# Analysis of a Hybrid Architecture for Cost-Effective Streaming Media Distribution

Dongyan Xu[‡], Heung-Keung Chai[†], Catherine Rosenberg[†], Sunil Kulkarni[†]
[‡]Department of Computer Sciences
[†]School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907
[‡]dxu@cs.purdue.edu, [†]{chaih, cath, sunilkul}@ecn.purdue.edu

## Abstract

To distribute video and audio data in real-time streaming mode, both CDN (Content Distributed Network) based and peer-to-peer based architectures have been proposed. However, each architecture has its limitations. CDN servers are expensive to deploy and maintain. The storage space and out-bound bandwidth allocated to each media file are limited and incur a cost. Current solutions to lowering such cost usually compromise the media quality delivered. On the other hand, a peer-to-peer architecture needs a sufficient number of 'seed' supplying peers to 'jumpstart' the system. Compared with a CDN server, a peer offers very low out-bound bandwidth. Furthermore, it is not clear how to fairly determine the contribution of each supplying peer.

In this paper, we propose a novel *hybrid architecture* which integrates CDN and peer-to-peer based streaming media distribution. The architecture is highly cost-effective: it significantly lowers the cost of CDN server resources, without compromising the media quality delivered. Furthermore, we propose a *limited contribution policy* for the supplying peers in the system, so that the streaming capacity of supplying peers is exploited on a limited and fair basis. We present an in-depth quantitative analysis of the hybrid system. The analysis is very well supported by our extensive simulation results.

## 1 Introduction

The proliferation of high-speed, broadband networking technologies has made real-time media streaming a reality. It is increasingly feasible to distribute video and audio data in real-time streaming mode. In fact, streaming media distribution has been an intensively studied research topic in the past few years. Among the most established architectures is the Content Distribution Network or CDN, where a large number of CDN servers are deployed at the edge of the Internet, and clients request media streaming service from their closest CDN servers. More recently, a peer-to-peer based media data distribution architecture is quickly gaining popularity, where clients store the data after receiving a streaming service, and act as supplying peers by streaming the data to other requesting clients (peers). However, we argue that both CDN-based and peer-to-peer based architectures have their advantages and disadvantages, and each architecture alone does not provide a cost-effective and scalable solution to streaming media distribution.

In a CDN for media streaming, data are first pushed to multiple CDN servers. Each CDN server serves clients within a network neighborhood. Besides being close to the clients, a CDN server also has large storage space and high out-bound bandwidth. Therefore, a CDN server is expected to deliver high-quality streaming media service to its clients. However, it is expensive to deploy and maintain CDN servers. The storage space and out-bound bandwidth allocated to each media file are limited, and they incur a cost to the content provider and/or clients - recall the recently imposed subscription fee for CNN.com Video. Some solutions try to lower this cost [16], by adaptively lowering the delivered media quality based on media popularity and request rate, and therefore lowering the storage space and bandwidth requirement. The downside of this solution is that it compromises the quality of service received by

individual clients - especially those 'nice' clients that have refrained from requesting the media data right after its release.

On the other hand, peer-to-peer media streaming exhibits a more de-centralized character: after clients receive the media data, they act as supplying peers and stream the data to other requesting clients[1] directly, without involving a CDN server. Such a peer-to-peer architecture exploits the aggregated and growing streaming capacity of individual supplying peers, and therefore provides a solution to the cost-quality tradeoff problem in CDN-based media streaming. However, the peer-to-peer architecture has its own problems. First, it needs a sufficient number of 'seed' supplying peers to 'jumpstart' the system. Second, compared with a CDN server, a peer offers very low out-bound bandwidth - probably lower than the media playback rate. Finally, it is not clear how to fairly determine the contribution of each supplying peer. For example, an 'early-bird' supplying peer may end up serving many more streaming sessions than a late-coming peer, which is not fair to the former. To the best of our knowledge, this problem has not been addressed in recent peer-to-peer literature.

In this paper, we propose a novel *hybrid architecture* that integrates CDN and peer-to-peer based streaming media distribution. In this architecture, the two streaming modes complement each other: On the release of a new media file, the CDN server will create the initial 'seed' supplying peers in its neighborhood. When the peer-to-peer streaming capacity grows to a certain level, the CDN server will stop the streaming service for this media file, and free the allocated storage space and out-bound bandwidth for other media data. We call this transition a 'CDN to peer-to-peer' *handoff*. Meanwhile, the CDN server naturally acts as the 'index server', hooking up requesting peers with supplying peers. Furthermore, we propose a *limited contribution policy* for each supplying peer: each peer only commits (1) a limited out-bound bandwidth to each streaming session and (2) a limited number of streaming sessions it will serve during its tenure as a supplying peer. This policy also realizes a contribution fairness among supplying peers: the higher the committed per session out-bound bandwidth, the fewer the committed sessions to serve.

The major contributions of this paper include the following: (1) The proposed hybrid architecture is highly cost-effective: it significantly lowers the cost of CDN server resources, without compromising the media quality delivered to the clients. (2) The limited contribution policy ensures that the streaming capacity of supplying peers is exploited on a limited and fair basis. (3) A discrete-time quantitative system analysis is performed. It leads to an in-depth understanding of the system dynamics and reveals the impact of the limited contribution policy on system performance. The analysis is very well supported by our extensive simulation results.

The rest of the paper is organized as follows. Section 2 presents the overall system architecture and operations. Section 3 presents our quantitative system analysis in detail. Section 4 presents simulation results. Section 5 compares our work with related work. Finally, Section 6 concludes this paper.

# 2 System Architecture and Operations

## 2.1 System Architecture

The proposed hybrid architecture is shown in Figure 1. We only show one CDN server[2] because we focus on the interaction between each CDN server and the clients it serves.

- The CDN server in our architecture plays two roles: the ordinary media streaming server and the peer-to-peer index server. For each media file, the CDN server stores the file before the 'CDN to peer-to-peer' handoff, and releases its allocated space after the handoff. In the meantime, the CDN server maintains a list of active supplying peers and their contribution fulfillment status. Note that both the handoff and the supplying peers are specific to each media file. Before the handoff, a streaming request may be served either by the CDN server or by a set of supplying peers selected by the CDN server, while after the handoff, the CDN server will only act as the index server of the corresponding media file.

- On the client side, each client requesting the media file has a three-phase life-cycle: (1) Before receiving the streaming service, the client is a *requesting peer*. (2) After receiving the streaming service, it becomes a *supplying peer* with a limited contribution commitment. (3) Finally, when its commitment has been fulfilled, it becomes a *finished peer*. We note that most current peer-to-peer systems do *not* define the third phase, i.e. they assume that a 'well-behaved' supplying peer will always contribute.

[1] The clients are also called requesting peers. We will use the terms 'peer' and 'client' interchangeably for the rest of the paper.

[2] The CDN server is a logical entity - it may consist of multiple physical servers.

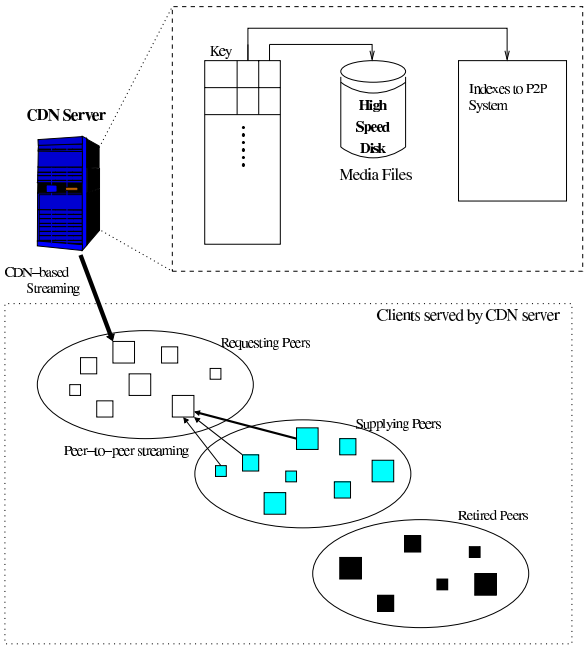


Figure 1: A hybrid architecture for streaming media distribution (different sizes of the peers indicate their different out-bound bandwidth contribution)

Different from CDN-based streaming, a peer-to-peer streaming session involves multiple supplying peers (as shown in Figure 1), each of them streaming a *subset* of the media data to the requesting peer. To ensure full media quality, the sum of their out-bound bandwidth contribution (possibly in different amounts) is at least the same as the media playback rate. In [20], we present an algorithm to assign a subset of the media data to each supplying peer, based on its out-bound bandwidth contribution. Our prototype implementation also demonstrates the feasibility of delivering full quality video from multiple supplying peers.

