Multimedia Systems (2006)
DOI 10.1007/s00530-006-0015-3

REGULAR PAPER

Dongyan Xu · Sunil Suresh Kulkarni · Catherine Rosenberg · Heung-Keung Chai

Analysis of a CDN–P2P hybrid architecture for cost-effective streaming media distribution

Abstract To distribute video and audio data in real-time streaming mode, two different technologies – Content Distribution Network (CDN) and Peer-to-Peer (P2P) – have been proposed. However, both technologies have their own limitations: CDN servers are expensive to deploy and maintain, and consequently incur a cost for media providers and/or clients for server capacity reservation. On the other hand, a P2P-based architecture requires sufficient number of *seed* supplying peers to *jumpstart* the distribution process. Compared with a CDN server, a peer usually offers much lower out-bound streaming rate and hence multiple peers must jointly stream a media data to a requesting peer. Furthermore, it is not clear how to determine how much a peer should contribute back to the system after receiving the media data, in order to sustain the overall media distribution capacity.

In this paper, we propose and analyze a novel *hybrid architecture* that integrates both CDN- and P2P-based streaming media distribution. The architecture is highly cost-effective: it significantly lowers the cost of CDN capacity reservation, without compromising the media quality delivered. In particular, we propose and compare different *limited contribution policies* for peers that request a media data, so that the streaming capacity of each peer can be exploited on a fair and limited basis. We present: (1) in-depth analysis of the proposed architecture under different contribution policies, and (2) extensive simulation results which validate the analysis. Our analytical and simulation results form a rigorous basis for the planning and dimensioning of the hybrid architecture.

Keywords Peer-to-peer systems · Multimedia streaming · Content distribution networks

D. Xu (✉)
Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA
E-mail: dxu@cs.purdue.edu

S. S. Kulkarni
Google Inc., Mountain View, CA 94043, USA
E-mail: sunilkul@gmail.com

C. Rosenberg
Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
E-mail: cath@ece.uwaterloo.ca

H.-K. Chai
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA
E-mail: chai@ecn.purdue.edu

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 technologies is the Content Distribution Network (CDN), where a 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, peer-to-peer (P2P)-based media distribution architectures have quickly gained popularity, where clients store the media data after the streaming service, and act as supplying peers by streaming the media data to other requesting clients (peers). However, we argue that both CDN- 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 architecture, a media file is first pushed to multiple CDN servers, each of which serves clients in its designated domain(s). A CDN server has dedicated storage space and out-bound bandwidth for high-quality media streaming. However, CDN servers are expensive to deploy and maintain. The server capacity (including processing power and out-bound bandwidth) that can be allocated to the distribution of one media file is limited, and it incurs a non-trivial cost to the provider and/or clients of this media file. For example, until recently, users had to pay a subscription fee to

view streaming videos on *CNN.com*. There exist solutions to CDN cost control, which adaptively degrade the media quality according to the rate of client requests, and therefore bounding the CDN server capacity requirement. The downside of such solutions is that they compromise the quality of service received by individual clients.

On the other hand, P2P media streaming exhibits a more de-centralized nature: After clients receive the media data, they will act as supplying peers and stream the data to other requesting clients (Note that we will use the terms *peers* and *clients* interchangeably for the rest of this paper). A P2P streaming session takes place between peers, without involving a CDN server. Such a P2P architecture exploits the *growing* aggregated streaming capacity of individual supplying peers, and therefore provides a more economical way to disseminate the media content among peers. However, the P2P architecture has its own problems. First, it needs a sufficient number of *seed* supplying peers to *jumpstart*. Second, compared with a CDN server, a peer is only able or willing to offer a much lower out-bound streaming rate; probably lower than the playback rate of the media data. Finally, it is not clear how much a peer should contribute back to the system in order to sustain the aggregated media distribution capacity, while maintaining fairness among peers. To the best of our knowledge, this problem has not been addressed in the context of P2P streaming media distribution.

In this paper, we propose a novel *hybrid architecture* that integrates CDN- and P2P-based streaming media distribution. In this architecture, the two streaming technologies complement each other: When a media file needs to be distributed to a community of clients, the file will first be distributed by a CDN server;[1] and a fraction of the CDN server capacity will be reserved for this media data. While fulfilling streaming requests from clients, the CDN server will create the *seed* supplying peers in its service area. Together, the CDN server and the dynamically created supplying peers serve the streaming media requests with much higher capacity than that of the server alone. More importantly, when the P2P streaming capacity grows to a certain level, the CDN server can even stop serving streaming sessions for this media file and let the peers take over the task. We call this transition a "CDN-to-P2P" *handoff*. After the handoff, the reserved CDN server capacity for this media data can be *released*, saving the cost for the provider and/or clients.

Both the *jumpstarting* and *growth* of media streaming capacity are key to the cost-effectiveness of the hybrid architecture; and they are critically dependent on the contributions made by individual peers. In this paper, we propose three *limited contribution policies*: Each peer, upon the receipt of the media file, becomes a supplying peer but only commits a limited out-bound streaming rate to each streaming session. A supplying peer is also committed to serve (1) a limited number of P2P streaming sessions, or (2) P2P streaming sessions within a limited period of time, or (3) a combination of both (1) and (2). After fulfilling its commitment, the supplying peer can discard the media file and retire from the distribution process. Each of these policies provides some level of fairness among the supplying peers: the higher the committed out-bound streaming rate, the lower the committed number of sessions and/or the shorter the period of service time.

The main contributions of this paper are as follows: (1) The proposed hybrid architecture combines CDN and P2P technologies with *integrated* capacity planning and runtime operations. (2) The suite of limited contribution policies advocate and reflect fairness toward peers. (3) The analysis and simulations reveal the impact of different policies and parameters on the progress, cost, and peer load of a media distribution process, and therefore provide a rigorous basis for the dimensioning of the hybrid architecture and for the design of other variations of the contribution policy.

The roadmap of the paper is as follows: Sect. 2 presents an overview of the hybrid architecture, including its components, operations, and policies. In Sect. 3, we present the *session-based* limited contribution policy, as well as a detailed quantitative analysis of the policy. In particular, we derive a close-form expression for calculating the *handoff* time when the CDN server capacity reservation can be released. The analysis will also be validated by simulation results in the same section. Observing that the session-based contribution policy has certain operational inconvenience as seen by the peers, we introduce the *time-based* limited contribution policy in Sect. 4, and present its quantitative analysis along with simulation results. From this, we realize that a better design of the contribution policy should integrate both session and time commitments. Hence, we introduce an *integrated* limited contribution policy in Sect. 5 and illustrate its advantage through simulations. In the same section, we present guidelines for the planning and dimensioning of the hybrid CDN–P2P architecture. Section 6 compares our work with related work. Finally, we conclude this paper in Sect. 7.

2 System architecture and operations

2.1 System architecture

The proposed hybrid architecture is shown in Fig. 1. We only show one CDN server[2] because we focus on the interaction between one CDN server and the clients in its service area/domain(s).

– The CDN server in our architecture plays two roles: (1) the actual media streaming server and (2) the P2P index server. For the distribution of a media file, the CDN server reserves a certain amount streaming capacity for a limited period of time. Throughout the distribution process, the CDN server also maintains a list of clients

[1] In general, more than one CDN server will be involved to serve clients in different domains. In this paper, we focus on *one* of the servers as well as the clients it serves.

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

registered for the media file, as well as a list of active supplying peers (among the registered clients) and their contribution fulfillment status. Note that both the handoff and the supplying peers are specific to a media file. Before the "CDN-to-P2P" 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 media file. In addition, the CDN server is in a good position to profile each peer's contract fulfillment behavior so that free-riding peers can be identified and blocked in the system.

– On the client side, each client registered for the media file has a multi-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) After its contribution commitment has been fulfilled, it becomes a *retired peer*. We note that many current P2P systems do *not* define the third phase, which may lead to overloaded peers and unfairness among peers.

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 streaming rate contribution (possibly in different amounts) is at least the same as the media playback rate. In our earlier work [1], we present an algorithm to assign a subset of the media data to each supplying peer,

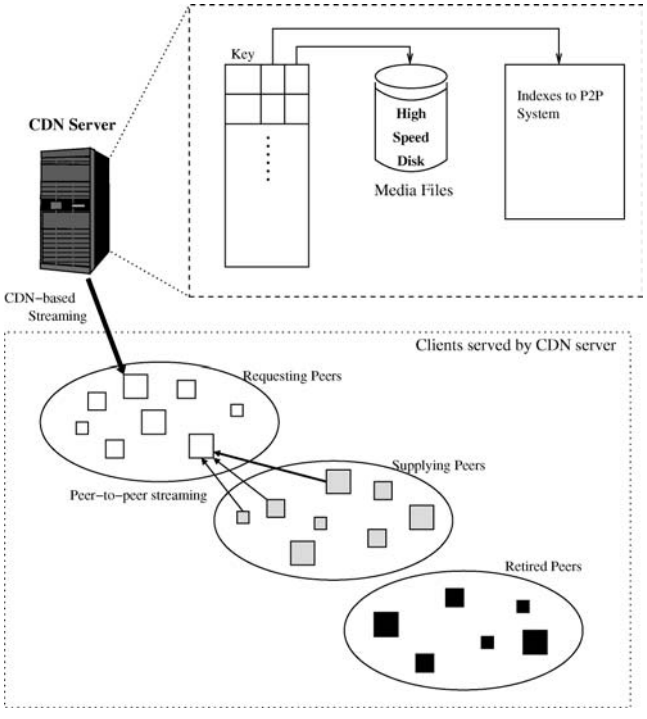

Fig. 1 The hybrid architecture for streaming media distribution (different size of peers indicating their different out-bound streaming rate contribution)

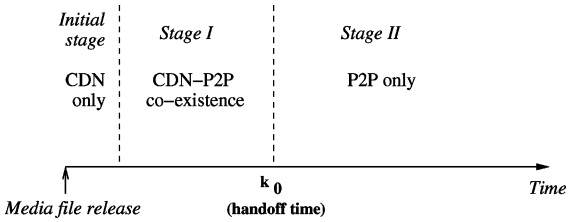

Fig. 2 Different stages of a media data distribution process

based on its out-bound streaming rate. Furthermore, our P2P streaming prototypes [2, 3] have demonstrated the feasibility of delivering full quality video by multiple supplying peers.

2.2 System operations

When the media file is first released, it is pushed to the 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 CDN server registers the client that has just received the streaming service as a supplying peer with a *limited contribution commitment* which includes: (1) a limited out-bound streaming rate for this media file and (2) a limited number of streaming sessions *or* a limited period of service time it will fulfill during its tenure as a supplying peer. To reflect fairness, the higher the amount in (1), the smaller the amount in (2) – a quantitative definition will be given in Sects. 3 and 4.

