufficient upload bandwidth resources.

Second, as more peers gradually join the system with positive gain of surplus capacities, the ratio of the number

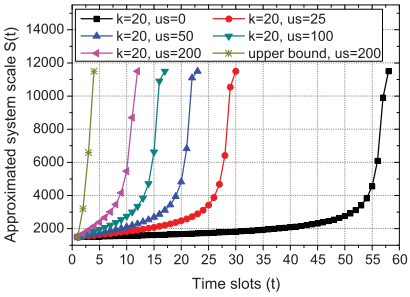


Fig. 1. Approximated system scale over time slots, with different amount of server capacities. We set the initial system scale M to 1,500 and flash crowd scale N to 10,000. The number of partners for new peers k is set to a typical value of 20. The relative server capacity u_s varies from 0 to 200. Others are set as $h = x = 5$.

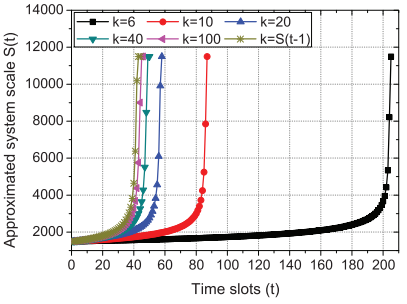


Fig. 3. Approximated system scale over time slots, with different settings of the number of partners for new peers k. We set the initial system scale M to 1,500 and flash crowd scale N to 10,000. The value of k varies from 6 to $S(t-1)$. Others are set as $u_s = 0, h = x = 5$.

of existing peers to the number of new peers $\alpha(t)$ continuously increases and the entire system capacity improves; thus the system scale ramps up more quickly.

Third, as expected, the system scale can be substantially improved with an additional amount of server capacity provisioned, especially during the initial time. However, we note that the improvement slows down with more server capacity provisioned, as demonstrated by the decreasing gaps between the curves.

We plot the peer join time distribution (i.e., the percentage of peers that joined the system in each time slot) in Fig. 2, which clearly illustrates that significant portion of peers could suffer from long startup delays during a flash crowd; and not surprisingly, the extra server(s) can help reduce the startup delays.

4.2 Sensitivity Analysis on Critical Factors

We next examine the effects of several critical factors indicated by Theorem 2.

First, Fig. 3 compares the approximated system scale over time slots, by varying the number of partners for new peers k. We observe that the system scale improves significantly as k increases in the range of typical settings that real-world systems use [3]. Equivalently, this implies that the time to accommodate a given scale of a flash crowd decreases substantially, as each new peer can explore with, and obtain video content from more partners. However, the improvement becomes marginal with the further increase of the k value, especially when the value of k is close to the size of current set of existing peers $S(t-1)$.

Second, we examine the effects of k by evaluating the time to accommodate different scales of a flash crowd, as shown in Fig. 4. We observe that: 1) When the flash crowd is less severe with respect to the initial system capacity (i.e., the demand to supply ratio of $(Nx)/(Mh)$ is relatively less stringent), results are relatively insensitive to different values of k. Specifically, the increase of k actually does not help (e.g., when the flash crowd scale $N = 4{,}000$, the time to accommodate it under different values of k stays nearly the same). This is because that a given initial system capacity can more easily accommodate a smaller degree of flash crowd, and there is far less competition for upload bandwidth resources in the system. 2) As the scale of the flash crowd increases, our results become more sensitive to different values of k, and there are remarkable improvements by increasing k. However, excessive increase of k brings relatively minor improvements, which is consistent with previous observation from Fig. 3.

Third, we examine the impact from the relative average peer surplus capacity h, the initial system scale M, and their correlation with k. Figs. 5 and 6 plot the time to accommodate a given scale of a flash crowd when h or M varies, respectively, under different settings of k. The increase of h or M can effectively reduce the time to accommodate flash crowd, as it improves the entire system capacity. In general, the more upload bandwidth resources exist in the system (though it takes time to utilize them), the less time it takes to accommodate a flash crowd. When the

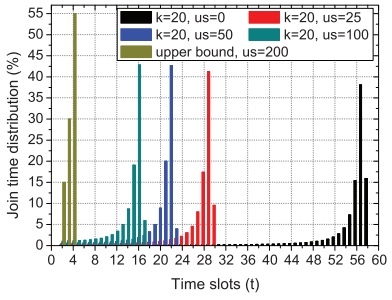


Fig. 2. Peer join time distribution versus time slots, with different amount of server capacities. We set the initial system scale M to 1,500 and flash crowd scale N to 10,000. The number of partners for new peers k is set to a typical value of 20. The relative server capacity u_s varies from 0 to 200. Others are set as $h = x = 5$.

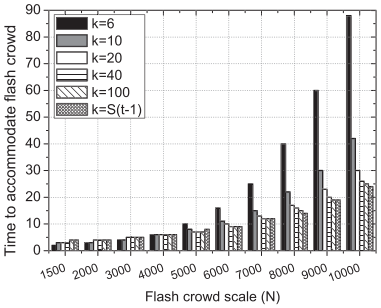


Fig. 4. Time to accommodate different scales of a flash crowd, under different settings of the number of partners for new peers k. We set the initial system scale M to 1,500. The value of k varies from 6 to $S(t-1)$. Others are set as $u_s = 0, h = 6, x = 5$.

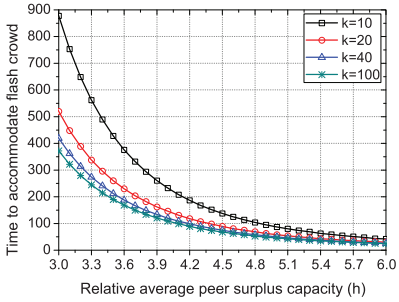


Fig. 5. Time to accommodate a flash crowd of $N = 10{,}000$ peers when relative average peer surplus capacity h varies, under different settings of the number of partners for new peers k. We set the initial system scale M to 1,500. The value of k varies from 10 to 100. Others are set as $u_s = 0, x = 5$.

upload bandwidth resource is relatively constrained (i.e., when h or M decreases), the performance gaps (in terms of time saved) between different settings of k are more profound, which is consistent with the observation in Fig. 4.

