Stochastic Fluid Theory for P2P Streaming Systems

Rakesh Kumar
Department of Electrical and
Computer Engineering,
Polytechnic University,
Brooklyn, NY, USA 11201
Email: rkumar04@utopia.poly.edu

Yong Liu
Department of Electrical and
Computer Engineering,
Polytechnic University,
Brooklyn, NY, USA 11201
Email: yongliu@poly.edu

Keith Ross
Department of Computer and
Information Science,
Polytechnic University,
Brooklyn, NY, USA 11201
Email: ross@poly.edu

Abstract—We develop a simple stochastic fluid model that seeks to expose the fundamental characteristics and limitations of P2P streaming systems. This model accounts for many of the essential features of a P2P streaming system, including the peers' real-time demand for content, peer churn (peers joining and leaving), peers with heterogeneous upload capacity, limited infrastructure capacity, and peer buffering and playback delay. The model is tractable, providing closed-form expressions which can be used to shed insight on the fundamental behavior of P2P streaming systems.

The model shows that performance is largely determined by a critical value. When the system is of moderate-to-large size, if a certain ratio of traffic loads exceeds the critical value, the system performs well; otherwise, the system performs poorly. Furthermore, large systems have better performance than small systems since they are more resilient to bandwidth fluctuations caused by peer churn. Finally, buffering can dramatically improve performance in the critical region, for both small and large systems. In particular, buffering can bring more improvement than can additional infrastructure bandwidth.

I. INTRODUCTION

With the widespread adoption of broadband residential access, live video streaming may be the next disruptive IP communication technology. As an indication of the potential of live video streaming, recently a commercial P2P streaming system broadcasted the 2006 Chinese New Year's celebration to over 200,000 users over the Internet at bit rate in the 400-800 kbps range [9], generating an aggregate bit rate in the vicinity of 100 gigabits/sec. In the future, we expect to see thousands of live video streaming channels available on the Internet, each with a bit rate of 500 kbps or more, each supporting tens of users to hundreds of thousands of users.

There are several classes of delivery architecture for live video streaming, including native IP multicast [6], application-level infrastructure overlays such as those provided by CDN companies [1], [7], and P2P architectures. Requiring minimal infrastructure, the P2P architectures offer the possibility of rapid deployment at lowest cost. P2P streaming architectures roughly fall into two categories: (i) multicast trees such as in end-system multicast [3]; (ii) and pull-driven P2P streaming such as CoolStreaming [18], PPLive [13] and PPStream [14].

Bearing strong similarities to BitTorrent [5], pull-driven P2P streaming architectures have the following characteristics:

- A video (live or stored) is divided into media chunks of about one second in duration, and the chunks are made available at an origin server.
- A peer, interested in viewing the video stream, obtains from the system a list of peers currently watching the video. The peer then establishes partner relationships (TCP connections) with the peers on the list.
- The peer requests media chunks from its partners and (possibly) from the server. But because the chunks have playback deadlines, a peer only requests chunks that can likely be received before their playback deadlines.
- Once a peer has obtained a chunk, it makes the chunk available for downloading by other peers.

An important characteristic of pull-driven P2P streaming is the lack of an (application-level) multicast tree - a characteristic particularly desirable for the highly dynamic, high-churn P2P environment [18]. Although pull-driven P2P streaming has similarities with BitTorrent, BitTorrent in itself is not a feasible delivery architecture, since it does not account for the real-time needs of streaming.

Several pull-driven P2P streaming systems have been successfully deployed to date, accommodating thousands of simultaneous users. Most of the these deployments have originated from China (including Hong Kong). The pioneer in the field, CoolStreaming, reported more than $4,000$ simultaneous users in 2003. More recently, a number of second-generation pull-driven P2P streaming systems have reported phenomenal success on their Web sites, advertising tens of thousands of simultaneous users who watch channels at rates between 300 kbps to 1 Mbps. These systems include PPLive [13], PPStream [14], VVSky [17], TVAnts [16] and FeiDian [8].

In P2P streaming systems, participating nodes are very heterogeneous, particularly in terms of the amount of upload bandwidth they contribute [9]. Today there are roughly two classes of peers participating in P2P streaming systems: broadband residential peers with DSL and cable access; and institutional peers with high-bandwidth Ethernet access. In addition to being heterogeneous, nodes churn, with peers randomly joining the system, watching the video for a random period of time, and then leaving the system. As the peers churn, both the system's demand for video as well as the system's

overall ability to supply video changes. Another important characteristic of P2P live streaming systems is that they can allow for small buffering delays, which potentially mitigate against the short-term bandwidth variations due to peer churn. Broadly speaking, a P2P streaming system performs well if all participating peers can continuously playback the video (without freezing or skipping) with a small playback delay.

In this paper we develop a simple stochastic fluid model that seeks to expose of the fundamental characteristics and limitations of P2P streaming systems. This model accounts for many of the essential features of a P2P streaming system, including the peers' real-time demand, peer churn (peers joining and leaving), peers with heterogeneous upload capacity, limited server upload capacity, and buffering and playback delay. Additionally, the model is tractable, providing closed-form expressions which can be used to shed insight on the fundamental behavior of P2P streaming systems. We use the stochastic fluid model to seek answers to the following questions.

- What are the key parameters that determine the performance of the P2P streaming system? Is there a threshold effect for which the performance switches from poor to excellent as the threshold is crossed?
- It has been observed that large P2P streaming systems generally perform better than small systems [9]. Why do large systems perform better?
- What happens to performance as the system scales? In particular, for a dynamic system with churn, what is the asymptotic performance of the system as the average number of participating peers become very large?
- Can buffering and playback lag significantly improve performance? If so, by how much and in what circumstances?
- Can we quantify the benefit of additional infrastructure resources? Will increasing the server upload rate significantly improve performance?
- Finally, how can admission control be applied to provide adequate service to all peers while minimizing the number of rejected peers?