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*; and the P2P streaming capacity grows (stage I in Fig. 2). When a streaming request arrives for a given media file, the CDN server first checks if there is a set of active supplying peers for this media file such that: (1) They are not currently serving another streaming session and (2) The sum of their out-bound streaming rate is no less than the media playback rate. If so, the request will be served by the set of supplying peers selected; otherwise, the request will be served by the CDN server itself. If both CDN and P2P do not have enough streaming capacity, the request will be rejected. In the presence of multiple requests, if the current streaming capacity is not sufficient to accommodate all requests, the CDN server will perform *admission control* by accepting a *subset* of the requests. Different admission policies may be applied. For example, the system may accept requests based on the amount of out-bound bandwidth, number of sessions, or amount of service time promised by requesters. On the other hand, if the current P2P streaming capacity exceeds the capacity requested, the system will perform *supplying peer selection* by choosing a subset of the available supplying peers to serve the requests. Different selection policies also exist, for example, based on the fulfillment status of contribution commitment. Both the senders (i.e. supplying peers) and receivers (i.e. requesting peers) confirm the transaction with the CDN server at the beginning and at the end of the media streaming session; and

they report problems to the server whenever a sender is no longer able to contribute during the streaming session (e.g., when the user exits the system or when the sender experiences network condition degradation). If the latter situation happens, the server will designate other peers as replacement senders. Our earlier P2P streaming prototypes [2, 3] are capable of such dynamic sender switch without incurring streaming quality degradation.

Finally, when the P2P streaming capacity for the media file becomes sufficiently large, CDN-based streaming will no longer be provided, so that the CDN server capacity reserved for this media file can be released. The handoff time k_0 is determined such that the P2P streaming capacity alone is sufficient to handle all subsequent streaming requests, with a zero expected rejection rate. 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). In the case where the peer contribution is in the form of *service time* (rather than number of sessions), there may be a *stage III* (not shown in Fig. 2) during which the CDN server will re-join the media distribution process, using a marginal capacity, to pick up the few "tail" streaming requests that cannot be accommodated by the P2P streaming capacity.

As to be shown in Sect. 3, a non-trivial analysis is needed to determine the handoff time k_0. 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 server capacity reserved for the media file will be held longer thus incurring higher cost. Furthermore, our analysis of k_0 will create a foundation for the modeling of a more complex media distribution scenario: The content provider releases new media files on a continuous basis. Operations of the hybrid architecture will then be decomposed into *cycles* with each cycle starting at the release of a new media file. Due to the limited CDN server and P2P streaming capacity, the release of new media files needs to be *controlled*, so that the system can absorb the peak demand for one media file before the release of a new one. The analysis of k_0 will help determining the inter-release duration for more efficient utilization of CDN and P2P capacity. In this paper, we will focus on an in-depth study of a *single* cycle, namely the distribution process of one media file.

3 Session-based contribution policy

In this section, we present an analysis of the CDN–P2P hybrid architecture, under the *session-based* limited contribution policy. Especially, the analysis will determine the handoff time k_0. The derivation of k_0 is complicated by the limited contribution policy, which creates a *dynamic* population of supplying peers. The analysis will capture the system dynamics, including the growth of P2P streaming capacity, the progress of the distribution process, and the fulfillment of peer contribution contracts.

To make the analysis tractable, we make the following assumptions:

- We assume honesty and "always-on" network connections for all peers: each peer will fulfill its limited contribution contract in terms of both its committed streaming rate and number of streaming sessions to serve. We also assume that each supplying peer has sufficient disk space to store the media file being distributed.
- We adopt a "flat rate" style peer contribution contract, with peers committing to the same number of sessions regardless of time. This is for the tractability of the analysis, as well as for the simplicity of the contribution policy.
- 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 roughly justified by the fact that all clients (peers) are within the same domain served by the CDN server.
- We assume that the peer population for each media data is finite and known, which can be justified by the registration/subscription requirement in many content distribution scenarios. The streaming requests are generated independently by each requesting peer with a given Poisson rate of λ requests per time unit and per client. This, in effect, produces a finite population model with a *time-varying* overall streaming request rate for the media file.

Our analysis is based on the above assumptions, some of which do not reflect reality. Instead, the analysis will serve as the basis of an extendible framework for the analysis of more dynamic and complex systems, such as systems with dynamic peer failure/departure [2], flash crowd effect [4, 5], or time-varying peer contribution contract. Especially, in Sect. 3.6, we will extend our basic analysis to account for the system's streaming capacity losses due to peers' incomplete contract fulfillment or on-line/off-line status change.

3.1 System parameters and metrics

The system parameters and the performance metrics are summarized in Table 1. Note that they are defined with respect to the distribution of a specific media file.

Upon the release of the media file, the CDN server reserves a streaming capacity that is equal to N_c concurrent full-quality streaming sessions. In the service area of the CDN server, the total number of clients registered for the media file is M_0. When a peer requests the media file, it will commit a *limited contribution* contract with the CDN server in terms of number of streaming sessions it will serve after getting the media file. Our architecture provides n options of limited contribution for the peers: Each option includes (1) an out-bound streaming rate committed to each session, which is equal to $\frac{1}{c_i}$ $(1 \leq i \leq n)$ of the media playback

Table 1 Definitions of system parameters and performance metrics

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 number of clients registered for the media file.
n	Number of peer classes.
p_i	Percentage of peers belonging to the ith class.
λ	Per-client request generation rate, in requests per minute.
c_i	The out-bound streaming rate 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.
k_0	The "CDN-to-P2P" handoff time.
$M(k)$	Number of remaining requesting peers at time k. $M(0) = M_0$
$S(k)$	Total *committed* P2P streaming capacity at time k, in number of *full* streaming sessions.
$N(k)$	*Instantaneous* P2P streaming capacity at time k, in number of *full* streaming sessions.

rate[3] and (2) a total of x_i sessions it will participate in to serve other peers. Correspondingly, the client population is divided into n classes: a class-i peer chooses option i. The percentage of class-i peers among all registered peers 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 streaming rate is equal to or greater than 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 λ.

The analysis is based on a discrete time scale in multiples of L, the duration of one streaming session, and is denoted by k. Even though this is a relatively coarse granularity of time, the analysis closely matches our simulation results based on a much finer time granularity as shown later in this section.

One of the main goals of our analysis is to compute the "CDN-to-P2P" handoff time k_0. More precisely, k_0 is defined as the time when the P2P streaming capacity alone is able to fulfill (expectedly) all subsequent streaming requests, so that the CDN server capacity N_c can be released. Starting from k_0, the request rejection rate should remain close to zero. To derive k_0, we need to derive the following quantities: (1) $N(k)$ – the total instantaneous P2P streaming capacity at time k, and (2) $M(k)$ – the number of remaining requesting peers at time k.

Unfortunately, the accurate form of $N(k)$ turns out to be extremely difficult – if at all possible, to derive. This is because $N(k)$ is not a deterministic quantity and is dependent on the random file request process and the progress of the contribution fulfillment of each *individual* supplying peer. To illustrate the difficulty in deriving $N(k)$, consider the example in Fig. 3. Suppose that 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, the four supplying peers 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, the value of $N(k+1)$ will be different, 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. 3b). However, if *Peer 3* and *Peer 4* are chosen, $N(k+1)$ will be $5/2 = 2.5$ (Fig. 3c). Different selections of supplying peers lead to different progress of their contribution contract fulfillment, making the value of $N(k+1)$ difficult to keep track of.

To get around the difficulty in deriving $N(k)$, we instead derive a *lower bound* of $N(k)$. First, we define $S(k)$ as the total *committed* (namely "instantaneous" + "future") P2P streaming capacity, in *full* sessions, at time k. $S(k)$ 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 are selected to serve *Peer 5*, $S(k+1)$ will be $S(k)-1+3/2 = 3.5$, as verified by Figs. 3b and 3(c). Second, 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*: In this 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. 3a' and 3b', 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. 3.3, by assuming that the fraction p_i of different peer classes remains the same and then taking the weighted average of x_i, we will derive a lower bound of $N(k)$ under multiple peer classes (i.e., with multiple x_i's).

The rest of this section is organized as follows: In Sect. 3.2, we derive the expected quantities of interest (such as $S(k)$, $N(k)$) in two stages: In stage I when $k \leq k_0$, the hybrid architecture provides both CDN-based streaming and P2P streaming, as the P2P streaming capacity alone is insufficient to serve the incoming requests. In stage II when $k > k_0$, the architecture only provides P2P streaming as the P2P streaming capacity is sufficient to serve the incoming requests alone. We determine k_0 in Sect. 3.3. We discuss the dimensioning among key parameters N_c, x_i and k_0 in Sect. 3.4. Finally, Sect. 3.5.1 presents the simulation results.

[3] For example, we let $c_i = 2^i$ in [1] and design an optimal media data assignment algorithm for supplying peers serving a streaming session. However, this does *not* need to be assumed in our analysis.

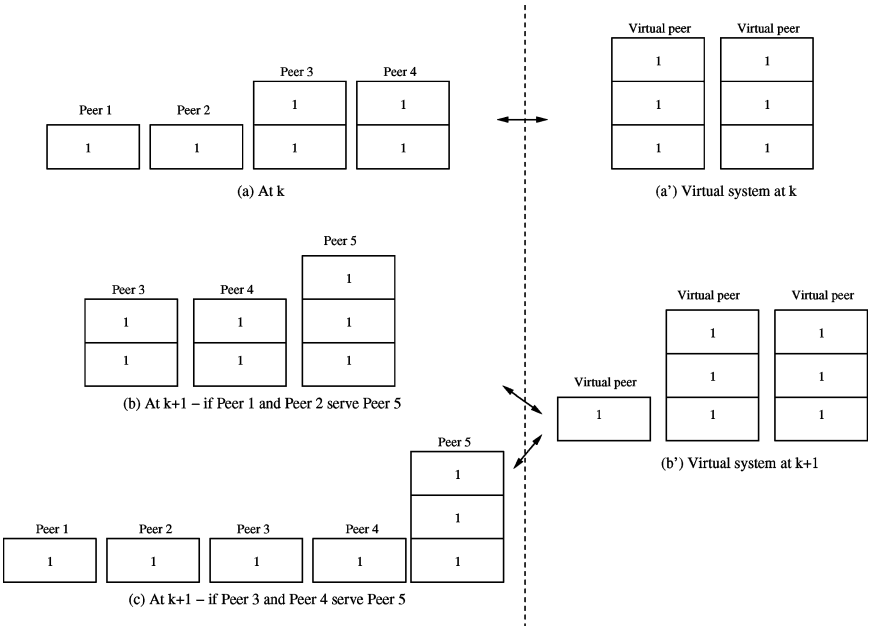

Fig. 3 Example illustrating the difficulty in tracking $N(k)$

3.2 Derivation of $S(k)$ and $M(k)$

$S(k)$ is the total committed P2P streaming capacity in number of full sessions (i.e. number of streaming requests that can be accommodated). $S(k)$ is defined by the following two-piece recursive equation:

$$S(k+1) = \begin{cases} (k \le k_0) \quad S(k) + \underbrace{N_c \sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j}\right)}_{\text{Term 1}} + \\ \underbrace{\frac{S(k)}{\sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j}\right)}}_{\text{Term 2}} \underbrace{\sum_{j=1}^{n} \frac{p_j}{c_j}}_{\text{Term 3}} \underbrace{\left[\sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j} - 1\right)\right]}_{\text{Term 4}} \\ (k > k_0) \quad S(k) + \lambda L M(k) \left[\sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j} - 1\right)\right] \end{cases} \quad (1)$$

