An Analytical Study of Peer-to-Peer Media Streaming Systems

Yi-Cheng Tu† Jianzhong Sun‡ Mohamed Hefeeda§ Sunil Prabhakar†

†Department of Computer Sciences
Purdue University
West Lafayette, IN, USA
{tuyc, sunil}@cs.purdue.edu

‡Department of Mathematics
University of North Carolina
Wilmington, NC, USA
sunj@uncw.edu

§School of Computing Science
Simon Fraser University
Surrey, BC, Canada
mhefeeda@cs.sfu.ca

ABSTRACT

Recent research efforts have demonstrated the great potential of building cost-effective media streaming systems on top of peer-to-peer (P2P) networks. A P2P media streaming architecture can reach a large streaming capacity that is difficult to achieve in conventional server-based streaming services. Hybrid streaming systems that combine the use of dedicated streaming servers and P2P networks were proposed to build on the advantages of both paradigms. However, the dynamics of such systems and the impact of various factors on system behavior are not totally clear. In this paper, we present an analytical framework to quantitatively study the features of a hybrid media streaming model. Based on this framework, we derive an equation to describe the capacity growth of a single-file streaming system. We then extend the analysis to multi-file scenarios. We also show how the system achieves optimal allocation of server bandwidth among different media objects. The unpredictable departure/failure of peers is a critical factor that affects the performance of P2P systems. We utilize the concept of *peer lifespan* to model peer failures. The original capacity growth equation is enhanced with coefficients generated from peer lifespans that follow an exponential distribution. We also propose a failure model under arbitrarily distributed peer lifespan. Results from large-scale simulations support our analysis.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and Principles

General Terms

performance

Technical Report CSD-TR-05-011, Department of Computer Sciences, Purdue University, 2005

Keywords

media streaming, peer-to-peer, media-on-demand, CDN

1. INTRODUCTION

Multimedia streaming over the Internet has become a reality with the development of media compression methods, high-throughput storage systems, and broadband networking technology. Attractive applications such as entertainment video-on-demand, digital libraries, and on-line news services built on top of real-time media streaming architectures are now publicly available. However, there are still many challenges towards building cost-effective, robust and scalable multimedia streaming systems [33] due to the stringent bandwidth, packet loss and delay requirements for media streaming.

A majority of media streaming systems follow a server-client design. The server deployed by service providers acts as the streaming entity and client devices controlled by the users act as passive receivers of media streams. In a large streaming system where user requests arrive at a high rate, a server has to support a large number of concurrent streaming sessions. Multiple servers or proxies can be deployed to increase total system capacity. In this design, media content is replicated on these proxies and clients receive streaming data from the closest proxy. There are two advantages of using proxies: (i) user requests are handled by all proxies with a combined capacity greater than the capacity provided by the single-server architecture; (ii) better QoS (in terms of latency and packet loss) in streaming due to the shortened packet delivery path. Such systems are sometimes called Content Distribution Networks (CDNs) [3].

The cost of maintaining a CDN is extremely high considering the massive CPU power, storage space and bandwidth needed. As the service becomes more popular, more servers have to be deployed. One approach to solve the above problem is motivated by the emerging concept of peer-to-peer (P2P) computing [17, 27, 8]. In a P2P system, there is no centralized entity controlling the behavior of peers. Instead, each peer contributes its share of resources and cooperates with other peers according to some predefined rules for communications. In the context of media streaming, a well-organized community of clients can significantly lower the service load of CDN servers by taking over some of the streaming tasks. The basic idea is to let clients that have acquired a media object act as streaming servers for subsequent requests to that object. One of the key features of a P2P streaming system is that its total capacity grows when the content it manages becomes more popular [35]. This is the most important difference between P2P and the server/client paradigms. To some extent, a P2P architecture can be viewed as an extreme case of a CDN: data are replicated on a large number of client nodes.

Hybrid media streaming systems that combine centralized servers and peer-to-peer networks have been proposed in [34] and [15, 14]. As compared to a P2P-only architecture, the hybrid streaming system can disseminate media content faster and responds quicker to requests. System performance in media streaming services is mainly bottlenecked by bandwidth [20]. Some operations such as directory management and searching that consume less bandwidth can be processed at a centralized server for efficiency reasons. Furthermore, peers are heterogeneous in the duration of their commitment to the community [29]: each peer could leave or fail at any time. In order to minimize the effects of this come-and-go behavior, servers can act as backup resource providers even when the P2P network has enough capacity. Servers are qualified to play this role because of their robustness.

The focus of this paper is to study the features of a hybrid media streaming architecture by mathematical analysis. We are primarily interested in the pattern of system capacity growth and the effects of various factors on that growth. The system capacity is defined as the total streaming bandwidth available from both servers as well as peers. Conclusions drawn from such analysis improve our understanding of system dynamics and provide guidelines for the design and realization of media delivery services based on hybrid architectures. In this paper, we propose a generic media streaming model that utilizes both a CDN and P2P network. The model differs from those found in [34] and [15] in that it is applicable to a more general environment and is more amenable to quantitative analysis of system performance. Using a deterministic discrete-time analysis approach, we derive the growth equation for system capacity. This equation is used to analyze the time threshold at which the system capacity is sufficient to handle the total load. We extend the results from a single-file system to a multi-file system. We also show that our streaming architecture achieves near-optimal performance in terms of the time needed for a complete load hand-over from servers to peers. Furthermore, peer failures are factored into the analytical model. We study peer failures by associating a 'lifespan' with each peer and analyzing the system performance under different lifespan distributions. We also present two enhancements of the analysis. In the first, we analyze a media streaming system in which a peer can start serving a file to others before completely receiving it. We show that this overlapping of receiving and serving can significantly accelerate the capacity growth and reduce the server-peer transition time. In the second enhancement, we study a general multi-file streaming system in which files may have different lengths and bit rates. We describe how the optimal transition time can be computed numerically. In addition, we evaluate several performance metrics by extensive simulations, the results of which confirm the validity of our analysis. We shall also see that most of our analysis apply directly to pure-P2P media systems without servers.

This paper continues with Section 2 by comparing our research with related work. Then we introduce the streaming model in Section 3. We start our analysis by studying single-file systems without failures in Section 4. Then we extend to multi-file systems (Section 5) and systems with peer failures (Section 6). Two extensions of the main results are presented in Section 7. Section 8 presents the simulation results. We conclude the paper with Section 9.

2. RELATED WORK

A review on Internet video streaming can be found in [33]. The key research areas of video streaming are identified and methodologies are discussed. Research on P2P computing was greatly motivated by the success of Gnutella[1] and Napster[2]. The general philosophy and current research efforts of P2P computing are introduced in [17] and [8]. Pastry [27], Chord [30], and CAN [25] are the most popular P2P searching/routing protocols. P2P applications built on top of these protocols are presented in [28] and [9]. Other topics of P2P research include system design [26], traffic measurement [29], and usage of coupons/incentives [16, 13].

In the context of peer-to-peer media streaming, both the CoopNet project [20] and the ZIGZAG prototype [31] explore how media streams should be delivered to many clients under the situation of flash crowd. Both projects concentrate on how to efficiently maintain a multicast tree in an environment where user behavior is unpredictable. CoopNet utilizes the method of *Multiple Description Coding* (MDC) to deal with the in-session departure/failure of streaming peers. Our system model differs from these efforts in the sense that we focus on the delivery of on-demand media instead of live media. Commercial content delivery services such as Allcast[3] and C-Star[4] are close in spirit to on-demand P2P media streaming. In [35], an algorithm that assigns media segments to different supplying peers and an admission protocol for requests are introduced. [18] emphasizes streaming protocol design. In their work, an RTP-like protocol with the features of rate control and packet synchronization is developed.

Research on hybrid media streaming architecture shown in [34] is directly related to our work. A similar P2P streaming architecture can be found in [15] and [14], in which efficient algorithms for dissemination of media content and economical analysis of P2P streaming services are presented. They show that, with small initial investment and the use of incentives, a large-scale and profitable media streaming service can be built.

Several recent efforts emphasize performance analysis of P2P networks. In [36], a branching model and a Markov chain model are used to study the system dynamics of a BitTorrent[5]-like file sharing network in its transient and steady states, respectively. They find that the capacity of such systems grows exponentially in transient and stabilizes at

[1]http://www.gnutella.com
[2]http://www.napster.com
[3]http://www.allcast.com
[4]http://www.centerspan.com
[5]http://www.bittorrent.com

steady state. The above work is extended in [23] where a fluid model is exploited to quantify system capacity at steady state so that explicit expressions of performance metric are obtained (vs. numerical results obtained in [36]). Furthermore, other features of the BitTorrent network such as downloading efficiency and incentives are discussed. In [24], downloading speed in similar systems are analyzed with consideration of network topology and peer heterogeneity. Our work differs from the above efforts in the following aspects:

1. We deal with P2P media streaming rather than file downloading. We study system capacity and transition time using a discrete-time analytical method;
2. We accomplish a quantitative analysis of the performance of a multi-file system and prove optimality in terms of the transition time of the system model. Ours is the only work that accomplishes this, to the best of our knowledge;
3. We explore the impact of peer failure under different failure models. Among them, the matrix model is not found in any other P2P research.

Among the above contributions, items 1 and 3 can be readily used to study pure-P2P streaming systems (i.e., those without servers). Our study greatly improves the analytical research on a hybrid streaming system presented in [34] as the latter only considers single-file system without failures. Finally, we extend the conference version of this paper [32] by relaxing several assumptions such that the analysis applies to more general cases of system operation. For instance, we study system performance under the effects of arbitrary distributions of peer lifespan (Section 6.2), shortened intersession delays (Section 7.1), and variable streaming length and bitrate of media objects (Section 7.2).

3. SYSTEM MODEL, ASSUMPTIONS, AND NOTATIONS

Our analysis is based on the media streaming infrastructure shown in Figure 1B. The model is similar to the hybrid structure proposed in [34] and [15] with some modifications. The main entities of the system are:

- **Directory Server.** The role of the directory server is to maintain an index of media location (i.e. what peers hold copies of the media). It is also responsible for processing queries.[6]
- **Server.**[7] A server holds a copy of all media files and is responsible for streaming when the requested media cannot be served through the P2P network. Each server has a fixed bandwidth. We assume a zero downtime for the servers.
- **Peer (client).** The set of user machines participating in the streaming system are known as peers. A peer asking for a media object is called a *requesting peer* and a peer that has acquired any media object(s) is called a *qualified peer*. Upon joining the system, each peer announces its maximum bandwidth and storage contribution. We assume honesty of peers in the contribution of their reported resources. In our model, peers admit any requests forwarded to them when they have available bandwidth. We do not specify the maximum number of streaming sessions a peer can support. The reason is that most peers have limited bandwidth [29] and can only support less than one session in practice. We divide peers into a number of classes based on their bandwidth contributions. The average bandwidth contribution of peers in all classes is α.
- **Media content.** The target resource a client requests. We can view this as a collection of media files. To simplify the analysis, we assume that all streams are Constant Bit Rate (CBR) media streams.

Figure 1B shows all entities in the hybrid streaming architecture and how they interact. For the sake of comparison, Figure 1A shows a CDN-based streaming architecture. Note that the main difference between the CDN architecture and the hybrid architecture is that client machines can be data senders in the latter. We understand that streaming protocols and softwares are essential parts of any streaming architecture but they are beyond the scope of this paper.