This paper is organized as follows. In Section 2 we introduce the basic model for P2P streaming with peer churn. In Section 3, we take a brief interlude and derive necessary and sufficient conditions for a churnless system. These conditions are not only central to our stochastic model with peer churn but are also of independent interest. In Section 4 we return to systems with churn. We first determine an explicit expression for the probability of degraded service. We then employ an asymptotic model to study large P2P streaming systems. In Section 5 we explore the potential for improvement with playback buffering and lag at the peers. In Section 6 we use the results from Section 4 to develop an effective admission control scheme for P2P streaming systems. We summarize the contributions of this paper and conclude in Section 7.

A. Related Work

To our knowledge, this is the first paper that presents an analytical model for P2P streaming systems (fluid or otherwise). Here, we briefly describe other papers that propose fluid models for P2P *download* systems. Qui and Srikant [15] developed and solved a fluid model for BitTorrent-like systems. The model accounts for churn, and views the number of seeds and leechers as fluid quantities. They develop simple differential equations for the fluids and solve the equations in steady state. Clevenot et al [4] develop a multiclass fluid model for BitTorrent systems. The multiclass fluid model leads to a non-linear system of differential equations with special structure. They prove the system of differential equations admits a unique stable equilibrium, which is computed in closed-form. The fluid models in [15] and [4] are not applicable to streaming systems with heterogeneous upload rates, since there is no notion of leechers transitioning to seeds.

There is also recent work in modeling the time it takes to distribute a file from seeds to leechers in *churnless download* systems. Mundinger et al have studied this problem for heterogeneous peers with infinite download capacity, both for chunk-based and fluid-based systems [12]. Kumar and Ross derived an explicit expression for the minimum download time in a general heterogenous fluid system with finite download rates. They also extended this result to multi-class systems with first and second-class leechers. Biersack et al used a chunk-based model to derive expressions for the distribution time for a several practical overlay topologies [2].

II. MODELING P2P STREAMING

In this section we provide our basic model and notation of P2P streaming. The video originates from a server node; denote by u_s (in bps) for the upload rate of the server. Let r denote the rate (in bps) of the video. The video is to be streamed to all participating peers.

We classify each peer as either a *super peer* or an *ordinary peer*. Super peers provide high-speed access rates in excess of a few Mbps; ordinary peers have residential broadband access, typically with upload rates of 500 kbps or less. In our model, all super peers have the same upload capacity u_1; and all ordinary peers have the same upload capacity u_2 with $u_2 < u_1$. Unless explicitly stated, we assume that $u_2 < r < u_1$. In other words, a super peer can upload at a rate higher than the video rate, and an ordinary peer's uploading capacity is smaller than the video rate. We often refer to super peers and ordinary peers as class-1 and class-2 peers, respectively. *Although we are assuming only two classes of the peers, the theory and results presented here can easily be extended to any number of classes, with each class having its own upload rate.* However, we shall see that a two-class model suffices to expose many of the key issues underlying P2P streaming.

Peers join and leave the P2P streaming system at random times. As in existing P2P streaming systems, whenever a peer joins the system and receives chunks of video, it is obligated to redistribute the chunks it receives [18] [13] [14]. Denote by λ_i for the rate at which class-i peers join the P2P streaming

system. Denote by $1/\mu_i$ for the average amount of time a class-i peer views the video (and hence sojourns in the system). We make no assumptions on the distribution of the peer sojourn (viewing) times. Peers from the two classes join the system as two independent Poison processes. Let $P_i(t)$ be the number of class-i peers in the system at time t. Clearly, $P_1(t)$ and $P_2(t)$ are two independent $M/G/\infty$ processes [10].

Having described the model for peer churn, we now turn to streaming. We adopt a fluid flow model and focus on the instantaneous rate at which peers receive and transfer bits. Initially we assume a bufferless system, that is, bits cannot be buffered before playback or before copying to other peers. (In Section 5, we extend the model to allow for buffers and playback lag.) In this bufferless model, a peer can playback the video whenever it receives fresh content bits at rate r. When all participating peers receive the video at rate r, we say that the system provides *universal streaming*. When the system is not providing universal streaming, we say that system operates in *degraded service mode*.

At any given instant of time, whether universal streaming can be accomplished or not depends of number of super peers and ordinary peers in the system at time t, that is, it is a function of $P_1(t)$ and $P_2(t)$. The more super peers in the system, the greater the average upload capacity per peer and the easier it is to accomplish universal streaming. Denote by $\Phi(P_1(t), P_2(t))$ for the maximal rate at which the system can deliver fresh content to *each* of the peers when the system is in state $(P_1(t), P_2(t))$. The function $\Phi(\cdot, \cdot)$ depends on the efficiency or the distribution scheme in the pull-driven P2P streaming protocol. Universal streaming occurs at time t if and only if $\Phi(P_1(t), P_2(t)) \geq r$. To complete the stochastic fluid model, we need to specify $\Phi(\cdot, \cdot)$, which we refer to as the *fluid function*. In the next section we provide an explicit expression for $\Phi(\cdot, \cdot)$ for an optimized system.

III. UNIVERSAL STREAMING FOR CHURNLESS SYSTEMS

In this section we seek to derive the maximum streaming rate r for which universal streaming is possible for a churnless system, that is, for a system in with a fixed set of peers. We do this for a system that is more general than that described in the previous section - namely, we consider general heterogeneous systems, with each peer having its own upload rate.

The results derived in this section will play a central role in the analysis of the stochastic system with churn. However, they are also of independent interest, as they provide simple and explicit expressions for the maximum rate of a churnless P2P streaming system.

Denote by n for the number of peers in the system and let u_i denote of the upload capacity of peer i for $i = 1, \ldots, n$. Viewing bits as fluid, bits arrive to the server at rate r. As the bits arrive to the server, they can be copied to one or more peers. As bits arrive to a peer, they can also be copied to one or more of the remaining peers. The aggregate bit rate out of the server cannot exceed u_s; the aggregate bit rate out of a peer cannot exceed u_i, $i = 1, \ldots, n$. The system can perform universal streaming if it is possible to copy and route the bits so that all peers receive fresh bits at rate r. We define the *maximum achievable rate* to be the maximum value of r such that the system can perform universal streaming.