In stage I when $k \le k_0$, the overall request generation rate is higher than the combined CDN–P2P streaming capacity. Hence during this stage, both the supplying peers and the CDN server are expected to be busy serving new streaming requests. The terms in the above equation are explained as follows:

- Term 1 is the total committed contribution from peers served by the CDN server during the interval $[k, k+1]$. (x_j/c_j) is the number of sessions (normalized to full session) contributed by a class-j peer. N_c is the reserved 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 time k. Recall that $S(k)$ is in number of full sessions. To estimate the number of the 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 fraction of a full streaming session that each supplying peer contributes. Therefore, the product of Term 2 and Term 3 is the expected number of P2P streaming sessions that can be accommodated at time k.
- Term 4 is the expected number of committed full sessions brought by each P2P streaming session during $[k, k+1]$. Note that the "-1" in Term 4 accounts for the fulfillment (and thus the loss) of one full session. Therefore, the product of Terms 2, 3 and 4 is the total committed contribution (in number of full sessions) from peers served by P2P streaming during interval $[k, k+1]$.

We now explain the equation for stage II when $k > k_0$: By definition of k_0, all streaming requests are expected to be accommodated by the P2P streaming capacity. Hence, during this stage, no streaming request is supposed to be rejected due to insufficient capacity. Hence, the growth of $S(k)$ in this stage is computed as the number of streaming requests $\lambda L M(k)$ multiplied by Term 4 in Eq. (1).

Based on $S(k)$, we derive $M(k)$, the number of remaining requesting peers at time k, as follows:

$$M(k+1) = \begin{cases} M(k) - \dfrac{S(k)\sum_{j=1}^{n}\left(\frac{p_j}{c_j}\right)}{\sum_{i=1}^{n} p_i\left(\frac{x_i}{c_i}\right)} - N_c, & (k \le k_0) \\ M(k)\cdot(1-\lambda L), & (k > k_0) \end{cases} \quad (2)$$

To simplify our presentation, we introduce two variables: r and ρ, as defined in the following equation:

$$r = \sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j}\right) \qquad \rho = \sum_{j=1}^{n} \left(\frac{p_j}{c_j}\right) \tag{3}$$

Note that r can be thought of as the normalized average "contribution ratio" of all supplying peers: the higher the r, the more contribution the peers will make, and vice versa. Similarly, ρ can be thought of as the average fraction of the full streaming rate that each supplying peer contributes during a P2P streaming session.

First, if $r \neq 1$, the close-form expressions for $S(k+1)$ and $M(k+1)$ in terms of r and ρ can be obtained as follows:

$$S(k) = \begin{cases} \frac{N_c}{\rho}\left(\frac{r^2}{r-1}\right)\left[\left(1+\rho\left(\frac{r-1}{r}\right)\right)^k - 1\right], & (k \leq k_0) \\ S(k_0) + M(k_0)\,(r-1)\left[1-(1-\lambda L)^{k-k_0}\right], & (k > k_0) \end{cases} \tag{4}$$

and

$$M(k) = \begin{cases} M_0 + \dfrac{kN_c - S(k)}{r-1}, & (k \leq k_0) \\ M(k_0)\,(1-\lambda L)^{k-k_0}, & (k > k_0) \end{cases} \tag{5}$$

For $r = 1$, the close-form expressions for $S(k+1)$ and $M(k+1)$ can be given as follows:

$$S(k) = \begin{cases} kN_c, & (k \leq k_0) \\ S(k_0), & (k > k_0) \end{cases} \tag{6}$$

and

$$M(k) = \begin{cases} M_0 - kN_c - \rho N_c \frac{k(k+1)}{2}, & (k \leq k_0) \\ M(k_0)\,(1-\lambda L)^{k-k_0}, & (k > k_0) \end{cases} \tag{7}$$

3.3 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. Thus, the reserved CDN server capacity can be released. k_0 can be derived by equating the number of requests in the interval $[k_0, k_0+1]$ to the instantaneous P2P streaming capacity $N(k)$ at k_0 multiplied by a conservative factor α.

$$\lambda L M(k_0) = \alpha N(k_0) \tag{8}$$

As shown in Sect. 3.1, $N(k)$ is difficult to derive. Instead, we can derive a lower bound of $N(k)$ based on $S(k)$ in Sect. 3.2. With the presence of multiple peer classes, the lower bound of $N(k)$ can be expressed as $\frac{\rho S(k)}{r}$: It is easy to verify that $\frac{\rho S(k)}{r}$ is the expected instantaneous P2P capacity of the "virtual system" with total committed capacity $S(k)$ *and* 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. (8) to derive an even *safer* k_0:

$$\lambda L M(k_0) = \alpha \frac{\rho S(k_0)}{r} \tag{9}$$

By replacing $M(k)$ using Eq. (5), Eq. (9) for $r \neq 1$ can be re-arranged as follows:

$$\lambda L\left(M_0 + \frac{k_0 N_c}{r-1}\right) = \left(\alpha\frac{\rho}{r} + \frac{\lambda L}{r-1}\right) S(k_0) \tag{10}$$

After a number of algebraic manipulations, we have

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

Observe that the above equation has the form:

$$a^{k_0} = bk_0 + d,$$

where

$$a = 1 + \rho\frac{r-1}{r}, \quad b = \frac{\lambda L}{\alpha r} + \frac{\rho(r-1)}{r^2}$$

and

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

Hence, k_0 can be solved as follows:

$$k_0 = \frac{-d\,\log(a) - b\,W\left(-\frac{\log(a)\,e^{\left(\frac{-d\log a}{b}\right)}}{b}\right)}{b\,\log(a)} \quad (r > 1) \tag{12}$$

where $W(\cdot)$ is the *Lambert's W-function*. A detailed definition of this function can be found in [6]. An equation similar to (11) for the case of $r = 1$ can be obtained as:

$$\lambda L\left[M_0 - k_0 N_c\left(1 + \frac{\rho(k_0+1)}{2}\right)\right] = \alpha\rho k_0 N_c \tag{13}$$

This equation can be re-arranged in the form of a quadratic: $k_0^2 + gk_0 - h = 0$, where $g = 2(\frac{\alpha}{\lambda L} + \frac{\lambda}{2} + \frac{1}{\rho L})$ and $h = 2\frac{M_0}{\rho N_c}$. Taking the positive root of the quadratic equation, we obtain the expression for k_0 as follows:

$$k_0 = \frac{-g + \sqrt{g^2 + 4h}}{2} \quad (r = 1) \tag{14}$$

3.4 Relation between k_0, r, and N_c

The derivation of k_0 provides a rigorous basis for the dimensioning of parameters in the hybrid architecture to save CDN server capacity cost without overloading the peers. In particular, we discuss the relation between k_0, r, and N_c. Note that N_c and x_i $(1 \leq i \leq n)$ are *tunable* system parameters in the hybrid architecture, and the "contribution ratio" r is defined as $r = \sum_{j=1}^{n} p_j(\frac{x_j}{c_j})$. The other parameters affecting

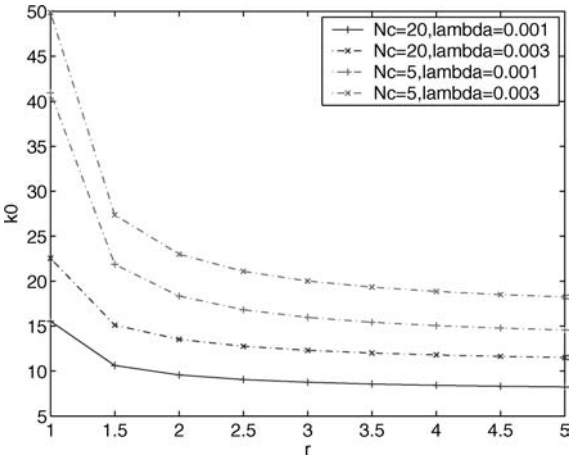

Fig. 4 k_0 as a function of r for different N_c and different λ (based on analysis)

k_0 are the registered client population M_0 and the per-client streaming request generation rate λ.

Based on Eqs. (12) and (14), Fig. 4 shows a numerical example illustrating the relation between k_0 and r, under different CDN server capacity ($N_c = 5,\ 20$, respectively) and different per-client request generation rate ($\lambda = 0.001,\ 0.003$, respectively). The values of other system parameters are fixed: $\alpha = 1$, $M_0 = 2000$, $L = 60$ min. We first observe that with r and N_c fixed, when λ increases, k_0 also increases, indicating that the "CDN-to-P2P" handoff will occur later rather than earlier. We also 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, it is less intuitive to realize that the effect of N_c will *diminish* as it increases. On the other hand, with the same N_c, k_0 decreases when r increases. Interestingly, however, as r increases, k_0 quickly levels off. This important observation justifies our limited contribution policy for supplying peers: beyond a certain point, any further increase in r will *not* yield significant improvement in the "CDN-to-P2P" handoff time. In other words, by imposing a relatively small r on the peers, we already achieve a reasonably early handoff time k_0.

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

$$\lim_{r\to\infty} k_0 = \frac{\log\left(\frac{\lambda L M_0}{\left(\alpha N_c + \lambda L \frac{N_c}{\rho}\right)} + 1\right)}{\log(1+\rho)} \tag{15}$$

In summary, there are three cases that need to be considered: If $0 < r < 1$, the peers will contribute less than they receive (in media data volume); and the system capacity will collapse if the CDN server stops serving streaming requests. If $r = 1$, the peers will contribute back the exact amount of media data they have received. Hence the total streaming capacity will *not* change after the CDN server stops serving streaming requests. In the case of $r > 1$, the peers will make extra contribution to the total streaming capacity. Hence, the capacity may grow even after k_0.

For now, we assume that $r \geq 1$, which can be ensured if $x_j \geq c_j$ $(1 \leq j \leq n)$. This will ensure that the committed contribution of each peer – if measured by the total volume of media data it sends out, is greater than or equal to the total volume of media data it receives (i.e., volume of the media file).

3.5 Simulations

In this section, we present extensive simulation results to validate our analytical results as well as to demonstrate the effectiveness of the hybrid architecture under the session-based contribution policy.

