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

Dongyan Xu\textsuperscript{a}, Heung-Keung Chai\textsuperscript{b}, Catherine Rosenberg\textsuperscript{b}, Sunil Kulkarni\textsuperscript{b}

\textsuperscript{a}Department of Computer Sciences
\textsuperscript{b}School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907

ABSTRACT

To distribute video and audio data in real-time streaming mode, both Content Distributed Network (CDN) based and peer-to-peer (P2P) 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 P2P 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 P2P 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.

Keywords: Content Distribution Networks, Peer-to-peer Systems, Media Streaming, Distributed Systems

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 (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 (P2P) 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 P2P 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\textsuperscript{1} 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

\textsuperscript{1}Authors' contact information:
\textsuperscript{a}dxu@cs.purdue.edu (Main correspondent)
\textsuperscript{b}\{chah, cath, sunilkul\}@ecn.purdue.edu
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, P2P 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\footnote{The clients are also called requesting peers. We will use the terms ‘peer’ and ‘client’ interchangeably for the rest of the paper.} directly, without involving a CDN server. Such P2P 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 P2P 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 literature on P2P-related researches.

In this paper, we propose a novel hybrid architecture that integrates CDN and P2P 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 P2P 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. We conclude this work in Sect. 6.

2. SYSTEM ARCHITECTURE AND OPERATIONS

2.1. System Architecture

The proposed hybrid architecture is shown in Fig. 1. We only show one CDN server\footnote{The CDN server is a logical entity - it may consist of multiple physical servers.} 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 P2P index server. For each media file, the CDN server stores the file before the ‘CDN to P2P’ 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 P2P systems do not define the third phase, i.e. they assume that a ‘well-behaved’ supplying peer will always contribute.

Different from CDN-based streaming, a P2P streaming session involves multiple supplying peers (as shown in Fig. 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 our previous work, 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

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{Different stages of the system}
\end{figure}
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 Fig. 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 Sect. 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 P2P based streaming co-exist, while the P2P streaming capacity grows (Stage I in Fig. 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 P2P streaming capacity is not available, the request will be rejected.

Finally, the CDN server decides to handoff from CDN streaming to P2P 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 P2P 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 Fig. 2).

As to be shown in Sect. 3, a non-trivial analysis is needed to determine the handoff time. If the handoff takes place too early, the P2P 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 P2P streaming capacity growth, supplying peer contribution fulfillment, and streaming request arrivals over time.

3. SYSTEM ANALYSIS

3.1. Assumptions

To make our analysis feasible, we make the following assumptions. 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. 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 Tab. 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.

---

1We realize that this assumption may not always be valid. One counter-example is the flash crowd scenario which requires alternative solutions.², ³
\textbf{Table 1. Definitions of system parameters and performance metrics}

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>Length of one streaming session in minutes.</td>
</tr>
<tr>
<td>$k$</td>
<td>Discrete time index, each unit has a length of $L$.</td>
</tr>
<tr>
<td>$N_c$</td>
<td>CDN server capacity allocated to the media file (in number of simultaneous streaming sessions).</td>
</tr>
<tr>
<td>$M_0$</td>
<td>Total client population.</td>
</tr>
<tr>
<td>$n$</td>
<td>Number of peer classes.</td>
</tr>
<tr>
<td>$p_i$</td>
<td>Percentage of peers in the $i^{th}$ class.</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Per-client request generation rate, in requests per minute.</td>
</tr>
<tr>
<td>$c_i$</td>
<td>An integer indicating that the out-bound bandwidth contributed by a class $i$ peer is $\frac{1}{c_i}$ of the media playback rate.</td>
</tr>
<tr>
<td>$x_i$</td>
<td>Number of sessions a class $i$ peer is committed to serve. $x_i &gt; c_i$.</td>
</tr>
<tr>
<td>$k_0$</td>
<td>The ‘CDN to P2P’ handoff time.</td>
</tr>
<tr>
<td>$M(k)$</td>
<td>Number of remaining requesting peers at time $k$. $M(0) = M_0$.</td>
</tr>
<tr>
<td>$S(k)$</td>
<td>Total committed P2P streaming capacity at time $k$, in number of full streaming sessions.</td>
</tr>
<tr>
<td>$N(k)$</td>
<td>Instantaneous P2P streaming capacity at time $k$, in number of full streaming sessions.</td>
</tr>
</tbody>
</table>

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 $p_i$. In a P2P 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 $\lambda$.