Theorem 1: The maximum achievable streaming rate, $r_{\max}$, is given by

$$r_{\max} = \min\{u_s, \frac{u_s + \sum_{1=1}^{n} u_i}{n}\}.$$

Proof: Denote

$$u(\mathcal{P}) = \sum_{1=1}^{n} u_i$$

Clearly, the maximum streaming rate cannot exceed the aggregate upload rate of the server; thus, $r_{\max} \leq u_s$. Furthermore, the maximum aggregate rate that bits can flow out the server and out of the n peers is bounded by $u_s + u(\mathcal{P})$. This maximum aggregate rate, if it could be achieved, needs to be distributed to the n peers. Thus the maximum streaming rate to an individual peer is also bounded by $u_s + u(\mathcal{P})/n$. Combining these two bounds gives

$$r_{\max} \leq \min\{u_s, \frac{u_s + u(\mathcal{P})}{n}\}. \tag{1}$$

It remains to show that if (i) $u_s \leq (u_s + u(\mathcal{P}))/n$, then the streaming rate $r = u_s$ can be supported; and if (ii) $u_s > (u_s + u(\mathcal{P}))/n$, then the streaming rate $r = (u_s + u(\mathcal{P}))/n$ can be supported.

Suppose $u_s \leq (u_s + u(\mathcal{P}))/n$. Consider a video stream of rate $r = u_s$. Divide this video stream into n substreams, with the ith substream having rate

$$s_i = \frac{u_i}{u(\mathcal{P})} u_s \text{ for all } i \in \mathcal{P}$$

Note that the aggregate rate of the n substreams is equal to the rate of the stream, that is,

$$\sum_{i=1}^{n} s_i = u_s = r.$$

We have the server copy the ith substream to the ith peer. Furthermore, because $(n-1)s_i \leq u_i$, we can have the ith peer copy its substream to each of the other $n-1$ peers. Thus each peer receives a substream directly from the server and also receives $n-1$ additional substreams from the other $n-1$ peers. The total rate at which peer i receives is

$$r_i = s_i + \sum_{k \neq i} s_k = u_s.$$

Hence the rate $r = u_s$ can be supported.

Now suppose $u_s > (u_s + u(\mathcal{P}))/n$. Consider a video stream of rate $r = (u_s + u(\mathcal{P}))/n$. Divide this video stream into $n+1$ substreams, with the ith substream, $i = 1, \ldots, n$, having rate $s_i = u_i/(n-1)$ and the $(n+1)$st substream having rate

$$s_{n+1} = (u_s - \frac{u(\mathcal{P})}{n-1})/n$$

Clearly $s_i \geq 0$ for all $i = 1, \ldots, n+1$. Now have the server copy two substreams to each peer i: the ith substream and the $(n+1)$st substream. The server can do this because

$$\sum_{i=1}^{n} (s_i + s_{n+1}) = u_s$$

Furthermore, have each peer i stream a copy of the ith substream to each of the $n-1$ other peers. Each peer i can do this because $(n-1)s_i = u_i$ for $i = 1, \ldots, n$. The total rate at which peer i receives is

$$r_i = s_i + s_{n+1} + \sum_{k \neq i} s_k = (u_s + u(\mathcal{P}))/n.$$

Hence the rate $r = (u_s + u(\mathcal{P}))/n$ can be supported. $\square$

We remark that this result can be extended to include finite download rates at each of the n peers. In particular, suppose peer i has download rate d_i. Let $d_{\min} = \min\{d_i : i = 1, \ldots, n\}$. Then

$$r_{\max} = \min\{u_s, \frac{u_s + u(\mathcal{P})}{n}, d_{\min}\} \qquad (2)$$

We omit the proof of (2) for brevity; see [11] for a related result concerning the minimum time to distribute a file to all peers in a P2P system.

A. Two-Class Model

Let us now return to the original model in which every peer is either a super peer or an ordinary peer. Denote by n_1 and n_2 the number of super peers and ordinary peers in the P2P streaming system. The following result is a consequence of Theorem 1 and will be used repeatedly in this paper:

Corollary 1: For any rate r such that $u_2 < r < u_1$, universal streaming is achievable by some fluid distribution scheme if and only if

$$r \leq \phi(n_1, n_2) \qquad (3)$$

where

$$\phi(n_1, n_2) = \min\{u_s, \frac{u_s + n_1 u_1 + n_2 u_2}{n_1 + n_2}\}.$$

IV. P2P STREAMING WITH PEER CHURN

We now return to original model as given in Section 2 with peers joining and departing at random times. Denote $\rho_i = \lambda_i/\mu_i$. Recall that with $P_i(t)$ denoting the number of active type-i peers at time t, the two stochastic processes $(P_1(t), t \geq 0)$ and $(P_2(t), t \geq 0)$ are independent M/G/∞ processes with arrival rates λ_1 and λ_2 and departure rates μ_1 and μ_2. Recall that to complete the stochastic fluid model, we need to specify a fluid function $\Phi(\cdot, \cdot)$. Henceforth, we use $\Phi(\cdot, \cdot) = \phi(\cdot, \cdot)$, where $\phi(\cdot, \cdot)$ is given in Corollary 1. Thus, we assume an optimized P2P distribution scheme. *We remark, however, that the theory developed in this paper can be easily extended to any relevant fluid function $\Phi(\cdot, \cdot) \leq \phi(\cdot, \cdot)$. For example, we could use $\Phi(\cdot, \cdot) = .8 \cdot \phi(\cdot, \cdot)$, modeling a P2P distribution scheme that is only 80% efficient.*

A natural question is, in steady-state, for what fraction of time do we have universal streaming? We refer to this fraction of time as the *universal streaming probability*. Let P_i be the random variable denoting the number of active type-i peers in steady state. It is well-known that P_i has a Poisson distribution with mean $E[P_i] = \rho_i$. From Corollary 1, we have

$$P(\text{universal streaming}) = P(P_1 \geq cP_2 - u'_s) \qquad (4)$$

where

$$c = \frac{r - u_2}{u_1 - r} \text{ and } u'_s = \frac{u_s}{u_1 - r}$$

Since P_1 and P_2 are independent Poison random variables, we can explicitly calculate the universal streaming probability as follows. Let $M = \lfloor \frac{u'_s}{c} \rfloor$. We have

$$P(P_1 \geq cP_2 - u'_s) = \sum_{l=0}^{\infty} P(P_1 \geq cl - u'_s | P_2 = l)P(P_2 = l)$$

$$= P(P_2 \leq M) + \sum_{l=M+1}^{\infty} P(P_1 \geq cl - u'_s)P(P_2 = l)$$

$$= P(P_2 \leq M) + \sum_{l=M+1}^{\infty} P(P_1 \geq \lceil cl - u'_s \rceil)P(P_2 = l)$$

$$= F_2(M) + \sum_{l=M+1}^{\infty} (1 - F_1(\lceil cl - u'_s \rceil) + f_1(\lceil cl - u'_s \rceil))f_2(l) \quad (5)$$

where

$$f_i(n) = \frac{e^{-\rho_i} \rho_i^n}{n!} \text{ and } F_i(n) = \sum_{l=0}^{n} f_i(n)$$

We will return to this result when we present numerical results at the end of this section.