In addition to the extreme flash crowd scenario, we further utilize Corollary 2 to examine the scale-time relationship in P2P live streaming systems under other typical peer arrival patterns. Fig. 7 plots the approximated system scale over time slots under a constant peer arrival rate $\lambda(t) = \lambda$. To make the result comparable to that of the extreme flash crowd scenario, the total number of new peers that arrive at the system (i.e., the flash crowd scale) is limited to the same as in Fig. 1, i.e., $\lambda\tau = N$, where N peers arrive at the system in a period of τ with a constant rate λ. We can see that our previous discussion on the scale-time relationship still holds for such a constant peer arrival pattern. More importantly, given a specific scale of flash crowd, the system scale under a constant peer arrival pattern grows more quickly compared to the extreme (pessimistic) flash crowd case (marked as $N = 10{,}000$). As the peer arrival rate λ decreases, the initial startup delays decrease remarkably. Specifically, when the peer arrival rate is close to the initial system scale (e.g., $\lambda = 2{,}000$ which is close to $M = 1{,}500$), the system is able to accommodate all the new peers shortly after they arrived. Additional observations on a more realistic pattern called exponentially decreasing peer arrival pattern [15], [21], [22] can be found in Appendix E, available in the online supplemental material.

Finally, our recent measurement study [4] shows that a considerable portion of peers undergoing a flash crowd

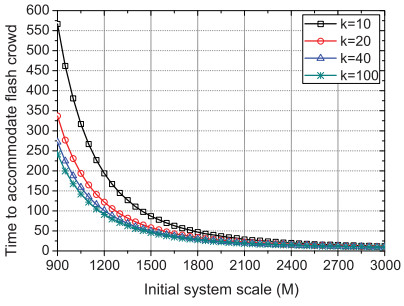


Fig. 6. Time to accommodate a flash crowd of $N = 10{,}000$ peers when the initial system scale M varies, under different settings of the number of partners for new peers k. The value of k varies from 10 to 100. Others are set as $u_s = 0, h = x = 5$.

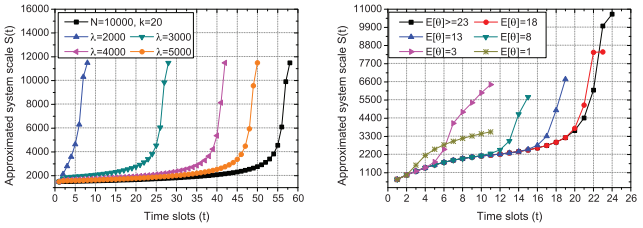


Fig. 7. Effects of peer arrival patterns and peer departures: (a) approximated system scale over time slots, under a constant peer arrival pattern $\lambda(t) = \lambda$. We set the initial system scale M to 1,500 and flash crowd scale N to 10,000, with the peer arrival rate λ varying from 2,000 to 5,000. (b) approximated system scale over time slots, under a constant peer arrival pattern $\lambda(t) = \lambda$ and different values of the expected peer impatience time threshold $E[\theta]$. We set the initial system scale M to 700 and flash crowd scale N to 10,000, with $\lambda = 1{,}000$ and $E[\theta]$ varying from 23 to 1 time slots. The number of partners for new peers k is set to a typical value of 20. Others are set as $u_s = 0, h = x = 5$.

could opt to leave the system due to impatience. Based on Corollary 3, Fig. 7 compares the approximated system scale over time slots with different values of the expected peer impatience time threshold $E[\theta]$, under a constant but relatively high peer arrival rate (with respect to the initial system scale). We observe that: 1) As the expected peer impatience time threshold $E[\theta]$ decreases, the system could scale up more quickly during the flash crowd with relatively more peers, especially later arrivals, experiencing shorter startup delays. For instance, the approximated system scale with $E[\theta] = 3$ quickly reaches a high level (6,600) at the 11th time slot, which is nearly 3 times of the system scale (2,200) with larger values of $E[\theta]$ (8, 13, 18 and >23, respectively). This implies that the departures of impatient peers during a flash crowd could alleviate the heavy competition for the limited pool of upload bandwidth resources, as reflected by $\alpha'(t, \theta)$ and $p'(t, k, h, \theta)$ in Corollary 3. 2) This is achieved at the cost of a reduced system scale in the steady-state due to the departures of impatient peers. For example, after all the new peers during the flash crowd either left or joined the system, the final system scale with smaller values of $E[\theta]$ is only around half (e.g., $E[\theta] = 3, 8, 13$) or even a quarter (e.g., $E[\theta] = 1$) of that with loose impatience thresholds (e.g., $E[\theta] \geq 23$).

4.3 Simulation Results

We have carried out simulations to validate the model and analytical results, under *heterogeneous* peer upload capacity and *asymmetric* load distribution among peers. The input to the simulator includes a set of existing peers in the initial system and a flash crowd of newly arriving peers yet to join the streaming, both of which consist of heterogeneous peers. Without loss of generality, we classify each peer as either a low-capacity one with $u_l < R$, or a moderate-capacity one with $u_m = R$, or a high-capacity one with $u_h > R$. Specifically, in our experiment we randomly choose 20 percent of peers with $u_h = 35$, 30 percent of peers with $u_m = R = 5$, and another 50 percent of peers with $u_l = 3$ to emulate the observed environments [3]. It is easy to check that the relative average peer surplus capacity is $h = 5$. These rates could be in units of 100 Kbps [7], so that high-capacity peers can represent enterprise/campus users with

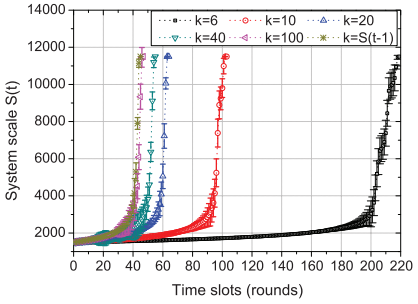


Fig. 8. Simulated system scale over time with 95 percent-level confidence intervals, under heterogeneous peer upload capacity and different settings of the number of partners for new peers k. We set the initial system scale M to 1,500 and flash crowd scale N to 10,000. The value of k varies from 6 to $S(t-1)$. Others are set as $u_s = 1, h = 5$.

