Performance Analysis of a Hybrid Media Streaming System

Yi-Cheng Tu[a], Jianzhong Sun[b] and Sunil Prabhakar[a]

[a]Department of Computer Sciences, Purdue University, 1398 Computer Science Building, West Lafayette, IN 47907-1398 USA

[b]Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067 USA

ABSTRACT

Recent research efforts have demonstrated the promising potential of building cost-effective media streaming systems on top of peer-to-peer (P2P) networks. A P2P media streaming architecture can reach large size and streaming capacity that are 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 behaviors 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 by solving an optimization problem. We also show that the system model achieves optimal allocation of server bandwidth among different media objects. The unpredictable departure/failure of peers is a critical factor that affects performance of P2P systems. To model peer failures in our system, we propose the concept of *peer lifespan*. The original equation is enhanced with coefficients generated from the distribution of peer lifespan. Results from large-scale simulations support our analysis.

Keywords: Peer-to-peer networks, Content Distribution Networks, hybrid system, media streaming, mathematical analysis

1. INTRODUCTION

Multimedia streaming over the Internet has become a reality with the development of efficient media compression methods, high-throughput storage systems and broadband networking technology. Attractive applications such as entertainment video-on-demand, digital library, and on-line news service built on top of real-time media streaming service are now available for the public. However, there are still many challenges towards building cost-effective, robust and scalable multimedia streaming systems[1] due to the high bandwidth, loss and delay requirements for media streaming.

A majority of media streaming architectures follows a server/client design. The server deployed by service providers acts as the streaming entity and clients controlled by the service users as passive receivers of media streams. In a large streaming system where user requests arrive at a high rate, the server has to support a large number of concurrent streaming sessions, each of which has its own bandwidth and QoS requirements. Thus, the total capacity of the system is limited to the out-bound bandwidth of the server. Multiple servers or proxies can be deployed on the edge of the Internet to increase total system capacity. In this design, media content are replicated on these proxies and clients receive streaming data from the closest proxy. There are two advantages of using proxies: user requests are handled by all proxies with a combined capacity greater than the single-server architecture; increased QoS (latency, loss) in streaming due to the shortened packet delivery path. Such systems are sometimes called Content Distribution Networks (CDNs).[2]

The cost of maintaining a CDN is extremely high considering the massive CPU power, storage space and output bandwidth each CDN server possesses. More servers have to be deployed with the increase of popularity

Further author information: (Send correspondence to Y.Tu)
Y. Tu: E-mail: tuyc@cs.purdue.edu, Telephone: 1 765 494 5005
J. Sun: E-mail: jsun@math.purdue.edu, Telephone: 1 765 430 1775
S. Prabhakar: E-mail: sunil@cs.purdue.edu, Telephone: 1 765 494 6008

of the service. One approach to solve the above problem is motivated by the emerging concept of peer-to-peer (P2P) computing.[3–5] In a P2P system, there is no centralized entity controlling the behaviors of each peer. Instead, each peer contributes its share of resources and cooperates with other peers following some predefined rules for communication and synchronization. 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 media. One of the nice features of P2P streaming systems is that its total capacity grows when the content it manages becomes more popular.[6] This is the most important difference between peer-to-peer and the server/client strategies. To some extent, a P2P architecture can be viewed as an extreme case of 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.[7, 8] As compared to a P2P-only architecture, the hybrid streaming system can disseminate media content faster and respond more quickly to requests. In media streaming systems, the bottle-necking resource is found to be the bandwidth.[9] Some of the less bandwidth-consuming operations such as directory management and searching are better processed in a centralized server for efficiency reasons. Furthermore, due to their robustness, servers also serve as great backup resource providers even when the P2P network has enough capacity. Peers are heterogeneous in the duration of their commitment to the community[10]: each peer could leave or fail at any time. How to minimize the effects of this come-and-go behavior is a research topic that has attracted a lot of attention. Our previous work[11] shows that the use of servers effectively enhanced the system's ability to recover from in-session peer failures.

The focus of this research is to study the features of a media streaming architecture by mathematical modeling. We are primarily interested in the pattern of capacity growth in the streaming model and the effects of various factors on the growth. Conclusions drawn from such analysis can improve our understanding of system dynamics of the proposed streaming architecture and provide guidelines for the design and realization of media delivery services based on the hybrid architecture. In this paper, we first propose a generic media streaming model that utilizes both a CDN and P2P network. The model differs from those found in Xu et al.[7] and Hefeeda et al.[8] in a number of ways such that it applies to more general environments but the complexity of quantitative analysis becomes lower. Based on this model and a discrete-time analysis approach, an exponential growth equation for the system capacity is derived. The same equation also gives the threshold time when the P2P network takes over all the streaming load. We extend the results from a single-file system to a multi-file system by transforming the analysis into an optimization problem. It is also shown that our streaming architecture achieves near-optimal performance in terms of the time needed for a complete load hand-over from servers to peers. Finally, the factor of peer failures is introduced into the mathematical model. We propose a new approach to model failures by associating a 'lifespan' value to each peer. To our knowledge, this is by far the most sophisticated and complete work on mathematical analysis of P2P multimedia systems. We also evaluate several performance metrics by simulation, the results of which confirm the validity of the quantitative analysis.

The major contributions of this paper are: 1. Based on a generic hybrid media streaming model, we derive an equation to describe the system capacity growth. 2. We accomplish a quantitative analysis on performance of hybrid streaming systems with multiple media objects. 3. We prove the optimality of our streaming architecture in terms of server bandwidth allocation and server-peer transition time. 4. As an enhancement to the above equation, we use *peer lifespan* to study the effects of peer failures and obtain exact solutions for system performance under such effects.