## 2.2 System Operations

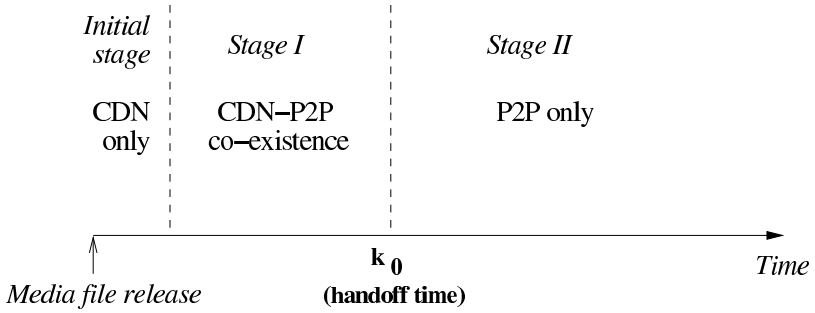


Figure 2: Different stages of the system

When a media file is first released, it is pushed to each CDN server. At the beginning, there are no supplying peers. The CDN server streams the media data to requesting clients (the initial stage in Figure 2). After a streaming session, the requesting client registers itself with the CDN server and becomes a supplying peer with a *limited contribution commitment* which includes: (1) a limited out-bound bandwidth contributed to each streaming session and (2) a limited number of streaming sessions it will serve during its tenure as a supplying peer. To ensure fairness, the higher the amount in (1), the smaller the number in (2) - a quantitative definition will be given in Section 3.

With the creation of supplying peers, the CDN server can divide the streaming load between itself and the supplying peers. This is the stage when the CDN and peer-to-peer based streaming co-exist, while the peer-to-peer streaming capacity grows (Stage I in Figure 2). When a streaming request arrives, the CDN server first checks if there are active supplying peers such that (1) they are not serving other streaming sessions and (2) the sum of their out-bound bandwidth is equal to the media playback rate. If so, the request will be served by the selected supplying peers; otherwise, the request will be served by the CDN server itself. If both CDN and peer-to-peer streaming capacity is not available, the request will be rejected.

Finally, the CDN server decides to handoff from CDN streaming to peer-to-peer streaming, so that the storage and bandwidth resources allocated to the media file can be freed. The handoff time $k_0$ is determined such that the peer-to-peer streaming capacity alone is sufficient to handle all subsequent requests for this media file, with rejection probability close to zero. After the handoff, the CDN server only acts as a directory server of this file, and the streaming will be performed by the supplying peers (Stage II in Figure 2).

As to be shown in Section 3, a non-trivial analysis is needed to determine the handoff time. If the handoff takes place too early, the peer-to-peer streaming capacity may not have grown to the sufficient level. On the other hand, if the handoff happens too late, the CDN resources for the media file will be held longer thus incurring additional cost. The handoff time calculation is further complicated by our limited contribution policy, which creates a *dynamic* population of supplying peers. Therefore, to determine the handoff time, we need an in-depth analysis of the system dynamics, capturing the peer-to-peer streaming capacity growth, supplying peer contribution fulfillment, and streaming request arrivals over time.

# 3 System Analysis

## 3.1 Assumptions

As a first step toward an in-depth understanding of the hybrid architecture with dynamic 'CDN to peer-to-peer' handoff, we make the following assumptions to make our analysis feasible. First, we assume honesty and 'always-on' network connection of all supplying peers: they will complete each of the committed streaming sessions. Second, for each streaming session, the intermediate network does not create additional bottleneck between the CDN server and the requesting peer, or between the supplying peers and the requesting peer, i.e. the bottleneck always lies in the out-bound link of the CDN server or of the supplying peers. This assumption can be partially justified by the fact that all clients (peers) considered are within the same domain served by the CDN server. Third, the streaming requests are generated independently by each requesting peer[3]. Finally, we assume that each client has sufficient storage to store the media file. Our analysis based on these assumptions will serve as a basic and extendible framework for the modeling of more dynamic and complex systems.

## 3.2 System Parameters and Metrics

The system parameters and performance metrics are summarized in Table 1. Note that they are defined with respect to the *same* media file.

Initially, we have a CDN server that has allocated a streaming capacity of $N_c$ full streaming sessions to the media file. The total client population served by the CDN server is $M_0$. After the media file is streamed to a requesting client, the client will register with the CDN server and commit a *limited contribution*. Our system provides $n$ options of limited contribution: each option includes (1) an out-bound bandwidth contributed to each session, which is equal to $\frac{1}{c_i}$ $(1 \leq i \leq n)$ of the media playback rate and (2) a total of $x_i$ sessions to serve. As to be shown in Section 3.3, the following must hold: $x_i > c_i$. Correspondingly, the client population is divided into $n$ classes: a class $i$ peer chooses option $i$. The percentage of class $i$ peers in the total population is denoted by $p_i$. Therefore we have $\sum_{i=1}^{n} p_i = 1$. In a peer-to-peer streaming session, the requesting peer will be served by a set of supplying peers whose sum of out-bound bandwidth is equal to the media playback rate. If a streaming request is rejected due to insufficient streaming capacity, the requesting peer will continue to generate requests for this media file with a per-client request generation rate $\lambda$.

We discretize the time scale in multiples of $L$, the duration of one streaming session, and $k$ is the discrete time index. We adopt this relatively coarse time granularity in order to make our system analysis feasible. As to be shown

[3]We realize that this assumption may not always be valid. One counter-example is the *flash crowd* scenario which requires alternative solutions such as the ones proposed in [4] and [14].

| Notation | Definition |
|---|---|
| $L$ | Length of one streaming session in minutes. |
| $k$ | Discrete time index, each unit has a length of $L$. |
| $N_c$ | CDN server capacity allocated to the media file (in number of simultaneous streaming sessions). |
| $M_0$ | Total client population. |
| $n$ | Number of peer classes. |
| $p_i$ | Percentage of peers in the $i^{th}$ class. |
| $\lambda$ | Per-client request generation rate, in requests per minute. |
| $c_i$ | An integer indicating that the out-bound bandwidth contributed by a class $i$ peer is $\frac{1}{c_i}$ of the media playback rate. |
| $x_i$ | Number of sessions a class $i$ peer is committed to serve. $x_i > c_i$. |
| $k_0$ | The 'CDN to peer-to-peer' handoff time. |
| $M(k)$ | Number of remaining requesting peers at time $k$. $M(0) = M_0$ |
| $S(k)$ | Total *committed* peer-to-peer streaming capacity at time $k$, in number of *full* streaming sessions. |
| $N(k)$ | *Instantaneous* peer-to-peer streaming capacity at time $k$, in number of *full* streaming sessions. |

Table 1: Definitions of system parameters and performance metrics

in Section 4, our analytical results match the results from our simulations, which have a minimum time unit of one minute.

One of the main goals of our analysis is to compute the 'CDN to peer-to-peer' handoff time $k_0$. $k_0$ is defined as time when the peer-to-peer streaming capacity alone is able to fulfill all subsequent streaming requests. Starting from $k_0$, the request rejection rate should remain close to zero. To derive $k_0$, we have to derive: (1) $N(k)$ - the instantaneous peer-to-peer streaming capacity at any time $k$ and (2) $M(k)$ - the number of remaining requesting peers at $k$. Unfortunately, the accurate form of $N(k)$ proves extremely difficult - if at all possible. This is because $N(k)$ is dependent on the progress of contribution fulfillment of each *individual* supplying peer.

To illustrate the difficulty in deriving $N(k)$, consider the example in Figure 3. Suppose at time $k$, there are four class 1 supplying peers: *Peer 1* to *Peer 4*. Let $c_1 = 2$ and $x_1 = 3$. Suppose at $k$, they still need to serve 1, 1, 2, and 2 sessions, respectively. Since $c_1 = 2$, $N(k) = 4/2 = 2$ (in number of *full* sessions). If a request by *Peer 5* arrives at $k$, then $N(k+1)$ will have different values, depending on which supplying peers among *Peers 1, 2, 3* and *4* are chosen to serve *Peer 5*. For example, if *Peer 1* and *Peer 2* are chosen, $N(k+1)$ will be $3/2 = 1.5$ (Figure 3(b)). However, if *Peer 3* and *Peer 4* are chosen, $N(k+1)$ will be $5/2 = 2.5$ (Figure 3(c)). Such a disparity is due to the different progress of contribution fulfillment of *Peers 1, 2, 3* and *4*.