In our model, the system operates as follows. When a peer requests a media object, it first sends out a query to the directory server. The directory server searches its local database and returns a list of available qualified peers and CDN servers to the requesting peer. We first choose CDN servers with available bandwidth as data senders. If all CDN servers are busy, the requesting peer chooses from the list of qualified peers a subset that satisfies the bandwidth and QoS requirements and the streaming starts.[8] The peers that act as data senders are called *supplying peers*. When the streaming is finished, the requesting peer becomes a qualified peer. If there is not enough bandwidth from both the servers and qualified peers, the request is rejected immediately (without waiting).

We model the arrival of streaming requests as a Poisson process with a time-invariant rate λ. We are interested in a system where the streaming capacity of the servers is small compared to the total capacity needed to handle all requests. In other words, servers act as 'seeds' for the media content and we expect that the streaming load will eventually be shifted to peers.

3.1 Assumptions

The system analysis is performed in a top-down manner. We start from a simple model with assumptions on the factors we are interested in and then enhance the results derived from the simple model by relaxing these assumptions.

[6] If we replace the directory server with non-centralized object lookup solutions (e.g. Pastry), our analysis still works as we focus on system throughput rather than lookup latency.

[7] In this paper, the terms 'streaming server', 'CDN server' and 'server' are used interchangeably.

[8] The mechanism of selecting the subset of peers to stream the media object is orthogonal to the analysis presented in this paper. We note that there are a number of ways to perform the selection. A simple mechanism is to choose suppliers that are topologically close to the receiver based on IP addresses. This can be done by clustering peers in the system using their IP addresses [14]. As another mechanism, the receiver could leverage Internet measurement infrastructures such as IDMaps [11] to measure relative distances between each potential supplier and itself. A third selection mechanism infers characteristics of the network paths connecting the potential suppliers. These characteristics are then used to select the best subset of suppliers [15].

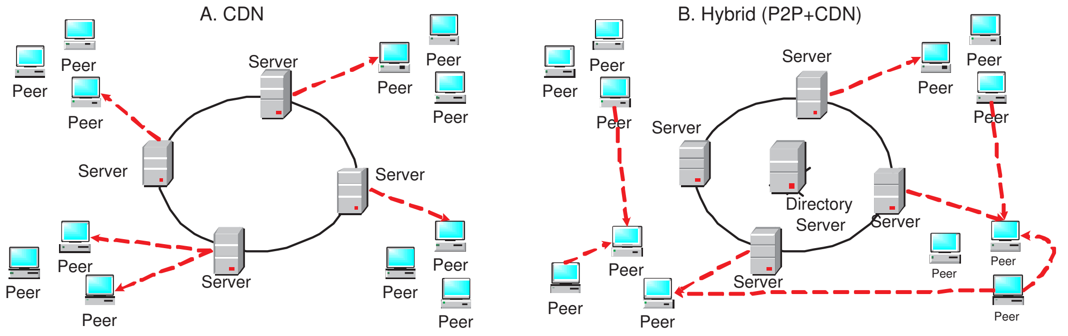

Figure 1: A comparison between CDN and hybrid media streaming architectures. Dotted lines show the directions of the streaming.

In the initial analysis, we make the following assumptions:

1. The system contains only one media file. In Section 5, the analysis is extended to multi-file systems;

2. Peers never fail. Peer failure is addressed in Section 6;

3. Requests are uniformly distributed among the peer population;

4. The bottleneck link for a streaming session can only be the upload link capacity of the data sender (CDN server or supplying peer); and

5. Each participating peer has infinite storage contribution. We shall see in Section 8 that only a small storage contribution is actually needed from each peer, which makes this a harmless assumption.

3.2 Metrics and Notations for Analysis

The total number of qualified peers and their bandwidth contributions are direct measures of system capacity. Our analysis focuses on these two metrics. Previous work [34] on hybrid streaming systems has shown that the streaming load can be fully taken over by peers after some time. This can be illustrated by Figure 2 where the tentative system bandwidth is plotted. As more and more peers become qualified peers, we have reason to believe that the system capacity grows over time (not necessarily linear growth as shown in Figure 2). Suppose the total capacity required to handle all requests is R, then at a certain point in time, the system capacity outgrows R. This time point is called *server-peer transition time* (denoted as k_0). Knowing k_0, we can modify the protocol to let the requesting peers obtain bandwidth from qualified peers first and use servers as backup sources of bandwidth after transition. For service providers, k_0 can be used, for example, to indicate when server resources can be reallocated to stream other media objects, or to determine the length of their contracts with resource vendors.

With the knowledge of k_0, we can modify our protocol by retiring the servers after k_0. This is illustrated by Figure 3 where tentative server bandwidth usage under such change of protocol is plotted. At time zero, all server bandwidth is free. Servers become fully loaded after a short initial stage due to massive demand. The system has to reject some requests until the servers generate enough loyal qualified

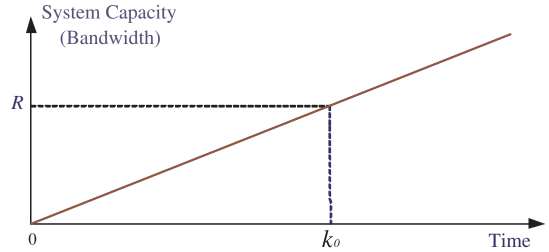

Figure 2: Proposed peer bandwidth growth and transition point in the hybrid streaming architecture.

peers. After that, servers no longer accept streaming requests. Thus, they are increasingly alleviated from streaming tasks until the system reaches a stage where only peers are needed for streaming (Peer-only stage). Note that the server bandwidth usage is greater than zero until the Peer-only stage begins at $k_0{}'$. This is due to the outstanding streaming sessions that are being served by the servers after k_0.

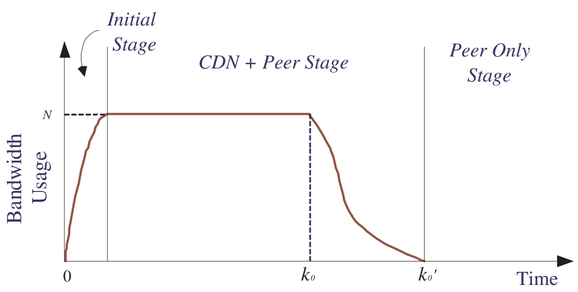

Figure 3: Bandwidth usage of CDN servers in the hybrid streaming architecture with retirement of servers at k_0.

Another metric we consider is *reject rate*. The reject rate at time x is defined as the ratio of total number of rejected requests to the total number of requests within a time interval $[x - \Delta x, x + \Delta x]$. Here we see that reject rate depends on the window size $2\Delta x$. So there is no unique value for it at any point in time. Intuitively, reject rate decreases as system capacity increases. With the knowledge of system capacity, we can derive the expected reject rate. In our experimental study (Section 8), we use zero reject rate as an indication of the system's accomplishing server-peer transition.

Symbols used in the analysis are listed in Table 1.

Table 1: Notations and Symbols.

Symbol	Definition
L	Length of one streaming session
k	Discrete time index, each has a length of L
N	Total server bandwidth
M	Total number of peers
λ	Request rate to the system, in requests per unit time
k_0	Server-Peer transition time, in number of streaming periods
T_0	Server-Peer transition time, in number of time units
$P(k)$	Number of qualified peers at interval k
$C(k)$	Total system capacity (i.e. bandwidth) at interval k
F	Total number of media files
b	Bandwidth required to stream a file
α	Average peer bandwidth contribution
z	Number of peer classes
h_i	Percentage of peers in the i–th class
c_i	Bandwidth contribution per peer of the i–th class

4. MAIN RESULT: SINGLE-FILE SYSTEM WITHOUT FAILURE

In this section, we focus our analysis on a system containing only one media file without considering peer failures. A salient feature of the media streaming application is that all streaming sessions of the same media object last for L time units. This feature can be leveraged to analyze (qualified) peer population in a discrete-time manner. Suppose, within the k-th time interval (Figure 4), the system initiates n streaming sessions $S_1, S_2, \ldots, S_n$. Then we are certain that the n requesting peers in these sessions will become qualified peers in the $(k+1)$–th interval as all sessions will terminate by the end of the next interval. To project the total number of qualified peers from period k to period $k+1$, we need to know how many sessions are initiated during period k.

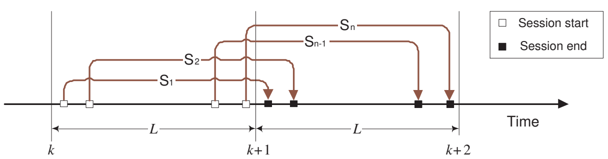

Figure 4: Streaming sessions within a time interval of length L.

First of all, the average peer bandwidth contribution of all classes can be obtained by $\alpha = \sum_{i=1}^{z} h_i c_i$. Due to assumption 5 in Section 3.1, this simplification does not affect our analysis on system bandwidth.

Before the server-peer transition time k_0, the bandwidth of both servers and qualified peers is fully utilized as demands are overwhelming. Therefore, the number of new qualified peers produced between two consecutive time intervals k and $k+1$ can be expressed as:

$$P(k+1) - P(k) = \frac{N}{b} + P(k)\frac{\alpha}{b}, \qquad 0 \le k \le k_0. \tag{1}$$

The two terms on the right-hand side of the above equation are the number of new qualified peers generated by servers and that of the qualified peers generated by previously qualified peers, respectively. Eq.(1) can be rewritten as

$$P(k+1) + \frac{N}{\alpha} = \left(P(k) + \frac{N}{\alpha}\right)\left(1 + \frac{\alpha}{b}\right).$$

Solving the above geometric progression with $P(0) = 0$, we get

$$P(k) = \frac{N}{\alpha}\left[\left(1+\frac{\alpha}{b}\right)^k - 1\right]. \tag{2}$$

We name the term $\frac{\alpha}{b}$ as the *capacity growth factor* of the system. The total system capacity (in terms of bandwidth) at interval k is thus given by:

$$C(k) = N + \alpha P(k) = N\left(1+\frac{\alpha}{b}\right)^k, \quad k \le k_0. \tag{3}$$

From the discussions in Section 3, we know that the total system capacity at k_0 is equal to the capacity required to serve all requests. The total capacity needed to satisfy all requests in L time units is $\lambda L b$. Therefore, we have the following equation to solve k_0:

$$N\left(1+\frac{\alpha}{b}\right)^{k_0} = \lambda L b. \tag{4}$$

In the above equation, N, α, λ, L, b are known constants, therefore we get k_0 as

$$k_0 = \log_{(1+\frac{\alpha}{b})}\left(\frac{\lambda L b}{N}\right) = \frac{\lg(\lambda L b) - \lg N}{\lg(1+\frac{\alpha}{b})}. \tag{5}$$

From Eq.(2), we also see that $P(k_0) = \frac{N}{\alpha}(\frac{\lambda L b}{N} - 1)$ peers are needed in addition to the servers so that total system bandwidth is able to handle all subsequent requests. Note that the unit for k_0 is time intervals rather than natural time units and it is only accurate to its integer part. For example, a k_0 value of 10.4 indicates the transition will be accomplished some time between the 10th and 11th streaming period.