This paper continues with Section 2 by introducing the streaming model. Section 3 is dedicated to system analysis of the proposed model. Section 4 presents the experimental results based on extensive simulations. Section 5 summarizes additional related work. We conclude the paper with Section 6.

2. THE HYBRID STREAMING MODEL

Our analysis is based on the media streaming infrastructure shown in Figure 1. The model is similar to the hybrid structure proposed in Xu *et al.*[7] and Hefeeda *et al.*[8] with some modifications. The main entities of the system are:

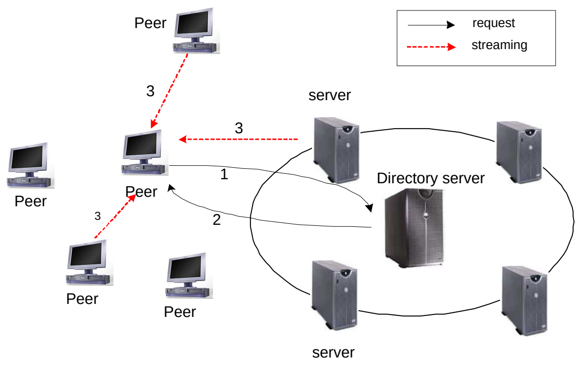

Figure 1. The architecture of the hybrid streaming system.

- **Directory Server.** The role of the directory server is to maintain peer/media information. It is also responsible for processing the user queries.
- **Server**[*]**.** It holds a copy of all media files and is responsible for streaming when the requested media cannot be serviced through the P2P network. Each server has a fixed bandwidth contribution. We assume a zero downtime for the servers.
- **Peer (client).** The set of user machines participating in the streaming system. The peer asking for a media object is called *requesting peer* and those that have acquired any media object(s) are called *qualified peers*. 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, the peers will admit any requests forwarded to it when it has available bandwidth. We do not specify the maximum number of streaming sessions a peer can support. The reason is: most peers have limited bandwidth contribution[10] and can only support less than one session in practice. Peers can be divided into a number of classes by their bandwidth contribution.
- **Media content.** The target resource a client requests. We can view it in the form of media files. To simplify analysis, we assume they are all Constant Bit Rate (CBR) media streams.

2.1. System Operations

When a peer requests a media object, it first sends out a query to the directory server (step 1 in Fig 1). The latter searches its local database for peers that have the media and returns a list of possible *supplying peers* to the *requesting peer* (step 2 in Fig 1). When no *supplying peers* are available, CDN servers are chosen. If CDN servers are also busy, the request is rejected. The *requesting peer* chooses from the list of supplying peers a subset that satisfies the bandwidth and QoS requirements and streaming starts (step 3 in Fig 1). When the streaming is successful, the *requesting peer* becomes a *qualified peer* of that media object. More details of the system model can be found in our previous work.[11]

3. MATHEMATICAL ANALYSIS

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 equation derived from the simple model by loosening these assumptions. In the initial analysis, we make the following assumptions:

1. The system contains only one media file. In Section 3.4 the analysis is extended to a multi-file system, in which all media files are of the same streaming length (time) and bit rate .

[*]In this paper, the terms ‘streaming server’, ‘CDN server’ and ‘server’ are used interchangeably.

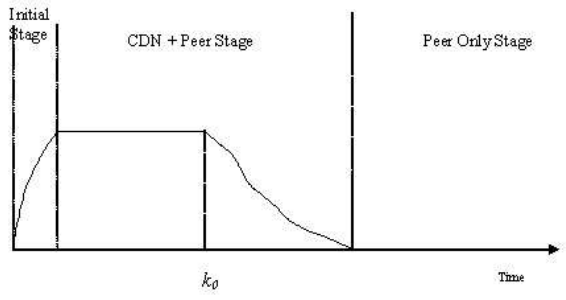

Figure 2. The bandwidth usage of the CDN servers in the hybrid system.

2. The bottleneck link for a streaming session can only be the upload link of the data sender (CDN server or supplying peer).
3. Each participating peer has infinite storage contribution. We will see in later sections that only minimal storage is actually needed from each peer.
4. Peers never fail. A model taking peer failure into account is addressed in Section 3.6.
5. Requests are uniformly distributed among the peer population.

3.2. Metrics and Notations for Analysis

Peer number and *peer bandwidth* are direct measures of system capacity. Our analysis is performed focusing on the change of these two metrics. Figure 2 illustrates the use of server bandwidth in a hybrid streaming system. Initially 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 peers. After that, the servers are increasingly alleviated from streaming tasks until the system reaches a stage where only peers are needed for streaming (Peer-only stage). It is important to find the time when the servers' streaming load starts to decrease monotonically. This time point is called *server-peer transition time* (denoted as k_0). Another metric for system capacity is the *reject rate*. The *instantaneous reject rate* decreases as system capacity increases and reaches 0 at k_0. The reject rate at any time x is defined as the ratio of total number of rejected requests to the total number of requests within the time interval $[x - \delta x, x]$. Suppose the mapping from time to reject rate results in a differential function $f(x)$, the instantaneous reject rate is simply given by $f'(x)$.