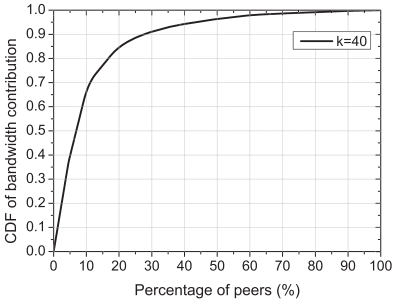


Fig. 9. CDF of peers' upload bandwidth contributions during the flash crowd, under simulation with heterogeneous peer upload capacity. We set the initial system scale M to 1,500, the flash crowd scale N to 10,000, and the number of partners for new peers $k = 40$. Others are set as $u_s = 1, h = 5$.

Ethernet access rates in excess of a few Mbps, while moderate/low-capacity ones represent broadband residential users (with DSL and cable access) typically with upload rates at 500 Kbps or less. Accordingly, the streaming content can be empirically presented as a series of fixed-length chunks (marked by sequence numbers) of size 10 Kb [3].

The simulator has a tracking server for peer registration and bootstrapping, which connects each peer in the initial system to k ($\in [6, 40]$ in accordance to practical systems [3]) other *randomly* chosen peers, so as to form the initial mesh overlay. The simulator runs in rounds during the flash crowd. In each round, peers independently perform partner selection using the random peering strategy described in Section 3.1. With incomplete knowledge on available content and bandwidth resources, neighboring peers then demand from and supply to one another in attempting to sustain the streaming. In particular, we relax the model assumption on *content availability* by introducing a simple yet representative *pull-based random* chunk scheduling policy [23] in the simulator, which is suggested to be near-optimal in terms of a peer upload capacity utilization and streaming performance [11], [23]. Specifically, each peer can buffer up to W of the aforementioned chunks (e.g., we set $W = 100$ in the simulation), and periodically exchange chunk availability information of the sliding streaming buffers (commonly referred to as buffer maps[3]) with its neighboring peers per round. Based on such information, each peer explicitly requests absent chunks in its current buffer from neighbors, starting from those chunks with less number of supplying neighbors. Among multiple neighbors holding the same absent chunk, the chunk will be assigned to one of the neighbors *randomly with the same probability* [23]. A peer obtaining sufficient upload bandwidth resource and half-loaded buffer is counted toward the system scale [3], [4]. Other parameters will be specified as part of the following experiments.

Fig. 8 shows the system scale evolution with 95 percent-level confidence intervals obtained by simulations, under heterogeneous peer upload capacity and different settings of the number of partners for new peers k. The parameters are set to be the same as in Fig. 3. We can see the experimental results of Fig. 8 match the analytical results of Fig. 3 in a wide range of settings of k, except that the respective ramp-up process under simulation tends to become slightly longer in Fig. 8. This demonstrates the *performance gap* accounted for by the relaxation of assumptions on content availability, homogeneous peer upload capacity, and homogeneous load distribution (see Fig. 9). Nevertheless, such an impact fades as the number of partners for peers increases from $k \leq 10$ to $k \geq 20$ (i.e., peers can explore with, and obtain absent chunks from more neighbors). This corroborates the validity of our theoretical model and the accuracy of associated approximations in characterizing the *asymptotic* scaling behavior of the system.

By capturing the distribution of upload bandwidth contributions from all peers during the flash crowd in Fig. 9, we observe highly uneven contributions from peers in the system. Specifically, 20 percent of peers, most of which are high-capacity ones, contribute more than 80 percent of the amount of upload bandwidth in combating with the flash crowd. This is consistent with the empirical observations in realistic P2P live streaming systems [24]. More importantly, even for such asymmetric load distribution among peers, simulation results in Fig. 8 are quite close to the theoretical analysis in Fig. 3.

We also compare the evolution of small-scale systems by varying M in Fig. 10, under both theoretical calculation and simulations with 95 percent-level confidence intervals. We start from an empty peer population size $M = 0$, with relative server upload capacity $u_s = 5$ as bootstrap. We can see that with smaller initial system scales ($M = 0, 100, 200$)

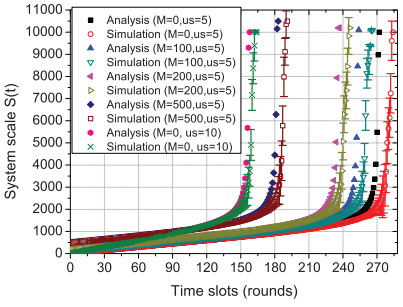


Fig. 10. Performance comparison of small-scale systems under theoretical analysis against simulations with 95 percent-level confidence intervals, where the initial system scale M varies from 0 to 500. We set the flash crowd scale N to 10,000, the number of partners for new peers $k = 20$, and the relative server upload capacity $u_s = 5$ or 10.

3. Similar to [3], we ignore the signaling overhead caused by the exchange of the buffer map since only one-bit is required for each chunk in the buffer map.