The above analysis (refer to Eq.(3)) shows an exponential growth pattern of system capacity, which is similar to the results of a recent study on P2P file sharing applications [36]. The expansion of our streaming system resembles the population growth of a biological species [21]. The latter is generally studied by the number of offsprings produced in generation(s). Our analysis is inspired by this idea: requesting peers can be regarded as the offspring of supplying peers and/or servers. [9] The discrete-time analysis approach is also used in [34] and similar results (with numerical solution for k_0) are found. We improve their analysis by giving a closed-form expression for the server-peer transition time.

[9] From now on, we sometimes use the word 'generation' to denote a streaming period with length L.

REMARK 4.1. *In practice, the capacity growth factor* $\frac{\alpha}{b}$ *is small (typically less than* 1.0*), because the average bandwidth contribution from a peer* (α) *is less than the bandwidth required to stream the media object* (b)*. As a result,* k_0 *is almost linearly related to* $\frac{\alpha}{b}$ *since*

$$k_0 = \frac{\lg(\lambda Lb) - \lg N}{\lg(1+\frac{\alpha}{b})} \approx \frac{\lg(\lambda Lb) - \lg N}{\frac{\alpha}{b}}.$$

This shows that k_0 *is more sensitive to* $\frac{\alpha}{b}$ *than to* λ *and* N*. The effect of* L *on* k_0 *is also similar to that of* λ *and* b*. Note that there is a super-linear relationship between* L *and the* absolute *transition time (denoted as* T_0*).* T_0 *is measured in natural time units rather than generations and is given by* $T_0 = Lk_0$*. In Section 7.2, we shall see more discussions on* T_0*.*

REMARK 4.2. *The value of* k_0 *obtained by solving Eq.(4) is the time at which the* total system capacity *can satisfy all subsequent requests. Strict server-peer transition occurs when the* peer capacity *alone is sufficient to serve all requests. Let us denote the strict transition time as* K_0*. After time* k_0*, system capacity grows linearly with rate* λLb *rather than exponentially. Therefore, we have* $N\left(1+\frac{\alpha}{b}\right)^{k_0} - N + \lambda Lb(K_0 - k_0) = \lambda Lb$ *and*

$$K_0 = k_0 + \frac{N}{\lambda Lb} = \frac{\lg(\lambda Lb) - \lg N}{\lg(1+\frac{\alpha}{b})} + \frac{N}{\lambda Lb}.$$

As server capacity is small compared to the requested load, the difference between k_0 *and* K_0 *is trivial. We focus our discussions on* k_0 *in the remainder of this paper.*

4.0.1 Dynamics of reject rate

As mentioned earlier, there is no unique way to quantify the reject rate as it depends on the size of time window (Δx) we use. Let us first discuss the scenario when window size is much smaller than L. An observation under such circumstances is: in early streaming periods, the majority (if not all) of streaming sessions start at the very beginning of that period. This is due to the heavy load put to the system: bandwidth is quickly utilized and all subsequent requests in that period will be rejected. As a result, the reject rate is close to zero at the beginning and reaches almost 100% until the end of the current streaming period. The above pattern repeats itself in every streaming period. However, within each cycle, the time when the reject rate is low becomes longer as system capacity grows. At period k_0, the reject rate for the whole streaming period will be low. In other words, k_0 can be viewed as the time after which no more fluctuations of reject rate can be observed. One thing to point out is that the fluctuations of reject rate can be smoothed out if we choose larger window sizes.

We also study the method to estimate reject rate at any time k. See Appendix B for details.

5. MULTI-FILE SYSTEM

In the previous section, we derived explicit expressions for the system capacity $C(k)$ and the server-peer transition time k_0 for a single-file system. In deriving these expressions, we used Eq.(1) to capture the increase in number of qualified peers in two consecutive time intervals. However, we cannot directly use Eq.(1) to study the dynamics of a multi-file system because of the interactions among peers holding and/or requesting more than one file. To clarify, consider a peer that has received a file f_1 in the past, i.e., it is considered a qualified peer for f_1. If that peer requests and receives another file f_2, it should not be counted as a qualified peer for both files because it may not have enough streaming capacity to serve both files at the same time. Intuitively, the rate of increase of the number of qualified peers will be smaller in a multi-file system than in a single-file system. Another problem is: when the growth of file-specific capacity are not well synchronized, we could also have long transition time. In this section, we analyze a multi-file system in which all files have the same length L and the same bit rate b. We first consider a simplified multi-file system model, for which we derive the optimal (i.e., shortest) server-peer transition time k_0 for the system and the conditions to achieve this optimal value. We then study a general multi-file system by analyzing the impact of the assumptions made in the simplified model.

In the simplified multi-file system, we divide the whole system into F virtual subsystems, each of which deals with only one file. Each individual subsystem is assigned a fixed share N_f of the total server bandwidth N. In other words, we divide the server capacity into F private channels. Naturally, each subsystem has its own request rate λ_f. Immediately, we have

$$\sum_{f=1}^{F} N_f = N, \text{ and } \sum_{f=1}^{F} \lambda_f = \lambda. \tag{6}$$

We further assume that the one-file subsystems are independent, i.e. a peer that has acquired file f will request no other files and remains a qualified peer of subsystem f forever. The whole system can then be viewed as F independent subsystems sharing the total server bandwidth N. The simple model differs from the original model by two factors: private channeling of the server bandwidth to individual files, and lack of interactions among subsystems. We discuss the effects of these factors in Section 5.1 and Section 5.2, respectively.

It is easy to see that the growth of each subsystem capacity follows Eq.(1) with N replaced by N_f and λ by λ_f. Therefore, the server-peer transition time for any single-file subsystem (denoted as $k_{0,f}$) can be obtained from Eq.(5) as follows:

$$k_{0,f} = \frac{\lg(\lambda_f Lb) - \lg N_f}{\lg(1+\frac{\alpha}{b})}. \tag{7}$$

Eq.(7) shows that the server-peer transition time in each subsystem depends only on the bandwidth allocation (N_f) and the per-file request rate (λ_f). Now we need to derive the system-level server-peer transition time k_0 from those of the subsystems. We can easily see that different allocations of server bandwidth may result in different server-peer transition times. Instead of deriving general expressions for k_0 as a function of N_f, we concentrate on the bandwidth allocations that lead to the optimal k_0. The problem of finding such allocation(s) can be formally stated as: for each media file f, how much server bandwidth is to be assigned (N_f) given the request rate of that file (λ_f) such that the system-level transition time (k_0) is minimized. One important observation is that the system-level transition time is the maximum value among those of all subsystems. This is because the whole system reaches the transition point only when all subsystems reach theirs. The problem can be further interpreted as an optimization subject to the constraints represented by Equations (6) and (7), with the objective function

$$\text{minimize} \max_{1 \le f \le F} \{k_{0,f}\}.$$

It is well-known [6] that the solution for the above optimization is obtained when all $k_{0,f}$ are the same, i.e.,

$$k_0 = k_{0,1} = k_{0,2} = \cdots = k_{0,F}.$$

Applying Eq.(7) to the above solution, we get

$$\frac{\lambda_1 Lb}{N_1} = \frac{\lambda_2 Lb}{N_2} = \cdots = \frac{\lambda_F Lb}{N_F} = \frac{\sum_{i=1}^{F} \lambda_i Lb}{\sum_{i=1}^{F} N_i} = \frac{Lb\lambda}{N}.$$

Hence for any file f, the optimal choice of N_f is

$$N_f = \frac{\lambda_f}{\lambda} N, \qquad f = 1, 2, \ldots, F. \tag{8}$$

In other words, the share of server bandwidth assigned to each single-file subsystem has to be proportional to the request rate of that file to achieve optimal k_0 at the system level. Now we can derive k_0 from Eq.(7) and Eq.(8):

$$k_0 = \frac{\lg(\lambda Lb) - \lg N}{\lg(1 + \frac{\alpha}{b})}. \tag{9}$$

Note that the above equation is the same as Eq.(5). From Eq.(9) we can also get the number of qualified peers for file f at time k_0:

$$P_f(k_0) = \frac{N_f}{\alpha}\left(\frac{\lambda_f Lb}{N_f} - 1\right) = \frac{\lambda_f Lb}{\alpha} - \frac{N_f}{\alpha}, \tag{10}$$

which is independent of M, and the same for fixed N_f, λ_f.

5.1 Optimality of the Original System Model.

The above result is important since it shows that the simple model is optimal in terms of server-peer transition time when the bandwidth allocation follows Eq.(8). Let us go back to the original system model. In this model, no private channels are assigned to individual files. Instead, requests come at random and can be admitted to any server channel that is available. We call this *statistical multiplexing* of server capacity. This makes N_f a random variable instead of a constant as in the modified model. For any file f, we model the request arrivals as a Poisson process with rate λ_f. The following theorem shows that the original system model is stochastically optimal.

THEOREM 5.1. *In a multi-file hybrid streaming system that performs statistical multiplexing of server capacity, the expectation of the server bandwidth utilized in streaming file f is* $E[N_f] = \frac{\lambda_f}{\lambda} N$*, which is the same as the optimal bandwidth allocation given by Eq.(8).*

PROOF. The server bandwidth can be viewed as $\frac{N}{b}$ channels, each of which can serve a streaming session. The F file-specific request streams can be viewed as a single Poisson stream with a fixed rate $\lambda = \sum_{f=1}^{F} \lambda_f$. Channel holding time for all requests is a constant L and requests do not stay in a waiting queue. With all these conditions, it follows that the CDN servers in our streaming system can be mapped to an Erlang loss system with $\frac{N}{b}$ service lines, arrival rate of λ, and service rate $\frac{1}{L}$ [7].

In the aggregate stream, the probability that a single request is to file f is $\frac{\lambda_f}{\lambda}$. According to well-established results in queuing theory ([7], page 84), the probability of rejection (blocking) is the same for all file-specific streams in such systems. Therefore, for any non-blocked request, the probability that it is to file f is still $\frac{\lambda_f}{\lambda}$. Consider any $\frac{N}{b}$ consecutive non-blocked requests in the aggregate stream, the number of requests to file f can be denoted as a random variable X_f, which follows a binomial distribution $B(\frac{N}{b}, \frac{\lambda_f}{\lambda})$. Therefore, the bandwidth consumed by file f is $N_f = bX_f$. It follows that

$$E[N_f] = bE[X_f] = b\frac{N}{b}\frac{\lambda_f}{\lambda} = \frac{\lambda_f}{\lambda}N.$$

□

From Theorem 5.1, we see that since $E[N_f]$ gets the optimal bandwidth given by Eq.(8), the server-peer transition time for the statistical multiplexing system will approximately achieve the optimal k_0 value in Eq.(9). The above proof requires familiarity with Erlang systems, a self-contained proof based on probability density functions of exponential distributions can be found in Appendix A. Note that Theorem 5.1 still holds true when we allow requests to wait in a queue, no matter what the queue size is. Here we skip the proof as it is not the model we use in this paper.