Symbols used in the analysis are listed below:

Symbol	Definition	Symbol	Definition
L	Length of one streaming session	k	Discrete time index, each with length L
N	Total server bandwidth	M	Total peer population
n	Number of peer classes	p_i	Percentage of peers in the i–th class
λ	Request rate to the system	c_i	Bandwidth contribution per peer of i–th class
k_0	Server-Peer transition time	$P(k)$	Number of usable peers at time period k
F	Total number of media files	b	Bandwidth required to stream a file

3.3. Capacity Growth of Single File System

We first focus on a system with only one media file. The expansion of our streaming system resembles the population growth of a biological species. The latter is generally studied by the number of offspring produced in generation(s). The same idea may be applied to our system analysis since the requesting peers can be regarded as the offspring of supplying peers and/or servers. This discrete-time analysis approach was used in Xu et al.[7]

and our analysis will follow this strategy. Starting from the end of the initial stage, the number of qualified peers produced between two consecutive generations k and $k+1$ can be expressed as:

$$P(k+1) - P(k) = \frac{N}{b} + P(k)\sum_{i=1}^{n} p_i \frac{c_i}{b}, \qquad 0 \le k \le k_0. \tag{1}$$

The two items on the right side of equation (1) are the number of new peers generated by the servers and qualified peers, respectively. For the convenience of analysis, we denote α as the average peer bandwidth contribution (i.e. $\alpha = \sum_{i=1}^{n} p_i\, c_i$). Equation (1) can be rewritten as $P(k+1) + \frac{N}{\alpha} = (P(k) + \frac{N}{\alpha})(1 + \frac{\alpha}{b})$.
Solving the above geometric progression with $P(0) = 0$, we get

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

We name the item $\frac{\alpha}{b}$ in above equations as *Capacity Growth Factor* of the system. The total system capacity at time k is $N + \alpha P(k) = N(1 + \frac{\alpha}{b})^k$, for $k \le k_0$. From the previous discussion we know that the instantaneous reject rate will be zero at time k_0, which also means the total system capacity (bandwidth) is no less than that required to service all the requests generated. The total bandwidth needed to satisfy all requests in L time units is $\lambda L b$. Therefore, we have the following equation to solve k_0: $N(1+\frac{\alpha}{b})^{k_0} = \lambda L b$. By our assumption, N, α, λ, L, b are constants, therefore we get k_0

$$k_0 = \log_{(1+\frac{\alpha}{b})}\left(\frac{\lambda L b}{N}\right) = \frac{\lg(\lambda L b) - \lg N}{\lg(1 + \frac{\alpha}{b})}. \tag{3}$$

From equation (2), we may also claim that $P(k_0) = \frac{N}{\alpha}(\frac{\lambda L b}{N} - 1)$ peers are needed in addition to the servers so that no requests will be rejected.
The above analysis shows an exponential growth model of system capacity. Similar results were found in Xu et al.[7] We improved their analysis by giving an exact solution for the server-peer transition time, which has not been accomplished by any previous work, to the best of our knowledge.

3.4. Multi-file System

We cannot directly use equation (1) to study the dynamics of a multi-file system because it is not clear how the capacity growth is affected by the introduction of more media objects. Intuitively, the rate of increase of the total number of qualified peers will be smaller in a multi-file system than in a single-file system given the same request rate. This is because peers holding multiple files will be counted multiple times in $P(k)$. To make the analysis of a multi-file system feasible, we propose a modified media streaming model that brings some additional features. Our analysis shows that, as a result of the modification, the media streaming system has optimal server-peer transition time. We will then prove that the original model without this modification achieves the same performance statistically.

Suppose the modification is to divide the whole system into F virtual subsystems, each of which deals with only one file. Each individual subsystem is assigned a fixed share of bandwidth N_f out of the total server bandwidth N and receives requests at rate λ_f. Immediately, we have

$$\sum_{f=1}^{F} N_f = N, \text{ and } \sum_{f=1}^{F} \lambda_f = \lambda. \tag{4}$$

The whole system can then be viewed as F independent subsystems sharing the total server bandwidth N, i.e. a peer that has accessed file f will not request any other files and remains a qualified peer of subsystem f forever. In other words, *frequency-sharing multiplexing* of the streaming channels provided by the servers is performed. The independence among file-specific dynamics is the major factor that distinguishes the modified model from the original one. In Section 3.5 we will show that the interactions among file-specific proliferation in our original model are negligible under reasonable assumptions.

It is easy to see that the proliferation of each subsystem's capacity follows equation (1) with N replaced by N_f and λ by λ_f. Therefore, the server-peer transition time for any single-file subsystem ($k_{0,f}$) can be obtained from equation (3) as the following:

$$k_{0,f} = \frac{\lg(\lambda_f Lb) - \lg N_f}{\lg(1 + \frac{\alpha}{b})}. \tag{5}$$

Optimal system-level transition time. Equation (5) 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. Different allocations of the server bandwidth may lead to different server-peer transition time. We first concentrate on the one(s) that gives the optimal value. 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. The reason for this is that the whole system reaches the transition point only when all one-file subsystems reach their own transition points. The problem can be further interpreted as an optimization subject to constraints represented by equations (4) and (5), with object function

$$k_0 = \min \max_{1 \le f \le F} k_{0,f}. \tag{6}$$

The solution for the above problem 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 equation (5) to above, we get $\frac{\lambda_1 Lb}{N_1} = \frac{\lambda_2 Lb}{N_2} = \cdots = \frac{\lambda_F Lb}{N_F} = \frac{\sum_{i=1}^{F} \lambda_i Lb}{\sum_{i=1}^{F} N_i} = \frac{Lb \sum_{i=1}^{F} \lambda_i}{N} = \frac{Lb\lambda}{N}$. Hence for any file f, the optimal choice of N_f is

$$N_f = \frac{\lambda_f}{\lambda} N, \qquad f = 1, 2 \cdots F. \tag{7}$$

In other words, for all single-file subsystems, the share of the server bandwidth N_f assigned has to be proportional to the request rate of that file to achieve optimal Server-Peer transition on the system level. Now, it is easy to derive k_0:

$$k_0 = \frac{\lg(\lambda Lb) - \lg N}{\lg(1 + \frac{\alpha}{b})}. \tag{8}$$

Note the above equation is the same as equation (3). From equation (8) we can also get the number of qualified peers for subsystem f at time k_0:

$$P_f(k_0) = \frac{N_f}{\alpha}\left(\frac{\lambda_f Lb}{N_f} - 1\right) = \frac{\lambda_f Lb}{\alpha} - \frac{N_f}{\alpha}, \qquad f = 1, 2 \cdots F, \tag{9}$$

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

Optimality of original system model. The above result is important since it shows that the modified model is optimal in terms of server-peer transition time when the bandwidth allocation follows Eq (7). However, it is impossible to predict the request rate in practice. According to our system model, requests come at random and are admitted or rejected based on the current bandwidth availability. We call this *statistical multiplexing* of server channels. This makes N_f a random variable instead of a constant. We have the following theorem to show that statistical multiplexing is approximately optimal.