A. Large System Analysis

We now scale the system by letting $\rho_1 \to \infty$ and $\rho_2 \to \infty$. It is natural to consider scaling regimes in which $\rho_1/\rho_2 = K$ for some constant K. We consider here a more generalized regime in which

$$\rho_1 = K\rho_2 + \beta\sqrt{\rho_2} \qquad (6)$$

for some $K > 0$ and some β (positive or negative). We will see that this more generalized scaling will enable us to glean additional insight into the fundamental characteristics of P2P streaming systems.
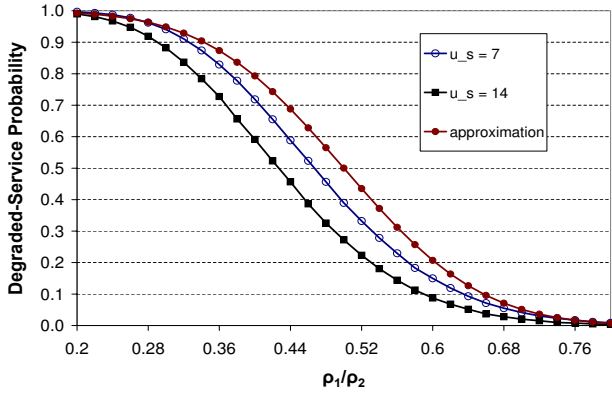
Theorem 2: In an asymptotic regime with $\rho_1 = K\rho_2 + \beta\sqrt{\rho_2}$, the asymptotic probability of universal streaming is given by

$$\lim_{\rho_2 \to \infty} P(P_1 \geq cP_2 - u'_s) = \begin{cases} 1 & K > c \\ F(\frac{-\beta}{\sqrt{c+c^2}}) & K = c \\ 0 & K < c \end{cases} \qquad (7)$$
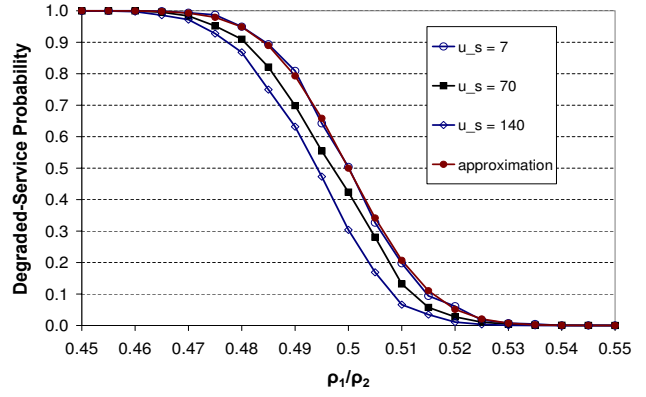
where $1 - F(\cdot)$ is the distribution function of the standard normal random variable.

Proof: Define the normalized random variables

$$X_1 = \frac{P_1 - \rho_1}{\sqrt{\rho_1}}, X_2 = \frac{P_2 - \rho_2}{\sqrt{\rho_2}}$$

| (a) Small System $\lambda_2 = 100$ | (b) Large System $\lambda_2 = 10000$ |

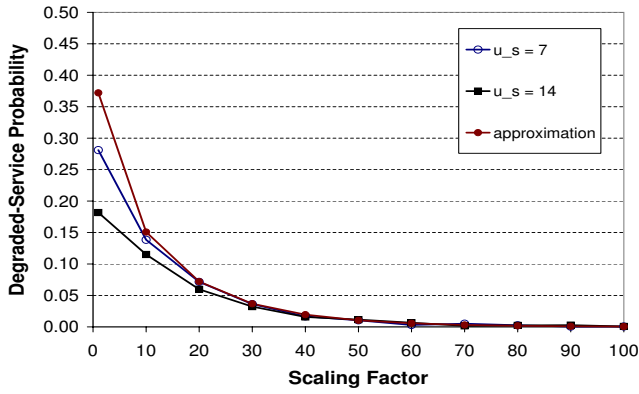Fig. 1. *Degraded-Service probability for small and large systems*

Fig. 2. *Degraded-Service probability with $\rho_1/\rho_2 = 0.54$ fixed. The system size is scaled in multiples of 10.*

Note that

$$P(P_1 \geq cP_2 - u'_s) = P(\sqrt{\rho_1}X_1 + \rho_1 \geq c(\sqrt{\rho_2}X_2 + \rho_2) - u'_s)$$

$$= P(\sqrt{K + \frac{\beta}{\sqrt{\rho_2}}}X_1 + (K-c)\sqrt{\rho_2} + \beta \geq cX_2 - \frac{u'_s}{\sqrt{\rho_2}})$$

Now, as $\rho_2 \to \infty$, clearly this probability goes to 1 if $K > c$ and goes to 0 if $K < c$. Now consider the case $K = c$. As $\rho_2 \to \infty$, this probability goes to

$$P(\sqrt{K}Z_1 - cZ_2 > -\beta)$$

where Z_1 and Z_2 are two independent standard normal random variables. But since a linear combination of independent normal random variables is also a normal random variable, we have

$$P(\sqrt{K}Z_1 - cZ_2 > -\beta) = F(\frac{-\beta}{\sqrt{c+c^2}}) \ \square$$

Theorem 2 indicates that P2P streaming systems exhibit a critical threshold. For a large system, if $\rho_1/\rho_2 + \epsilon < c$, then the performs poorly, rarely providing universal streaming. On the other hand, if $\rho_1/\rho_2 > c + \epsilon$, then the system almost always

provides universal streaming. This critical threshold c plays an important role in the design and operation of P2P streaming systems. We say that when $\rho_1/\rho_2 \approx c$, we say that system operates in the *critical region*.