To get round to the difficulty in deriving $N(k)$, we instead derive a lower bound of $N(k)$. We first define $S(k)$ as the total *committed* peer-to-peer streaming capacity (in *full* sessions) at $k$. And $S(k)$ is much easier to model. In the example in Figure 3, $S(k)$ is $(1+1+2+2)/2 = 3$. At $k+1$, no matter which supplying peers serve *Peer 5*, $S(k+1)$ will be $S(k) - 1 + 3/2 = 3.5$, as shown in Figures 3(b) and 3(c). Furthermore, we observe that $S(k)/x_1$ is a lower bound of $N(k)$. The former is in fact the instantaneous streaming capacity of the following *virtual system*: *at most* one virtual supplying peer has fewer than $x_1 = 3$ sessions to serve. The virtual system at $k$ and $k+1$ is shown in Figures 3(a') and 3(b'), respectively. It is easy to see that the virtual system has the minimum number of supplying peers (and therefore the lowest $N(k)$) among systems with the same $S(k)$. In Section 3.4, we will derive the lower bound of $N(k)$ under multiple $x_i$'s.

The rest of our analysis is organized as follows: In Section 3.3, we analyze the hybrid system in two stages: (1) stage I when $k < k_0$ and (2) stage II when $k \geq k_0$. When $k < k_0$, the hybrid system offers both CDN-based streaming and peer-to-peer streaming, and the total streaming capacity is lower than what is requested by the requesting peers. For stage I, we will first derive $S(k)$ and then derive $M(k)$. When $k > k_0$, the system only offers peer-to-peer streaming, and the total streaming capacity is higher than the capacity requested. For stage II, we will first derive $M(k)$ and then derive $S(k)$. We will use $S_I(k)$, $S_{II}(k)$, $M_I(k)$ and $M_{II}(k)$ to denote the $S(k)$ and $M(k)$ in stages I and II, respectively. With $S(k)$ and $M(k)$, we determine $k_0$ in Section 3.4. Finally, we discuss the relation among parameters $N_c$, $x_i$ and $k_0$ in Section 3.5.

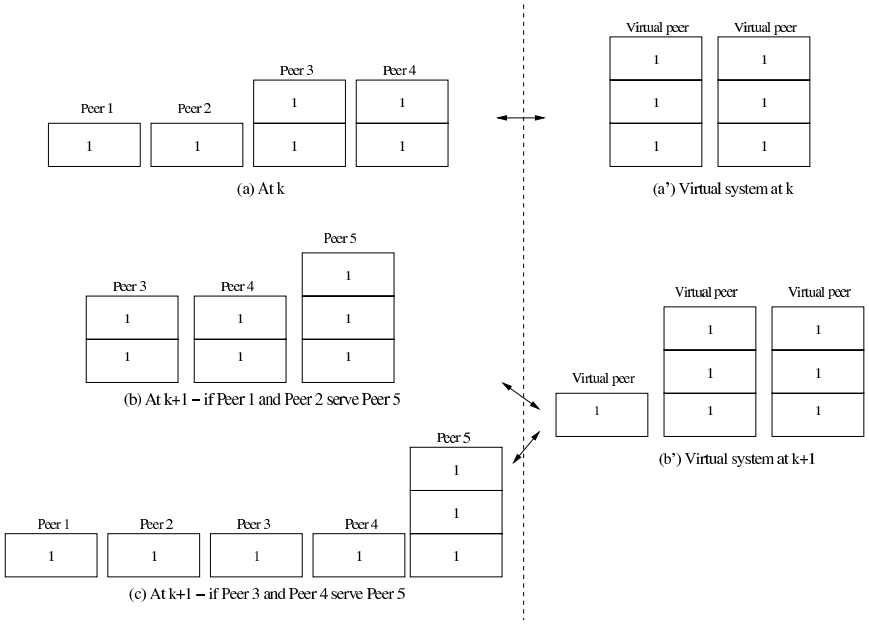


Figure 3: Example illustrating the difficulty in tracking $N(k)$

### 3.3 Derivation of $S(k)$ and $M(k)$

$S(k)$ is the total committed peer-to-peer streaming capacity in number of full sessions, and it can be defined as:

$$S(k+1) = \begin{cases} S(k) + N_c \sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j}\right) + \left(\frac{S(k)}{\sum_{j=1}^{n} p_j \cdot \left(\frac{x_j}{c_j}\right)}\right) \left[\sum_{j=1}^{n} \frac{p_j}{c_j}\right] \left[\sum_{j=1}^{n} p_j \cdot \left(\frac{x_j}{c_j} - 1\right)\right] & \text{if } k < k_0 \\ S(k) + \lambda L M(k) \left[\sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j} - 1\right)\right] & \text{if } k \geq k_0 \end{cases}$$

We first explain the equation for $S(k)$ when $k < k_0$, i.e. $S_I(k)$. The four terms in the equation are explained as follows:

For $k < k_0$:

$$S_I(k+1) = S_I(k) + \underbrace{N_c \sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j}\right)}_{Term\ 1} + \underbrace{\left(\frac{S_I(k)}{\sum_{j=1}^{n} p_j \cdot \left(\frac{x_j}{c_j}\right)}\right)}_{Term\ 2} \underbrace{\left[\sum_{j=1}^{n} \frac{p_j}{c_j}\right]}_{Term\ 3} \underbrace{\left[\sum_{j=1}^{n} p_j \cdot \left(\frac{x_j}{c_j} - 1\right)\right]}_{Term\ 4} \quad (1)$$

- Term 1 is the total committed contribution from peers served by the CDN server during interval $[k, k+1]$. $(\frac{x_j}{c_j})$ is the number of sessions (normalized to *full* sessions) contributed by a class $j$ peer. $N_c$ is the CDN server capacity - therefore it is also the number of peers served by the CDN server during $[k, k+1]$.
- Term 2 is the expected number of supplying peers at $k$. Recall that $S(k)$ is in number of full sessions. To estimate

the number of supplying peers, we need to divide $S(k)$ by the average number of full sessions contributed by each supplying peer.

- Term 3 is the average number of session (normalized to full session) contributed by each supplying peer at any time. This term is multiplied by Term 2 to estimate the total instantaneous peer-to-peer streaming capacity at $k$.
- Term 4 is the average number of sessions (normalized to full sessions) contributed by each peer that receives a peer-to-peer streaming session during $[k, k+1]$. Therefore, the product of Terms 2, 3 and 4 is the total committed contribution (in number of full sessions) from peers served by peer-to-peer streaming during $[k, k+1]$.

We then explain the equation for $S_{II}(k)$:

$$S_{II}(k+1) = S_{II}(k) + \lambda L M_{II}(k) \sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j} - 1\right) \tag{2}$$

In stage II, since the peer-to-peer streaming capacity is higher than the streaming capacity requested, the growth of $S_{II}(k)$ during $[k, k+1]$ is computed as the number of streaming requests $\lambda L M_{II}(k)$ multiplied by the same Term 4 in $S_I(k)$ .

Based on $S(k)$, we can easily define $M(k)$, the remaining number of requesting peers at $k$ in the following equation:

$$M(k+1) = \begin{cases} M(k) - \dfrac{S(k)}{\sum_{i=1}^{n} p_i \left(\frac{x_i}{c_i}\right)} \sum_{j=1}^{n} \left(\frac{p_j}{c_j}\right) - N_C, & k < k_0 \\ M(k) \cdot (1 - \lambda L), & k \geq k_0 \end{cases}$$

To simplify subsequent analysis, we introduce two additional variables $r$ and $\rho$ in Table 2.

| Notation | Definition |
|---|---|
| $r$ | $\sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j}\right)$ |
| $\rho$ | $\sum_{j=1}^{n} \left(\frac{p_j}{c_j}\right)$ |

Table 2: Definitions of $r$ and $\rho$

To derive the closed form of $S_I(k)$, we rewrite Equation (1) as:

$$S_I(k+1) = S_I(k) \left(1 + \rho \left(\frac{r-1}{r}\right)\right) + rN_c \tag{3}$$

Equation (3) can then be solved as:

$$S_I(k) = \frac{N_c}{\rho} \left(\frac{r^2}{r-1}\right) \left[\left(1 + \rho \left(\frac{r-1}{r}\right)\right)^k - 1\right] \tag{4}$$

With (4), the closed form of $M_I(k)$ can then be solved as:

$$M_I(k) = M_0 + kN_c \left(\frac{r}{r-1} - 1\right) - \frac{N_c}{\rho} \left(\frac{r}{r-1}\right)^2 \left[\left(1 + \rho \left(\frac{r-1}{r}\right)\right)^k - 1\right] \tag{5}$$