Theorem 3.1 (Statistical Multiplexing of Server Bandwidth). *In a hybrid streaming system with total server bandwidth N and homogeneous streaming bitrate b and duration L for all media files, if the requests to any file f are uniformly distributed in the system waiting queue with rate λ_f, then the expected consumption of server bandwidth for file f is $\frac{\lambda_f}{\lambda} N$.*

Proof: The server can handle $\frac{N}{b}$ requests concurrently. By the assumption that requests are uniformly distributed in the request queue with rate λ_f for file f, the number of outstanding streaming sessions for f follows a binomial distribution $B(N/b, \lambda_f/\lambda)$. Therefore, N_f follows the distribution $b\,B(N/b, \lambda_f/\lambda)$, which is a dilation of a binomial distribution, and $E[N_f] = b\frac{N}{b}\frac{\lambda_f}{\lambda} = \frac{\lambda_f}{\lambda}N$, which is the same as Eq (7). □

From Theorem 3.1, we can see that since $E[N_f]$ gets the optimized bandwidth, the server-peer transition time for the statistical multiplexing system will approximately achieve the optimum value in Eq (8).

More rigorously, we can use standard statistical tools to estimate k_0 for the statistical multiplexing case. Assume the distribution of the request is uniform among the peers, and the per-file streaming bandwidth is b, then the amount of bandwidth allocated to subsystem f is $b\,B(N/b, \lambda_f/\lambda)$, its variance is

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

Since N is relatively large, we could have used the F–distribution to make an estimation. However, the estimates from F–distribution may not be as tight as the following analysis using confidence intervals.

If the random variable N_f falls into an interval, say, $N_f \in (E[N_f] - 0.05\sigma, E[N_f] + 0.05\sigma)$, we have $N_f \geq E[N_f] - 0.05\sigma = \frac{\lambda_f}{\lambda}N - 0.05\sigma$, then from Eq (5), we get $k_0 \leq \frac{\lg(\lambda_f Lb) - \lg(\frac{\lambda_f}{\lambda}N - 0.05\sigma)}{\lg(1+\frac{\alpha}{b})}$. This gives an upper bound of k_0 taking the variance of randomly distributed N_f into consideration and is good only for relatively large $\frac{bN\lambda_f}{\lambda}$. When $\frac{bN\lambda_f}{\lambda}$ is small (< 10, for example), any small deviation will be out of the 95% percent confidence interval and the analysis fails.

3.5. Dependence Among Subsystems

In previous analysis, we made an assumption on the running independence of different subsystems. However, interactions exist among these virtual subsystems in the original model. As described earlier, the problem comes from those peers that access more than one media objects. 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 the CDN servers or supplying peers, it will be included as a qualified peer in the virtual system related to that file. This is equivalent to counting the same peer multiple times on the whole-system level. Given the complexity of the situation, we are unable to quantify the interactions among virtual systems, but we can give a loose (upper) bound of the level of such interactions via analysis. The following shows that k_0 is only slightly larger than that given by equation (3).

We assume that, at any time, the per-client probability of requesting any file f is the same as for all members in the peer community. That is to say, any request to file f comes uniformly from all M potential clients, there is no such tendency that a peer already holding file f_1 has a better chance to ask for file f_2.

Let us go back to the analysis of multi-file system, reconsider the growing equation for peer number in Eq (1), and take the peer-wise interactions into account, we claim:

$$P_f(k+1) = P_f(k) + \beta_{k,f}(\frac{N_f}{b} + P_f(k)\frac{\alpha}{b}), \qquad 0 \leq k \leq k_0 \tag{10}$$

where $\beta_{k,f} (0 < \beta_{k,f} < 1)$ is an coefficient for the "valid" proliferation of the subsystem, or the percentage of the peers holding only file f during time interval $[k, k+1]$. We denote these peers *valid peers* to file f. If a peer holds other media object(s) when it gets file f, it is called a *invalid peer* to f.
Suppose at time k, $P_f(k)$ is the number of valid peers. By the assumption of uniform distribution of requests, the probability of getting a request from a peer that holds a copy of another file g is less than $\frac{P_g(k)}{M}$. Since g could be any of the $F-1$ media files other than f, the total probability of having an invalid peer is bounded by $\gamma_{k,f} = \sum_{g\neq f}\frac{P_g(k)}{M}$, which includes the portion of peers that should not be counted into the contributors of the subsystem accordingly. So the probability of the newly-generated peers containing only file f is $1 - \sum_{g\neq f}\frac{P_g(k)}{M}$. At any time up to time k_0, Eq (9) gives the upper bound for the number of valid peers in subsystem g, 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 M. Thus, $\gamma_{k,f} \leq \frac{\lambda Lb - N}{M\alpha}$. Now if the pool size is large enough, i.e. the request rate is small compared to M, we have $\beta_{k,f} \geq 1 - \frac{\lambda Lb-N}{M\alpha} \approx 1$. Let $\beta = 1 - \frac{\lambda Lb - N}{M\alpha} \approx 1$, from Eq (10), we have $P_f(k+1) \geq P_f(k) + \beta(\frac{N_f}{b} + P_f(k)\frac{\alpha}{b})$. Following the same procedures as in derivation of Eq (8) and (9), we get $P_f(k) \geq \frac{N_f}{\alpha}\left((1+\frac{\alpha\beta}{b})^k - 1\right)$, therefore $k_0 \leq \frac{\lg(\lambda Lb) - \lg N}{\lg(1+\frac{\alpha\beta}{b})}$.