Theorem 2 also leads to useful and simple approximation for medium and large systems when operating in the critical region Given ρ_1, ρ_2 and $c = (r - u_2)/(u_1 - r)$, we set $K = c$ and solve for β in (6):

$$\beta = \frac{\rho_1 - c\rho_2}{\sqrt{\rho_2}}.$$

We then plug this expression for β into $F(-\beta/\sqrt{c+c^2})$ from Theorem 2 and obtain the explicit approximation:

$$\text{P(universal streaming)} \approx F(\frac{\rho_1 - c\rho_2/\sqrt{\rho_2}}{\sqrt{c+c^2}}). \qquad (8)$$

B. Numerical Results and Insights

We now explore how the equation (5) and the approximation (8) can be used to study the performance of P2P streaming systems. We do this for two systems: a "small system" with a number of concurrent peers in the vicinity of 75; and a large system with the number of concurrent peers in the vicinity of 7,500.

For both the small and large systems, we use the rates $r = 3$, $u_2 = 1$ and $u_2 = 7$. These rates could be, for example, in units of 100 kbps. The chosen rates reflect current streaming rates, residential upload rates, and enterprise/university rates. For this choice of rates, the video rate is three times the upload rate of the ordinary peers; and the upload rate of the superpowers is 7 times that of the ordinary peers. (For example, most PPLive channels are currently in the vicinity of 400 kbps, which is about 2-4 times the upload rate of many residential broadband peers. University access rates vary, depending on traffic and university-to-ISP bandwidths; for most cases, we expect u_2/u_1 to be in the 5 to 20 range.) These values give $c = 0.5$ for the critical factor.

In these numerical examples, we use different server rates u_s, all multiples of the u_1. By using different multiples for the server rate, we explore how additional infrastructure resources can potentially improve performance.
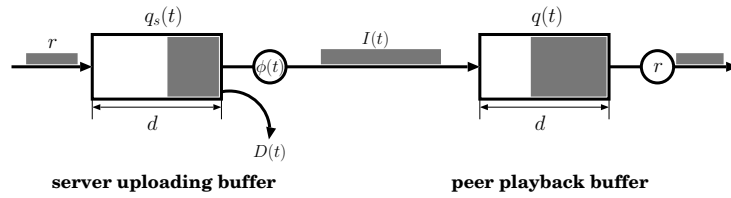
Fig. 3. *Tandem Fluid Queueing Model for P2P Streaming Buffering*

We set throughout $1/\mu_1 = 1/\mu_2 = 0.5$ hours. Thus, we suppose that super peers and ordinary peers sojourn in the system on average for 30 minutes. For the small and large system, we fix the arrival rate of the ordinary peers, λ_2, and vary the arrival rate of the super peers in vicinity of $\lambda_2/2$. For the small system, we set the arrival rate of ordinary peers to $\lambda_2 = 100/hr$, so that the average number of ordinary peers in the system is $\rho_2 = \lambda_2/\mu_2 = 50$ and the average number of super peers, ρ_1, is in the vicinity of 25. For the large system, we set the arrival rate of ordinary peers to $\lambda_2 = 10,000/hr$, so that the average number of ordinary peers in the system is $\rho_2 = \lambda_2/\mu_2 = 5,000$ and the average number of super peers, ρ_1, is in the vicinity of $2,500$.

Figure 1(a) shows the probability of degraded service as a function of ρ_1/ρ_2 for the small system. Three curves are shown: one for the approximation and two for the exact value with two different server rates. As expected, performance improves as the arrival rate of super peers increases (equivalently, as ρ_1/ρ_2 increases). We observe that by doubling the infrastructure resources for a server bandwidth of $u_s = 7$ to $u_s = 14$, we can obtain significant performance gains. These improvements, however, diminish when operating outside of the critical region (for example, when either $\rho_1/\rho_2 > 0.7$ or $\rho_1/\rho_2 < 0.3$). We also observe from this figure that although the approximation (8) follows the general performance trend, it significantly over estimates the probability of degraded service throughout the critical region. This is to be expected, since the approximation is derived from a large-system asymptotic analysis.

Figure 1(b) shows the probability of degraded service as a function of ρ_1/ρ_2 for the large system. Four curves are shown: one for the approximation and three for the exact value with three different server rates. As expected, performance improves as the arrival rate of super peers increases (equivalently, as ρ_1/ρ_2 increases). For the large system, the critical region is very pronounced. For each of three values of μ_s, when $\rho_1/\rho_2 < .47$ the system almost always operates in the degraded service mode. When $\rho_1/\rho_2 > .47$, the system almost always provides universal service. Figure 1(b) also shows that doubling or tripling the infrastructure resources (that is, μ_s) have little impact on performance. For this large system, we need to increase u_s by a factor of 10 to 100 to get significant gains. Finally, the figure also shows that the approximation is very accurate for large systems. In fact, the approximation is accurate even outside of the critical region.

We also investigate the probability of degraded service as the system is scaled from small to large. For this we fix $\rho_1/\rho_2 = 0.54$. As before the small size system is chosen to be with $\lambda_2 = 100$. The given ratio $\rho_1/\rho_2 = 0.54$ yields $\lambda_1 = 54$. From this the system is linearly scaled with a scaling factor increased from 1 to 100 in multiples of 10. Figure 2 shows the probability of degraded service with respect to this scaling factor. We again see that the role of the server upload rate diminishes as the system scales.

As mentioned in the Introduction, it has been observed that P2P streaming systems with large peer populations perform better than systems with small populations [13]. This general claim is collaborated by Figures 1 and 2. For example, consider the case of $\mu_s = 7$. For the large system with $\rho_1/\rho_2 > .53$, universal streaming is essentially always provided. However, for the small system, with $\rho_1/\rho_2 = .53$, degraded service occurs more than 25% time; even at $\rho_1/\rho_2 = .68$, degraded service occurs more than 5% time for the small system. Thus, the large system provides universal service over a much wider range of system parameters. This can be explained as follows. When a super peer leaves the system, the impact will be relatively small in a large system as there is an averaging effect due to the large number of super peers in the system. However, in a small system, the loss of a super peer can have a dramatic effect, moving the system from the critical region to the overloaded region.