In fact, it is more convenient to rewrite $M_I(k)$ as a function of $S_I(k)$:

$$M_I(k) = M_0 + \frac{kN_c - S_I(k)}{r-1} \tag{6}$$

The derivation of $M_{II}(k)$ is simpler. Note that by definition, $M_{II}(k_0) = M_I(k_0)$. Therefore, $M_{II}(k)$ can be solved as:

$$M_{II}(k) = M_I(k_0)\left[1-\lambda L\right]^{k-k_0}, \quad k \geq k_0 \tag{7}$$

With (7), it is easy to derive $S_{II}(k)$:

$$S_{II}(k) = S_I(k_0) + M_I(k_0)(r-1)\left[1-(1-\lambda L)^{k-k_0}\right], \quad k \geq k_0 \tag{8}$$

One key observation is that in order for $S_I(k)$ to grow instead of decline, we must have $r > 1$ in Equation (4). Since $r = \sum_{j=1}^{n} p_j(\frac{x_j}{c_j})$, $r$ can be thought of as the average 'contribution ratio' of all supplying peers: the higher the $r$, the more the contribution they make. Recall that in our limited contribution policy (Section 3.2), we require that $x_j > c_j$ $(1 \leq j \leq n)$. Now it is clear that $x_j > c_j$ will lead to $r > 1$, which will guarantee the growth of $S_I(k)$. Otherwise, if we let $x_j = c_j$, we will have $r = 1$, and $S_I(k)$ will collapse. Intuitively, $x_j > c_j$ means that the committed contribution of each peer - if measured by the total volume of data it sends out, should be greater than the total volume of data it receives (i.e. the volume of the media file). In other words, the peers need to make a 'net' contribution to the system.

### 3.4 Derivation of Handoff Time $k_0$

By definition, starting at time $k_0$, the peer-to-peer streaming capacity alone will be able to handle all subsequent streaming requests without the CDN server. $k_0$ can be derived by equating the number of requests in interval $[k_0-1, k_0]$ to the instantaneous peer-to-peer streaming capacity at $k_0$ multiplied by a conservative factor $\alpha$ $(0 < \alpha \leq 1)$:

$$\lambda L M_I(k_0) = \alpha N_I(k_0) \tag{9}$$

Recall that in Section 3.2, $N_I(k)$ is shown difficult to derive. Instead, we suggest a lower bound of $N_I(k)$ computed from $S_I(k)$. With the presence of multiple peer classes, the lower bound of $N_I(k)$ can be expressed as $\frac{\rho S_I(k)}{r}$. It is easy to verify that $\frac{\rho S_I(k)}{r}$ is the instantaneous peer-to-peer capacity of the 'virtual system' with total committed capacity $S_I(k)$, but with at most one class $j$ $(1 \leq j \leq n)$ supplying peer having fewer than $x_j$ sessions to serve. Therefore, we will use the following equation instead of (9) to derive $k_0$:

$$\lambda L M_I(k_0) = \alpha \frac{\rho S_I(k_0)}{r} \tag{10}$$

By replacing $M_I(k)$ using (6), (10) can be re-arranged as:

$$\lambda L\left[M_0 + \frac{k_0 N_c}{r-1}\right] = \left(\alpha\frac{\rho}{r} + \frac{\lambda L}{r-1}\right) S_I(k_0)$$

By replacing $S_I(k)$ using (4), we have:

$$\frac{\lambda L\left[M_0 + \frac{k_0 N_c}{r-1}\right]}{\left(\alpha\frac{\rho}{r} + \frac{\lambda L}{r-1}\right)} = \left(\frac{r^2}{r-1}\right)\frac{N_c}{\rho}\left[\left(1+\rho\frac{r-1}{r}\right)^{k_0} - 1\right]$$

$$\Rightarrow \left(1+\rho\frac{r-1}{r}\right)^{k_0} = \frac{\lambda L\left[M_0 + \frac{k_0 N_c}{r-1}\right]}{\left(\alpha\left(\frac{r}{r-1}\right)N_c + \lambda L\left(\frac{r}{r-1}\right)^2\frac{N_c}{\rho}\right)} + 1 \tag{11}$$

Observe that (11) has the form of $a^{k_0} = bk_0 + c$, where $a = \left(1+\rho\frac{r-1}{r}\right)$, $b = \frac{\left(\frac{\lambda L N_c}{r-1}\right)}{\left(\alpha\left(\frac{r}{r-1}\right)N_c+\lambda L\left(\frac{r}{r-1}\right)^2\frac{\lambda L N_c}{\rho}\right)}$ and $c = \frac{\lambda L M_0}{\left(\alpha\left(\frac{r}{r-1}\right)N_c+\lambda L\left(\frac{r}{r-1}\right)^2\frac{N_c}{\rho}\right)} + 1$. $k_0$ can be solved as follows:

$$k_0 = \frac{c\cdot\log(a) - b\cdot W\left(-\frac{\log(a)\cdot e^{\left(\frac{c\log a}{b}\right)}}{b}\right)}{b\cdot\log(a)} \tag{12}$$

$W(\cdot)$ is the *Lambert's W-function*. Detailed definition of this function is given in [8] and omitted in this paper.

The following Lemma shows that after the handoff at $k_0$, the instantaneous peer-to-peer streaming capacity is expected to satisfy all subsequent streaming requests.

**Lemma 1:** For any $k \geq k_0$, we have $\lambda L M_{II}(k) \leq \alpha N_{II}(k)$.

**Proof:** First, we observe that $M_{II}(k)$ is a decreasing function, due to the finite client population. Hence we have $\lambda L M_{II}(k) \leq \lambda L M_I(k_0)$ for any $k \geq k_0$. On the other hand, we observe that $S_{II}(k)$ is an increasing function. Hence we have $\alpha\frac{\rho S_I(k_0)}{r} \leq \alpha\frac{\rho S_{II}(k)}{r}$. Since $\frac{\rho S_{II}(k)}{r}$ is the lower bound of $N_{II}(k)$, we have $\alpha\frac{\rho S_{II}(k)}{r} \leq \alpha N_{II}(k)$. Finally, the following holds for $k > k_0$:

$$\lambda L M_{II}(k) \leq \lambda L M_I(k_0) = \alpha\frac{\rho S_I(k_0)}{r} \leq \alpha\frac{\rho S_{II}(k)}{r} \leq \alpha N_{II}(k) \tag{13}$$

## 3.5 Relation between $k_0$, $r$ and $N_c$

Having derived $k_0$, we now discuss the relation between $k_0$, $r$, and $N_c$. Note that $N_c$ and $x_i$ $(1 \leq i \leq n)$ are the *tunable* system parameters, and the 'contribution ratio' $r = \sum_{j=1}^{n} p_j(\frac{x_j}{c_j})$.

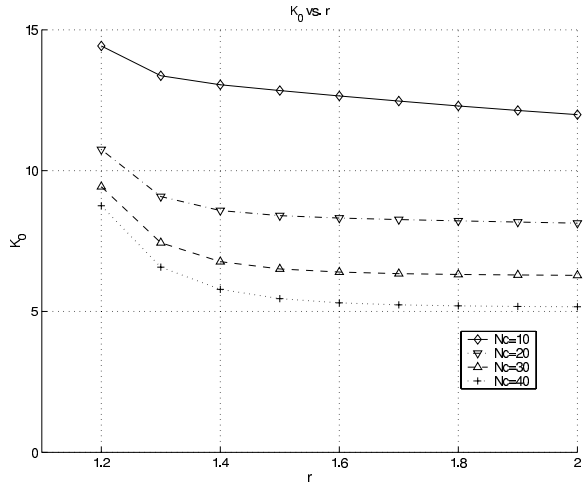


Figure 4: Relation between $k_0$ and $r$ under different $N_c$

Based on Equation (12), Figure 4 shows the relation between $k_0$ and $r$, under different CDN streaming capacity $N_c$. The values of $\lambda$, $\alpha$, $M_0$ and $L$ are fixed and the same as in our simulation (Section 4.1). We observe that for a fixed $r$, the higher the $N_c$, the lower the $k_0$ (i.e. the sooner the handoff can take place). However, the effect of $N_c$ diminishes as $N_c$ increases. On the other hand, with the same $N_c$, $k_0$ decreases with the increase of $r$. However, as $r$ increases, $k_0$ quickly levels off. This justifies our limited contribution policy for the supplying peers: beyond a certain point, any further increase in $r$ will *not* bring about a much earlier handoff. In other words, by setting a relatively low $r$, we already achieve a reasonably early handoff time $k_0$.