3.6. Failure Model

P2P systems are intrinsically dynamic. In this section, we study the effects of peer failure on the capacity growth of our media streaming system. In our model, each qualified peer is associated with a real number capturing the uptime of this peer. We call this the *lifespan* of the peer. We can use a random variable X to represent peer lifespan. We may also assume, within each unit streaming period, the number of surviving qualified peers is proportional to that at the beginning of the streaming period with a ratio (*survival rate*) $\gamma < 1$. For any streaming period k, the survival rate γ peers can be interpreted as a conditional probability (assuming peers fail independently):

$$\gamma = Pr\{X \geq T + L \mid X > T\} \tag{11}$$

where T is the starting time of the streaming period. Generally, it is difficult to obtain γ from Eq (11) 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 (or, in other words, γ is related to the number of generation k). We denote the survival probability of a peer with age i (in number of generations) as γ_i. 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. If we use a k-dimensional vector to represent the age structure and a k-column vector for the age-specific survival rates, we have

$$\gamma = (x_1, x_2, \ldots, x_k)\begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_k \end{pmatrix}$$

where x_i is the ratio of number of peers that are i generations old to $P(k)$.

The above formula brings additional complexity to our analysis. Fortunately, it is reasonable to assume that the lifespan of peers follows an exponential distribution.[10] As a result, we can use formula (11) to calculate a uniform γ value for all k. Since exponential distribution is memoryless, the probability for any peer to live longer than $T + t$ time units (t is an arbitrary time period) given it is alive at time T is e^{-ts}, where $1/s$ is the average lifespan of all peers. Apply this to Eq (11), we have $\gamma = e^{-Ls}$.

With a fixed survival rate γ, system analysis can be extended with the consideration of peer failures. At the beginning of period $k+1$, the inherited qualified peers from period k is $\gamma P(k)$. Consider this case for the single file system, then Eq (1) becomes

$$P(k+1) = P(k)\gamma + \frac{\gamma N}{b} + P(k)\gamma \sum_{i=1}^{n} p_i \frac{c_i}{b}, \qquad 0 \leq k \leq k_0. \tag{12}$$

All items on the right side of above equation have coefficient γ because we assume a client peer may also fail during a streaming session. Rewriting the equation, get $P(k+1) + \frac{\gamma N}{b\theta} = (P(k) + \frac{\gamma N}{b\theta})\gamma(1 + \frac{\alpha}{b})$, where θ is the new Capacity Growth Factor and $\theta = \gamma(1 + \frac{\alpha}{b}) - 1 \neq 0$, then Eq (2) and Eq (3) become

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

and

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

respectively. We can see Eq (2) and Eq (3) are special cases of Eq (13) and Eq (14) when we take $\gamma = 1$. Since $\gamma < 1$, from Eq (14), we have

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

After having Eq (14), we can follow the same analysis as in Section 3.4 to derive the upper bound for multi-file system without interaction and get $\frac{\lambda_1 Lb}{N_1} = \frac{\lambda_2 Lb}{N_2} = \cdots = \frac{\lambda_F Lb}{N_F} = \frac{\sum_{i=1}^{F} \lambda_i Lb}{\sum_{i=1}^{F} N_i} = \frac{Lb \sum_{i=1}^{F} \lambda_i}{N} = \frac{Lb\lambda}{N}$, i.e. we have the optimal choice of N_f as

$$N_f = \frac{\lambda_f}{\lambda} N, \qquad f = 1, 2 \cdots F, \tag{15}$$

which is the same as in Eq (7). Thus, we have 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 3.4 and 3.5 can be derived.

4. EXPERIMENTAL RESULTS

4.1. System Parameters

We studied the dynamics of the proposed hybrid media streaming system by simulation. Unless specified otherwise, the example system contains a pool of 200,000 peers ($M = 200000$) and 100 media objects ($F = 100$). The playback bitrate for all video objects is $b = 800Kbps$ and length is one hour ($L = 3600$, 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 α/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.

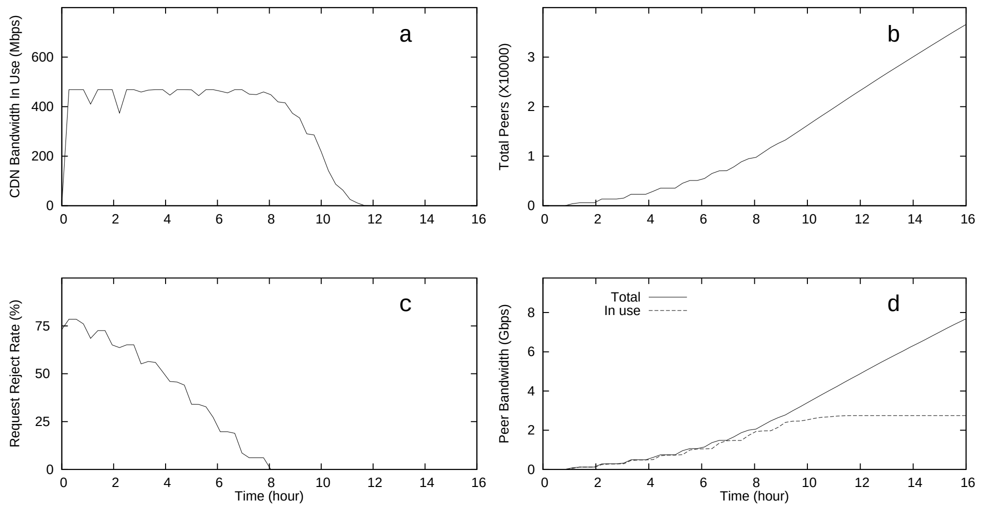

Figure 3. Performance of a typical hybrid media streaming system. a. Bandwidth usage of CDN servers; b. Number of qualified peers; c. System reject rate; d. Peer capacity.