The analysis is done on a discrete time scale in multiples of $L$, the duration of one streaming session, and is denoted by $k$. As to be shown in Sect. 4, our analytical results match the results from our simulations, which have a time unit of one minute.

One of the main goals of our analysis is to compute the ‘CDN to P2P’ handoff time $k_0$. $k_0$ is defined as time when the P2P 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 P2P 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 Fig. 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 $3/2 = 1.5$ (Fig. 3(b)). However, if Peer 3 and Peer 4 are chosen, $N(k+1)$ will be $5/2 = 2.5$ (Fig. 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 P2P streaming capacity (in full sessions) at $k$. We will show that $S(k)$ is much easier to model. Consider the example in Fig. 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 Figs. 3(a) and 3(b). 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 Figs. 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 Sect. 3.4, we will derive the lower bound of $N(k)$ under multiple $x_i$'s.

The rest of this section is organized as follows: In Sect. 3.3, we show the corresponding equations for two stages. In stage I when $k < k_0$, the hybrid system offers both CDN-based streaming and P2P streaming, and the total streaming capacity is lower than what is requested by the requesting peers. The system enters stage II when $k \geq k_0$. In this stage, the system only offers P2P streaming as the total streaming capacity is higher than the capacity requested. 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 Sect. 3.4. Finally, we discuss the relation among parameters $N_c, x_i$ and $k_0$ in Sect. 3.5.

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

Due to space constraint, we omit some intermediate steps and only present results that are the most important. Detailed derivation and explanations of individual equations can be found in our technical report. 5

$S(k)$ is the total committed P2P streaming capacity in number of full sessions, and it is defined piece-wise as:

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

Base on $S(k)$, we define $M(k)$, the remaining number of requesting peers at $k$ in the following equation:
\[
M(k + 1) = \begin{cases} 
M(k) - \sum_{i=1}^{n} \frac{S_i}{p_i} \sum_{j=1}^{n} \left( \frac{p_j}{c_j} \right) - N_C, & k < k_0 \\
M(k) \cdot (1 - \lambda L), & k \geq k_0
\end{cases}
\]

To simplify the analysis that follows, we introduce two variables: \( r \) and \( \rho \), as defined in Tab. 2. \( S_I(k + 1) \) is now

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

**Table 2. Definitions of \( r \) and \( \rho \)**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
<td>( \sum_{j=1}^{n} \frac{p_j}{c_j} )</td>
</tr>
<tr>
<td>( \rho )</td>
<td>( \sum_{j=1}^{n} \frac{p_j}{c_j} )</td>
</tr>
</tbody>
</table>

One key observation is that \( r \) can be thought of as the normalized average ‘contribution ratio’ of all supplying peers: the higher the \( r \), the more the contribution peers make, and vice versa. Naturally, there are 3 cases to consider: if \( 0 < r < 1 \), then the peers are contributing less than they receive and the system capacity is decreasing. In the \( r = 1 \) case, the peers are contributing the exact amount of sessions they received, and when \( r > 1 \), peers are making extra contributions to the capacity of the system.

For \( r = 1 \) case, Eq. (1) is simply \( S_I(k) = k N_c \). If \( r \neq 1 \), the equation is solved as:

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

In this paper, we will assume \( r > 1 \) for all subsequent analysis. By using Eq. (2), the closed form of \( M_I(k) \) can then be obtained as a function of \( S_I(k) \):

\[
M_I(k) = M_0 + \frac{k N_c - S_I(k)}{r - 1}, r > 1. \tag{3}
\]

Note that by definition, \( M_{II}(k_0) = M_{I}(k_0) \). Therefore, \( M_{II}(k) \) can be solved as:

\[
M_{II}(k) = M_I(k_0) \left( 1 - \lambda L \right)^{k - k_0}, \; k \geq k_0. \tag{4}
\]

With Eq. (4), it is easy to derive \( S_{II}(k) \):

\[
S_{II}(k) = S_I(k_0) + M_I(k_0) (r - 1) \left[ 1 - (1 - \lambda L)^{k - k_0} \right], \; k \geq k_0. \tag{5}
\]
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 $S_I(k)$. Intuitively, this means the committed contribution of each peer - if measured by the total volume of data it sends out, is greater than the total volume of data it receives (i.e. the volume of the media file).