In fact, if we let $r \to \infty$ in (12), we can derive the *earliest* possible handoff time as:

$$\lim_{r\to\infty} k_0 = \frac{\log\left(\frac{\lambda L M_0}{\left(\alpha N_c+\lambda L\frac{N_c}{\rho}\right)}+1\right)}{\log(1+\rho)} \tag{14}$$

# 4 Simulations

In this section, we present our simulation results to demonstrate the effectiveness of the hybrid architecture, as well as to confirm our analytical results.

## 4.1 Simulation Setup

We simulate a hybrid system with one CDN server and a client population of $M_0 = 2000$. Initially, the CDN server allocates sufficient storage space as well as an out-bound bandwidth capable of serving $N_c = 20$ simultaneous streaming sessions for the given media file. The duration of each streaming session is $L = 60$ minutes. Each client makes streaming request for this media file independently, with a per client Poisson request generation rate $\lambda = 0.001$ request/minute. The 2000 clients belong to $n = 3$ classes: for class 1, $c_1 = 2$ and $x_1 = 3$; for class 2, $c_2 = 4$ and $x_2 = 6$; for class 3, $c_3 = 8$ and $x_3 = 12$. The percentages of class 1, 2 and 3 peers are 20%, 50%, and 30%, respectively (therefore, $r = \frac{3}{2} * 20\% + \frac{6}{4} * 50\% + \frac{12}{8} * 30\% = 1.5$). The factor $\alpha$ used in the calculation of $k_0$ is 1.0. Finally, the minimum time unit in our simulation is one minute, contrary to the coarse time granularity in our analysis.

## 4.2 Simulation Results

We present simulation results in the following five performance metrics: (1) number of remaining requesting peers $M(k)$, (2) total committed peer-to-peer streaming capacity $S(k)$, (3) instantaneous peer-to-peer streaming capacity $N(k)$, (4) streaming request rejection rate, and (5) number of sessions a supplying peer actually fulfills. For each of these metrics, we will first compare the simulation results with our analytical results. We next compare the performance under *different* handoff times and validate our choice of $k_0$. We then compare the performance under our standard configuration (i.e. handoff at $k_0$ and $x_i$'s set as in Section 4.1) and in the case when $x_i = c_i$ ($i = 1, 2, 3$). This will confirm our observation in Section 3.3 that the system capacity will *not* grow if $x_i = c_i$. Finally, we show that there is no significant performance gain if $x_i = \infty$ (i.e. supplying peers never retire), justifying our limited contribution policy for each supplying peer.

**(1) Remaining requesting peers** $M(k)$ This performance metric indicates how fast the media file is distributed to the clients. Figure 5 shows the decrease of $M(k)$ during the first 60 hours after the media file is released.

In figure 5(a), the simulation results closely match the numerical results based on our analysis. It indicates the validity of our derivation of $M(k)$, despite the coarse time granularity used in our analysis. Figure 5(b) shows the impact of different handoff times. $k_0 = 9$ is the calculated handoff time. $k_0/4$ and $k_0/2$ represent handoff times which are too early, while $2k_0$ represents a handoff time which is too late. As a comparison, we also show the performance when there is no peer-to-peer streaming ('CDN only'). From Figure 5(b), we observe that the extreme case of 'CDN only' results in linear and the slowest decrease of $M(k)$ (i.e. the slowest progress of media distribution). For our hybrid architecture, a handoff earlier than $k_0$ is premature ($k_0/4$ or $k_0/2$): it leads to a slower progress of media distribution. On the other hand, a handoff after $k_0$ does *not* help speeding up the media distribution progress: the curves for 'handoff at $k_0$' and the curve for 'handoff at $2k_0$' almost overlap.

In Figure 5(c), we show the negative impact of setting $x_i = c_i$ ($i = 1, 2, 3$). In this case, if the handoff still takes place at $k_0$, the number of remaining requesting peers will exhibit a much slower and more linear decrease. Even if the CDN server stays longer and does not hand off until $2k_0$, the decrease in the number of remaining requesting peers is still slower than under our standard configuration, not to mention the additional cost of the CDN server during $[k_0, 2k_0]$. In Figure 5(d), we show the performance when $x_i = \infty$ ($i = 1, 2, 3$), i.e. the supplying peers never retire. We can see that a system with infinite $x_i$'s does not result in significant speed-up of media distribution progress. This justifies the limited contribution policy in our hybrid architecture.

**(2) Total committed peer-to-peer streaming capacity** $S(k)$ $S(k)$ represents the total 'reserve' of peer-to-peer streaming capacity in the system. Note again that it is *not* the instantaneous streaming capacity. Figure 6 shows the growth of $S(k)$ during the first 60 hours.

In Figure 6(a), the numerical results from our analysis again match our simulation results. Figure 6(b) shows the impact of different hand-off times on the growth of $S(k)$. An early handoff (at $k_0/4$ or $k_0/2$) results in slower growth of $S(k)$, which partly explains the slower progress in media distribution shown in Figure 5(b). On the other hand, we observe that a late handoff (at $2k_0$) does not significantly increase the total committed peer-to-peer streaming capacity.

In Figure 6(c), we show that the total committed streaming capacity does *not* grow, if $x_i = c_i$ ($i = 1, 2, 3$). Instead, $S(k)$ *decreases* with the elapse of time. More specifically, we notice that the handoff times are very close to the turning

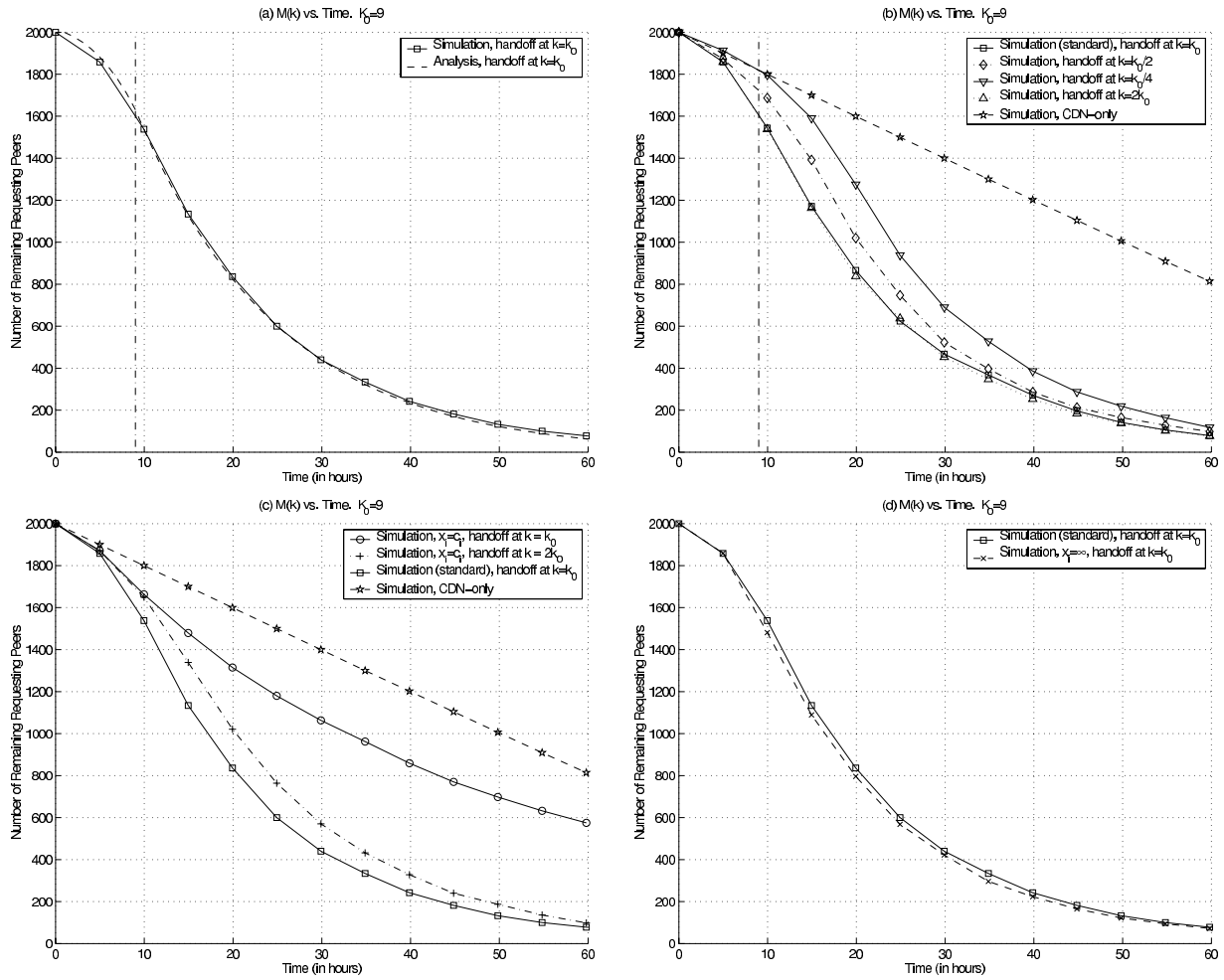


Figure 5: Number of remaining requesting peers

points when $S(k)$ begins to decrease. This shows that if $x_i = c_i$ $(i = 1, 2, 3)$, the initial increase in $S(k)$ is only due to the presence of CDN streaming capacity. Once the CDN server stops providing streaming service, $S(k)$ will collapse. This partly explains the slower progress in media distribution shown in Figure 5(c).