V. Buffering and Playback Lags

In Section 4 we observed that one of the fundamental characteristics of P2P streaming systems is that peer churn introduces fluctuations in the available upload bandwidth. In traditional client-server streaming applications, buffering and playback delays are commonly used to mitigate against fluctuating available bandwidth. In this section, we investigate whether buffering can also improve performance in P2P streaming systems.

The basic idea of buffering in P2P streaming systems is to build up reservoirs of content in the peers' playback buffers which is consumed when the upload rate falls below the live streaming rate. Unlike in video-on-demand systems, the upload rate in a P2P streaming system is not only constrained by its aggregate uploading capacity $u_s + u_1 P_1(t) + u_2 P_2(t)$ but also by the rate at which fresh content is generated. For a live streaming session, fresh content is generated by the source at a fixed rate r. If all peers want to have real-time playback, the uploading rate of new content to a single peer can never exceed r. In order to have reservoirs of content at the peers, it is therefore necessary for peers to have playback lags so that
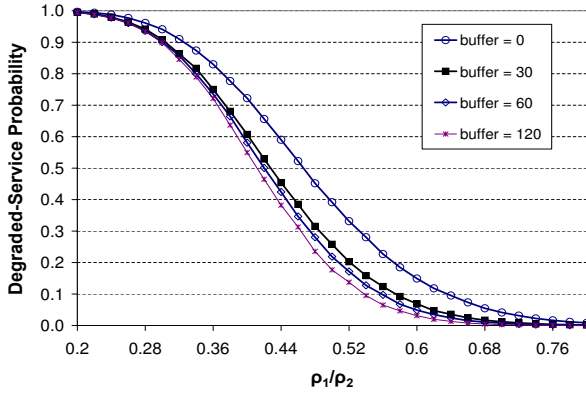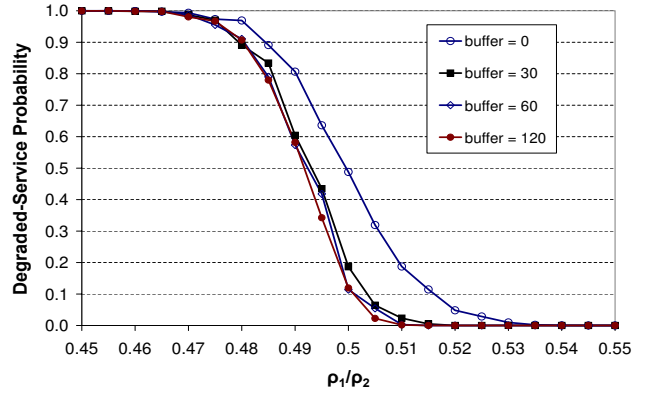
Fig. 4. *Buffering: Degraded-Service probability for small and large systems*

they can *pre-fetch* content before playback when the average uploading rate in the system exceeds r.

To illustrate the potential benefits of buffering, we now extend the system of Section 4 by placing at each peer a playback buffer that can hold up to d seconds of video. We also introduce a server upload buffer. The buffered system operates as follows. The server is feed content at rate r and uses the P2P system to distribute the content to all peers. Before starting playback, each peer first fills its playback buffer. Once playback starts, each peer playback buffer is drained at rate r and the server buffer continues to be fed at rate r. Under the optimized content distribution algorithm described in the previous section, the maximum rate at which a peer can download new content, both directly from the server and indirectly from other peers, at time t is

$$\phi(t) = \min\left\{ u_s, \frac{u_s + u_1 P_1(t) + u_2 P_2(t)}{P_1(t) + P_2(t)} \right\}. \qquad (9)$$

We call $\phi(t)$ the available bandwidth for a peer at time t. $\phi(t)$ is also the rate at which the server can pump out new content to the whole P2P streaming system. Note $\phi(t)$ is *not* the aggregate rate of traffic uploaded from the server to all peers. For example, if the server simultaneously uploads to two peers at the full live streaming rate r, the aggregate traffic rate is $2r$, but the aggregate new content rate pumped into the system is just r.

When the available bandwidth $\phi(t)$ is below r, the server cannot pump out new content at the full video rate. Consequently, the server buffer level increases at rate $r - \phi(t)$. At the same time, a peer's video download rate is lower than the video playback rate. Therefore, the peer playback buffer level decreases at the same rate $r - \phi(t)$. Thus, as the reservoir in the peer's buffer decreases, the reservoir in the server's buffer increases. If later the available bandwidth $\phi(t)$ becomes greater than r, then content in the server's reservoir is transferred over to the peer buffer. In this manner, buffering at the server can help to mitigate the effects of a fluctuating upload bandwidth $\phi(t)$ due to peer churn.

We require that a peer always keep the same playback lag d. In other words, content that arrives to a peer after its deadline
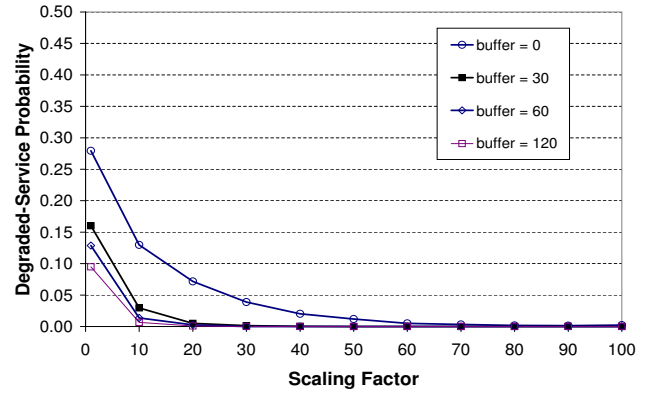

Fig. 5. *Buffering: Degraded-Service probability with $\rho_1/\rho_2 = 0.54$ fixed. The system size is scaled in multiples of 10.*

will be skipped. One mechanism to achieve content skipping is to set the server buffer size to d, and to have the server drop at rate $r - \phi(t)$ from the *head* of the buffer when the server buffer is full. In this way, we ensure that all the content in the server uploading buffer is always "fresh" enough to meet the peers' playback deadlines. Our extended system uses this mechanism.