3.5.1 Simulation setup

We simulate a hybrid system with one CDN server and $M_0 = 2000$ clients registered for a media file. Unless stated otherwise, the simulation parameters are set as follows: Initially, the CDN server reserves a capacity for serving $N_c = 20$ simultaneous streaming sessions for the media file to be distributed. The duration of each streaming session is $L = 60$ min. Each client makes a 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, $r = \frac{3}{2} \times 20\% + \frac{6}{4} \times 50\% + \frac{12}{8} \times 30\% = 1.5$). The factor α used in the calculation of k_0 is 1.0. Finally, contrary to the coarse time granularity in our analysis, the time unit in our simulation is 1 min.

In this section, we only present representative simulation results based on the above parameter values. Similar results have also been obtained from simulations under different suites of parameter values.

3.5.2 Simulation results

We measure various performance metrics such as the number of remaining requesting peers $M(k)$ at time k, the total committed P2P streaming capacity $S(k)$, the instantaneous P2P streaming capacity $N(k)$, and the request rejection rate. For each of these metrics, we will present the simulation results and wherever necessary we will compare them with our analytical results.

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 h after the media file is released. The two curves in this figure indicate that the simulation results closely match the numerical results based on our analysis. The results (plus the results based on other sets of simulation parameter values) validate our derivation of $M(k)$ and $S(k)$ in Sect. 3.2, despite the

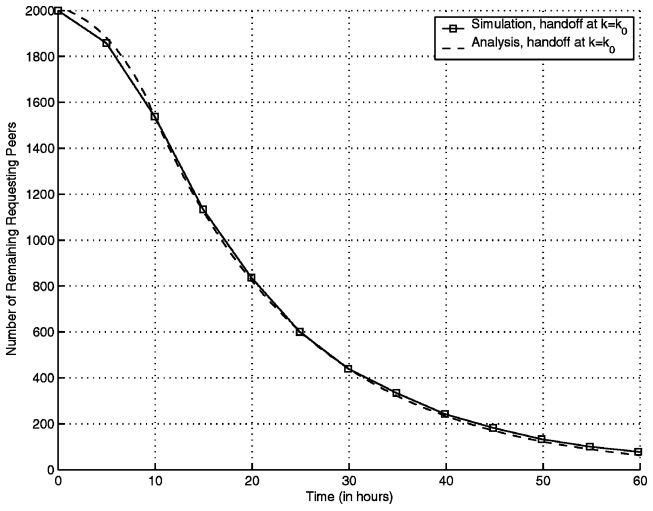

Fig. 5 Number of remaining requesting peers (from simulation and analysis, respectively)

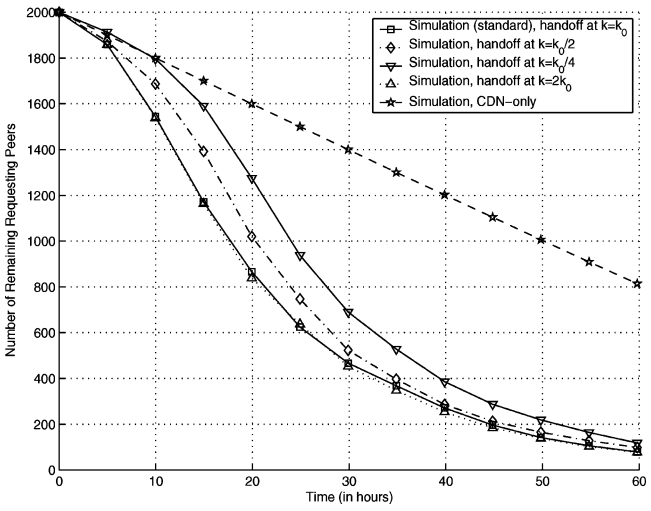

Fig. 7 Impact of handoff time on $M(k)$

coarse time granularity used in the analysis. They also validate some of the other assumptions made in the analysis.

To understand the impact of the system parameters, Fig. 6 presents similar curves for the following four cases: $\lambda = 0.001, 0.003$; and $N_c = 5, 20$. From Fig. 6, it can be seen that the "trend" of $M(k)$ remains the same in all cases. If the amount of reserved CDN capacity (N_c) is higher, then $M(k)$ will decline more rapidly, indicating a faster jumpstart of P2P streaming capacity. However, this will only happen up to a certain threshold – once the reserved CDN capacity N_c exceeds the threshold, any further increase of N_c will *not* change the curve of $M(k)$ significantly. We also notice from the figure that as the request rate λ increases, $M(k)$ declines more rapidly, reflecting the effectiveness of the CDN in jumpstarting P2P streaming capacity, as well as the *intrinsic* adaptivity of the P2P streaming capacity to time-varying demand.

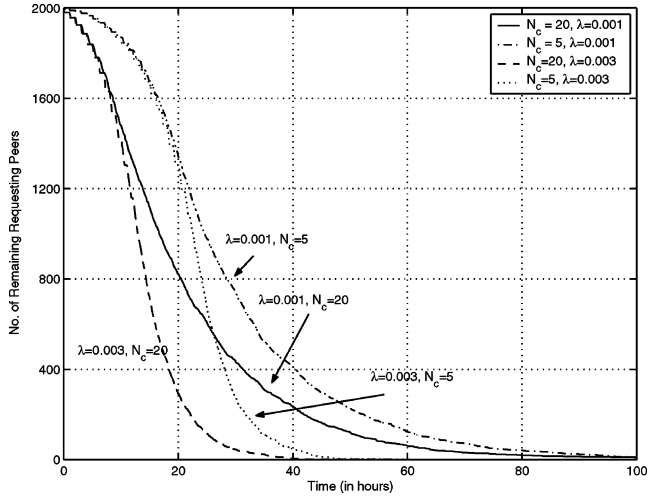

Fig. 6 $M(k)$ under different system parameters

Figure 7 shows the impact of different handoff time. $k_0 \approx 11$ is the calculated handoff time. We set handoff times equal to $k_0/4$ and $k_0/2$ representing early handoff situations; and we set a handoff time equal to $2k_0$ to represent a late handoff. As a comparison, and to demonstrate the benefit of the proposed hybrid architecture, we also present the curve when there is no P2P streaming ("CDN only"). From Fig. 7, we observe that the extreme case of "CDN only" results in an almost linear and the slowest decrease of $M(k)$ (i.e., the slowest progress of media distribution). For our hybrid architecture, a handoff time 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 time later than k_0 does *not* help speeding up the media distribution progress: the curve for "handoff at k_0" and the curve for "handoff at $2k_0$" almost overlap, although the latter will reserve the CDN server capacity for twice as long as the former.

Impact of contribution ratio r on media distribution progress. Figures 8 and 9 demonstrate the impact of r on the progress of media distribution. In these two figures, the remaining number of peers, $M(k)$, is plotted against time for $N_c = 5$ and for $N_c = 20$, respectively. We observe that if the reserved CDN server capacity is relatively high ($N_c = 20$), there will not be much of a difference between the curves for $r = 1$ and for $r = \infty$ (Fig. 8). However, if the reserved CDN server capacity is relatively low ($N_c = 5$), the impact of contribution ratio r on $M(k)$ will be more distinctive. This can be observed in Fig. 9 where there is clear difference among the curves for $r = 1$, $r = 1.5$, and $r = \infty$.

Figure 8 also plots the remaining number of requesting peers when $r = 0$ and $N_c = 60$. From the figure, we notice that the 90% *completion time* of the media distribution process (i.e., when $M(k) = 200$) for this case is 40 h, which is the same as in the case of $r = 1.5$ and $N_c = 20$. This further demonstrates the cost-effectiveness of our hybrid architec-

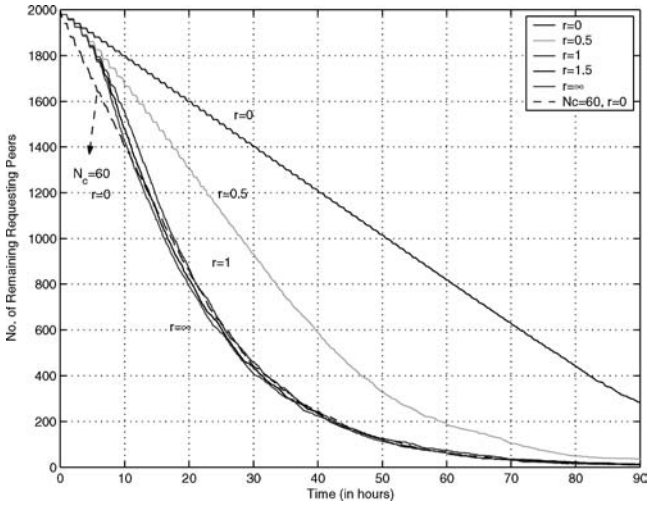

Fig. 8 Impact of r on $M(k)$ ($N_c = 20$)

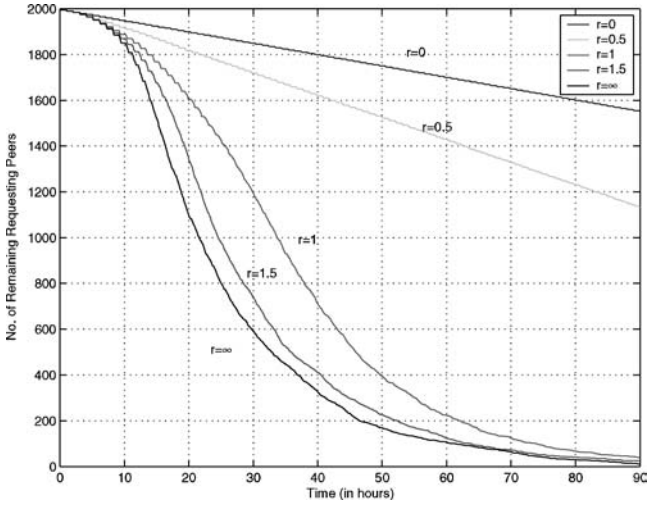

Fig. 9 Impact of r on $M(k)$ ($N_c = 5$)