**(3) Instantaneous peer-to-peer streaming capacity** $N(k)$ The instantaneous streaming capacity represents the truly usable streaming capacity at any time $k$. In our analysis, we do not derive an accurate form for $N(k)$. Instead, we derive its lower bound as $\frac{\rho S(k)}{r}$. Figure 7 shows the growth of $N(k)$ in the simulation.

Figure 7(a) shows (i) the actual $N(k)$ in the simulation, (ii) the actual $\frac{\rho S(k)}{r}$ in the simulation, and (iii) the numerical $\frac{\rho S(k)}{r}$ from our analysis. We first notice that the curves for (ii) and (iii) match very well, again justifying the validity of our analysis. In addition, we also observe the actual $N(k)$ is constantly higher than $\frac{\rho S(k)}{r}$ - the lower bound of $N(k)$. Although the lower bound is not a tight one, the difference is not significant especially during the first stage when $k < k_0$. Moreover, by using the lower bound of $N(k)$, our calculation of $k_0$ is more conservative and safer.

In Figure 7(b), we show the impact of handoff time on the growth of $N(k)$. If the handoff happens too early (at $k_0/4$ or $k_0/2$), the growth of $N(k)$ will be slower, and $N(k)$ will be constantly lower than in the case of handoff at $k_0$. On the other hand, if the handoff happens too late (at $2k_0$), there will be no significant increase in $N(k)$. Since $N(k)$ is the instantaneously available streaming capacity, our results explain the impact of handoff time on the progress of media data distribution shown in Figure 5(b), as well as on the streaming request rejection rate to be shown in Figure 8(a).

Figure 7(c) shows the negative impact of having $x_i = c_i$ $(i = 1, 2, 3)$. After the handoff time (at $k_0$ or $2k_0$), the

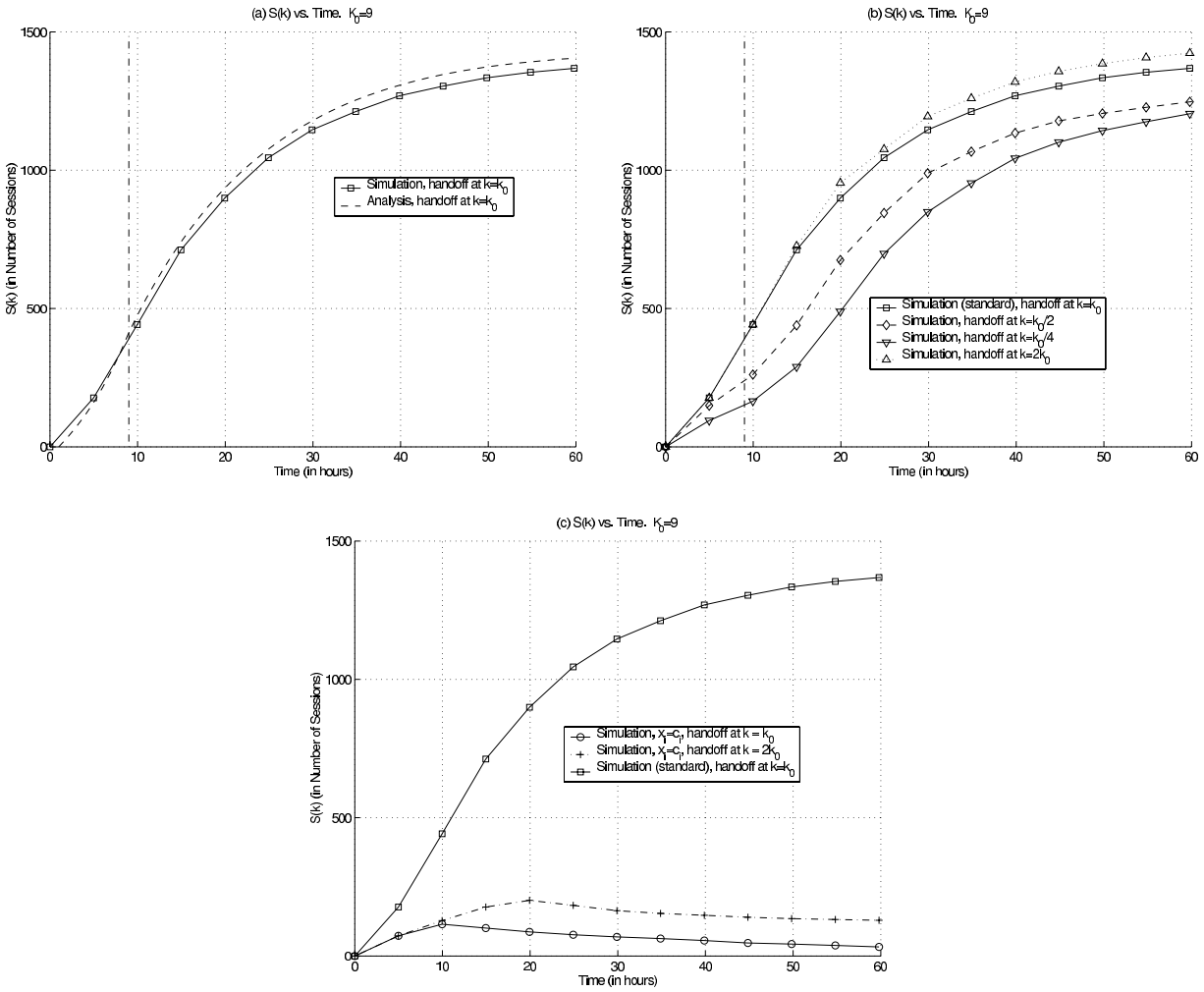


Figure 6: Total committed peer-to-peer streaming capacity $S(k)$

instantaneous streaming capacity begins to decline, without the support from CDN-based streaming. In figure 7(d), the system with $x_i = \infty$ $(i = 1, 2, 3)$ accumulates more instantaneous peer-to-peer streaming capacity than under our standard simulation configuration. However, this higher capacity does not translate into faster media distribution (Figure 5(d)). The reason is that the $N(k)$ under our standard configuration is already sufficient to handle the streaming requests. Therefore, much of the accumulated $N(k)$ in the case of $x_i = \infty$ will be left unused.

**(4) Request rejection rate** To reveal more details of the system dynamics, we show the per hour streaming request rejection rate in Figure 8.

In Figure 8(a), we show the per hour request rejection rate under different handoff times. As a comparison, we also show the rejection rate in the case without peer-to-peer streaming ('CDN only'). The CDN only system results in the slowest decrease in request rejection rate, due to the fixed and limited CDN streaming capacity. For our hybrid architecture, if the handoff time is too early, the rejection rate will remain high many hours after the handoff. If the handoff takes place at $k_0$, the rejection rate will drop to 0% only three hours after handoff. However, if the handoff happens at $2k_0$, the rejection rate will virtually be the same as in the case of handoff at $k_0$. Once again, this demonstrates the cost-effective choice of $k_0$.

In Figure 8(b), we show the request rejection rates when $x_i = c_i$ $(i = 1, 2, 3)$. We observe that if the handoff happens at $k_0$, the rejection rate will *not* drop over time, due to the decrease in peer-to-peer streaming capacity (as shown in Figures 6(c) and 7(c)). Even if the handoff takes place at $2k_0$, the drop in rejection rate is still considerably slower than in our standard simulation configuration, not to mention the additional cost of CDN server in $[k_0, 2k_0]$.