Under the optimized content distribution algorithm and homogeneous peer playback policy, all peers download and playback content synchronously. The interaction between the content buffer level on the server and the playback buffer level on any single peer in the streaming system can be modeled as a tandem queue system illustrated in Figure 3. This system has two fluid queues in tandem: the server upload queue and a peer playback queue. Denote by $q_s(t)$ and $q(t)$ for the contents in the server queue and the peer queue, respectively. The server queue is filled in at the constant video source rate r and is drained at rate $\phi(t)$ when the buffer is non-empty and at the rate r when the buffer is empty; thus the output rate of the server queue at time t is

$$I(t) = \phi(t)1(q_s(t) > 0) + r1(q_s(t) = 0).$$

As discussed above, when the server queue is full and

$r > \phi(t)$, content is removed from the head of the server queue while fresh content continues to join the tail of the queue. The content is removed from the server queue at rate $D(t) = [r - \phi(t)]1(q_s(t) = d)$. Note that we always have $q_s(t) + q(t) = rd$ after playback begins. Under the optimized distribution algorithm, all peers download new content, both directly from the server and indirectly from other peers, at the output rate of the server queue, namely, at rate $I(t)$. The peer queue is drained at the playback rate r when the buffer is non-empty. Skipping occurs if $q(t) = 0$ and $I(t) < r$.

A. Simulation Results and Insights

This tandem fluid queue system exactly models our fluid P2P streaming system with peer playback buffers and a server uploading buffer. It is of interest to compare the performance of this buffered system to that the bufferless system studied in Section 4. The buffered system does not appear to be analytically tractable, so we instead appeal to simulation.

For the simulation results reported here, we use the same parameters used in Section 4: $r = 3$, $u_2 = 1$, $u_2 = 7$, and $1/\mu_1 = 1/\mu_2 = 0.5$ hours. We fix the server bandwidth to $u_s = 7$ in these experiments.

Figure 4(a) shows the probability of degraded service for a small system at different buffer capacities of 0, 30, 60 and 120 seconds of video. We observe that the system enjoys significant performance gains by adding only a 30 second lag to the playback. Additional gains are possibly by further increasing the playback lag. Comparing 4(a) with 1(a) we see that playback lags can achieve the same performance gains as infrastructure increases without the cost of added infrastructure.

Figure 4(b) shows the probability of degraded service for a large size system at different buffer capacities. We observe that in the critical region, playback lags can bring a dramatic improvement in performance. For example, when operating at $\rho_1/\rho_2 = c = 0.5$, without buffering the probability of degraded service is approximately 50% With buffering, this probability drops to approximately 5%. We also observe that a 30-second lag is sufficient to obtain almost all the potential buffering gains from buffering. Furthermore, on comparing 4(b) and 1(b), we observe that having a small lag of 30 seconds is more effective than deploying 10 times the server capacity for this large system.

We also investigate the probability of degraded service as the system is scaled from small to large. For this we fix $\rho_1/\rho_2 = 0.54$. As before the small size system is chosen to be with $\lambda_2 = 100$. The given ratio $\rho_1/\rho_2 = 0.54$ yields $\lambda_1 = 54$. From this the system is linearly scaled with a scaling factor increased from 1 to 100 in multiples of 10. Figure 5 shows the probability of degraded service with respect to this scaling factor. We again observe the advantage of deploying even small size buffers while the system is operating in the critical region. (When the system gets very large, we operate in the under-loaded region, in which the probability of degraded service becomes almost zero with and without buffering.)

Duration	Buffer = 0	Buffer = 30	Buffer = 60
= 0	0.61	0.74	0.78
$\in (0, 18]$	0.0065	0.00056	0.00047
$\in (18, 36]$	0.007	0.00086	0.00068

TABLE I
SMALL SIZE SYSTEM: FIRST COLUMN SHOWS DEGRADED-SERVICE DURATION INTERVALS. OTHER COLUMNS SHOW PROBABILITY OF THE SYSTEM BEING IN THAT PARTICULAR DEGRADED-SERVICE INTERVAL FOR DIFFERENT BUFFER SIZES. ALL TIME UNITS ARE IN SECONDS.

Duration	Buffer = 0	Buffer = 30	Buffer = 60
= 0	0.51	0.81	0.89
$\in (0, 18]$	0.023	0.0017	0.0012
$\in (18, 36]$	0.009	0.0011	0.0009

TABLE II
LARGE SIZE SYSTEM: FIRST COLUMN SHOWS DEGRADED-SERVICE DURATION INTERVALS. OTHER COLUMNS SHOW PROBABILITY OF THE SYSTEM BEING IN THAT PARTICULAR DEGRADED-SERVICE INTERVAL FOR DIFFERENT BUFFER SIZES. ALL TIME UNITS ARE IN SECONDS.

B. Distribution of Degraded-Service Durations

Using our simulation tool, we also investigate the distribution of the degraded-service durations for both bufferless and buffered systems. We fix the ratio $\rho_1/\rho_2 = 0.5$ and with system parameters of the small and large systems as before, we simulate the fraction of time the system spends in degraded-service of different durations. Table I and II show the probabilities of finding the system in degraded-service duration of (i) = 0 (universal streaming), (ii) duration between 0 and 18 seconds and (iii) duration between 18 and 36 seconds. These probabilities are shown for three different buffer sizes of 0 seconds (bufferless), 30 seconds and 60 seconds. One can observe that the probability of finding the system (both large and small) in degraded service decreases by a factor of 10 by increasing the buffer sizes. Further, the probability of universal streaming increases substantially as buffer sizes are increased.

VI. THE UNDER-CAPACITY REGION

We learned in Section 4 that, at any instant of time, a P2P streaming system operates in one of three regions: (i) the over-capacity region for which $\rho_1/\rho_2 > c + \epsilon$; (ii) the critical region for which $\rho_1/\rho_2 \approx c$; (iii) and the under-capacity region for which $\rho_1/\rho_2 < c + \epsilon$. The value of ϵ depends on the size of the system, being smaller for big systems.