Theorem 5.1 only shows the expectation of N_f without considering the effects of the variance of N_f on k_0. We can use standard statistical tools to estimate k_0 with the consideration of such variances. First of all, the variance of N_f is

$$\sigma^2 = b^2\frac{N}{b}\frac{\lambda_f}{\lambda}\left(1 - \frac{\lambda_f}{\lambda}\right) = \frac{bN\lambda_f}{\lambda}\left(1 - \frac{\lambda_f}{\lambda}\right).$$

We now can analyze N_f using confidence intervals. [10] If the random variable N_f falls into an 95% interval, say, $N_f \in (E[N_f] - 0.05\sigma, E[N_f] + 0.05\sigma)$, we get

$$N_f \ge E[N_f] - 0.05\sigma = \frac{\lambda_f}{\lambda}N - 0.05\sigma.$$

By Eq.(7), we thus have

$$k_0 \le \frac{\lg(\lambda_f Lb) - \lg(\frac{\lambda_f}{\lambda}N - 0.05\sigma)}{\lg(1 + \frac{\alpha}{b})}.$$

This result gives an upper bound for k_0 taking the variance of randomly distributed N_f into consideration. Since the effects of σ on k_0 are diluted by the log function, k_0 is not sensitive to the variance of N_f. As long as σ is relatively large (i.e., $\frac{bN\lambda_f}{\lambda}$ is not so small), k_0 is very close to our result in Eq.(7). Otherwise, any small deviation will be out of the 95% confidence interval and the analysis fails.

5.2 Dependence Among Subsystems

In the analysis of a multi-file system, we assume that different subsystems grow independently. However, interactions exist among these virtual subsystems in the original model. As described earlier in this section, the problem comes from those peers that access more than one media object. In computing the server-peer transition time, we focus on the growth of system capacity in terms of bandwidth. Every time a peer obtains a certain media file from

[10] Since N is relatively large, we could also use the F-distribution to estimate N_f. However, the estimates from the F-distribution may not be as tight as those obtained using confidence intervals.

the CDN servers or supplying peers, it will be included as a qualified peer in the virtual subsystem related to that file. This is equivalent to counting the same peer multiple times on the whole-system level. It is difficult to quantify such interactions among media files. In this section, we give an upper bound of the level of these interactions. The following analysis shows that k_0 is only slightly larger than the value given by Eq.(5).

We assume that the probability of requesting any file f is the same for all peers. That is, any request to file f comes uniformly from all M potential clients, i.e., there is no tendency that a peer already holding file A has a better chance to ask for file B. Let us go back to the analysis of multi-file systems, reconsider Eq.(1), and take the peer interactions into account, we have:

$$P_f(k+1) = P_f(k) + \beta_{k,f}\left(\frac{N_f}{b} + P_f(k)\frac{\alpha}{b}\right), \qquad (11)$$

where $\beta_{k,f}(0 < \beta_{k,f} \leq 1)$ is a coefficient for the 'valid' proliferation in the subsystem of file f. This means that $\beta_{k,f}$ is the probability that peers in this subsystem hold only file f during time interval k. We call such peers as *valid peers* of file f. If a peer holds other media file(s) when it acquires file f, it is called an *invalid peer* of media f. Suppose at time k, $P_f(k)$ is the number of valid peers. By the assumption of uniformly-distributed requests, the probability of getting a request from a peer that holds another file g is at most $\frac{P_g(k)}{M}$. Since g could be any of the $F-1$ files other than f, the total probability of having an invalid peer (denoted by $\delta_{k,f}$) is

$$\delta_{k,f} \;\leq\; \sum_{g\neq f}\frac{P_g(k)}{M},$$

which captures the portion of peers that should not be counted as contributors in the subsystem of file f. At any time up to k_0, Eq.(10) gives an upper bound for the number of valid peers in any subsystem g, and we have

$$\sum_{g\neq f} P_g(k) \leq \sum_{g=1}^{F}\frac{\lambda_f Lb}{\alpha} - \frac{\lambda_f N}{\alpha\lambda}\left(\frac{\lambda_f Lb}{N} - 1\right) \leq \frac{\lambda Lb}{\alpha} - \frac{N}{\alpha},$$

which is independent of the total peer population M. Thus, $\delta_{k,f} \leq \frac{\lambda Lb - N}{M\alpha}$. Now, the probability of having valid peers for file f is $\beta_{k,f} = 1 - \delta_{k,f} \geq 1 - \frac{\lambda Lb - N}{M\alpha}$. Note that if the pool size of peers is large enough, i.e., the request rate is small compared to M, we have $\beta_{k,f} \geq 1 - \frac{\lambda Lb - N}{M\alpha} \approx 1$. From Eq.(11), we get

$$P_f(k+1) \geq P_f(k) + \left(1 - \frac{\lambda Lb - N}{M\alpha}\right)\left(\frac{N_f}{b} + P_f(k)\frac{\alpha}{b}\right).$$

Following the same procedures as in the derivation of Eq.(9) and Eq.(10) and setting $\beta = 1 - \frac{\lambda Lb - N}{M\alpha}$, we obtain

$$P_f(k) \geq \frac{N_f}{\alpha}\left[(1 + \frac{\alpha\beta}{b})^k - 1\right],$$

which immediately leads to

$$k_0 \leq \frac{\lg(\lambda Lb) - \lg N}{\lg(1 + \frac{\alpha\beta}{b})}.$$

6. IMPACT OF PEER FAILURE

P2P systems are intrinsically dynamic [29]. A major difference between a peer and a server is that a peer's commitment to the community is not guaranteed: it may leave the network at any time. In this section, we study the effects of peer failure on the capacity growth of our media streaming system. Peer failure in our analysis means that a peer leaves the system permanently [2]. We start by presenting a simple peer failure model that is based on experimental studies performed by other researchers. Then we present a general peer failure model that considers an arbitrary distribution for peer lifespan.

6.1 Simple Model for Peer Failure

In this failure model, we assume that at the end of each streaming period (i.e., generation), the number of surviving qualified peers is proportional to the number of surviving qualified peers at the beginning of that streaming period. The proportionality factor is called the *survival rate* and is denoted by γ ($\gamma < 1$). We discuss how to determine γ later in this section.

At generation $k+1$, the number of inherited qualified peers from generation k is $\gamma P(k)$. Consider the case for the single-file system, Eq.(1) becomes

$$P(k+1) = \gamma P(k) + \frac{N}{b} + \gamma P(k)\frac{\alpha}{b}, \qquad 0 \leq k \leq k_0. \quad (12)$$

Rewriting the above equation, we have

$$P(k+1) + \frac{N}{b\theta} = \left(P(k) + \frac{N}{b\theta}\right)\gamma\left(1 + \frac{\alpha}{b}\right),$$

where θ is the new capacity growth factor and $\theta = \gamma(1 + \frac{\alpha}{b}) - 1 \neq 0$.
Then Eq.(2) and Eq.(5) become

$$P(k) = \frac{N}{b\theta}\left[\gamma^k\left(1 + \frac{\alpha}{b}\right)^k - 1\right] = \frac{N}{b}\cdot\frac{\gamma^k(1+\frac{\alpha}{b})^k - 1}{\gamma(1+\frac{\alpha}{b}) - 1} \quad (13)$$

and

$$k_0 = \frac{\lg\left(\frac{b(\gamma-1)+\gamma\alpha}{\alpha\gamma}\left(\frac{\lambda Lb}{N} - 1\right) + 1\right)}{\lg\gamma(1+\frac{\alpha}{b})}, \qquad (14)$$

respectively. Note that Eq.(2) and Eq.(5) are special cases of Eq.(13) and Eq.(14) when $\gamma = 1$. To guarantee positive growth of the system capacity, we must have $\theta > 0$ and therefore $\gamma > \frac{b}{\alpha+b}$.

Once we have Eq.(14), we can follow the same analysis as in Section 5 to derive the upper bound for multi-file systems and get $\frac{\lambda_1 Lb}{N_1} = \frac{\lambda_2 Lb}{N_2} = \cdots = \frac{\lambda_F Lb}{N_F} = \frac{\sum_{i=1}^F \lambda_i Lb}{\sum_{i=1}^F N_i} = \frac{Lb\sum_{i=1}^F \lambda_i}{N} = \frac{Lb\lambda}{N}$. That is, we have the optimal choice of N_f as $N_f = \frac{\lambda_f}{\lambda}N$ ($f = 1, 2, \ldots, F$), which is the same as in Eq.(8). Thus, we get the same equation as Eq.(14) for the system-level transition time of a multi-file system with peer failures. Once we have the above equations, the other results in Section 5 can be derived accordingly.

REMARK 6.1. *Since $\gamma < 1$, from Eq.(14), we have*

$$k_0 = \frac{\lg\left(\frac{b(\gamma-1)+\gamma\alpha}{\alpha\gamma}\cdot\frac{\lambda Lb}{N} - \frac{b(1-\gamma)}{\alpha\gamma}\right)}{\lg\gamma(1+\frac{\alpha}{b})} < \frac{\lg\left(\frac{b(\gamma-1)+\gamma\alpha}{\alpha\gamma}\cdot\frac{\lambda Lb}{N}\right)}{\lg\gamma(1+\frac{\alpha}{b})}.$$

If γ is close to 1, then

$$\frac{\lg \frac{\lambda L b}{N}}{\lg(1+\frac{\alpha}{b})} \le k_0 \le \frac{\lg\left(\frac{b(\gamma-1)+\gamma\alpha}{\alpha\gamma}\cdot\frac{\lambda L b}{N}\right)}{\lg \gamma(1+\frac{\alpha}{b})} \approx \frac{\lg \frac{\lambda L b}{N} + \frac{b(\gamma-1)}{\alpha\gamma}}{\lg(1+\frac{\alpha}{b}) + \lg \gamma},$$

The dominant part is $\frac{\lg \frac{\lambda L b}{N}}{\lg(1+\frac{\alpha}{b})}$*, hence* k_0 *is not very sensitive to* γ*. Therefore, we can still use* $\frac{\lg \frac{\lambda L b}{N}}{\lg(1+\frac{\alpha}{b})}$ *to estimate* k_0*.*

REMARK 6.2. *In practice,* γ *can be factored into to* $\frac{\alpha}{b}$*, as* β *in Section 5.2. In [34], this idea was used to analyze a single-file system.*

6.1.1 Computing the survival rate γ

To determine the survival rate γ, we associate with each qualified peer a random variable X to model its *lifespan.* Assuming peers fail independently, then the survival rate γ can be interpreted as the conditional probability:

$$\gamma = Pr\{X \ge T + L \mid X > T\}, \tag{15}$$

where T is the starting time of any streaming period k. Generally, it is difficult to solve Eq.(12) using Eq.(15) when the peer lifespan follows an arbitrary statistical distribution. The reason is that the above probability for any individual peer depends on its age T. If we consider all living peers at streaming period k as a whole, γ is determined by the *age structure* of all $P(k)$ peers. In other words, γ is a variable that is related to k. Two large-scale measurement studies analyzed the lifespan of peers ([29] and [5]). According to [29], the lifespan of peers approximately follows an exponential distribution. Since the exponential distribution is memoryless, the probability for any peer to live beyond time $T+t$ given it is alive at time T is e^{-tq}, where $1/q$ is the average lifespan of all peers. Thus, in our case, $\gamma = e^{-Lq}$.

The measurement study in [5] indicates that a long-tail Pareto distribution is a better fit for the lifespan data collected from more than 500,000 peers. Since Pareto distribution is not memoryless, computing γ from Eq.(15) is not easy. However, it is shown in [10] that long-tail distributions can accurately be approximated by a weighted sum of a small number of exponential distributions over a finite time interval. The maximum peer lifespan can be assumed to be a sufficiently large (e.g., 100 days) but finite value for all practical purposes. Therefore, we can approximate the distribution of the peer lifespan as $\sum_{i=1}^{m} w_i e^{-\mu_i t}$ where the parameters m, w_i and μ_i are estimated using the recursive algorithm described in [10]. Using this approximate distribution for peer lifespan in Eq.(15), the survival rate is given by: $\gamma = \sum_{i=1}^{m} w_i e^{-\mu_i L}$.