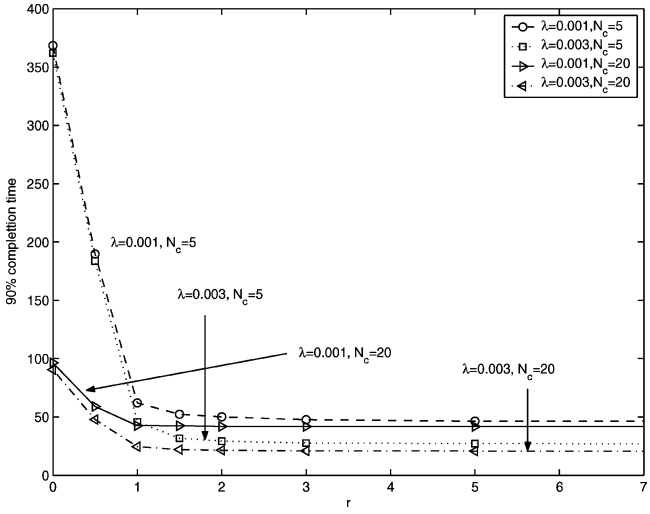

Fig. 10 90% completion time (hours) versus r

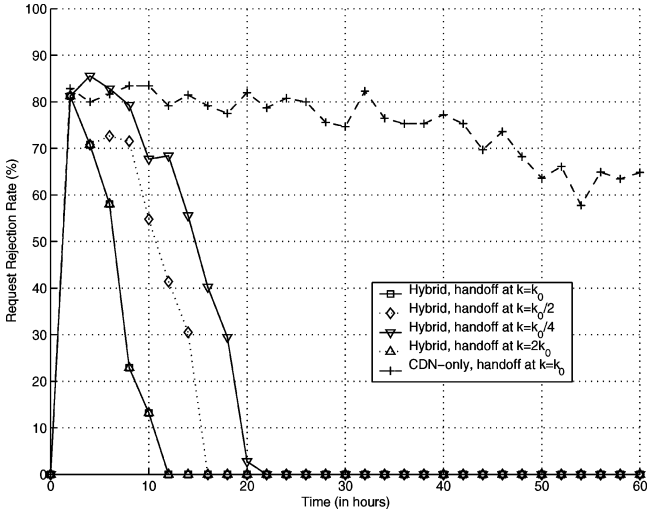

Fig. 11 Per hour request rejection rate over time

ture. With a limited contribution ratio of $r = 1.5$, we only need to reserve a CDN capacity of $N_c = 20$ for a period of 11 h ($k_0 \approx 11$), contrary to $N_c = 60$ for at least a period of 40 h without P2P streaming, in order to achieve the same progress of media distribution.

Figure 10 reveals the impact of λ and N_c on the 90% distribution completion time, which is an indication of how fast the media file is distributed among the peers. Among the four cases considered, the earliest 90% completion time is obtained in the case of $N_c = 20$ and $\lambda = 0.003$. It is important to note that extra-large values of r do not provide much reduction in the 90% completion time. Most of the reduction happens when the value of r increases from 0 (i.e., CDN only with no P2P streaming) to 2.0. This indicates that a limited contribution from the peers is necessary and *sufficient* for the media distribution process, contrary to the intuition that the supplying peers need to serve as long as possible.

Streaming request rejection rate. To illustrate the system dynamics in more detail, we record the per-hour streaming request rejection rate in Fig. 11, 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 the request rejection rate, due to the fixed and limited CDN streaming capacity. For our hybrid architecture, if the handoff time is too early (at $k_0/2$), the rejection rate will remain high many hours after the handoff. With a late handoff (at $2k_0$), the rejection rate is almost the same as in the case of handoff at k_0. Once again, this demonstrates the importance of determining k_0 for both media distribution progress and for CDN server capacity saving.

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

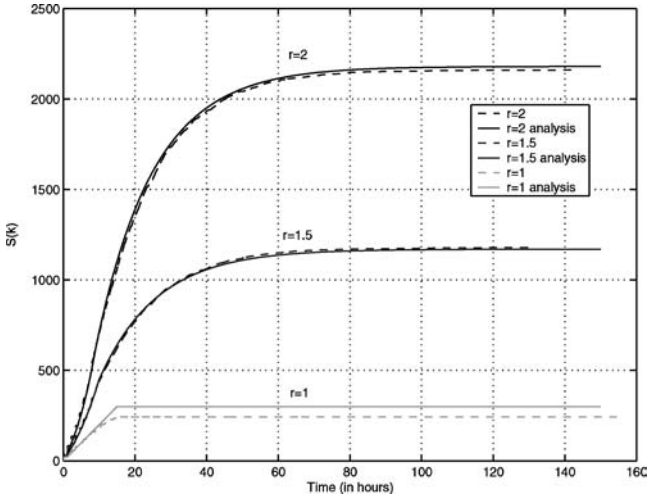

Fig. 12 S(k) under different r (from simulation and analysis)

capacity. The instantaneous streaming capacity, $N(k)$, represents the truly usable streaming capacity (number of full rate media streaming sessions) at time k. In our analysis, we do not derive an accurate form for $N(k)$. Instead, we have derived its lower bound as $\frac{\rho S(k)}{r}$. Figure 12 shows the growth of $S(k)$ based on both simulation and analysis results. Once again, we find that the results from analysis and simulation match closely. Similar match is also observed for $N(k)$.

Figure 12 leads to one important observation: With the elapse of time, more and more peers are served and $S(k)$, the total committed P2P streaming capacity, will keep on growing (if $r > 1$). However, such a large P2P streaming capacity will *not* be consumed, due to the rapid decrease of the remaining requesting peers in the meantime. This proves to be a waste of streaming capacity; and it makes the peer contribution commitment unfulfilled. Indeed, there is no need for those peers to stay committed since there will not be as many requests in the future. The *session-based* contribution policy is the reason for such undesirable feature, which motivates us to design alternative contribution policies that can *bound* the duration in which a peer is required to remain committed to the media file. These alternative policies will be presented in Sects. 4 and 5.

In summary, the simulation results confirm the validity of our analysis. In particular, the hybrid architecture under the limited contribution policy proves to be highly cost-effective: It achieves faster media distribution progress and lower request rejection rate, compared with a "CDN only" architecture. Moreover, it lowers the media distribution cost by only requiring to reserve a reasonable amount of CDN server capacity, for a *limited* period of time. Finally, an undesirable feature due to the session-based contribution policy is that the P2P streaming capacity tends to build up toward the *end* of the media distribution process and never get used. In Sect. 4, we will present a *time-based* contribution policy that eliminates the unnecessary P2P streaming capacity accumulation.

3.6 Modeling streaming capacity losses

Till now we have analyzed the dynamics of an idealistic hybrid CDN–P2P system under the assumptions stated at the beginning of Sect. 3. In particular, we assume peers' full "honesty" in contributing back to the system. We also assume that the peers are always on-line and peers' network connections are always in good condition. However, if one or more of the above assumptions do not hold, the system will suffer from a loss of streaming capacity. To take into account system capacity losses in our analysis, we in this section introduce another parameter l ($0 \le l \le 1$) termed as the *loss factor* and in companion η ($= 1 - l$) as the *effective capacity factor*. More specifically, if a peer commits a total of x sessions, we expect that only ηx sessions will actually be fulfilled by the peer.

l (or η) integrates various factors leading to streaming capacity losses, including peer dishonesty in contract fulfillment, sender status change (from "on-line" to "off-line") during a session, and sender network condition degradation. We note that the CDN server will detect and block free-riders that do not make any contribution to the system. However, the system has a certain level of tolerance for dishonest peers that only contribute a portion (η) of their commitment. We also note that a sender may not always serve the entire streaming session because of its status change or network degradation.[4] In addition, l might be class-specific as well: If l_i represents the loss factor for peers in class i, then we can calculate the overall loss factor l for the system as a weighted average: $l = \frac{\sum_i l_i x_i p_i}{\sum_i p_i x_i}$. Revising the analysis in Sect. 3.2, we can write:

$$S(k+1) = \begin{cases} (k \le k_0) \quad S(k) + \underbrace{N_c \sum_{j=1}^{n} p_j \left(\frac{\eta x_j}{c_j}\right)}_{\text{Term 1}} + \\ \qquad \underbrace{\frac{S(k)}{\sum_{j=1}^{n} p_j \left(\frac{x_j}{c_j}\right)}}_{\text{Term 2}} \underbrace{\left[\sum_{j=1}^{n} \frac{p_j}{c_j}\right]}_{\text{Term 3}} \underbrace{\left[\sum_{j=1}^{n} p_j \left(\frac{\eta x_j}{c_j} - 1\right)\right]}_{\text{Term 4}} \\ (k > k_0) \quad S(k) + \lambda L M(k) \left[\sum_{j=1}^{n} p_j \left(\frac{\eta x_j}{c_j} - 1\right)\right] \end{cases} \tag{16}$$

Even though the recursive definition of $S(k)$ has changed, the recursive definition of $M(k)$ given in Eq. (2) remains the same. For completeness, we rewrite the recursive definitions for $S(k)$ and $M(k)$.

$$S(k+1) = \begin{cases} S(k) + \eta r N_c + \frac{(\eta r - 1)\rho S(k)}{r} & (k \le k_0) \\ S(k) + \lambda L M(k)(\eta r - 1) & (k > k_0) \end{cases} \tag{17}$$

$$M(k+1) = \begin{cases} M(k) - \frac{S(k)}{r}\rho - N_c & (k \le k_0) \\ M(k)\,(1 - \lambda L) & (k > k_0) \end{cases} \tag{18}$$

[4] Dynamic sender replacement for sustained streaming quality has been implemented in our P2P streaming system prototypes GnuStream [3] and PROMISE [2].

These equations can be solved to yield the following close-form solutions:

$$S(k) = \begin{cases} \dfrac{N_c}{\eta\rho}\left(\dfrac{\eta^2 r^2}{\eta r - 1}\right)\left[\left(1+\eta\rho\left(\dfrac{\eta r-1}{\eta r}\right)\right)^k - 1\right], \\ \quad (k \le k_0,\ \eta r > 1) \\ \\ S(k_0) + M(k_0)\,(\eta r - 1)\left[1-(1-\lambda L)^{k-k_0}\right], \\ \quad (k > k_0,\ \eta r > 1) \end{cases} \tag{19}$$

$$M(k) = \begin{cases} M_0 + \dfrac{kN_c - S(k)}{\eta r - 1}, \quad (k \le k_0,\ \eta r > 1) \\ \\ M(k_0)\,(1-\lambda L)^{k-k_0},\ (k > k_0,\ \eta r > 1) \end{cases} \tag{20}$$

From these close-form expressions we can derive the k_0 for the CDN–P2P hybrid system with streaming capacity losses:

$$k_0 = \frac{-d\,\log(a) - b\,W\left(-\frac{\log(a)\,e^{\left(\frac{-d\log a}{b}\right)}}{b}\right)}{b\,\log(a)} \quad (\eta r > 1) \tag{21}$$

where

$$a = 1 + \eta\rho\,\frac{\eta r - 1}{\eta r}, \quad b = \frac{\lambda L}{\eta\alpha r} + \frac{\rho(\eta r - 1)}{\eta r^2},$$

and

$$d = 1 + \frac{\lambda L M_0}{\left(\alpha\left(\frac{\eta r}{\eta r - 1}\right)N_c + \lambda L\left(\frac{\eta r}{\eta r - 1}\right)^2 \frac{N_c}{\eta\rho}\right)}.$$

Figure 13 shows the impact of the loss factor l (or the effective capacity factor $\eta = 1 - l$) on the handoff time k_0; while Fig. 14 shows the effect of η on the system capacity $S(k)$ and the number of remaining requesting peers $M(k)$. In Fig. 13, when $r = 2$, the handoff time k_0 will have to be extended from 9.5 h to 12.5 h if the effective capacity factor η drops from 1.0 to 0.8. If η drops to 0.6, k_0 will be further postponed to 17.5 h. The decline of $S(k)$ growth rate shown in Fig. 14 confirms the negative impact of η. The effect on $M(k)$ (remaining requesting peers) is also noticeable but less significant than that on $S(k)$. The results from this loss-aware analysis will help determining the degree of peer contract *unfulfillment* that the system can tolerate.