4.2. System Dynamics

In Figure 3, various metrics of the system described above are plotted. The two graphs on the left column (Figure 3a and 3c) are bandwidth usage of servers and system reject rate, respectively. The server bandwidth use curve shown here has similar shape to the one proposed in Figure 2. The short initial stage was followed by a period lasting over 11 hours when peers and CDN servers coexist as streaming data senders. After 11.94 hours, all the media delivery tasks were taken over by the peers and the server bandwidth consumption becomes 0. As expected, CDN bandwidth used at the first part of the CDN+Peer stage is close to the highest possible value (480Mbps). Starting from the 8th hour, the CDN bandwidth usage goes down. On the other hand, the reject rate of system requests (Figure 3c) keeps decreasing during the CDN+Peer stage till it reaches zero at 8.06 hours. The time for zero-rejection is very close to the time when the streaming load on the servers started to decrease.

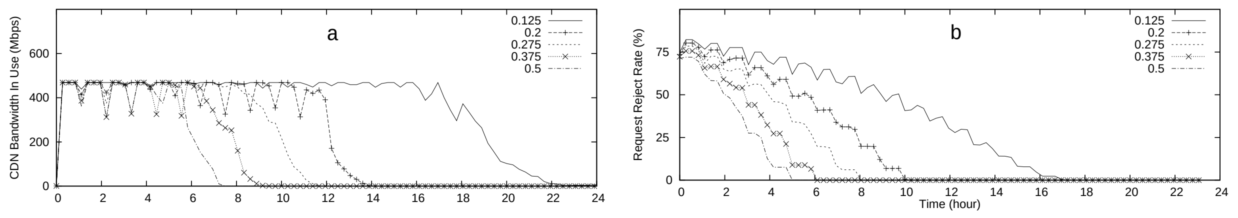

Figure 4. System performance under different capacity growth factors. a. Bandwidth use of CDN servers; b. Smoothed system reject rate.

Therefore, the k_0 value from this simulation setup is 8. Note the Instantaneous Reject Rate mentioned in Section 3 cannot be obtained from experiments. The data plotted in Figure 3c is an approximation by calculating the reject rate within a time window with a certain width. The window size we used for the simulation shown in Figure 3 was 8000 seconds. Smaller window sizes yielded less smooth curves (graphs not shown). No matter what window size we used, the same k_0 value was observed.

The growth of system capacity is illustrated by the change of total number of qualified peers (Figure 3b) and the bandwidth contribution from these peers (Figure 3d). Both peer number and peer bandwidth show geometric growth at the first 8 hours and linear growth afterwards, as expected. This is because the CDN servers and qualified peers are fully-loaded only before k_0. From Fig 3d we can also see that the deviation of the in-use peer bandwidth from total peer bandwidth increases monotonically. At about the 10th hour, the peer bandwidth use reaches a stable stage.

4.3. Effects of Peer Bandwidth Contribution and Request Rate

The effects of the system capacity growth factor ($\frac{\alpha}{b}$) and request rate (λ) on performance were also investigated. Figure 4 shows the CDN bandwidth use and reject rate under different choices of α while all other parameters remained unchanged. The legend in Figure 4 indicates the capacity growth factor of individual simulations. A four-fold increase of capacity growth factor (from 0.125 to 0.5) significantly shortened the CDN-Peer transition time from 17 hours to 5 hours. This means that k_0 is almost linearly related to the capacity growth factor $\frac{\alpha}{b}$. This is consistent with the analytical results given by Eq (3): since $\frac{\alpha}{b}$ is close to 0, we can consider

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

.

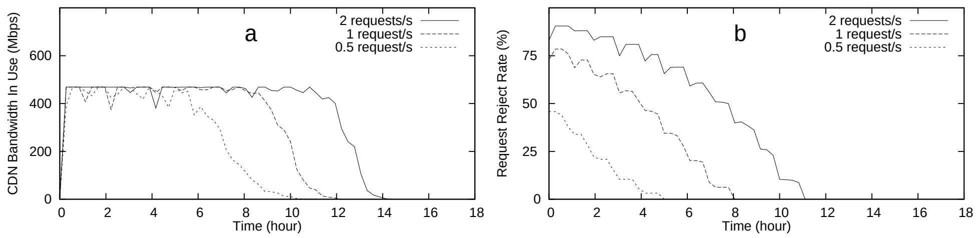

Figure 5. System performance under different request rate. a. Bandwidth use of CDN servers; b. Smoothed system reject rate.

A similar set of experiments were designed to study the impact of system request rate (λ) on k_0. The results for three request rates, 0.5, 1, and 2 requests per second, are shown in Figure 5. The value of k_0 observed increases as the request rate increases. However, the change of k_0 due to the change request rate is less dramatic than that caused by the change of the capacity growth factor. When the request rate increases 20 times to 10

Table 1. Theoretical and experimental values of k_0 under different scenarios.