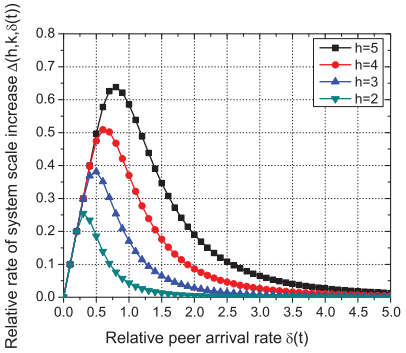


Fig. 11. The relative rate of system scale increase $\Delta(h,k,\delta(t))$ as a function of the relative peer arrival rate $\delta(t)$, under different settings of the relative average peer surplus capacity h. The number of partners for new peers k is set to a typical value of 20. Others are set as $x=5, u_s=0$.

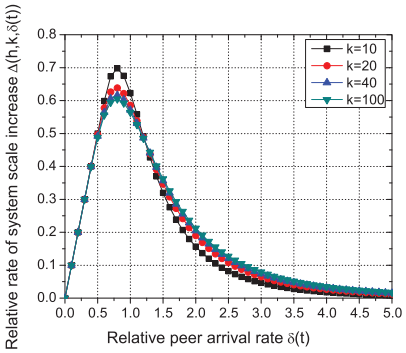


Fig. 12. The relative rate of a system scale increase $\Delta(h,k,\delta(t))$ as a function of the relative peer arrival rate $\delta(t)$, under different settings of the number of partners for new peers k. Others are set as $h=x=5, u_s=0$.

relative to the flash crowd scale ($N=10{,}000$), the system ramps up much slower in both theory and experiments. In particular, the impact of content availability and chunk scheduling on small-scale systems is more evident under flash crowd, as shown by the performance gap between analytical and simulation results. Not surprisingly, a larger population size ($M=500$) or a further increase of the server capacity ($u_s=10$) can help accelerate the initial ramp up process, and then simulation results asymptotically conform to analytical results. This implies that simple pull-based chunk scheduling policies only lead to a small degree of performance impact when the system reaches a reasonable scale with sufficient server bandwidth, as consisted with both the theory [11] and realistic P2P streaming systems [23].

5 Population Control for Alleviating Flash Crowds

The design implication from our previous analysis of the scale-time relationship is that we can *trade peer startup delays in the initial stage of a flash crowd to achieve better system scale*. To this end, we proceed to explore the design and potential benefits of *population control* for alleviating flash crowds.

5.1 Inherent Relationship between Rate of System Scale Increase and Peer Arrivals

We first identify the critical criteria for population control decisions, by analyzing an inherent relationship between the rate of system scale increase and the rate of peer arrivals in P2P live streaming systems.

Corollary 4. *For a P2P live streaming system with a given relative average peer surplus capacity h, and each peer having partial knowledge of the system and a random partner selection strategy as assumed in Theorem 2, then given any specific peer arrival pattern $\lambda(t)$, there exists an inherent relationship between the* relative rate of system scale increase $\Delta(h,k,\delta(t))$ *and* relative peer arrival rate $\delta(t)$, *as characterized by the following function:*

$$\Delta(h,k,\delta(t))=\delta(t)\sum_{i=x}^{k}C_k^i\left(\frac{h}{k\delta(t)}\right)^i\left(1-\frac{h}{k\delta(t)}\right)^{k-i},\qquad(7)$$

where the relative rate of system scale increase is defined as $\Delta(h,k,\delta(t))\triangleq(E[S(t)]-S(t-1))/S(t-1)$, and the relative peer arrival rate is defined as $\delta(t)\triangleq 1/\alpha(t)=(\int_0^t\lambda(\tau)\,\mathrm{d}\,\tau+M-S(t-1))/S(t-1)$, i.e., the ratio of the number of new peers versus that of existing peers in the system at the beginning of the tth time slot.

Remark. The proof of Corollary 4 is given in Appendix F, available in the online supplemental material. Corollary 4 with (7) essentially describes *how fast the P2P live streaming system with a certain capacity can scale up, in response to a certain peer arrival intensity imposed by a certain peer arrival pattern.* In particular, by viewing the rate of system scale increase $\Delta(h,k,\delta(t))$ as a function of the parameters h, k, and $\delta(t)$, the relationship in (7) is general in any P2P live streaming system. This is because both $\Delta(h,k,\delta(t))$ and $\delta(t)$ are abstracted as relative ratios that are independent of specific system population sizes and specific peer arrival patterns. As we will elaborate in Section 5.2, this relationship can serve as the theoretical principle of population control design for alleviating flash crowds.

To clearly illustrate the relationship in Corollary 4 and its implication on population control design, Figs. 11 and 12 depict the relative rate of system scale increase $\Delta(h,k,\delta(t))$ as a function of the relative peer arrival rate $\delta(t)$, under different settings of the relative average peer surplus capacity h and the number of partners for new peers k, respectively.

We find that in general, *as the relative peer arrival rate increases (i.e., more new peers[4] relative to the current system scale request to join the system), the relative rate of a system scale increase does not grow monotonously, regardless of the setting of the relative average peer surplus capacity h and number of partners for new peers k.* In particular, given a certain system capacity represented by h and k, there exists a peak relative rate of system scale increase $\overline{\Delta(h,k)}$ when the relative peer arrival rate reaches a critical *resiliency threshold* $\overline{\delta(h,k)}$. Beyond this threshold, the rate of a system scale increase will fall significantly. This clearly demonstrates the motivation and potential benefits

4. Without loss of generality, those new peers at the beginning of the tth time slot include both newly arrived ones at the beginning of the tth time slot and others that arrived in previous time slots but not yet joined the system.

of population control: during a flash crowd wherein the peer arrival rate exceeds the resiliency threshold, we can adaptively admit (or delay) a proper portion of new peers from joining the system, so as to attain the peak rate of a system scale increase. Otherwise, the rate of a system scale increase could deteriorate quickly due to a heavy resource competition, leading to a poor system scalability and excessive peer startup delays. Henceforth, we refer to $(\overline{\delta(h,k)}, \overline{\Delta(h,k)})$ as the *optimal control point* under given (h,k), which is the criteria for making population control decisions.

