RZ 3781 (# Z1006-002) 06/29/10
Computer Science/Mathematics 10 pages

Research Report

Content Retrieval Delay Driven by Caching Policy and Source Selection

Mathias Björkqvist and Lydia Y. Chen

IBM Research – Zurich
8803 Rüschlikon
Switzerland
{mbj,yic}@zurich.ibm.com

Content Retrieval Delay Driven by Caching Policy and Source Selection

Mathias Björkqvist
IBM Research Zurich Laboratory,
8803 Rüschlikon, Switzerland
Email: mbj@zurich.ibm.com

Lydia Y. Chen
IBM Research Zurich Laboratory,
8803 Rüschlikon, Switzerland
Email: yic@zurich.ibm.com

Abstract*—In this paper we study content retrieval delay in a hybrid content distribution system, e.g., emerging content cloud [1], where a requested content item can be vertically retrieved from the central server and horizontally retrieved from network nodes. The content retrieval delay depends on the load intensities of the retrieval sources, which have asymmetric system properties such as bandwidth and cache capacity. The retrieval traffic is generated due to heterogeneous content availability, i.e., content diffusion resulting from the applied caching policies, and the selection of retrieval sources. To optimize the retrieval delay, the advantages of the network nodes should be utilized while also leveraging the caching and retrieval capacity of the server. We present analytical models to evaluate the content retrieval delay under two retrieval selection strategies, i.e., *Bernoulli* and *Shortest-Queue*, and three caching policies: *selfish*, *altruistic*, and our proposed hybrid caching policy which partitions the content items into three categories, each employing different caching schemes. The traffic loads and latency of a given combination of source selection and caching policy is derived based on the content diffusion and distribution in the entire system. The simulation and analytical results show that satisfactory content retrieval delay is achieved when the retrieval selection is load-aware and the caching policies can effectively utilize the cache storage and retrieval capacity of both the network nodes and the server. In particular, the proposed hybrid caching policy combined with Shortest-Queue selection is shown to scale with various network configurations and to adapt to the load changes in our experiment results.

I. Introduction

Content distribution systems hosting wide varieties of content items have become ever popular in recent years. A hybrid architecture consisting of a central server and network/peer nodes has been shown to effectively deliver content items to end-users [4], [5]. Network and storage resources needs to be deployed efficiently in order to provide satisfactory retrieval delay from the network nodes and the server, which have asymmetric retrieval and caching capabilities. Such a system design is very applicable and relevant to today's emerging content cloud systems, such as Akamai [2] and Amazon CloudFront [1]. Retrieving a content item from a network node has the advantage of better scalability with increasing total retrieval traffic, as shown in peer-to-peer-like systems [16], whereas the central server has the advantage of larger bandwidth compared to a network node. Due to buffer limits for content caching, distributed network nodes can usually cache and serve a subset of the content items available in the system, whereas the central server caches all the content items. Caching policies manage the content availability in the network nodes, thereby also deciding the retrieval traffic. Moreover, the distribution of the retrieval traffic is also shaped by how a retrieval source is selected. As a result, the retrieval delay of a hybrid system hinges on the intensity of retrieval loads due to caching policy as well as source selection.

Caching policies have been widely adopted in various computing systems to improve the content/data retrieval latency [13], [18], [21]. Conventionally, a popularity-based caching policy is applied to optimize the hit ratio for a single node, so that its average latency of retrieving content items is minimized. Analytical caching studies [7], [10] focus on scalability analyses of the caching capacity, especially in purely hierarchal systems. The caching strategy in a distributed system is also referred to as a "duplication" policy [17], which maintains multiple copies of content items either in a distributed or a centralized manner, depending on the synchronization and communication overhead. The performance of a caching strategy depends on the local content popularity, but also on the content distribution among the remaining network nodes. As a hybrid system consists of both hierarchical and P2P-like content retrievals, a well-designed caching policy needs to consider the popularity of content items, the limits on the caching capacity, as well as the retrieval capacities of the server and the network nodes.

Peer-to-peer like systems [15], [20] have been analytically proven to be highly scalable in terms of content provisioning when the number of peer/network nodes increases. A larger number of nodes generates more retrieval traffic, but simultaneously the peer retrieval capacity also grows. On the other hand, the complexity of the peer retrieval traffic and the management overhead also increases with the number of peers. Earlier studies on task assignment [8], [11] have shown that a load-aware source selection strategy can benefit from highly distributed systems, whereas the performance of a system using load-oblivious source selection deteriorates when the number of peer nodes increases. The performance difference between different strategies of source selection grows with the peer traffic intensity. Most analytical derivations are based on homogenous systems, meaning that all nodes store all the content items; however, the system considered here has heterogeneous nodes, which is due to the content availability being managed by caching policies at each node separately. As source selection and caching policies are clearly interrelated, both should be load-aware with regard to the peer network and server retrieval in order to minimize the retrieval delay.