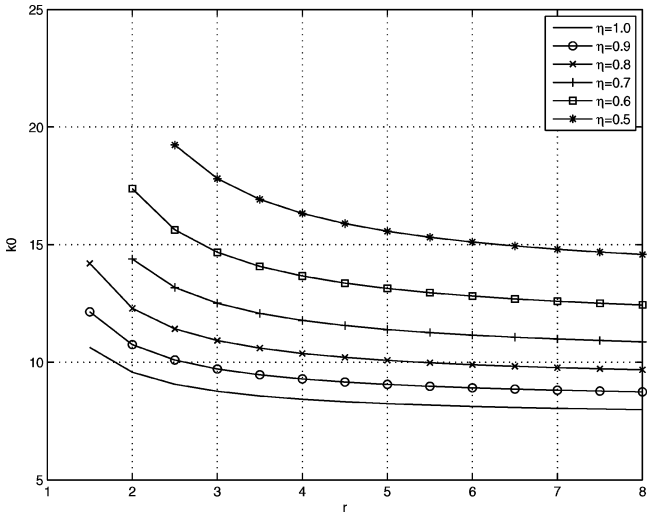

Fig. 13 k_0 as a function of r under different η ($N_c = 20$ and $\lambda = 0.001$, other parameters the same as in Sect. 3.5)

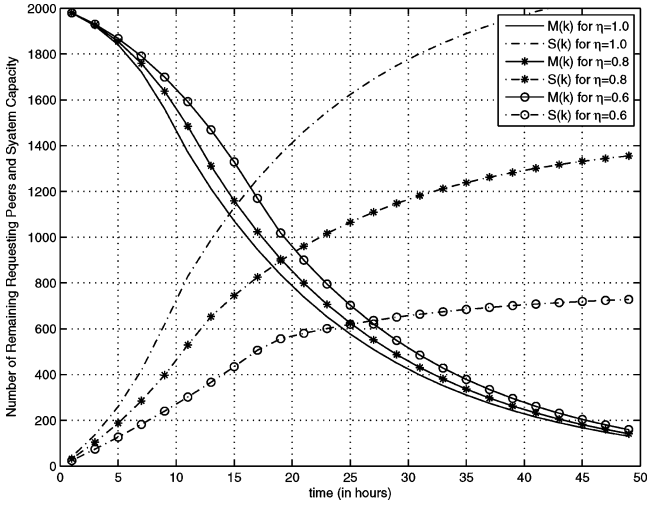

Fig. 14 Number of remaining requesting peers $M(k)$ and system capacity $S(k)$ ($r = 2$)

4 Time-based contribution policy

In this section, we propose an alternative form of limited contribution policy for supplying peers. In Sect. 3, we have used the number of streaming sessions as the form of contribution contract and it is realized that a supplying peer may be held up for an excessively long period of time in order to fulfill its commitment (and may never be able to fulfill it). A convenient and natural form of commitment is simply the duration of service (as a supplying peer) imposed on each peer. The committed *service time* is denoted by t – in number of periods each equal to the duration of one media streaming session. During its service time, a supplying peer is required to serve streaming requests at its committed streaming rate (with the same definition of c_i as in the session-based policy) whenever selected. We denote its contribution ratio as $\hat{r}_i = t_i/c_i$. The intuition behind the time-based contribution policy is that a peer may serve a maximum of t_i sessions during its committed time period, and that c_i such sessions amount to the same volume as that of the media file being distributed. Once the service time is over, the supplying peer retires no matter how many sessions it has actually served.

Such policy is easier to be accepted by peers due to the time bounded contract. However, the fallout of this policy is that the accumulated P2P streaming capacity starts to *decrease* toward the end of the "P2P only" stage (i.e.,

stage II) due to retiring peers. Hence under this policy, the CDN server may need to re-join the distribution process after the P2P-only stage. We denote the time when the CDN server restarts serving incoming requests as k_1. At this time, streaming requests will be served by both CDN and P2P again. Although the CDN server capacity requirement after k_1 is expected to be marginal compared with N_c at the beginning, we need to dimension the system parameters so that k_1 is not too early.

4.1 Derivation of k_1

We analyze the time-based contribution policy to derive an estimate of k_1, the time when the CDN server has to re-join the media distribution process. We make the following assumption to make our analysis tractable: all peers belong to the *same* class with the same committed out-bound streaming rate that is equal to $\frac{1}{c}$ of the media playback rate. Each supplying peer has a service time of t, as defined earlier in this section.

We first note that the handoff time k_0 for this new model can be obtained using the same formula as in the session-based model by taking $x = t$ (note that t is normalized – hence a peer will serve at most t sessions). During stage I of the system, as discussed in Sect. 3, most of the supplying peers are always busy due to the high overall streaming request rate. Hence, we expect that most of the supplying peers in stage I will serve close to t sessions. Therefore, it is also appropriate to use Eq. (12) to determine k_0 under the time-based contribution policy. Once k_0 is calculated, we are interested in calculating the time k_1, when the transition from stage II (P2P only) to a new stage III (CDN and P2P, not shown in Fig. 2) takes place.

We define two new quantities: Let $n(k)$ be the number of active supplying peers at time k and let $P(k)$ be the number of peers that are served starting at time k. At time k, the supplying peers which were served at time $k-t$ will retire from the system. An additional $P(k) = \lambda M(k-1)$ peers will receive the media file and will become supplying peers at time $k+1$. Hence, we can write the expected system dynamics *after* time k_0 as follows: For $k > k_0$

$$\begin{aligned} n(k) &= n(k-1) + \lambda M(k-1) - P(k-t) \\ P(k) &= \lambda M(k-1) \end{aligned} \tag{22}$$

Note that the above equations assume that at time k, the streaming capacity $N(k)$ is sufficient to satisfy all the incoming media streaming requests. Now for simplicity of the analysis, we also assume that all the peers that are active at time k_0 expire at time $k_0 + t$. We can calculate the value of $M(k_0)$ (denoted as M) from the analysis in the previous section. Hence, the above equations can be re-written as follows:

$$\begin{aligned} n(k_0+t) &= M\lambda L(1 + (1-\lambda L) + \cdots + (1-\lambda L)^{t-1}) \\ M(k_0+t) &= M(1-\lambda L)^t \end{aligned} \tag{23}$$

Similarly, the equations for subsequent time instances $k > k_0 + t$ can be written as follows:

$$\begin{aligned} n(k) &= M\lambda L(1-\lambda L)^{k-t-k_0}(1 + \cdots + (1-\lambda L)^{t-1}) \\ M(k) &= M(1-\lambda L)^{k-k_0} \end{aligned} \tag{24}$$

Similar to the definition of k_0 in Eq. (8) , one obvious approach is to define k_1 as the earliest time after k_0, such that $\frac{n(k)}{c} = \beta\lambda L M(k)$ where β is some constant. Unfortunately, this approach does not result in an expression for k_1, because while solving $\frac{n(k)}{c} = \beta\lambda L M(k)$, the factor involving k on both sides gets cancelled! In fact, let $\beta = 1$, we can show through simple algebraic manipulation that, if t satisfies:

$$t > \frac{\log(1+\lambda L c)}{-\log(1-\lambda L)} \tag{25}$$

then we have $\frac{n(k)}{c} > \lambda L M(k)$ for *any* $k > k_0 + t$, indicating that the P2P streaming capacity will always be greater than the overall request arrival rate.

Based on the above analysis, it seems that the CDN server will *not* have to re-join the media distribution process, as long as t is properly set. However, the analysis, which is based on the expected case, does not reveal the hidden fact that the overall stream request rate $\lambda L M(k)$ can be dangerously close to the P2P streaming capacity $\frac{n(k)}{c}$, as confirmed by our simulations. For this reason, we revise the definition of k_1 as the time when the P2P streaming capacity is equal to $\lambda L M(k)$ *plus* the standard deviation of the incoming streaming requests. Thus, we have the following revised equation:

$$\frac{n(k)}{c} = \lambda L M(k) + \sqrt{\lambda L M(k)} \tag{26}$$

Plugging in Eq. (24), we have:

$$\begin{aligned} &\frac{M\lambda L}{c}(1-\lambda L)^{k_1-t-k_0}(1 + \cdots + (1-\lambda L)^{t-1}) \\ &= M\lambda L(1-\lambda L)^{k_1-k_0} + \sqrt{M\lambda L(1-\lambda L)^{k_1-k_0}} \end{aligned} \tag{27}$$

Solving Eq. (27), we have:

$$\begin{aligned} k_1 &= k_0 + 2t \\ &+ \frac{2\log c + \log\lambda L - \log M - 2\log[1-(1+c\lambda L)(1-\lambda L)^t]}{\log(1-\lambda L)} \end{aligned} \tag{28}$$

In the more general case when the peers belong to multiple classes, a weighted average of their service time commitment ($\sum_i p_i t_i$) can replace t and the above equation can be solved for k_1.