Also, benefiting from the generality of Corollary 4, the optimal control point is essentially an abstract and general criteria (i.e., in the form of relative ratios) that is independent of specific system population sizes and peer arrival patterns (although the specific maximum number of new peers that could join the system over the tth time slot apparently depends on the existing number of peers in previous time slots, i.e., $S(t-1) \times \overline{\Delta(h,k)}$). In practice, given a typical range of h and k based on specific system design choices and empirical measurements, we can precompute a set of optimal control points for each typical setting of (h,k), even though it is hard to directly obtain closed-form expressions of $(\overline{\delta(h,k)}, \overline{\Delta(h,k)})$ by maximizing (7). This is useful for simplifying the population control implementation with reduced computational overhead, as we will further discuss in Section 5.3.

In addition, we have observed that the optimal control point is very sensitive to the relative average peer surplus capacity h (i.e., both $\overline{\Delta(h,k)}$ and $\overline{\delta(h,k)}$ increase as h increases), while relatively insensitive to the number of partners for new peers k. We will take a closer look at such an effect when we examine the robustness and sensitivity of population control in Appendix H, available in the online supplemental material.

5.2 Population Control: An Ideal Case

Based on the above analysis, the natural question is: *what is the ideal case of population control, and what is its fundamental limit in improving the system scale?*

By assuming accurate knowledge on the number of existing peers in the system $S(t-1)$, the relative average peer surplus capacity h and the number of partners for new peers k to be available, we can design an ideal population control procedure in Algorithm 1. Specifically, based on the optimal control point $(\overline{\delta(h,k)}, \overline{\Delta(h,k)})$ illustrated in Figs. 11 and 12, the intuition of population control is very simple: as long as the number of new peers relative to the current system scale does not exceed $\overline{\delta(h,k)}$, all new peers can be admitted. Otherwise, population control will attempt to attain the peak rate of system scale increase $\overline{\Delta(h,k)}$ by admitting a portion of new peers according to $\overline{\delta(h,k)}$ (alternatively, by delaying a counterpart portion of new peers from joining the system for certain time slots). As any advantage may come with tradeoffs, an obvious cost of population control is a portion of delayed peers during the flash crowd. We will demonstrate in Section 5.3 that, this is a worthy choice for improving the overall system scalability which eventually compensates for such delays.

Algorithm 1. An ideal population control procedure with accurate knowledge on the number of existing peers in the system, the relative average peer surplus capacity h, and the number of partners for new peers k.

Definition: $U(t) \triangleq$ the number of new peers that have not yet joined the system at the end of the t-th time slot. $(\overline{\delta(h,k)}, \overline{\Delta(h,k)}) \triangleq$ the optimal control point based on Corollary 4.
Initialization: $S(0) \leftarrow M,\ U(0) \leftarrow 0$
while $S(t) < S(0) + \int_0^t \lambda(\tau)\mathrm{d}\tau$ **do**
  if $\lambda(t) + U(t-1) \le S(t-1)\overline{\delta(h,k)}$ **then**
    all the new peers can be admitted, thus
    $\delta(t) \leftarrow [U(t-1) + \lambda(t)]/S(t-1)$
    $S(t) \leftarrow [1 + \Delta(h,k,\delta(t))] \times S(t-1)$
    $U(t) \leftarrow U(t-1) + S(t-1) + \lambda(t) - S(t)$
  else
    admit a portion of new peers according to $\overline{\delta(h,k)}$ (alternatively, by delaying a corresponding portion of new peers from joining the system for one time slot), thus
    $S(t) \leftarrow (1 + \overline{\Delta(h,k)}) \times S(t-1)$
    $U(t) \leftarrow (1 + \overline{\delta(h,k)}) \times S(t-1) - S(t)$
  end if
end while

Theorem 3. *For a P2P live streaming system with a given relative average peer surplus capacity h, and each peer having partial knowledge of the system and a random partner selection strategy as assumed in Theorem 2, then given any specific peer arrival pattern $\lambda(t)$ during a flash crowd, the system can potentially scale over time* exponentially *under ideal population control in Algorithm 1.*

Remark. The proof can be found in Appendix G, available in the online supplemental material. Theorem 3 indicates the fundamental limit of population control for alleviating flash crowds. In particular, with ideal population control in Algorithm 1, the system can potentially scale over time in an exponential manner, even if a partial knowledge of peers and their competition for the limited upload bandwidth resources are taken into account. This will be quantitatively demonstrated in Section 5.3.

Such an ideal population control procedure in Algorithm 1 with its limit in Theorem 3 can provide a benchmark for designing practical population control algorithms. However, it essentially tracks and regulates the system evolution by relying on accurate knowledge on $S(t-1)$, h, and k, which is infeasible to be achieved in practical systems. Interested readers are referred to Appendix H, available in the online supplemental material, for investigating *how robust and sensitive population control could be, with respect to h, k, and $S(t-1)$, in order to explore if it is feasible to design simple and practical population control mechanisms with approximated knowledge.*

5.3 Population Control: Simple Framework and Guidelines

Based on insights from Algorithm 1 in Section 5.2 and Theorem 4 in Appendix H, available in the online

supplemental material, we propose a simple population control framework in Algorithm 2. Rather than dictating specific implementation of population control that highly depends on and varies across different P2P live streaming systems, we focus on a common and flexible framework with key guidelines. Compared to the ideal population control procedure in Algorithm 1, the salient features of the framework in Algorithm 2 in terms of simplicity, flexibility, and practicability include:

Algorithm 2. A simple and flexible framework for alleviating flash crowds in P2P live streaming systems, based on key principles from Algorithm 1 and Theorem 4 in Appendix H, available in the online supplemental material.