6.2 General Model for Peer Failure

The previous section presented a peer failure model for two commonly conceived distributions for peer lifespan. Although these two distributions are corroborated by extensive measurement studies, they are system and environment specific. Therefore, they may not be applicable to other P2P systems, or their accuracy may degrade under different conditions. In this section, we develop a more general peer failure model that is not restricted to particular peer lifespan distributions.

With a general lifespan distribution, we lose the nice feature of representing the survival probability within any time interval as a constant. What we can do is to divide the continuous *peer age* (i.e., the length of time since a peer becomes online) into discrete *age classes* and associate a survival probability p with each age class. In our analysis, we use age classes with length L. Without loss of generality, we ignore peers with age more than m, and $m < k_0$. We denote the survival probability from age k to age $k+1$ as p_k. Suppose we obtain the probability density function of peer lifespan $f(x)$ from measurement studies. Based on Eq.(15), we have

$$p_k = \frac{\int_{k+1}^{\infty} f(x)dx}{\int_{k}^{\infty} f(x)dx}.$$

The overall survival rate γ at time k is thus determined by the *age structure* of all $P(k)$ qualified peers, i.e.,

$$\gamma = [x_0, x_1, \ldots, x_m][p_0, p_1, \ldots, p_m]^T,$$

where x_i is the percentage of peers of age i among the $P(k)$ qualified peers.

Following the same strategy as in Section 4, we develop a model to project the number of qualified peers from time k to $k+1$. Define $n_{x\,k}$ as the number of peers with age x at generation k, where $x \ge 0$, and $k \ge 0$ are integers. We have the following relations between different age groups in the single-file system:

$$\frac{N}{b} + \frac{\alpha}{b}\sum_{x=0}^{m} n_{x\,k} = n_{0\,k+1}, \qquad k = 0, 1, \ldots, k_0 - 1,$$

$$p_x n_{x\,k} = n_{x+1\,k+1}, \quad x = 0, 1, \ldots, m-1, \ k = 0, 1, \ldots, k_0.$$

The first relation shows that the number of qualified peers of age 0 at time $k+1$ is the sum of those generated by servers and by peers of all other age classes. The second relation shows that the peers of age $x+1$ at time $k+1$ are those of age x at time k and survived through time period k with probability p_x. Employing matrix notation, the system capacity can be expressed as

$$\vec{B} + \mathbf{A}\vec{n}_k = \vec{n}_{k+1}, \tag{16}$$

where

$$\vec{B} = \begin{bmatrix} \frac{N}{b} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ \vec{n}_k = \begin{bmatrix} n_{0\,k} \\ n_{1\,k} \\ \vdots \\ n_{m\,k} \end{bmatrix}, \ \vec{n}_{k+1} = \begin{bmatrix} n_{0\,k+1} \\ n_{1\,k+1} \\ \vdots \\ n_{m\,k+1} \end{bmatrix},$$

and

$$\mathbf{A} = \begin{bmatrix} \frac{\alpha}{b} & \frac{\alpha}{b} & \cdots & \frac{\alpha}{b} & \frac{\alpha}{b} \\ p_0 & 0 & \cdots & 0 & 0 \\ 0 & p_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & p_{m-1} & 0 \end{bmatrix}.$$

Note that the above formula has a similar form as Eq.(1). Set $p_{(k)} = p_0 p_1 \cdots p_k \quad (k \ge 0)$ and $p_{(-1)} = 1$. We also define the following column vector

$$\vec{v} = \frac{N}{-b + \alpha \sum_{k=0}^{m} p_{(k-1)}} \left[1, p_{(0)}, p_{(1)}, \ldots, p_{(m-1)}\right]^T.$$

We explain more about this vector later. For vector $\vec{v}$, it can be verified that $\mathbf{A}\vec{v} - \vec{v} = \vec{B}$. Plugging it into Eq.(16), we obtain

$$\vec{n}_{k+1} + \vec{v} = \mathbf{A}\left(\vec{n}_k + \vec{v}\right), \quad \vec{n}_0 = \vec{0}. \tag{17}$$

Solving the above equation, we get:

$$\vec{n}_k = (\mathbf{A}^k - I)\vec{v}, \quad k = 1, 2, \ldots, k_0, \tag{18}$$

where I is the $(m+1) \times (m+1)$ identity matrix. This can be viewed as a matrix counterpart of Eq.(2). We now can see that the purpose of introducing vector $\vec{v}$ is to get the nice form shown in Eq.(17) such that $\vec{n}_k$ can be solved as an exponential function of k.

Now let us discuss under what conditions the system capacity grows positively. For convenience, we let $\psi = -1 + \frac{\alpha}{b}\sum_{k=0}^{m} p_{(k-1)}$, which immediately gives

$$\vec{v} = \frac{N}{b\psi}\left[1, p_{(0)}, p_{(1)}, \ldots, p_{(m-1)}\right]^T. \tag{19}$$

Obviously, the elements in $\vec{v}$ should all be positive to achieve positive capacity growth. Therefore, we must have $\psi > 0$. We may also see that, to guarantee positive growth, the largest eigenvalue of $\mathbf{A}$ should be greater than 1. According to the Perron-Frobenius theorem in linear algebra [12], since the matrix $\mathbf{A}$ is nonnegative and irreducible, it has at least one positive eigenvalue and the largest such eigenvalue is greater than or equal to the modulus of any of its complex roots. Furthermore, according to Descartes' rule of signs, there is only one such real eigenvalue.

To find the server-peer transition time k_0 (assuming the above conditions hold such that the capacity grows positively), we need to solve

$$\frac{N}{b} + \frac{\alpha}{b}\sum_{x=0}^{m} n_{x\,k_0} = \lambda L,$$

or equivalently, solve

$$\frac{N}{b} + \frac{\alpha}{b}(\mathbf{A}^{k_0} - I)\vec{v} = \lambda L.$$

Divide both sides by $\frac{N}{b}$, and recall Eq.(19), we obtain

$$1 + \frac{1}{\psi}\frac{\alpha}{b}(\mathbf{A}^{k_0} - I)\left[1, p_{(0)}, p_{(1)}, \ldots, p_{(m-1)}\right]^T = \frac{\lambda b L}{N}. \tag{20}$$

Since the system grows exponentially, the left-hand side of Eq.(20) is proportional to the increased population of the system, it is an increasing function of k_0, and we will have a unique k_0 satisfying Eq.(20). Unfortunately, we could not find a simple explicit expression for k_0. Instead, we can calculate k_0 numerically. Moreover, since the right-hand side depends on the ratio of the bandwidth and the request rate rather than their absolute values, the solution of k_0 will depend only on this ratio. Hence, if we generalize the single-file solution to multi-file systems, where each media object has it own request rate λ_f, we will have the same result as in Eq.(8), i.e. the optimal bandwidth is proportional to the request rate.

7. OTHER EXTENSIONS

7.1 Acceleration of Capacity Growth

In the previous sections, we assume that a peer can only serve others after it finishes receiving the entire stream. In practice, it is feasible to allow peers to start serving others after receiving the first few blocks of the media file. Formally, we set a delay d ($d \leq L$) after which a requesting peer in a streaming session can serve as a supplying peer in a new session. In this section, we study how the inter-session delay d affects system performance.

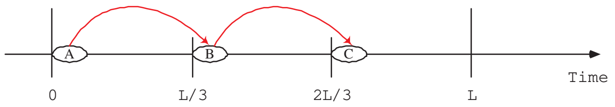

Figure 5: Streaming sessions initiated in a generation when $d = L/3$.

For simplicity, we only consider the situation of $d = L/n$ where n is a positive integer. When n is not an integer, we can estimate k_0 using the two neighboring integers of n. Revisiting the arguments used to generate Eq.(1) in Section 4, an important observation is: within one streaming period, a newly started session will have the chance to start $n-1$ rounds of new sessions. Figure 5 shows an example of $n = 3$ and we use cluster A to represent the chunk of sessions initiated at the beginning of a streaming period. The requesting peer in a session in cluster A could be a supplying peer in a new session that starts in the beginning of time interval $[\frac{L}{3}, \frac{2L}{3}]$ (cluster B). The requesting peer of a cluster B session can in turn be a supplying peer in a session in cluster C. On average, one session in cluster A can give rise to $\frac{\alpha}{b}$ sessions in cluster B and $\frac{\alpha}{b} \cdot \frac{\alpha}{b} = \frac{\alpha^2}{b^2}$ sessions in cluster C. Knowing this, the number of sessions initiated in the period shown in Figure 5 is $|A| + |B| + |C| = |A|(1 + \frac{\alpha}{b} + \frac{\alpha^2}{b^2})$ where $|A|$, $|B|$, $|C|$ are the numbers of sessions in cluster A, B, and C, respectively. Generalizing this idea, Eq.(1) can be written as:

$$P(k+1) - P(k) = \left(\frac{N}{b} + P(k)\frac{\alpha}{b}\right)\Phi_n\,,$$

$$\Phi_n = 1 + \frac{\alpha}{b} + \frac{\alpha^2}{b^2} + \cdots + \frac{\alpha^{n-1}}{b^{n-1}}\,.$$

with $0 \leq k \leq k_0$, and $P(0) = 0$. Similar to the derivation of Eq.(2), we obtain

$$P(k) = \frac{N}{\alpha}\left[\left(1 + \frac{\alpha}{b}\Phi_n\right)^k - 1\right].$$

The server-peer transition time becomes

$$k_0 = \log_{(1+\frac{\alpha}{b}\Phi_n)}\left(\frac{\lambda L b}{N}\right) = \frac{\lg(\lambda L b) - \lg N}{\lg(1 + \frac{\alpha}{b}\Phi_n)}. \tag{21}$$

Comparing this with Eq.(5), we see that the improvement is significant because the system capacity growth factor increases to $1 + \frac{\alpha}{b}\Phi_n$ while other factors remain unchanged. A smaller delay d leads to a smaller k_0 value. However, this does not mean it is always better to choose a smaller d (i.e., a larger n) value. Since we have $\frac{\alpha}{b} < 1$ in practice, k_0 converges rapidly as n increases: it is well-known that for $\Phi_n = \frac{1-(\frac{\alpha}{b})^n}{1-\frac{\alpha}{b}}$, we have $\lim_{n\to\infty}\Phi_n = \frac{b}{b-\alpha}$. It follows that we get

$$\lim_{n\to\infty} k_0 = \frac{\lg(\lambda L b) - \lg N}{\lg(1 + \frac{\alpha}{b-\alpha})}.$$

From the point of view of streaming protocol design, an excessively short delay is also infeasible: there will inevitably be some delays in setting up the network connections, and some buffer time is needed for maintaining QoS in any streaming sessions.