Parameter		k_0	
α/b	λ(reqs/s)	*Calculated value* (h)	*Observed value* (h)
0.125	1	0.17+15.2	16.94
0.275	1	0.17+7.38	8.06
0.5	1	0.17+4.41	5.27
0.275	10	0.02+16.8	17.27
0.275	2	0.08+10.23	11.11
0.275	0.5	0.34+4.52	5.27

requests/second, a k_0 of 17.27 hours was obtained. The effects of streaming length (L) and bitrate (b) are similar to those of request rate (data not shown).

To verify the validity of our analytical model, k_0 values obtained from simulation experiments were compared with theoretical values (Table 1). The first item of the theoretical values shown in the table indicates the length of the initial stage while the second item the value calculated using Eq (3). The observed values are only slightly higher than relevant theoretical values. Considering the k_0 given by Eq (3) should be an integer number of time intervals with length L, the difference between theoretical value and simulation result is trivial for most cases.

Notice the total number of media files (F) is 100 for all the simulated systems. Therefore, our conclusion that the same equation can be applied to both single-file and multi-file systems is confirmed by our experiments.

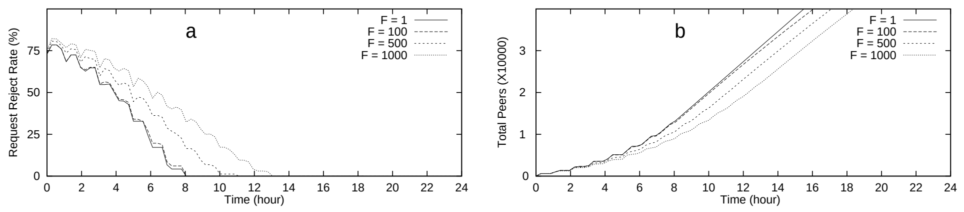

Figure 6. System performance under different number of media files. a. Smoothed system reject rate; b. Number of qualified peers.

4.4. Performance Affected by Number of Media Objects

As specified in the system analysis, the theoretical value of the server-peer transition time k_0 in a multi-file environment is the same for a system with only one video file. However, k_0 will be greater than the ideal value in a real-world situation. Figure 6 shows how the system performs under different media pool size.

The simulations in Figure 6 were run under four different values of F. The curves for experiments with total file number 1 and 100 are almost identical. The performance of a system conformed to the analytical results until the total file number goes beyond 120. After that, the transition time increases (Fig 6a).

As discussed in Section 3, two factors could account for the long transition time in a multi-file system: peers that acquired multiple files and lack of synchronization in the growth of per-file capacity. The effects of the first factor were investigated in the same set of experiments by recording the storage usage of qualified peers periodically. In Table 2, the number of qualified peers at transition time is listed by their storage consumption. For example, there were 12209 peers holding 1 media file (valid peer) and 54 holding 2 files for the simulation with 250 files. The values in the last column (β) are the ratios of the number of valid peers to all qualified peers. These ratios can be viewed as the lower bound of $\beta_{k,f}$ in Eq (10). According to Table 2, all β values are close enough to 1 so that the effects of the first factor mentioned above on k_0 are very small. Another conclusion we

Table 2. Storage usage of peers at transition time.

		# of media files stored				
Experiment	k_0 *(h)*	1	2	3	4	β
$F = 1$	8	10319	0	0	0	1.000
$F = 50$	8	11844	33	0	0	0.997
$F = 100$	8	10245	23	0	0	0.998
$F = 250$	9	12209	54	0	0	0.991
$F = 500$	11	16694	158	1	0	0.981
$F = 1000$	13	19260	324	1	0	0.967

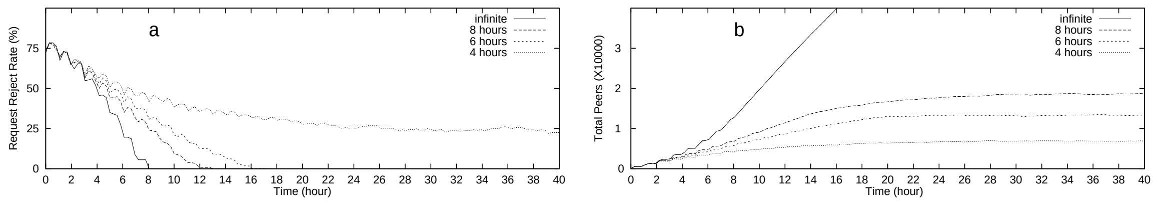

Figure 7. Effects of peer failure on system dynamics. a. Smoothed system reject rate; b. Total number of qualified peers.

may draw from Table 2 is that the space contribution of peers can be made minimal without affecting system performance.

Now it is clear that the second factor accounts for the degraded performance. According to Section 3.4, when F is big, $\frac{bN\lambda_f}{\lambda}$ 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. From the above experiments, $\frac{N}{Fb}$ has to be at least 5 for the system to get near-optimal transition time.

4.5. Negative Effects of Peer Failures

The introduction of finite peer lifespans significantly reduced the speed of system proliferation (Fig 7). We experimented on three simulation systems with different average peer lifespan (8, 6 and 4 hours). In all tested systems, peer lifespan is exponentially distributed. A system with no peer failures (infinite peer lifespan) was used as control. As the average lifespan of peers increases, the reject rate drops more dramatically (Fig 7a) and the system converges to server-peer transition faster. For the system with average lifespan of 4 hours, the peers fail too early to serve other peers so it never reaches the transition point. We also plotted the capacity growth of all systems (Fig 7b). The system with longer average peer lifespan grows faster than those with shorter peer life. For the case of 4 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.