Definition: $I(t) \triangleq$ the number of distinct peers issuing request to join the system at the beginning of the t-th time slot. $(\overline{\delta(h,k)}, \overline{\Delta(h,k)}) \triangleq$ the optimal control point that can be pre-computed according to typical (h,k), as discussed later.
Initialization: $S'(0) \leftarrow \varepsilon M$, where $\varepsilon \in (0,1]$ is a tunable parameter for making conservative estimate of the initial system scale.
if $I(t) \leq S'(t-1)\overline{\delta(h,k)}$ **then**
  all the new peers can be admitted, thus
  $\delta'(t) \leftarrow I(t)/S'(t-1)$
  $S'(t) \leftarrow [1 + \Delta(h,k,\delta'(t))] \times S'(t-1)$
else
  admit a number of $S'(t-1)\overline{\delta(h,k)}$ new peers, thus
  $S'(t) \leftarrow (1 + \overline{\Delta(h,k)}) \times S'(t-1)$
end if

First, it does not need to continuously track both the newly arrived peers $\lambda(t)$ and those that arrived in previous time slots but not yet joined the system $U(t-1)$. Instead, it can simply monitor the total number of requests issued by distinct new peers $I(t)$, which can be readily captured via the tracking server deployed in most existing P2P live streaming systems [1], [3] for peer registration and bootstrapping.

Second, it does not rely on accurate knowledge of the number of existing peers in the system $S(t-1)$, which is hard to obtain in large-scale P2P live streaming systems during a flash crowd. Instead, it allows the use of a conservative estimate $S'(t-1)$ to achieve proven robustness based on Theorem 4 in Appendix H, available in the online supplemental material. Specifically, Theorem 4 implies that we can make a conservative estimate of the initial system scale M by tuning $\varepsilon \in (0,1]$ in Algorithm 2, at the early stage of a flash crowd (e.g., just after a new live broadcast has been released). Then, a guaranteed rate of system scale increase can be preserved under population control. Furthermore, system designers can jointly consider the $S'(t)$ predicted by Algorithm 2 and empirical measurements (e.g., consider peer departures during the flash crowd) to adjust and ensure the subsequent conservative estimation of $S'(t)$.

As an illustration, Fig. 13 compares the simulated system scale over time during a flash crowd, under no population control, ideal population control in Algorithm 1 with accurate knowledge, and the simplified framework in Algorithm 2 with conservative and over estimation of the initial system scale, respectively. It shows that ideal

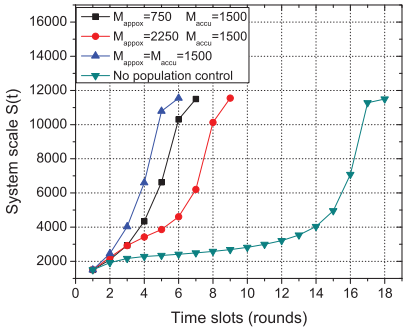


Fig. 13. System scale over time slots during a flash crowd, under no population control, ideal population control with accurate knowledge, the simplified population control framework with a conservative estimate ($\varepsilon = 0.5$), and an overestimate of the initial system scale ($\varepsilon = 1.5$), respectively. The accurate initial system scale is $M = 1{,}500$ and the scale of the extreme flash crowd is $N = 10{,}000$. Others are set as $k = 20$, $h = x = 5$, $u_s = 0$.

population control with accurate knowledge outperforms the other alternatives for alleviating a flash crowd, as it can help to attain the peak rate of a system scale increase over time in an exponential manner (Theorem 3). Although the population control framework with a conservative estimate of the initial system scale (e.g., $\varepsilon = 0.5$) limits the number of admitted new peers to the system at an early stage, a guaranteed rate of system scale increase can help the system to scale up rapidly in subsequent time. This leads to comparable results to that of ideal population control. While an overestimate of the initial system scale (e.g., $\varepsilon = 1.5$) can degrade the effectiveness of population control, it still substantially improves the system scalability during the flash crowd, compared to the case of no population control. Similar results are observed for general peer arrival patterns besides the extreme flash crowd scenario.

Third, as the optimal control point $(\overline{\delta(h,k)}, \overline{\Delta(h,k)})$ is abstracted as relative ratios based on the generality of Corollary 4, it is independent of specific system population sizes and peer arrival patterns. This enables the use of precomputed optimal control points in our framework, so as to simplify the implementation with reduced real-time computational overhead. For example, given a typical range of h and k based on specific system design choices and empirical measurements, we can precompute a set of optimal control points for each typical setting of (h,k). And the set of precomputed optimal control points can be organized and stored as a map, with different pairs of (h,k) as indices. Then, such a set can be periodically updated in a coarse-grained manner according to potential variations in h and k in practical P2P live streaming systems.[5]

Last but not the least, our framework is also extensible to incorporate other criteria for population control decisions. For example, it is suggested by Kumar et al. [7] that it would be beneficial for a P2P live streaming system operating in the undercapacity region (e.g., in a flash crowd) to preferentially admit high-capacity peers. Consistent with this, Figs. 11, 16,

5. Such an update is supposed to be infrequent, given that: 1) the number of partners for peers k in practical P2P live streaming systems is usually configured within a typical range [1], [3], beyond which the system performance is insensitive to k. 2) Typical peer surplus capacities h can be largely covered via existing measurement statistics on the distribution of user upload bandwidth capacities (e.g., DSL, Cable, and Ethernet) [19].

and 17 (in Appendix H, available in the online supplemental material) also demonstrate that the peer surplus upload capacity h has significant impact on the system scalability and the optimal control point. Note that our framework generally instructs the sweet spot for the number of admitted new peers $A(t) \leq \min\{S'(t-1)\overline{\delta(h,k)}, I(t)\}$, without enforcing which set of new peers $\mathcal{A}(t)$ to be preferentially admitted (or delayed). In practice, the streaming service providers can preferentially admit high-capacity peers, which can enable the system to withstand a higher peer arrival intensity during a flash crowd. Besides, the streaming service providers may also incorporate service differentiation [25] into population control decisions, depending on whether their systems are offered free of charge or as a billable service (e.g., determine $\mathcal{A}(t)$ by prioritizing paying users over free users). Overall, how to incorporate various criteria and practical concerns of P2P live streaming systems to refine and optimize population control decisions is an open problem for future research.