Over larger time scales, the values of ρ_1 and ρ_2 will likely change, possibly causing the system to drift from one of the three regions to another. From Section IV we learned that when in the over-capacity region, system performance should be universally good. We showed in Section V that system performance is also good in the critical region when buffering and playback lags are employed. However, in the under-capacity region, even with large buffers, performance will generally be poor, with universal streaming rarely occurring.

When operating in the under-capacity region, we have a number of options including:

- Apply admission control to peers which provide relatively little upload capacity (ordinary peers in our two-class model). This requires detecting when the system begins to enter the under-capacity region, and then rejecting an appropriate fraction of the low-bandwidth peers that want to join the system.
- Apply some form of scalable coding technique, so that the rate of the video r decreases as the system begins to enter the under-capacity region. Both layered video and multiple-description video are possibly candidates for the scalable video.
- Provide good service to as many of the low-capacity peers as possible. We design the distribution scheme so that high-capacity peers always get the video at rate r; we also try to provide as many low-capacity peers as possible with rate r. The remaining peers receive nothing.

Designing robust schemes to handle transitions from one operating region to another remains an important research problem and will be considered in a future paper.

VII. Summary of Contribution and Lessons Learned

In this paper we have developed tractable and relevant analytical models for P2P streaming systems. The contributions of this paper include:

- *Conditions for universal streaming for churnless systems.* For churnless systems with heterogenous upload and download capacities, we have derived a simple necessary and sufficient condition for the existence of a fluid distribution scheme that achieves universal streaming.
- *A novel model for P2P streaming systems with peer churn.* The model leads to an explicit expression for the probability of degraded service. We used the expression to investigate the performance of small and large P2P streaming systems.
- *Asymptotic analysis of P2P streaming systems.* We have studied analytically the performance of a P2P streaming system as the joining rate of new peers becomes very large. We obtained an explicit expression for the probability of degraded service for large systems. We used that expression to develop an accurate and simple approximation for degraded service for all operating regions for moderate and large systems.
- *An extended model with buffers and playback lag.* This exact buffer model, although not analytically tractable, is amenable to efficient simulation. We built a simple simulation tool to analyze the affects of buffering as well as the distribution of degraded-service durations.

The models developed and analyzed in this paper have led to a number of important lessons for P2P streaming systems:

1) Peer churn introduces fluctuations in the available upload bandwidth. Specifically, when a super peer leaves or when an ordinary node joins, the available bandwidth to a peer (averaged over all peers) decreases; similarly, when a super peer joins or when an ordinary node leaves, the available bandwidth to a peer increases.

2) The performance of the system is largely determined by a critical value. For systems of moderate-to-large size, if the ratio of average number of super peers to average number of ordinary peers (ρ_1/ρ_2) exceeds the critical value c, the system performs well; otherwise, the system performs poorly.

3) Big systems have better performance. Big systems are more resilient to bandwidth fluctuations caused by peer churn. Big systems therefore have robust performance over a wider range of traffic loads (that is, ρ_1/ρ_2 values).

4) Buffering can dramatically improve performance in the critical region, for both small and large systems. It can bring more improvement than that provided with additional infrastructure bandwidth.

5) Special attention must be given when operating in the under- capacity region. Both admission control and scalable video hold promise for dealing with the under-capacity region.

References

[1] *"Akamai,"* http://www.akamai.com
[2] E.W. Biersack, P. Rodriguez and P. Felber, *"Performance Analysis of Peer-to-Peer Networks for File Distribution,"* In proceedings of Quality of Future Internet Services (QOFIS04), September 2004, Barcelona, Spain.
[3] Y. Chu, S. G. Rao and H. Zhang, *"A Case for End System Multicast,"* ACM SIGMETRICS 2000, June 2000, Santa Clara, CA, USA.
[4] F. Clevenot, P. Nain and K.W. Ross, *"Multiclass P2P Networks: Static Resource Allocation for Bandwidth for Service Differentiation and Bandwidth Diversity,"* Performance 2005, Juan-les-Pins, 2005.
[5] B. Cohen, *"Incentives Build Robustness in BitTorrent,"* First Workshop on Economics of Peer-to-Peer Systems, June 2003, Berkeley, CA.
[6] S. Deering and D. Cheriton, *"Multicast Routing in Datagram Internetworks and Extended LANs,"* ACM TRANSACTIONS on Computer Systems, Pages 85-111, May 1990.
[7] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw and D. Stodolsky, *"A Transport Layer for Live Streaming in a Content Delivery Network"*, IEEE PROCEEDINGS, volume 92, pages 1408- 1419, September 2004.
[8] *"FeiDian,"* http://tv.net9.org
[9] X. Hei, C. Liang, J. Liang, Y. Liu and K. W. Ross, *"A Measurement Study of a Large-Scale P2P IPTV System,"* Submitted.
[10] L. Kleinrock, *"Queueing Systems, vol. I: Theory,"* John Wiley and Sons, 1975.
[11] R. Kumar, K.W. Ross, *"Peer Assisted File Distribution: The Minimum Distribution Time,"* To appear in First IEEE Workshop on Hot Topics in Web Systems and Technologies, HOTWEB 2006, November 2006, Boston, MA , USA.
[12] J. Mundinger, R. R. Weber and G. Weiss, *"Analysis of Peer-to-Peer File Dissemination amongst Users of Different Upload Capacities,"* Performance Evaluation Review, Performance 2005 Issue.
[13] *"PPLive,"* http://www.pplive.com
[14] *"PPStream,"* http://www.ppstream.com
[15] D. Qiu and S. Srikant, *"Modeling and Performance Analysis of BitTorrent-Like Perr-to-Peer networks,"* ACM SIGCOMM 2004, August 2004, Portland, Oregon.
[16] *"TVAnts,"* http://www.tvants.com
[17] *"VVSky,"* http://www.vvsky.com.cn
[18] X. Zhang, J. Liu, B. Li and T-S. P. Yum, *"CoolStreaming: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming,"* IEEE INFOCOM 2005, March 2005, Miami, FL, USA.