7.2 General Multi-file System

In the analysis of multi-file systems (Section 5), we assume that all media files require the same bandwidth b and have the same length L. Our analysis can be generalized to media files with different bandwidth requirements and streaming lengths. Assume that a media file f requires bandwidth b_f, and streaming length L_f. Then the proliferation of each subsystem capacity follows Eq.(1) with N replaced by N_f and b by b_f. Therefore, the server-peer transition time for any single-file subsystem ($k_{0,f}$) can be obtained from Eq.(5) as:

$$k_{0,f} = \frac{\lg(\lambda_f L_f b_f) - \lg N_f}{\lg(1 + \frac{\alpha}{b_f})} . \tag{22}$$

Note that the time unit for every subsystem is its streaming length L_f, so that the objective function of the optimization problem becomes:

$$\text{minimize} \max_{1 \le f \le F} \{L_f k_{0,f}\} . \tag{23}$$

If we allow $k_{0,f}$ to have continuous solutions, the solution of the above optimization problem is achieved when $L_f k_{0,f}$ are equal to each other for each f. Recall that T_0 is the optimal transition time counted as natural time units, we have $T_0 = k_{0,f} L_f$, $\forall f$, which is the same as

$$\begin{aligned} T_0 &= \frac{\lg(\lambda_1 L_1 b_1) - \lg N_1}{\lg(1 + \frac{\alpha}{b_1})} L_1 = \frac{\lg(\lambda_2 L_2 b_2) - \lg N_2}{\lg(1 + \frac{\alpha}{b_2})} L_2 \\ &= \cdots = \frac{\lg(\lambda_F L_F b_F) - \lg N_F}{\lg(1 + \frac{\alpha}{b_F})} L_F \end{aligned}$$

where $N_1, N_2, \cdots, N_F$ are the unknowns. With the condition $\sum_{f=1}^{F} N_f = N$, we could solve $N_1, N_2, \cdots, N_F$ by iteration methods. That is, we first solve T_0 by representing N_f through T_0 and plugging it into the sum condition. Since $N_f = \lambda_f L_f b_f \left(\frac{b_f}{b_f+\alpha}\right)^{T_0/L_f}$, we have

$$\sum_{f=1}^{F} \lambda_f L_f b_f \left(\frac{b_f}{b_f + \alpha}\right)^{T_0/L_f} = N.$$

The left-hand side of the above equation monotonically decreases about T_0. This means the equation has a unique root. There are many quick iteration techniques to solve T_0 such as the bisection method, Newton's method, and the secant method, details of which can be found in general numerical analysis texts such as [4]. Note our original system model is not optimal under such conditions therefore we have to assign private channels to files based on the solution of the above equation to achieve the shortest T_0.

REMARK 7.1. *The above analysis can be further generalized to the scenario where each media file attracts a different group of peers with different bandwidth contributions (i.e., the quantity α in Eq.(22) is replaced by a file-specific item α_f). Under such changes, T_0 can still be solved by the same iteration methods. Furthermore, this generalization can also be applied to analysis in Section 6.2.*

8. EXPERIMENTAL RESULTS

We study the dynamics of the proposed hybrid media streaming system by extensive simulations. We implement our media streaming simulator using the Tool Command Language (TCL)[11].

8.1 System parameters

Unless specified otherwise, the example system contains a pool of 200,000 peers ($M = 200,000$) and 100 media objects ($F = 100$). The playback bit rate for all video objects is $b = 800Kbps$ and length is one hour ($L = 3,600$, basic time unit is *second*). Bandwidth contribution of peers is: 5% of the peers with 800Kbps, 10% with 400Kbps, 55% with 200Kbps, and 30% with 100Kbps, which translates into an $\frac{\alpha}{b}$ value of 0.275. System receives requests at a rate (λ) of 1 request per second. Total server bandwidth is 480Mbps. This is roughly the bandwidth of 10 T3 lines and covers 600 concurrent streaming sessions in our case.

8.2 Dynamics of Single-File Systems

In Figure 6, various metrics of the simulated system are plotted. After a short initial stage, server bandwidth usage (Figure 6a) is close to the maximum value all the time. We plot reject rates resulted from two window sizes, 8000 seconds and 1000 seconds, in Figure 6c. For the one with smaller window size, reject rate fluctuates between zero and one. These fluctuations almost disappear in the experiment using windows of size 8000. For both experiments, the reject rate stays at zero after 8.06 hours. According to Section 4.0.1, the time point 8.06 can be regarded as the k_0 value for this experiment as no fluctuations occur afterwards. We also test windows with other sizes but the same k_0 value is observed. In Figure 6c we also plot reject rate calculated by the Erlang B formula and our experimental data are very close to the theoretical values.

The growth of system capacity is illustrated by the change of total number of qualified peers (Figure 6b) and bandwidth contributions of these peers (Figure 6d). Both peer number and peer bandwidth show geometric growth at the first 8 hours and linear growth afterwards. One thing to point out is that the curves for system capacity growth are not smooth. They tend to appear as step functions of time, as illustrated by the small graph in Figure 6. The height of steps increases as time goes by, thus an exponential growth is achieved. From Figure 6d we can also see that the system capacity reaches the requested bandwidth at about the 8th hour, which confirms our conclusion about transition time drawn from reject rate data. Peer bandwidth usage first increases and then stabilizes (after transition point) at exactly the same level with the requested bandwidth.

8.2.1 Model validation

To verify the validity of our analytical model, k_0 values obtained from simulations are compared with theoretical values derived from our analysis (Table 2). Each observed value is the mean of four experiments. Although the difference between theoretical value and simulation result is small in all cases, we also see that the observed values tend to be greater than the theoretical ones. The reasons for this are: (i) the time we report in Table 2 are in natural time units (hour). In practice, there are delays between the release and re-occupation of bandwidth so the actual streaming length is longer than L; (ii) as system capacity is not a smooth function of time, the fractional part of k_0 given by Eq.(5) is

[11] http://tcl.sourceforge.net

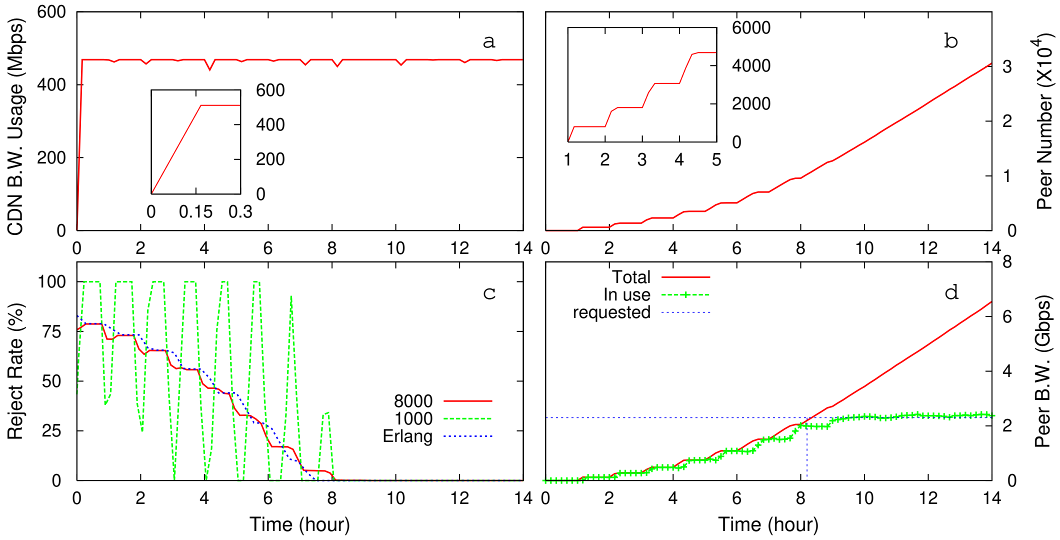

Figure 6: Performance of a typical hybrid media streaming system. a. Bandwidth usage of CDN servers; b. Number of qualified peers; c. System reject rate; d. Peer capacity.

Table 2: Theoretical and experimental values of k_0 under different scenarios

Parameters		Transition Time	
α/b	λ(reqs/s)	*Calculated* (h)	*Observed* (h)
0.125	1	15.2	16.11
0.275	1	7.38	8.14
0.5	1	4.41	5.11
0.275	10	16.8	17.27
0.275	2	10.23	11.08
0.275	0.5	4.52	5.06

not accurate. We should take its ceiling integer in interpreting the theoretical value of k_0; (iii) Only integer number of new qualified peers will be generated each cycle thus some bandwidth will not be utilized.

8.2.2 *Effects of peer bandwidth contribution and request rate*

The impact of parameters $\frac{\alpha}{b}$ and λ on performance is also investigated. Figure 7a and 7b show the reject rate and total peer bandwidth under different choices of α while all other parameters remain unchanged. The legends in Figure 7a and Figure 7b indicate the $\frac{\alpha}{b}$ values of individual simulations. A six-fold increase of $\frac{\alpha}{b}$ (from 0.125 to 0.75) significantly shortened the server-peer transition time from 16.0 hours to 4.0 hours. This shows that k_0 is almost linearly related to $\frac{\alpha}{b}$ (recall Remark 4.1).

A similar set of experiments are designed to study the impact of system request rate (λ) on k_0. The results for four request rates, 0.5, 1.0, 2.0, and 4.0 requests per second, are shown in Figures 7c and 7d. The value of k_0 observed increases as the request rate increases. However, the change of k_0 due to the change of λ is less dramatic than that caused by the change of $\frac{\alpha}{b}$. When λ increases 20 times to 10 requests/second, a k_0 of 17.27 hours was obtained (data not plotted). The effects of streaming length (L) and bitrate (b) are similar to those of request rate (data not shown).

8.3 Performance of Multi-file Systems

As specified in Section 5, the theoretical value of the server-peer transition time k_0 in a multi-file environment is the same for a single-file system. Figure 8 shows how the system performs under different F values. First of all, we can see that the results for experiments with total file number 1 and 100 are almost identical. In this set of experiments, the observed k_0 value does not change until the total number of files goes beyond 120. Note that the number of media files is 100 for all the simulations whose results are presented in Table 2.

However, k_0 is found to be greater than the ideal value when F further increases (Fig 8a). According to Section 5, two factors could account for the long transition time in a multi-file system: (i) peers that acquired multiple files, and (ii) lack of synchronization in the growth of per-file capacity. We investigate the effects of the first factor in the same set of experiments by recording the storage usage of qualified peers. In Table 3, the number of qualified peers at transition time is listed by their storage consumption. For example, there are 11268 peers holding one media file (valid peer) and 31 peers holding two files for the simulation with 250 files. The β values in the last column are ratios of the number of valid peers to all qualified peers, which can be viewed as the lower bound of $\beta_{k,f}$ in Eq.(11). All β values shown are very close to 1.0. Another conclusion we may draw from Table 3 is that the space contribution of peers can be made minimal without affecting system performance.