4.2 Relation between k_1, $\hat{r}$, and N_c

To illustrate the relation between key system parameters, Fig. 15 shows the numerical value of k_1 versus different service time commitment (i.e., $\hat{r} = t/c$), based on Eq. (28). In

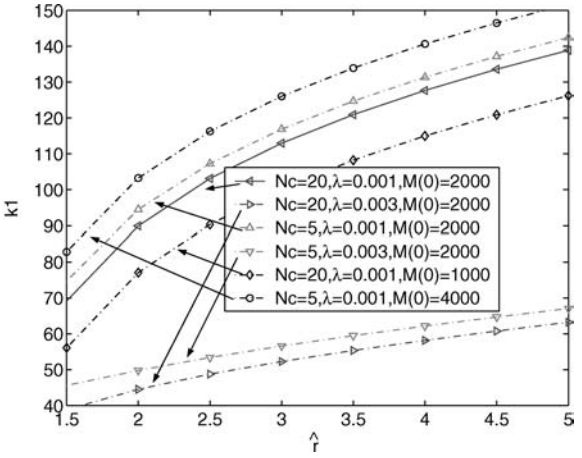

Fig. 15 k_1 versus service time commitment $\hat{r} = \frac{t}{c}$ (based on analysis)

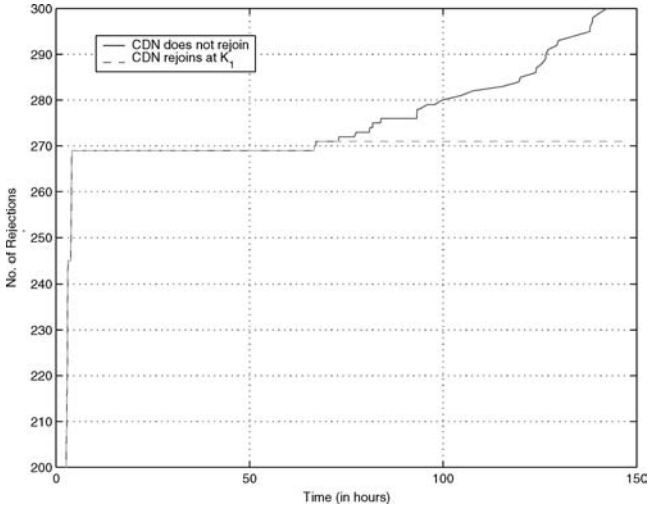

Fig. 16 Time-based contribution policy: cumulative number of request rejections over time, with and without CDN server re-join ($k_1 \approx 70$)

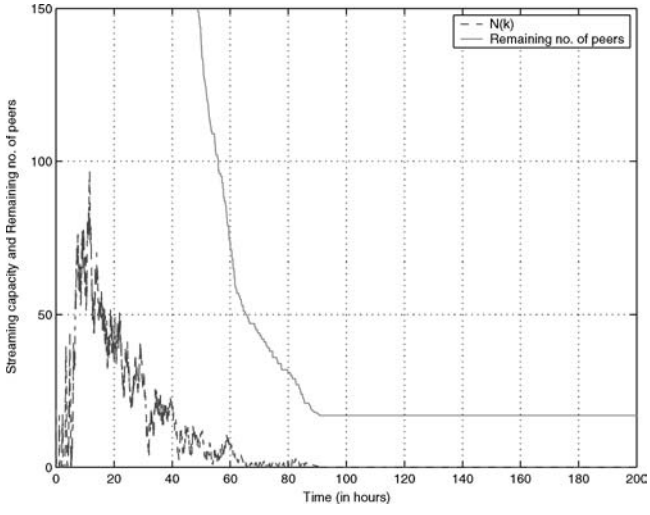

Fig. 17 Time-based contribution policy: $N(k)$ and remaining number of requesting peers, without CDN server re-join

this example, all peers belong to the same class with $c = 2$. From the figure, we notice that the per client request rate λ has a larger impact on the value of k_1 than the reserved CDN server capacity N_c. This can be explained as follows: With a higher λ, most peers request and receive the media file earlier. Hence, these peers leave the system earlier. This implies that the time at which the P2P streaming capacity becomes insufficient comes earlier too. On the other hand, the reserved CDN server capacity N_c does not have much influence on the value of k_1 because after k_0, the CDN server will not serve any requests for a sustained period of time (i.e., $k_1 - k_0$). The difference in k_1 under different values of N_c is indirectly due to the difference in the respective k_0 values. We have also plotted k_1 for different values of the initial registered client population M_0. As the client population increases, there is an increased chance that the system will have enough supplying peers even after a long time. Hence, k_1 increases with the number of registered clients. Finally, the longer the service time commitment t, the later the time k_1.

4.3 Simulation results

We perform simulations to evaluate the time-based contribution policy. The simulation parameters are the same as in the simulations for the session-based policy (Sect. 3.5), *except* that the peers belong to the same class with $c = 2$ and $t = 3$ (thus $\hat{r} = \frac{t}{c} = 1.5$).

Figure 16 shows the *cumulative* number of streaming request rejections over time, with and without the CDN server re-joining at $k_1 \approx 70$. We observe that after the "CDN-to-P2P" handoff (at $k_0 \approx 10$), there is a 60-h period when there is no streaming request rejection. However, sometime around $k = 70$, if the CDN server does *not* re-join, the number of rejections will start to increase again. This indicates that the P2P streaming capacity cannot always accommodate the incoming requests beyond k_1, confirming our analysis. On the other hand, if the CDN server re-joins at $k = 70$, the number of rejections will *stop* increasing. We also observe that the CDN server capacity required after the re-join is $N_c' = 1$, which is significantly lower than the original reserve of $N_c = 20$.

Figure 17 plots the P2P streaming capacity ($N(k) = \frac{n(k)}{c}$) versus the remaining number of requesting peers ($M(k)$), *without* the CDN server's re-join. It confirms the fact that under the time-based policy, the P2P streaming capacity may become too close to the streaming request rate thus missing some requests, even though the expectation of the former is mathematically higher than that of the latter. In our simulation, the P2P streaming capacity virtually becomes zero at $k \approx 90$, and there is no way to accommodate the remaining requesting peers by P2P streaming only.

In summary, the time-based peer contribution policy bounds the service time of each supplying peer. However, it leads to more limited P2P streaming capacity toward the

end of the media distribution process and therefore requires a re-join of the CDN server (with marginal capacity requirement). The higher the peer service time commitment t, the later the CDN server re-join will take place (at time k_1). A dilemma now arises: To postpone k_1 (thus minimizing CDN server involvement), a larger t is desirable. However, t may become so large that it forces a supplying peer to serve *more sessions* than under the session-based policy, especially during stage I (i.e., before k_0) when the streaming demand is high.

5 Integrated contribution policy

In the previous section, we present the time-based contribution policy and identify a dilemma in determining the service time commitment for supplying peers. Earlier, in Sect. 3, we propose the session-based contribution policy and note the "unused committed sessions" problem. By comparing and contrasting the two policies, we realize that the problems with the two policies are mutually exclusive, suggesting that it is possible to design an *integrated* contribution policy that combines the session-based and time-based policies and offsets their respective problems.

Specifically, each peer will commit an out-bound streaming rate as in the previous two policies. Based on its class, the peer will commit a number of streaming sessions x_i, *as well as* a service time t_i, according the session-based and time-based policies, respectively. As a supplying peer, it will be free to retire, as soon as it fulfills *either* the session commitment *or* the service time commitment – whichever happens earlier. At the beginning, when media streaming demand is high, supplying peers tend to finish their session commitment first. Toward the end of the media distribution process, supplying peers are likely to finish their service time commitment earlier.

5.1 Simulation results

We perform simulations to evaluate the integrated contribution policy. We use the same parameters: $N_c = 20$, $\lambda = 0.001$, $M_0 = 2000$, and $c_i = 2, 4, 8$ with proportions $p_i = 0.2,\ 0.5,\ 0.3$, respectively. The session-based contribution ratio $r = \sum_{i=1}^{3} p_i(\frac{x_i}{c_i})$ is set as 1, 1.5, and 2 in different simulations. The time-based contribution ratio $\hat{r} = \sum_{i=1}^{3} p_i(\frac{t_i}{c_i})$ is set as 3 (namely, $t_i = 6, 12, 24$) in all simulations. k_0 is computed as per Eq. (12) or (14). k_1 is then calculated using Eq. (28) with $t = \sum_i p_i t_i$. The key observations from the simulation results are as follows:

– Recall that the problem with the session-based contribution policy is the unused streaming capacity accumulated toward the end of the media distribution process. Under the integrated contribution policy, this problem is solved by integrating the time-based contribution policy: Fig. 18 plots the P2P streaming capacity versus time for different values of r. We see that the capacity begins to decrease (rather than to accumulate) after the initial growth period. This is due to the fact that more and more supplying peers fulfill their service time commitment earlier than their session commitment, with the decreasing demand for streaming. Furthermore, the higher the value of r, the earlier the decline trend is exhibited, which corresponds to an earlier handoff time k_0.

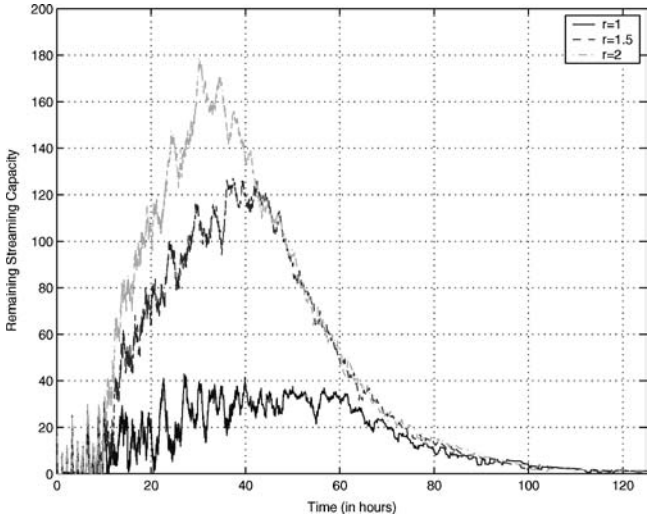

Fig. 18 Integrated contribution policy: remaining streaming capacity over time

– The problem with the time-based contribution policy is that the dilemma in determining the service time commitment t: a large t helps to postpone the re-join time of the CDN server. However, a large t unfairly punishes the *early* supplying peers, because they may have to serve more than x_i sessions as in the session-based policy. Under the integrated policy, this problem is solved: In the simulations, we observe that *almost all* supplying peers that start before k_0 fulfill their session-based commitment first, so that they can retire well *before* their service time commitment is fulfilled.
Figure 19 plots the number of cumulative streaming request rejections versus time for different values of r. From the figure, there are few rejections toward the end of the media distribution process. Interestingly, we observe that for $r = 1.5$ and $r = 2$, the media distribution process may be completed (i.e., all clients are served) *before* the computed CDN server re-join time (around $k = 110$), in which case the CDN server actually does *not* have to re-join.

In summary, the integrated contribution policy overcomes the problems with the session-based and time-based policies. Under the integrated policy, the early supplying peers fulfill their session commitment and retire before their service time is up; while the later supplying peers serve fewer than their committed number of sessions but stay in the system for a longer yet finite period of service time. The

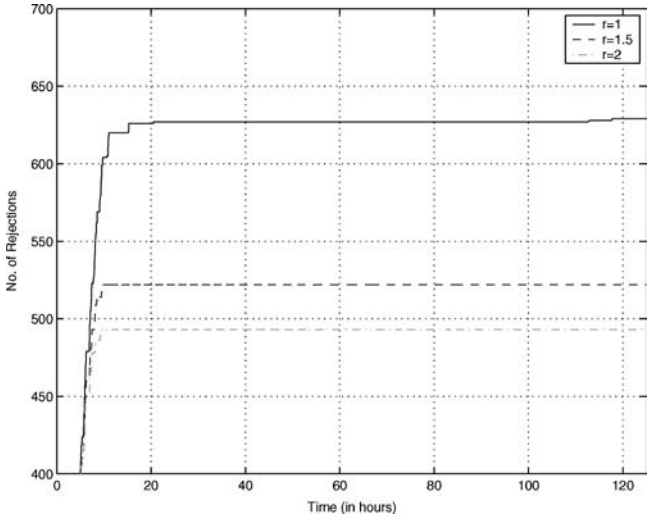

Fig. 19 Integrated contribution policy: cumulative number of rejections over time

integrated policy achieves fast media distribution and low streaming request rejection as in the session-based policy, yet it strikes a better balance among peers with respect to their contribution fairness.