6 Conclusion

Through a mathematical framework we developed, this paper has studied the inherent relationship between time and scale in P2P live streaming systems during a flash crowd, and investigated design principles of population control to alleviate a flash crowd. We have derived an upper bound on the system scale over time, which demonstrated that: having enough upload bandwidth resources is insufficient to sustain a flash crowd, as it takes time for peers to locate available resources. The design implication from our analysis is that we can trade peer startup delays for better system scale, via proper population control in the initial stage of a flash crowd. By identifying critical criteria for population control decisions, we have designed an ideal population control procedure and demonstrated its fundamental limit in improving the system scale. This has further led to a simple and flexible population control framework with practical guidelines. In addition, our analysis also brought an in-depth understanding on the effects of partial knowledge of peers and their competition for a limited pool of upload bandwidth resources, as well as peer departures due to impatience.

Acknowledgments

The research was support in part by a grant from The National Natural Science Foundation of China (NSFC) under grant No. 61103176, by a grant from the NSFC under grant No. 61133006, by a grant from the National Science and Technology Major Project of the Ministry of Science and Technology of China under grant No. 2010ZX-03004-001-03.

References

[1] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Trans. Multimedia,* vol. 9, no. 8, pp. 1672-1687, Dec. 2007.

[2] B. Li, S. Xie, G. Keung, J. Liu, I. Stoica, H. Zhang, and X. Zhang, "An Empirical Study of the Coolstreaming+ System," *IEEE J. Selected Areas in Comm.,* vol. 25, no. 9, pp. 1627-1639, Dec. 2007.

[3] B. Li, S. Xie, Y. Qu, Y. Keung, C. Lin, J. Liu, and X. Zhang, "Inside the New Coolstreaming: Principles, Measurements and Performance Implications," *Proc. IEEE INFOCOM,* Apr. 2008.

[4] B. Li, 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," *Proc. IEEE Globecom,* Nov. 2008.

[5] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky," *Proc. 17th ACM Int'l Conf. Multimedia,* Oct. 2009.

[6] X. Zhang, J. Liu, B. Li, and T. Yum, "CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming," *Proc. IEEE INFOCOM,* Mar. 2005.

[7] R. Kumar, Y. Liu, and K.W. Ross, "Stochastic Fluid Theory for P2P Streaming Systems," *Proc. IEEE INFOCOM,* Apr. 2007.

[8] Y. Liu, "On the Minimum Delay Peer-to-Peer Video Streaming: How Realtime Can It Be?," *Proc. 15th Int'l Conf. Multimedia,* Sept. 2007.

[9] T. Bonald, L. Massoulie, F. Mathieu, D. Perino, and A. Twigg, "Epidemic Live Streaming: Optimal Performance Trade-Offs," *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems,* June 2008.

[10] S. Liu, R.Z. Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance Bounds for Peer-Assisted Live Streaming," *Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems,* June 2008.

[11] C. Feng, B. Li, and B. Li, "Understanding the Performance Gap between Pull-Based Mesh Streaming Protocols and Fundamental Limits," *Proc. IEEE INFOCOM,* Apr. 2009.

[12] Y. Zhou, D. Chiu, and J. Lui, "A Simple Model for Analyzing P2P Streaming Protocols," *Proc. IEEE Int'l Conf. Network Protocols (ICNP),* Oct. 2007.

[13] X. Yang and G. de Veciana, "Service Capacity of Peer to Peer Networks," *Proc. IEEE INFOCOM,* Mar. 2004.

[14] D. Rubenstein and S. Sahu, "Can Unstructured P2P Protocols Survive Flash Crowds?," *IEEE Trans. Networking,* vol. 13, no. 3, pp. 501-512, June 2005.

[15] T. Hoßfeld and K. Leibnitz, "Modeling and Evaluation of an Online TV Recording Service," *Proc. Ninth Ann. Workshop Math. Performance Modeling and Analysis,* June 2007.

[16] F. Simatos, P. Robert, and F. Guillemin, "A Queueing System for Modeling a File Sharing Principle," *ACM SIGMETRICS Performance Evaluation Rev.,* vol. 36, no. 1, pp. 181-192, 2008.

[17] M. Wang, L. Xu, and B. Ramamurthy, "Providing Statistically Guaranteed Streaming Quality for Peer-to-Peer Live Streaming," *Prof. 18th Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* pp. 127-132, 2009.

[18] C. Wu, B. Li, and S. Zhao, "Multi-Channel Live P2P Streaming: Refocusing on Servers," *Proc. IEEE INFOCOM,* Apr. 2008.

[19] C. Huang, J. Li, and K. Ross, "Can Internet Video-on-Demand be Profitable?," *Proc. ACM SIGCOMM,* Aug. 2007.

[20] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-points," *Proc. ACM SIGCOMM,* Aug. 2004.

[21] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "A Performance Study of BitTorrent-Like Peer-to-Peer Systems," *IEEE J. Selected Areas in Comm.,* vol. 25, no. 1, pp. 155-169, Jan. 2007.

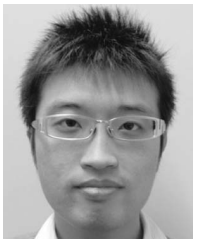
[22] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-Like Systems," *Proc. Fifth ACM SIGCOMM Conf. Internet Measurement (IMC),* Oct. 2005.