Now it is clear that factor (ii) above accounts for the degraded performance. According to Section 5.1, when F is large, $\frac{bN\lambda_f}{\lambda}$ is small and k_0 deviates from theoretical value. In other words, the average number of sessions allocated to each file ($\frac{N}{Fb}$) cannot be too small. Another way to interpret this is: when $\frac{N}{Fb}$ is too small (e.g. $F = 500$ in Fig 8), there is no guarantee that each file can occupy at least one server channel. Hence, the files take turns in using the server band-

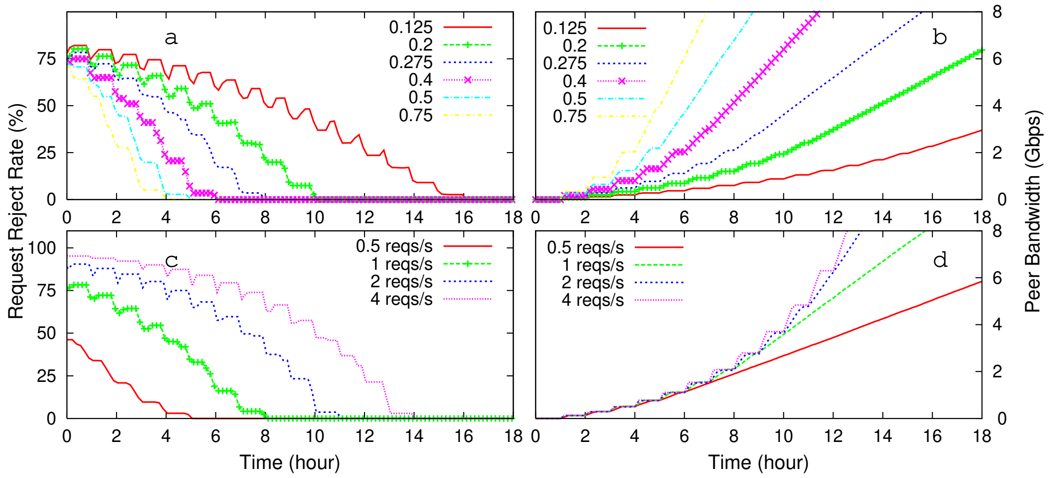

Figure 7: System performance under different capacity growth factors (a, b) and request rates (c, d). a,c: Smoothed system reject rate; b,d: Peer bandwidth.

Table 3: Storage usage of peers at transition time

Experiment	k_0 (h)	# of media files stored				β
		1	2	3	4	
$F = 1$	8	10895	0	0	0	1.000
$F = 50$	8	10791	21	0	0	0.998
$F = 100$	8	10678	21	0	0	0.998
$F = 250$	9	11268	31	0	0	0.997
$F = 500$	11	13808	89	0	0	0.994
$F = 1000$	13	14896	196	1	0	0.987

width and transition is delayed. For the above experiments, we see that $\frac{N}{Fb}$ has to be at least 5 for the system to get near-optimal transition time.

8.3.1 *Impact of access patterns*

All previous experiments were based on a uniform access model where the access rates of all files are the same. However, requests are more likely to skew toward a subset of the media objects [1] and this helps our system achieve earlier transition. We test the effects of various file access patterns on the system performance. In Fig 9, two skewed access models are studied: the b/c model where c% of the requests are made to b% of the media objects [19], and the famous Zipf model. Specifically, we test the b/c model with two sets of parameters (20/80 and 10/90) and one Zipf model with power 1.0. [12]. The total number of media is 2000 for all four experiments. According to Figure 9, systems with the skewed access patterns accomplish server-peer transition significantly earlier than that with the uniform model. Among them, the most skewed one (the 10/90 model) achieves the smallest k_0. The Zipf model reached the transition point later than both 20/80 and 10/90 models. Our interpretation for this is: under a skewed access pattern, most files are infrequently accessed. When the frequently-accessed files finish transition, the system reject reaches zero since the servers are able to handle requests to the less popular files.

[12] A list of references on Zipf's Law can be found in http://linkage.rockefeller.edu/wli/zipf

8.4 Systems with Peer Failures

The introduction of finite peer lifespans significantly reduces the speed of system proliferation (Figure 10). We experiment with three simulations with different average peer lifespan – eight, six and four hours. In all tests, peer lifespan is exponentially distributed. A system with no peer failures (i.e. infinite peer lifespan) is used as control. As the average lifespan of peers increases, the reject rate drops more dramatically (Figure 10a) and the system accomplishes server-peer transition faster. For the system with average lifespan of 4.0 hours, the peers fail too early to serve other peers so that it never reaches a transition point. The above results are confirmed by capacity growth of all tested systems plotted in Figure 10b. The system with longer average peer lifespan grows faster than those with shorter lifespan. For the case of lifespan four hours, the system never reaches the required number of qualified peer to service all coming requests. For the systems simulated, the calculated threshold value of survival rate γ to guarantee positive capacity growth is 0.7843, which also means the peers should have an average lifespan of at least 4.12 hours. This explains why the capacity of the system with average lifespan of 4 hours does not grow.

8.5 Service Acceleration

We verify our conclusions about service acceleration (Section 7.1) by allowing requesting peers to act as supplying peers before the whole media stream is delivered. We test inter-session delays with different values (from $L/5$ to $L/2$, and L as the control). From Figure 11, we can see that the usage of inter-session delays shorter than L does accelerate the server-peer transition: k_0 decreases from 8.0 to about 6.0 when delay changes from L to $L/2$. Further decrease of k_0 can also be observed when d gets even smaller. However, this effect on k_0 quickly diminishes: the results of $d = L/4$ and $d = L/5$ are almost identical. We test different sequences of random inputs and very similar results are obtained and the k_0 values observed match closely to the theoretical value given by Eq.(21)(data not shown).

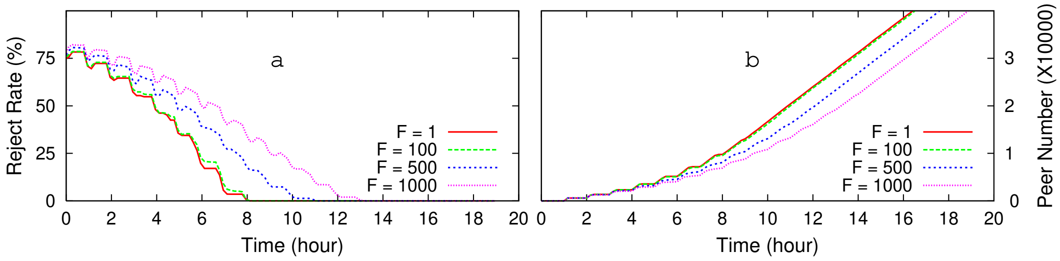

Figure 8: System performance under different number of media files. a. Smoothed system reject rate; b. Number of qualified peers.

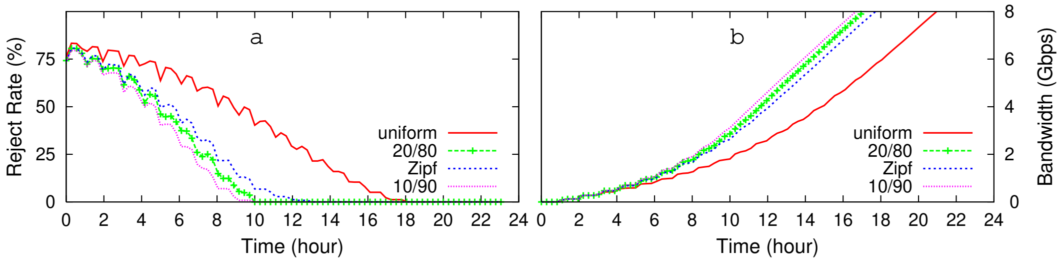

Figure 9: Effects of file access pattern on system dynamics. a. Smoothed system reject rate; b. Peer bandwidth

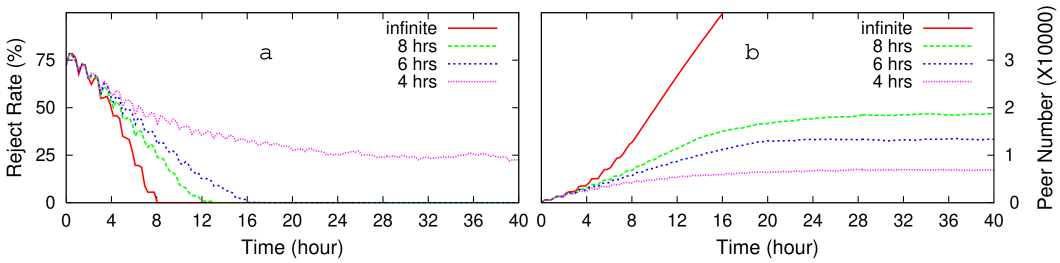

Figure 10: Effects of peer failure on system dynamics. a. Smoothed system reject rate; b. Total number of qualified peers.

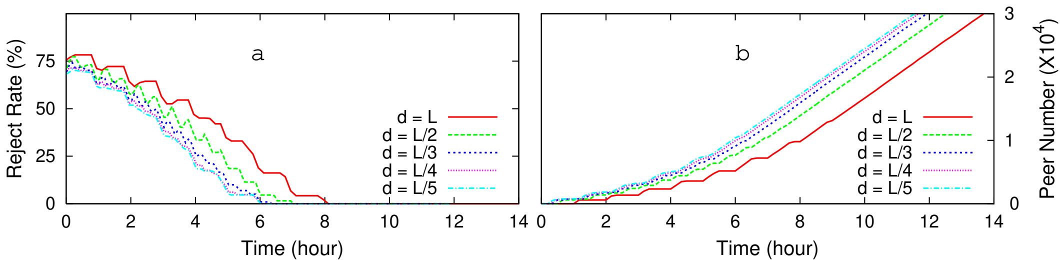

Figure 11: System under different inter-session delays. a. Smoothed system reject rate; b. Total number of qualified peers.

9. CONCLUSIONS AND FUTURE WORK

In this paper, we studied the capacity growth of P2P media streaming systems using a discrete-time analytical model. The capacity is defined as the total streaming bandwidth available from both servers and peers that previously acquired the media files. Based on the analytical model, we found that the capacity of such systems increases exponentially with time. Knowing the exact pattern of the system capacity growth enabled us to determine the moment at which the streaming load can be shifted from servers to peers. We obtained explicit expressions to determine this moment, which we call the server-peer transition time k_0. The server-peer transition time can be used by the system operator to decide when to reallocate server resources. We analyzed the capacity and server-peer transition time for single- and multi-file streaming systems. We showed that the equations derived for single-file systems can approximate the behavior of multi-file systems, within some boundary conditions.

We extended our analysis to quantify the effects of peer failures on the system performance. In particular, we model the unreliability and limited commitment of peers to the system by a *lifespan* random variable. We derived explicit expressions for the server-peer transition time using two commonly-known lifespan distributions in the literature: exponential and Pareto. Furthermore, we considered a general peer failure model in which the lifespan could follow any arbitrary distribution. Although we did not obtain explicit expressions in this case, we showed how the solution can be computed using simple numerical methods. Our analysis gives explicit expressions of performance metrics on most of the scenarios considered. Otherwise, numerical solutions are obtained. In addition, we conducted extensive simulation experiments which: (i) validated our analytical conclusions, and (ii) studied the effects of changing several parameters, e.g., request rate, average peer bandwidth contribution, and average peer lifespan on system performance.

Finally, our results from the analysis and the simulation experiments leads to better understanding of the operation of P2P media streaming systems. To that end, we have the following comments and suggestions for designers of P2P media streaming systems:

1. The system performance is most sensitive to the capacity growth factor, which is the average peer bandwidth contribution (α) divided by the bandwidth required to stream the media file (b). Therefore, we should concentrate on maintaining a large $\frac{\alpha}{b}$ value. One idea is to give higher priorities to peers with higher bandwidth contributions. Specifically, we could reserve some server bandwidth for high-capacity peers to enable them to get early admission to the system;

2. Peers receiving a media file should be allowed to serve others before receiving the entire file. However, a receiving peer does not need to become a serving peer too early in the session. As a rule of thumb: a delay of about one-third of the session length would yield better performance in terms of faster system capacity growth rates;

3. Peer failure negatively affects the system capacity by decreasing the capacity growth factor. Thus, maintaining a sufficiently large P2P community is the key to success. Incentive mechanisms could be used to encourage peers to stay longer in the system.