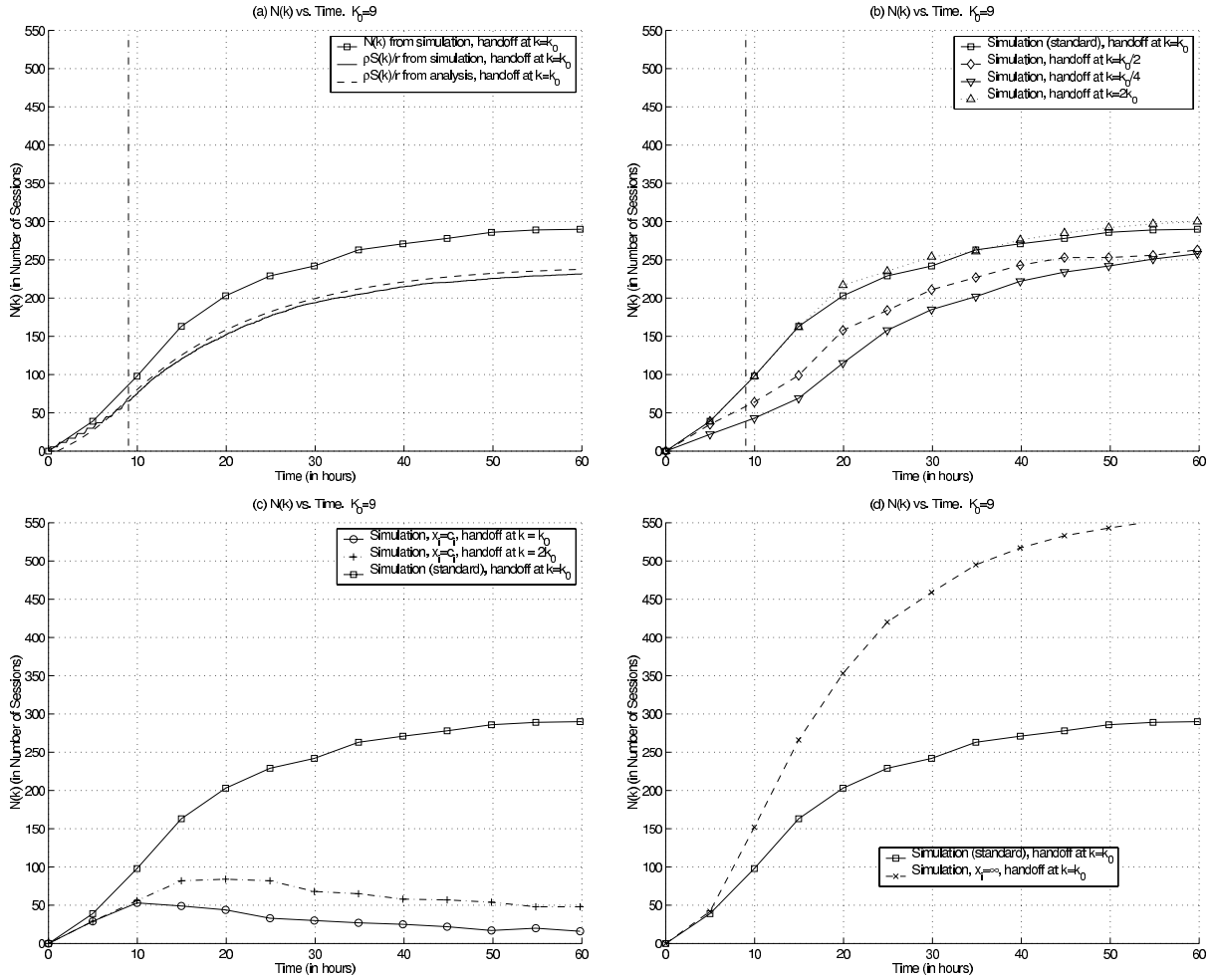


Figure 7: Instantaneous peer-to-peer streaming capacity $N(k)$

This once again confirms our observation that letting $x_i = c_i$ will lead to the collapse (instead of growth) of the hybrid system. In Figure 8(c), we compare the request rejection rate under our standard configuration and under infinite $x_i$'s. From the results, we realize that having infinite $x_i$'s does not significantly help lowering the rejection rate. This again confirms that the higher $N(k)$ under infinite $x_i$'s (Figure 7(d)) is not necessary, due to the declining streaming request rate.

**(5) Actual number of sessions served** Although each class $i$ supplying peer is committed to participate in $x_i$ streaming sessions, it may not be able to fulfill the commitment during the first 60 hours, due to the decreasing streaming request rate. For each class of supplying peers, Figure 9 shows the cumulative distribution of the actual number of sessions served: Figure 9(a) shows the results under our standard simulation configuration, while Figure 9(b) shows the results when $x_i$ is set to infinity. In the *latter* case, we observe that more than 40% of the class 1 supplying peers serve more than 3 sessions. However, under our standard configuration, we have $x_1 = 3$, indicating that any class 1 supplying peer will never serve more than 3 sessions. We also notice that in the latter case, more than 20% of the class 2 supplying peers serve more than 6 sessions - the corresponding $x_2$ under our standard configuration. The results reveal another undesirable consequence of letting $x_i = \infty$: it leads to unfairness toward peers contributing higher out-bound bandwidth (such as Class 1 and Class 2 peers in our simulation).

In summary, our simulation results confirms the validity of our analysis. Furthermore, our hybrid architecture with the limited contribution policy proves to be highly cost-effective: it results in faster media distribution progress and lower request rejection rate. Meanwhile, it lowers the CDN server cost by freeing the resources for the media file after handoff; and it only exploits the streaming capacity of supplying peers on a fair and limited basis.