In this study, we provide an analytical framework and develop a simulator to investigate the retrieval latency in the aforementioned hybrid content distribution system. The total retrieval delay is the weighted average of the server retrieval delay and the peer network retrieval delay, which are derived according to the source selection strategy and the caching policy. We consider two retrieval source selection strategies, Bernoulli selection and Shortest-Queue selection. The heterogeneous distribution of content items driven by caching policies is used for approximating the retrieval delay from multiple sources. Two caching policies are used as performance benchmarks for the proposed hybrid caching: a selfish policy where other network nodes are not considered, and an altruistic policy which adopts the proportional replication policy from [17]. Our proposed hybrid caching scheme partitions content items into three classes: gold, silver and bronze, with items in each class managed by different caching schemes. Specifically, items in the silver class are managed locally either by a collaborative LRU (Least Recently Used) scheme, referred to as H-cLRU, or by a random discarding scheme, referred to as H-Rnd.

To the best of our knowledge, this is the first study to have detailed investigations of the retrieval delay in a hybrid system under different caching policies as well as different source selection strategies. The contributions of this paper are three-fold: i) we provide an analytical solution for evaluating the interaction of caching policies and source selection strategies of a hybrid content distribution system; ii) we approximate the multi-queue delay under a heterogeneous server; and iii) we propose a hybrid caching policy, which can adapt to varying buffer limits as well as different network loads and capacity.

The remainder of this paper is organized as follows: The system specification and the description of the selfish and altruistic caching policies are given in Section II. The retrieval delay model is described in Section III. The proposed hybrid caching policy and its analysis are explained in Section IV. Section V contains the experimental results. The related work is presented in Section VI, and Section VII concludes the work.

II. System, Retrieval Selection, and Caching Strategies

The content distribution system considered consists of a central server and N second tier nodes, referred to as the peer network, connected to the server and each other. In the remainder of this study, we use network nodes and peer nodes interchangeably when referring to the second tier nodes. Fig. 1 depicts the system schematics. The server has a buffer capacity that is sufficiently large to store all the available content items. The network nodes have limited storage capacity B_p, and they require a content management policy to decide which content items to store and which to discard. The content management policies are further described later in this section and in Section III.

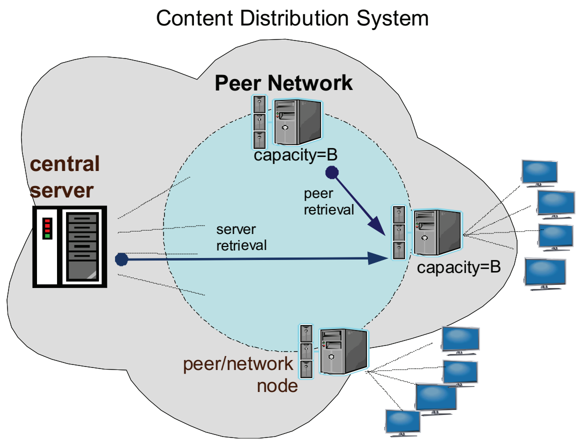


Fig. 1. System schematics of a hybrid content distribution system.

End users first send content retrieval requests to the network node to which they are connected. Each end user is connected to exactly one peer node. There is a total of K unique content items, denoted by subscript $k = \{1 \dots K\}$. The total rate of requests received by a node from end users is λ, and the arrivals follow a Poisson distribution. All content requests are assumed to be uniformly distributed among the peer nodes. We also adopt the common observation [14] that the popularity of content item k, r_k, follows a Zipf distribution, $r_k = \frac{1}{k^\alpha}$. To facilitate further analysis, we normalize r_k, so that $\sum_k r_k = 1$. A single node thus receives requests for content item k at the rate λr_k, $\forall k$. Note that the popularity is assumed to be known by the off-line profiling. Moreover, we present an on-line statistic estimation in Subsection V-C.

Upon receiving a request for a content item, a node satisfies the request from the local cache. If the content item is not available locally, the node queries the rest of the peer nodes in order to retrieve it. Among the network nodes with the requested content item, one is chosen according to the source selection criterions described in the following subsection. If a requested content item is not cached in any of the peer nodes, it will be retrieved from the server. Every network node can sustain C_p horizontal peer retrieval connections, and the server has C_s vertical connections. In this paper we assume that the server has a higher total bandwidth, and thus $C_s > C_p$.