[23] M. Zhang, Q. Zhang, L. Sun, and S. Yang, "Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?," *IEEE J. Selected Areas in Comm.,* vol. 25, no. 9, pp. 1678-1694, Dec. 2007.

[24] S. Xie, B. Li, G. Keung, and X. Zhang, "Coolstreaming: Design, Theory, and Practice," *IEEE Trans. Multimedia,* vol. 9, no. 8, pp. 1661-1671, Dec. 2007.

[25] A. Magnetto, R. Gaeta, M. Grangetto, and M. Sereno, "TURINstream: A Totally pUsh, Robust, and effIcieNt P2P Video Streaming Architecture," *IEEE Trans. Multimedia,* vol. 12, no. 8, pp. 901-914, Dec. 2010.

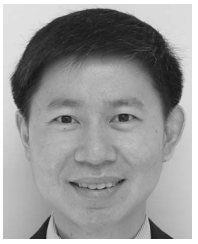
[26] R. Sibly and J. Hone, "Population Growth Rate and Its Determinants: An Overview," *Philosophical Trans. Royal Soc. B,* vol. 357, no. 1425, pp. 1153-1170, 2002.



Fangming Liu (S'08-M'11) received the BEngr degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2005; and the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology in 2011, where he was awarded with the Overseas Research Award and Postgraduate Excellence Scholarships. He is currently an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. In 2007, he worked as a research assistant at the Department of Computer Science and Engineering, Chinese University of Hong Kong. From August 2009 to February 2010, he was a visiting student at the Computer Engineering Group, Department of Electrical and Computer Engineering, University of Toronto, Canada. Since 2010, he has also collaborated with the ChinaCache Content Delivery Network Research Institute in Tsinghua University, Beijing. His research interests are in the area of peer-to-peer networks, rich-media distribution, cloud computing, and large-scale data center networking. He is a member of the IEEE, the IEEE Communications Society, and the ACM.

Bo Li (S'89-M'92-SM'99-F'11) received the BEngr degree in computer science from Tsinghua University, Beijing, and the PhD degree in the electrical and computer engineering from the University of Massachusetts at Amherst. He is a professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He was with IBM Networking System, Research Triangle Park, between 1993 and 1996. He was an adjunct researcher at Microsoft Research Asia (MSRA) (1999-2006), where he spent his sabbatical leave (2003-2004). He was with Microsoft Advanced Technology Center (ATC) in the summers of 2007 and 2008. His works have resulted in more than 220 publications. He has made original contributions on internet proxy placement, capacity provisioning in wireless networks, routing in WDM optical networks, and internet video streaming. He is best known for a series of works on a system called Coolstreaming (Google entries over 1,000,000 in 2008 and Google scholar citations over 800), which attracted millions of download and was credited as the first large-scale Peer-to-Peer live video streaming system in the world. His recent work on the peer-assisted online hosting system, FS2You (2007-2009) (Google entries 800,000 in 2009) has also attracted millions of downloads worldwide. He received two best paper awards from the IEEE. He received the Young Investigator Award from Natural Science Foundation of China (NFSC) in 2005. He has been an editor or guest editor for 17 IEEE/ACM journals and magazines, and he was involved in organizing 50 conferences. He was the co-TPC chair for IEEE Infocom'04. He was a distinguished lecturer in IEEE Communications Society (2006-2007). He is a fellow of the IEEE.

Lili Zhong received the BEngr degree from Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2008, and the MPhil degree from Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2011. Her research interests include peer-to-peer networking and cloud computing.



Baochun Li (S'98-M'00-SM'05) received the BEngr degree from Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 1995, and the MS and PhD degrees from the Department of Computer Science, University of Illinois at Urbana-Champaign, in 1997 and 2000, respectively. Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently a professor. He holds the Bell University Laboratories endowed chair in Computer Engineering since August 2005. In 2000, he was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems. In 2009, he was the recipient of the Multimedia Communications Best Paper Award from the IEEE Communications Society. His research interests include large-scale multimedia systems, peer-to-peer networks, applications of network coding, and wireless networks. He is a senior member of the IEEE, and a member of the ACM.

Hai Jin received the PhD degree in computer engineering from HUST in 1994. He is a Cheung Kung Scholars chair professor of computer science and engineering at the Huazhong University of Science and Technology (HUST) in China. He is now a dean of the School of Computer Science and Technology at HUST. He is the chief scientist of ChinaGrid, the largest grid computing project in China, and the chief scientist of National 973 Basic Research Program Project of Virtualization Technology of Computing System. He worked at The University of Hong Kong between 1998 and 2000, and as a visiting scholar at the University of Southern California between 1999 and 2000. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He was awarded Excellent Youth Award from the National Science Foundation of China in 2001. He is the member of Grid Forum Steering Group (GFSG). He has coauthored 15 books and published more than 400 research papers. His research interests include computer architecture, virtualization technology, cluster computing, and grid computing, peer-to-peer computing, network storage, and network security. He is the steering committee chair of International Conference on Grid and Pervasive Computing (GPC), Asia-Pacific Services Computing Conference (APSCC), International Conference on Frontier of Computer Science and Technology (FCST), and Annual ChinaGrid Conference. He is a member of the steering committee of the IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid), the IFIP International Conference on Network and Parallel Computing (NPC), and the International Conference on Grid and Cooperative Computing (GCC), International Conference on Autonomic and Trusted Computing (ATC), International Conference on Ubiquitous Intelligence and Computing (UIC). He is a senior member of the IEEE and a member of the ACM.

Xiaofei Liao received the PhD degree in computer science and engineering from Huazhong University of Science and Technology (HUST), China, in 2005. He is now an associate professor in school of Computer Science and Engineering at HUST. He has served as a reviewer for many conferences and journal papers. His research interests are in the areas of virtualization technology for computing system, peer-to-peer system, cluster computing and streaming services. He is a member of the IEEE and the IEEE Computer Society.