### 3.4. Derivation of Handoff Time $k_0$

By definition, starting at time $k_0$, the P2P streaming capacity alone should 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 P2P streaming capacity at $k_0$ multiplied by a conservative factor $\alpha$ ($0 < \alpha \leq 1$):

$$\lambda L M_I(k_0) = \alpha N_I(k_0). \quad (6)$$

As shown in Sect. 3.2, $N_I(k)$ is 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 P2P 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 Eq. (6) to derive $k_0$:

$$\lambda L M_I(k_0) = \frac{\rho S_I(k_0)}{r}. \quad (7)$$

By replacing $M_I(k)$ using Eq. (3), Eq. (7) can be re-arranged as:

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

After some algebraic manipulations, we have

$$k_0 = \frac{-c \cdot \log (a) - b \cdot W \left( -\frac{\log (a) - e^{\frac{c \log a}{b}}}{b} \right)}{b \cdot \log (a)}. \quad (8)$$

In eq. (8), $a = (1 + \rho \frac{r-1}{r})$, $b = \frac{\frac{\lambda L M_0}{(\frac{\lambda L M_0}{N_c}) N_c + \lambda L (\frac{\lambda L M_0}{N_c}) \frac{r}{r-1}}}{\alpha (\frac{\lambda L M_0}{N_c}) N_c + \lambda L (\frac{\lambda L M_0}{N_c}) \frac{r}{r-1}}$ and $c = \frac{\lambda L M_0}{\alpha (\frac{\lambda L M_0}{N_c}) N_c + \lambda L (\frac{\lambda L M_0}{N_c}) \frac{r}{r-1}} + 1$. $W(\cdot)$ is the Lambert’s W-function. Detailed definition of this function can be found in the references and omitted in this paper.

### 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’ $r = \sum_{j=1}^{n} p_j (\frac{x_j}{c_j})$.

Based on Eq. (8), Fig. 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 remain 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 $r$. As $r$ increases, $k_0$ quickly levels off and this justifies our limited contribution policy for the supplying peers: beyond a certain point, any further increase in $r$ will not yield any improvement on the handoff time. 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 Eq. (8), we can derive the earliest possible handoff time as

$$\lim_{{r \to \infty}} k_0 = \frac{\log \left( \frac{\lambda L M_0}{\alpha N_c + \lambda L \frac{r}{r-1}} + 1 \right)}{\log (1 + \rho)}. \quad (9)$$
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, \( r = \frac{2}{2} \times 20\% + \frac{4}{4} \times 50\% + \frac{8}{8} \times 30\% = 1.5 \)). The factor \( \alpha \) used in the calculation of \( k_0 \) is 1.0. Finally, the 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 P2P streaming capacity \( S(k) \), 3. instantaneous P2P 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 Sect. 4.1) and in the case when \( x_i = c_i \) \((i = 1, 2, 3)\). 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 Fig. 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 P2P streaming ("CDN only"). From Fig. 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 Fig. 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 Fig. 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 P2P streaming capacity \( S(k) \) \( S(k) \) represents the total ‘reserve’ of P2P 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.

Our analysis results match surprisingly well with the results obtained from the simulations and the relevant figure is available in our technical report.\(^5\) Figure 6(a) 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 Fig. 5(b). On the other hand, we observe that a late handoff (at 2k₀) does not significantly increase the total committed P2P streaming capacity.

In Fig. 6(b), we show that the total committed streaming capacity does not grow, if xᵢ = cᵢ (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 points when S(k) begins to decrease. This shows that if xᵢ = cᵢ (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 Fig. 5(c).