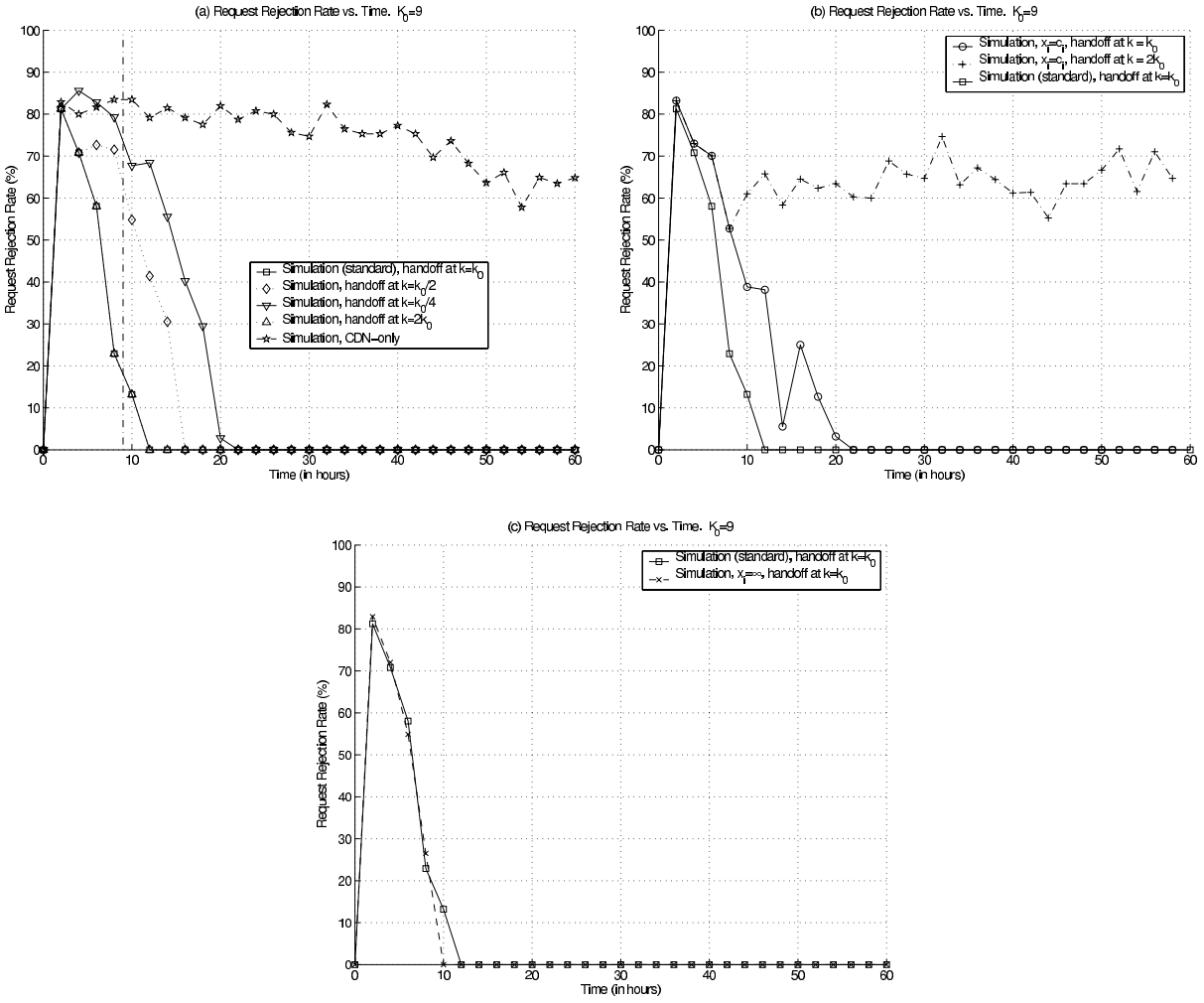


Figure 8: Per hour request rejection rate

# 5 Related Work

Content Distribution Networks have been successfully deployed on the Internet such as the one operated by Akamai. Technical issues of CDN have also been extensively studied. For example, [7] studies the efficient transport of content from their original sources to the multiple CDN servers; [11] addresses the problem of object replication and placement in a CDN; [6] discusses the dynamic brokering of CDN server capacity; and [5] presents flexible media data coding for CDNs. Our work instead focuses on improving the 'last-hop' distribution in a CDN, i.e. media streaming from a leave CDN server to the clients it serves.

Peer-to-peer systems have attracted tremendous research attention in the past two years. In [19], a detailed measurement study on the two most popular peer-to-peer systems - Napster [3] and Gnutella [2] is presented. New experimental peer-to-peer systems have also been presented. Examples include CFS [9] on top of Chord [15] and PAST [18] on top of Pastry [17]: the lower substrate provides peer-to-peer lookup service, while the upper layer provides peer-to-peer storage service. These systems do not specifically target streaming media distribution (although they can potentially be programmed to do so). One difference between our architecture and these systems lies in the lookup mechanism. We choose to use the CDN server as the centralized index server because it is a natural candidate.

More recently, peer-to-peer media streaming systems have also been proposed. C-star [1] is a commercial system which enables media streaming from multiple suppliers to one receiver. However, its cost-effectiveness is not clear due to a lack of technical details. In [13], the feasibility of streaming media from multiple senders is also reported.

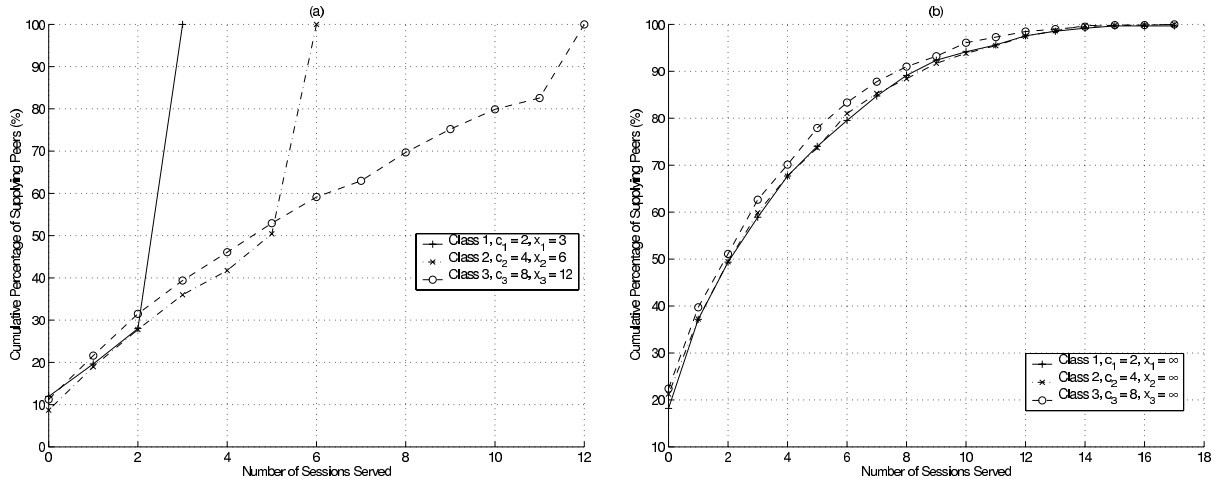


Figure 9: Cumulative distribution of the actual number of sessions served during the first 60 hours

However, the paper does not address the media *distribution* issue. CoopNet [14] is a scalable mechanism where clients cooperate to distribute streaming media content when the CDN server is overloaded. Like our architecture, it also advocates the co-existence of CDN and peer-to-peer based media distribution. However, CoopNet does not lead to a permanent 'CDN to peer-to-peer' handoff, due to its purpose to handle the temporary 'flash crowd' situation, as well as a more dynamic client participation. PeerCast [10] is an approach to streaming live media via an overlay tree formed by the clients themselves. PeerCast may not co-exist with a CDN system, nor does it introduce any policy to realize limited and fair peer contribution. In [12], a programming platform is presented to support better implementation of peer-to-peer multimedia services, including the implementation of our hybrid architecture. Finally, our earlier work [20] solves the problem of assigning media data to supplying peers in a peer-to-peer streaming session. However, it assumes that the supplying peers will stay in the system indefinitely, and it does not consider the integration of CDN and peer-to-peer media distribution.

# 6 Conclusion

We have presented a hybrid architecture which integrates the CDN and peer-to-peer based streaming media distribution. Under this architecture, we also propose a limited contribution policy for supplying peers, so that the streaming capacity of supplying peers can be exploited on a limited and fair basis. We perform an in-depth analysis of the system dynamics which involves a 'CDN to peer-to-peer' handoff. Both our analysis and simulation results demonstrate that the architecture is highly cost-effective. Our ongoing work involves the extension to the current analysis framework, so that we can model more complex and dynamic scenarios in the hybrid architecture.

# References

[1] C-star. *http://www.centerspan.com/*.

[2] Gnutella. *http://gnutella.wego.com*.

[3] Napster. *http://www.napster.com*.

[4] D. Rubenstein A. Stavrou and S. Sahu. A Lightweight, Robust P2P System to Handle Flash Crowds. *Columbia University Technical Report EE020321-1*, February 2002.

[5] J. Apostolopoulos, T. Wong, S. Wee, and D. Tan. On Multiple Description Streaming with Content Delivery Networks. In *Proceedings of IEEE INFOCOM 2002*, June 2002.

[6] A. Biliris, C. Cranor, F. Douglis, M. Rabinovich, S. Sibal, O. Spatscheck, and W. Sturm. CDN Brokering. *Proceedings of the International Workshop on Web Caching and Content Distribution (WCW 2001)*, June 2001.

[7] Y. Chawathe. Scattercast: an Architecture for Internet Broadcast Distribution as an Infrastructure Service. *Ph.D. Thesis, University of California at Berkeley*, 2000.

[8] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. E. Knuth. On Lambert's W function. *Adv. Computational Maths.*, 1996.

[9] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-Area Cooperative Storage with CFS. *Proceedings of ACM SOSP 2001*, October 2001.

[10] H. Deshpande, M. Bawa, and H. Garcia-Molina. Streaming Live Media over Peers. *Stanford Database Group Technical Report (2002-21)*, March 2002.

[11] J. Kangasharju, J. Roberts, and K. Ross. Object Replication Strategies in Content Distribution Networks. *Computer Communications*, 25(4), 2002.

[12] R. Lienhart, M. Holliman, Y. Chen, I. Kozintsev, and M. Yeung. Improving Media Services on P2P Networks. *IEEE Internet Computing*, January 2002.

[13] T. P. Nguyen and A. Zakhor. Distributed Video Streaming Over Internet. *Proceedings of SPIE/ACM MMCN 2002*, January 2002.

[14] V. N. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. *Proceedings of NOSSDAV 2002*, May 2002.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. *Proceedings of ACM SIGCOMM 2001*, August 2001.

[16] R. Rejaie and J. Kangasharju. Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming. *Proceedings of NOSSDAV 2001*, June 2001.

[17] A. Rowstron and P. Druschel. Pastry: Scalable Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. *Proceedings of IFIP/ACM Middleware 2001*, November 2001.

[18] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility. *Proceedings of ACM SOSP 2001*, October 2001.

[19] S. Saroiu, P. Gummadi, and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. *Proceedings of SPIE/ACM MMCN2002*, January 2002.

[20] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On Peer-to-Peer Media Streaming. *Proceedings of IEEE ICDCS 2002*, July 2002.