4. System performance can deteriorate when too many media files are introduced in the system. To some extent, this may be mitigated by synchronizing the growth of file-specific subsystems using private channels, especially when files are of different lengths and bitrates.

This study can be extended in several directions. For example, it could be interesting to analyze the strategies proposed above (e.g., fast track channels for high-capacity peers and providing incentives for peers to stay longer) and how these strategies enhance system performance. Since the focus of our analysis was on the early stages of system operation, another extension can be studying the system dynamics after the server-peer transition point. At the individual peer level, we may need to design file replacement policies especially when peers request many files and have limited storage capacity. It is important to design these replacement policies with minimal or no global information about other peers in the system. Furthermore, the impact of these policies on the system performance need to be studied. Other possible research directions include QoS management and handling security and integrity concerns in P2P media systems.

10. REFERENCES

[1] M. F. Arlitt and C. L. Williamson. Internet Web Server: Workload Characterization and Performance Implications. *IEEE/ACM Transactions on Networking*, 5(5):631–645, 1997.

[2] R. Bhagwan, S. Savage, and G.M. Voelker. Understanding availability. In *Proceedings of 2nd International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2003.

[3] A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal, O. Spastcheck, and W. Sturm. CDN brokering. *Journal of Computer Communications*, 25(4):393–402, March 2002.

[4] Richard Burden and J. D. Faires. *Numerical Analysis*. Brooks/Cole Publishing, 2001.

[5] Fabian Bustamante and Yi Qiao. Friendships that last: Peer lifespan and its role in p2p protocols. In *Proceedings of International Workshop on Web Content Caching and Distribution*, September-October 2003.

[6] Edwin K. P. Chong and Stanislaw H. Żak. *An Introduction to Optimization*. John Wiley & Sons, New York, 2001.

[7] Robert B. Cooper. *Introduction to Queueing Theory*. North Holland, New York, 1981.

[8] J. Crowcroft and I. Pratt. Peer to Peer: peering into the future. In *Networking Tutorials*, pages 1–19, May 2002.

[9] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proceedings of ACM Symposium on Operating Systems Principles*, pages 202–215, October 2001.

[10] Anja Feldmann and Ward Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze

network performance models. In *Proceedings of IEEE INFOCOM*, pages 1096–1104, April 1997.

[11] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A global internet host distance estimation service. *IEEE Transactions on Networking*, 9(5):525–540, October 2001.

[12] F. R. Gantmacher. *Applications of the Theory of Matrices*. Interscience, New York, 1959.

[13] P. Golle, K.Leylton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. In *Proceedings of ACM Conference on Electronic Commerce (EC)*, pages 264–267, October 2001.

[14] M. Hefeeda, B. Bhargava, and D. Yau. A hybrid architecture for cost-effective on-demand media streaming. *Journal of Computer Networks*, 44(3):353–382, 2004.

[15] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. Promise: Peer-to-peer media streaming using collectcast. In *Proceedings of ACM Multimedia*, pages 45–54, November 2003.

[16] B. Horne, B. Pinkas, and T. Sander. Escrow services and incentives in peer-to-peer networks. In *Proceedings of ACM Conference on Electronic Commerce (EC)*, pages 85–94, October 2001.

[17] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Rihard, S. Rollins, and Z. Xu. Peer-to-Peer Computing. Technical Report HPL-2002-57, HP Labs, 2002.

[18] T. Nguyen and A. Zakhor. Distributed Video Streaming over Internet. In *Proceedings of SPIE/ACM MMCN*, pages 186–195, January 2002.

[19] M. Nicola and M. Jarke. Performance Modeling of Distributed and Replicated Databases. *IEEE Trans. Knowledge and Data Engineering*, 12(4):645–672, July/August 2000.

[20] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of NOSSDAV*, pages 177–186, May 2002.

[21] E. C. Pielou. *Mathematical Ecology*. John Wiley & Sons, New York, 1976.

[22] L. Qiao and S. Qiao. A robust and efficient algorithm for evaluating Erlang B formula. Technical Report CAS98-03, Department of Computing and Software, McMaster University, Canada, 1998.

[23] Dongyu Qiu and R. Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *Proceedings of ACM SIGCOMM*, pages 367–377, August 2004.

[24] K. K. Ramachandran and B. Sikdar. An analytical framework for modeling peer-to-peer networks. In *Proceedings of IEEE INFOCOM*, March 2005.

[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM*, pages 161–172, August 2001.

[26] M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network. *IEEE Internet Computing Journal*, 6(1):50–57, 2002.

[27] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[28] A. Rowstron and P. Druschel. Storage management in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of ACM SOSP*, pages 188–201, October 2001.

[29] S. Saroiu, K. P. Gummadi, and S. D. Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts. *Springer/ACM Multimedia Systems*, 9(2):170–184, August 2003.

[30] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM*, pages 149–160, August 2001.

[31] D. A. Tran, K. A. Hua, and T. T. Do. ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. In *Proceedings of IEEE INFOCOM*, pages 1283–1292, April 2003.

[32] Yi-Cheng Tu, Jianzhong Sun, and Sunil Prabhakar. Performance analysis of a hybrid media streaming system. In *Proceedings of SPIE/ACM MMCN*, pages 69–82, January 2004.

[33] D. Wu, Y. T. Hou, W. Zhu, Y-Q. Zhang, and J. M. Peha. Streaming Video over the Internet: Approaches and Directions. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(1):365–386, February 2001.

[34] D. Xu, H-K. Chai, C. Rosenberg, and S. Kulkarni. Analysis of a hybrid architecture for cost-effective streaming media distribution. In *Proceedings of SPIE/ACM MMCN*, January 2003.

[35] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On peer-to-peer media streaming. In *Proceedings of IEEE ICDCS*, pages 363–371, July 2002.

[36] Xiangying Yang and Gustavo de Veciana. Service capacity of peer to peer networks. In *Proceedings of IEEE INFOCOM*, pages 2242–2252, March 2004.

APPENDIX

A. A SELF-CONTAINED PROOF OF THEOREM 5.1

PROOF. This proof is based on Lemma A.1, which shows that at any time a CDN server channel becomes vacant, the probability that it will be occupied by a request to file f is $\frac{\lambda_f}{\lambda}$. Then the number of channels used by file f follows a binomial distribution and the same arguments in the previous proof in Section 5.1 hold true. $\square$

LEMMA A.1. *Suppose we have F independent Poisson streams with arrival rates $\lambda_1, \lambda_2, \ldots, \lambda_F$. At any time t, the probability that the first event that arrives after t is from stream f $(f = 1, 2, \ldots, F)$ is $\frac{\lambda_f}{\lambda_1+\lambda_2+\cdots+\lambda_F}$.*

PROOF. We start the proof by considering the case of $F = 2$: there are two Poisson streams with parameters λ_1 and λ_2. We know that the inter-arrival time of a Poisson stream follows an exponential distribution with the same parameter. As Poisson processes are memoryless, we can denote the arrival times of the two streams after any time point t as X_1 and X_2. The joint distribution of X_1 and X_2 is

Table 4: Theoretical reject rate at k_0

Offered Load	10	50	100	500	1000	5000	10000
Rejection Probability	.2146	.1048	.0757	.0349	.0248	0.0112	.0079

$F(x_1, x_2) = \lambda_1 e^{-\lambda_1 x_1} \lambda_2 e^{-\lambda_2 x_2}$. Without loss of generality, the probability that the next event comes from stream one is

$$\begin{aligned} Pr\{X_1 < X_2\} &= \int_0^{+\infty} \int_{x_1}^{+\infty} \lambda_1 \lambda_2 e^{-(\lambda_1 x_1 + \lambda_2 x_2)} dx_2 dx_1 \\ &= \int_0^{+\infty} \lambda_1 \lambda_2 e^{-\lambda_1 x_1} \left(\int_{x_1}^{+\infty} e^{-\lambda_2 x_2} dx_2 \right) dx_1 \\ &= \int_0^{+\infty} \lambda_1 e^{-(\lambda_1 + \lambda_2) x_1} dx_1 \\ &= \frac{\lambda_1}{\lambda_1 + \lambda_2} \int_0^{+\infty} (\lambda_1 + \lambda_2) e^{-(\lambda_1 + \lambda_2) x_1} dx_1 \\ &= \frac{\lambda_1}{\lambda_1 + \lambda_2}. \end{aligned}$$

The above results can be easily generalized to n ($n > 2$) streams. For arrival times $X_1, X_2, \ldots, X_n$, the probability of X_1 being the smallest one can be obtained from the above solution:

$$Pr\left\{X_1 < \min\{X_2, X_3, \ldots, X_n\}\right\} = \frac{\lambda_1}{\lambda_1 + \sum_{i=2}^{n} \lambda_i} = \frac{\lambda_1}{\lambda}.$$

In the above reasoning we use the following result: the random variable $X^* = \min\left\{X_2, X_3, \ldots, X_n\right\}$ follows a exponential distribution with parameter $\lambda_2 + \lambda_3 + \cdots + \lambda_n$. This is easy to obtain as follows:

$$\begin{aligned} Pr\{X^* > x\} &= Pr\{X_2 \geq x\} Pr\{X_3 \geq x\} \cdots Pr\{X_n \geq x\} \\ &= e^{-\lambda_2 x} e^{-\lambda_3 x} \cdots e^{-\lambda_n x} \\ &= e^{-(\lambda_2 + \lambda_3 + \cdots + \lambda_n) x}. \end{aligned}$$

$\square$

B. ESTIMATION OF REJECT RATE

At the end of any period k, the system has a total bandwidth of $N(1 + \frac{\alpha}{b})^k$, meaning the number of concurrent sessions it can support is

$$s = \frac{N}{b} \left(1 + \frac{\alpha}{b}\right)^k.$$

Given this, the instantaneous reject rate can be calculated. As requests are rejected immediately without waiting in a queue, our streaming system can be modeled as an Erlang loss system [7] with s service lines, an arrival rate of λ, and service rate of $\frac{1}{L}$. The probability of request rejection (blocking) is given by the *Erlang B formula* as:

$$B(s, a) = \frac{a^s / s!}{\sum_{i=0}^{s} a^i / i!} \tag{24}$$

where $a = \lambda L$ is called the *offered load* of the system. There are many efficient algorithms [22] developed for evaluating the Erlang B formula.

An obstacle in using the Erlang loss model in our analysis is that system capacity is time-variant in our model while a fixed s value is assumed in the Erlang model. However, the applicability of Erlang model in our analysis can be justified by the following observation: in each streaming period, the majority (if not all) of streaming sessions start at the very beginning of that period. This is due to the heavy load put to the system. As a result, the system capacity jumps to a higher level at the beginning and stays unchanged for the rest of the streaming period. In other words, the system capacity is more like a step function than a smooth exponential function of time. Therefore, we can safely map the system to an Erlang loss system within each streaming period. Some of the values of $B(s, a)$ under the condition $s = a$ (exactly what happens at k_0) are listed in Table 4. We can see that the expected reject rate is very close to zero when s or a is large (what we expect in a busy streaming service). We verify our estimations of reject rate by Erlang B in Figure 6.

ACKONWLEDGEMENTS

This work is supported by NSF grant IIS-9985019. The authors would like to thank Dr. Dongyan Xu, from whose seminar class offered in Purdue University this research is initiated, and the anonymous reviewers in MMCN for their helpful comments. We would also like to thank Mr. Shan Lei, who shared with us many valuable insights at the early stage of this research, and Mr. Gang Shen, for his contributive advises.