5. RELATED WORK

A good review on Internet video streaming can be found in Wu et al.[1] The key research areas of video streaming are identified and methodologies were discussed. Research on P2P computing was greatly motivated by the success of Gnutella[†] and Napster[‡]. The general philosophy and current research efforts of P2P computing are introduced in Milojicic et al.[3] and Crowcroft et al.[5] Pastry,[4] Chord,[12] and CAN[13] are the most popular P2P

[†]http://www.gnutella.com
[‡]http://www.napster.com

searching/routing protocols. P2P storage applications built on top of these protocols include PAST,[14] CFS[15] and OceanStore.[16]

In the context of peer-to-peer media streaming, both the CoopNet project[9] and the ZIGZAG prototype[17] explore the way to deliver media streams 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 leave/failure of streaming peers. Our system model differs from these efforts in the sense that we are focusing on the delivery of on-demand media instead of live media. Commercial content delivery services such as Allcast[§] and C-Star[¶] are close in spirit to P2P media streaming.

Research on hybrid media streaming architecture[7] that combines CDN and P2P networks is directly related to our work. In some sense, our research is performed as an extension to the analysis presented in Xu et al.[7] In this paper, we present an improved analytical framework that leads to quantitative conclusions on the growth of both single-file and multi-file systems. We also study the impact of peer failures on the performance of our media streaming architecture. A similar P2P streaming architecture can be found in Hefeeda et al,[8, 18] in which efficient algorithms for dissemination of media content and economical analysis of P2P streaming service 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. In our previous work,[11] we proposed a failure-resistant protocol called *Redundant Multi-Channel Streaming Protocol (RMCSP)*. The major concern of this protocol is to deal with in-session peer failures by replacing the failed supplying peer(s) by the server(s).

In another work by Xu et al,[6] an algorithm that assigns media segments to different supplying peers and an admission protocol for requests are introduced. Another research project by Nguyen and Zakhor[19] is more closely related to our research in the aspect of streaming protocol design. In their work, an RTP-like protocol with the features of rate control and packet synchronization was developed. The use of incentives and coupons in P2P services are discussed in Horne et al.[20] and Golle et al.[21]

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have shown that the capacity of the hybrid streaming system grows exponentially. From quantitative analysis and simulation, we found that the server-peer transition time k_0 is most sensitive to the Capacity Growth Factor $\frac{\alpha}{b}$. Within a boundary, the same equation can be used to describe the behaviors of single-file and multi-file systems. When there are too many files, system performance can be deteriorated and the extent of such performance degradation was analyzed. Our analytical framework was also used to study the effects of peer failures on the capacity growth. We found that peer failures negatively affect system capacity by decreasing the Capacity Growth Factor.

Future work involves extension of our analysis to capture more general scenarios of system operations. For example, various statistical distributions of peer lifespan. Strategies to increase Capacity Growth Factor and peer commitment duration are also worth in-depth research. While discounts, incentives, or coupons can be awarded to peers based on their contribution, efficient accountability is needed. Other possible research topics include refinement of streaming protocols, QoS management of media, security and integrity issues.

ACKNOWLEDGMENTS

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 for their inspirations and 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.

[§]http://www.allcast.com
[¶]http://www.centerspan.com

REFERENCES

1. 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**, February 2001.
2. A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal, O. Spastcheck, and W. Sturm, "CDN Brokering," in *Proceedings of International Workshop on Web Caching and Content Distribution (WCW 2001)*, June 2001.
3. D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Rihard, S. Rollins, and Z. Xu, "Peer-to-Peer Computing," Tech. Rep. HPL-2002-57, HP Labs, 2002.
4. 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)*, November 2001.
5. J. Crowcroft and I. Pratt, "Peer to Peer: peering into the future," in *Proceedings of the IFIP-TC6 Networks 2002 Conference*, May 2002.
6. D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava, "On Peer-to-Peer Media Streaming," in *Proceedings of IEEE ICDCS 2002*, July 2002.
7. 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 Multimedia Computing and Networking (MMCN 2003)*, 2003.
8. M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava, "A Peer-to-Peer Media Streaming System Using Collectcast." to appear in *Proceedings of ACM Multimedia 2003.*
9. V. N. Padmanabhan and K. Sripanidkulchai, "The Case for Cooperative Networking," in *Proceedings of he First International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.
10. S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in *Proceedings of SPIE/ACM Multimedia Computing and Networking (MMCN 2002)*, 2002.
11. Y.-C. Tu and S. Lei, "Towards Cost-effective On-demand Continuous Media Service: A Peer-to-Peer Approach," Tech. Rep. CSD TR 03-023, Purdue University, 2003.
12. 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 '01*, August 2001.
13. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in *Proceedings of the ACM SIGCOMM '01*, August 2001.
14. A. Rowstron and P. Druschel, "Storage management in past, a large-scale, persistent peer-to-peer storage utility," in *Proceedings of 18th ACM Symposium on Operating Systems Principles*, October 2001.
15. F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-Area Cooperative Storage with CFS," in *Proceedings of ACM SOSP 2001*, October 2000.
16. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, and S. Rhea, "Oceanstore: An architecture for global-scale persistent storage," in *Proceedings of 9th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, November 2000.
17. 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*, pp. 1283–1292, April 2003.
18. M. Hefeeda, B. Bhargava, and D. Yau, "A Cost -Effective Architecture for On-Demand Media Streaming." to appear in *Journal of Computer Networks.*
19. T. Nguyen and A. Zakhor, "Distributed Video Streaming over Internet," in *Proceedings of SPIE/ACM MMCN 2002*, January 2002.
20. 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'01)*, October 2001.
21. P. Golle, K.Leylton-Brown, and I. Mironov, "Incentives for sharing in peer-to-peer networks," in *Proceedings of Second Workshop on Electronic Commerce (WELCOM'01)*, November 2001.