(3) Instantaneous P2P 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 \( \frac{dS(k)}{dr} \). 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 \( \frac{dS(k)}{dr} \) in the simulation, and (iii) the numerical \( \frac{dS(k)}{dr} \) 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 \( \frac{dS(k)}{dr} \) - 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 < k₀. Moreover, by using the lower bound of N(k), our calculation of k₀ is more conservative and safer.

In Fig. 7(b), we show the impact of handoff time on the growth of N(k). If the handoff happens too early (at k₀/4 or k₀/2), the growth of N(k) will be slower, and N(k) will be constantly lower than in the case of handoff at k₀. On the other hand, if the handoff happens too late (at 2k₀), there will be no significant increase in N(k). Since N(k) is the instantaneous available streaming capacity, our results explain the impact of handoff time on the progress of media data distribution shown in Fig. 5(b), as well as on the streaming request rejection rate to be shown in Fig. 8(a).

Figure 7(c) shows the negative impact of having xᵢ = cᵢ (i = 1, 2, 3). After the handoff time (at k₀ or 2k₀), the instantaneous streaming capacity begins to decline, without the support from CDN-based streaming. In Fig. 7(d), the system with xᵢ = \( \infty \) (i = 1, 2, 3) accumulates more instantaneous P2P streaming capacity than under our standard simulation configuration. However, this higher capacity does not translate into faster media distribution (Fig. 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 xᵢ = \( \infty \) 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 Fig. 8.
In Fig. 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 P2P 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 2\( k_0 \), 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 Fig. 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 P2P streaming capacity (as shown in Figs. 6(c) and 7(c)). Even if the handoff takes place at 2\( k_0 \), 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 [\( k_0, 2k_0 ] \). From extensive simulation results, we compare the request rejection rate under our standard configuration and under infinite \( x_i \)'s. We realize that having infinite \( x_i \)'s does not significantly help lowering the rejection rate. Indeed, this can be seen from Fig. 5(d): there is no significant differences in the number of remaining peers in the ‘standard’ settings and in the case where \( x_i = \infty \) at all times. Again, this confirms that the higher \( N(k) \) under infinite \( x_i \)'s (Fig. 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, Fig. 9 shows the cumulative distribution of the actual number of sessions served: Figure 9(a) shows the results under our standard simulation configuration, while Fig. 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.
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, work by Yatin et al.\(^7\) studies the efficient transport of content from their original sources to the multiple CDN servers; Kangasharju et al.\(^8\) address the problem of object replication and placement in a CDN; Bliiris et al.\(^9\) discusses the dynamic brokering of CDN server capacity; and Apostolopoulos et al.\(^10\) present 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.

P2P systems have attracted tremendous research attention in the past two years. In the work by Saroiu et al.,\(^11\) a detailed measurement study on the two most popular P2P systems - Napster\(^12\) and Gnutella\(^13\) is presented. New experimental P2P systems have also been presented. Examples include CFS\(^14\) on top of Chord\(^15\) and PAST\(^16\) on top of Pastry\(^17\): the lower substrate provides P2P lookup service, while the upper layer provides P2P 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, P2P media streaming systems have also been proposed. C-star\(^18\) 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 the work by Nguyen and Zakhor,\(^19\) the feasibility of streaming media from multiple senders is also reported. However, the paper does not address the media distribution issue. CoopNet\(^4\) 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 P2P based media distribution. However, CoopNet does not lead to a permanent ‘CDN to P2P’ handoff, due to its purpose to handle the temporary ‘flash crowd’ situation, as well as a more dynamic client participation. PeerCast\(^20\) 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. A programming platform is presented by Lienhart et al.\(^21\) to support better implementation of P2P multimedia services, including the implementation of our hybrid architecture. Finally, our earlier work\(^2\) solves the problem of assigning media data to supplying peers in a P2P 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 P2P media distribution.

6. CONCLUSION

We have presented a hybrid architecture which integrates the CDN and P2P 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 P2P’ 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.

ACKNOWLEDGMENTS

This work has been supported in part by the grant from the 21st Century Research and Technology Fund entitled ‘Entertainment Video Over the Internet’ and an exploratory research grant from the e-Enterprise Center at Discovery Park entitled ‘e-Courier: a Service for Enterprise Data Delivery’.

REFERENCES