5.2 Dimensioning of the hybrid architecture

By now, we have shown that the CDN–P2P hybrid architecture requires careful dimensioning, due to the different contribution policies, interrelated system parameters, and their impacts on multiple performance metrics. Our analysis and simulations provide a rigorous basis to guide such dimensioning. For example, a possible "workflow" of system parameter dimensioning is as follows: Given the values of parameters M_0, λ, c_i, p_i, N_c, as well as a target k_0 we wish to achieve, we can derive the peer session commitment x_i as well as the tolerable per peer capacity loss factor l, according to the analysis in Sect. 3. If the integrated contribution policy is adopted, we can further derive the peer service time commitment t_i in order to achieve a target k_1[5] based on the analysis in Sect. 4. Such dimensioning may not succeed in one try – if so, adjustment of parameters (for example, k_0, k_1, and/or N_c) may be performed, subject to certain constraints such as the overall cost and the maximum session/time contribution from each peer. Also note that we need to redo system dimensioning whenever key system parameters change – for example, when a new media file is released or when the per-client file request rate λ deviates from the initial estimate.

6 Related work

Content Distribution Networks have been successfully deployed on the Internet, an example being the one operated by *Akamai*. Technical issues in CDN have also been extensively studied. For example, Chawathe et al. [7] study 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. Biliris et al. [9] discuss 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., the streaming of media from each CDN server to the clients it serves.

P2P systems have attracted tremendous research attention in the past few years. Saroiu et al. [11] present the first detailed measurement study of two popular P2P systems, namely Napster and Gnutella. A number of distributed P2P lookup services have been proposed, such as CAN [12], Chord [13], Pastry [14], and Tapestry [15]. In our architecture, for centralized management and accountability, we adopt the "Napster"-like scheme by using the CDN server as the index server – a natural choice in a hybrid CDN–P2P system.

More recently, P2P media streaming systems have also been proposed. C-star (*www.centerspan.com*) is a commercial system which enables media streaming from multiple suppliers to one receiver. Nguyen et al. [16] show the feasibility of streaming media based on multiple senders. However, they do not address the media *distribution* and system *dimensioning* issues. PALS [17] achieves quality adaptation based on layer-encoded media for P2P streaming from multiple sender peers to a receiver peer. CoopNet [4, 18] is a scalable mechanism allowing peers to cooperate to distribute streaming media content when the CDN server is overloaded. Our work also aims at reducing the load of the CDN server. It complements CoopNet by proposing "CDN-to-P2P" handoff, limited contribution policies, as well as detailed analysis of system dynamics. A programming platform is presented by Lienhart et al. [19] to support P2P multimedia service development.

There is a body of results for *multicast* streaming to multiple requesters, known as application level multicast streaming [20–23]. Narada [21] maintains and optimizes a mesh among multiple receivers, upon which a multicast tree is built for the streaming session. In PeerCast [20], a new receiver joins a streaming session by traversing the distribution tree starting at the root, till it reaches a node with a sufficient remaining capacity. Both NICE [24] and ZIGZAG [23] adopt hierarchical distribution trees, and therefore scale up to a large number of requesters. In another category of P2P multicast streaming, each requester collects data from multiple upstream peers [18, 25–27]. For example, CoopNet [18] employs multi-description data coding and constructs multiple distribution trees (one tree for each description) spanning all participants. In P2P multicast streaming, it is generally assumed that a peer, acting as a relay, contributes an outbound streaming rate that is at least equal to the full streaming rate. Less effort has been devoted to P2P streaming to an individual requesting peer, under the conditions that supplying peers are heterogeneous and each willing to contribute

[5] As shown in the simulations in this section, if the target k_1 is sufficiently late, it is possible that the media distribution is completed before k_1 and therefore the CDN server does not have to re-join.

only a *fraction* of the streaming rate. In addition, the media distribution process is asynchronous, different from the *synchronous* nature of P2P multicast streaming.

In [28], a P2P file sharing system is modeled as a multiclass closed queuing network. This allows for the analysis of system throughput dynamics under various configurations of the peer community. Our analysis instead models the dynamics of a streaming media distribution system. The free riding problem has been studied in [29] (through a measurement study) and [30] (through a game-theoretic analysis). Both [29] and [30] advocate the use of payment mechanisms in order to motivate the peers with incentives to contribute to the system. Instead of an abstract payment model, our work proposes much simpler peer contribution policies for rigorous planning and dimensioning in a more disciplined peer community.

7 Conclusion

We have presented a hybrid architecture for cost-effective streaming media distribution. The architecture combines two complementary technologies: CDN and P2P. For this architecture, we present three limited contribution policies for supplying peers, so that the streaming capacity of the supplying peers can be aggregated and exploited on a limited basis: In the session-based policy, peers commit to participate in a limited number of streaming sessions. In the time-based policy, peers commit to serve during a limited period of service time. The integrated policy combines the session-based and time-based contribution commitments, and allows peers to retire when either commitment is fulfilled. We have conducted a comprehensive study of the hybrid architecture: In-depth analysis of the system dynamics is presented, revealing the impact of different policies and parameters on the progress, cost, and peer load of the media distribution process. Extensive simulations are also performed to validate the analysis and to reveal interesting observations. Both our analytical and simulation results lead to systematic guidelines for the planning and dimensioning of the proposed hybrid architecture.

References

1. Xu, D., Hefeeda, M., Hambrusch, S., Bhargava, B.: On peer-to-peer media streaming. In: Proceedings of IEEE ICDCS 2002 (2002)
2. Hefeeda, M., Habib, A., Botev, B., Xu, D., Bhargava, B.: Promise: peer-to-peer media streaming using collectcast. In: Proceedings of ACM Multimedia 2003 (2003)
3. Jiang, X., Dong, Y., Xu, D., Bhargava, B.: GnuStream: a P2P media streaming system prototype. In: Proceedings of IEEE ICME 2003 (2003)
4. Padmanabhan, V.N., Wang, H., Chou, P., Sripanidkulchai, K.: Distributing streaming media content using cooperative networking. In: Proceedings of NOSSDAV 2002 (2002)
5. Stavrou, A., Rubenstein, D., Sahu, S.: A lightweight, robust P2P system to handle flash crowds. In: Proceedings of IEEE ICNP 2002 (2002)
6. Corless, R., Gonnet, G., Hare, D., Jeffrey, D., Knuth, D.E.: On Lambert's W function. Adv. Comput. Maths. (1996)
7. Chawathe, Y.: Scattercast: An architecture for internet broadcast distribution as an infrastructure service. Ph.D. Thesis, University of California, Berkeley (2000)
8. Kangasharju, J., Roberts, J., Ross, K.: Object replication strategies in content distribution networks. Comp. Commun. **25**(4), 376–383 (2002)
9. Biliris, A., Cranor, C., Douglis, F., Rabinovich, M., Sibal, S., Spatscheck, O., Sturm, W.: CDN brokering. In: Proceeding of the International Workshop on Web Caching and Content Distribution (WCW 2001) (2001)
10. Apostolopoulos, J., Wong, T., Wee, S., Tan, D.: On multiple description streaming with content delivery networks. In: Proceedings of IEEE INFOCOM 2002 (2002)
11. Saroiu, S., Gummadi, P., Gribble, S.: A measurement study of peer-to-peer file sharing systems. In: Proceedings of SPIE/ACM MMCN2002 (2002)
12. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proceedings of ACM SIGCOMM 2001 (2001)
13. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of ACM SIGCOMM 2001 (2001)
14. Rowstron, A., Druschel, P.: Pastry: Scalable distributed object location and routing for large-scale peer-to-peer systems. In: Proceedings of IFIP/ACM Middleware 2001 (2001)
15. Zhao, B., Kubiatowicz, J., Joseph, A.: Tapestry: an infrastructure for fault-tolerant wide-area location and routing. UC Berkeley Computer Science Technical Report (CSD-01-1141) (2001)
16. Nguyen, T.P., Zakhor, A.: Distributed video streaming over Internet. In: Proceedings of SPIE/ACM MMCN 2002 (2002)
17. Rejaie, R., Ortega, A.: PALS: peer-to-peer adaptive layered streaming. In: Proceedings of ACM NOSSDAV 2003 (2003)
18. Padmanabhan, V., Wang, H., Chou, P.: Resilient peer-to-peer streaming. In: Proceedings of IEEE ICNP 2003 (2003)
19. Lienhart, R., Holliman, M., Chen, Y., Kozintsev, I., Yeung, M.: Improving Media Services on P2P Networks. IEEE Internet Computing (2002)
20. Bawa, M., Deshpande, H., Garcia-Molina, H.: Transience of peers and streaming media. In: Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets-I) (2002)
21. Chu, Y., Rao, S., Seshan, S., Zhang, H.: A case for end system multicast. IEEE J. Select. Areas Commun. (JSAC) **20**(8), 1456–1471 (2002)
22. Shi, S., Turner, J.: Routing in overlay multicast networks. In: Proceedings of IEEE INFOCOM'02 (2002)
23. Tran, D., Hua, K., Do, T.: Zigzag: An efficient peer-to-peer scheme for media streaming. In: Proceedings of IEEE INFOCOM'03 (2003)
24. Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable application layer multicast. In: Proceedings of ACM SIGCOMM'02 (2002)
25. Castro, M., Druschel, P., Kermarrec, A., Nandi, A., Rowstron, A., Singh, A.: SplitStream: high-bandwidth content distribution in a cooperative environment. In: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03) (2003)
26. Kostic, D., Rodriguez, A., Albrecht, J., Vahdat, A.: Bullet: high bandwidth data dissemination using an overlay mesh. In: Proceedings of ACM SOSP 2003 (2003)
27. Snoeren, A., Conley, K., Gifford, D.: Mesh-based content routing using XML. In: Proceedings of ACM SOSP 2001 (2001)
28. Ge, Z., Figueiredo, D., Jaiswal, S., Kurose, J., Towsley, D.: Modeling peer-peer file sharing systems. In: Proceedings of IEEE INFOCOM'03 (2003)
29. Adar, E., Huberman, B.: Free riding on gnutella. First Monday **5**(10) (2000)
30. Golle, P., Leylton-Brown, K., Mironov, I.: Incentives for sharing in peer-to-peer networks. In: Proceeding of the Second Workshop on Electronic Commerce (WELCOM'01) (2001)