The retrieval latency of a content item from a network node and the server is exponentially distributed with means $\mu_p =$ and μ_s, respectively. When the number of retrieval requests exceeds the connection limits, i.e., C_p or C_s, the request will join the respective waiting queue, from which requests are served following first-come, first-served scheduling. The content retrieval delay is therefore computed as the sum of the retrieval time and the queueing time in the retrieval channels. As it is out of scope for this study, we exclude any delay arising from the process of querying network nodes to find out if they are storing a desired content item from our calculations.

A. Caching Policies

We summarize the optimal caching policies with respect to system and network architecture.

1) Selfish Caching in a Hierarchical System: In a conventional hierarchical content distribution system, content caching is thus designed to optimize for local requests only. We refer to this policy that optimizes the caching for local requests only as *selfish*. Each peer node statically stores the B_p most popular content items with request rate $r_k, k = \{1 \dots B_p\}$. As every node keeps the same set of content items, there is no peer retrieval under such a policy. Requests for content items that are not cached locally will be satisfied by the server. It is straightforward to show that the static selfish policy generates the upper bound of the server load compared to other caching policies. We omit the formal proof due to page limit. One can expect the performance of a system using selfish caching to degrade with an increasing number of peer nodes. This is due to the increasing load on the server, whose retrieval capacity is constant.

2) Altruistic Caching in a P2P-like System: Tewari and Kleinrock [17] showed that "proportional" replication, which keeps the number of content items in the system proportional to their request rates, can result in minimal content retrieval delays in a purely peer-to-peer system under certain assumptions. We refer to the proportional replication policy as *altruistic*. Here, the buffer space in all peer nodes is centrally controlled as a single, large buffer. All the peers collaborate tightly to maximize the overall peer network performance. As server retrieval is not explicitly considered when using this policy, it is possible that the server capacity may be underutilized. Consequently, applying an altruistic policy in a hybrid system can provide the upper bound of the peer network size and load. In the following, we implement the idea of proportional replication proposed by [16]. First, we compute n_k*, the optimal number of copies of content item k in the peer network, which in principle is proportional to the request rate r_k and which satisfies the following equations.

$$0 \leq n_k \leq N,$$
$$\sum_k n_k = NB.$$

As a node does not keep more than one copy of content item k, the maximum number of content item k is thus N. The total number of content items cached in the system is equal to the total system capacity, NB. After determining the number of copies of each content item, we use the principle of load balancing to distribute those copies among all nodes.

Note that Tewari and Kleinrock in [17] showed that proportional replication can be roughly achieved by *collaborative LRU* caching when the request rate distribution has a low Zipf exponent. Each peer node maintains a list of the most recently used B content items. The list is updated by local requests as well as peer requests. When a local request is not a hit and it is retrieved from a peer, both the local and peer LRU-lists are updated. One variant of our proposed hybrid caching policies in Section IV applies collaborative caching on part of the content items.

3) Caching in a Hybrid System: Dynamic collaborative caching policies were proposed to minimize the retrieval traffic and bandwidth consumption of the peer network as well as the server [4], [5]. The collaborative caching policies acquire different degrees of information about the distribution of the content cached by peers. Consequently, the decision of whether or not to cache a content item needs to consider the trade-off between the local hit ratio, the peer hit ratio and the server hit ratio. However, the server and peer network retrieval capacities are usually not caching criteria considered in optimizing retrieval delay. In Section IV, we propose a load-aware hybrid caching policy which combines the merits of the existing selfish, altruistic and collaborative caching policies. This hybrid caching policy partitions the content items into different classes, to which different caching policies are applied in order to optimize the retrieval delay. We list key notations and corresponding definitions in Table I.

TABLE I
NOTATION AND DEFINITIONS

Notation	Definitions
K	number of contents
N	number of peer nodes
B	buffer space in peer nodes
r_k	request rate of content item k
λ	total request arrival rate per node
$\mu^{\{s,p\}}$	content retrieval rate from a connection of server/peer
$C_{\{s,p\}}$	number of retrieval connections per server/peer
π_k	steady state buffer occupancy of content item k
$\Psi^{l,s,p}$	local, peer, and server hit ratio
$D^{\{s,p\}}$	average delay at peer network and server

B. Source Selection

Source selection strategies have been well studied in the context of optimizing response time in web-server and P2P systems [4], [12]. The selection can be load-oblivious or load-aware selection, the latter of which has been shown optimal in minimizing response time in generic multi-queue systems. In this paper, we use Bernoulli and Shortest-Queue selection among the network nodes, which are heterogeneous because of caching different content items.

1) Bernoulli Selection: From n_k peers having content item k, one is selected with the probability of $\frac{1}{n_k}$. In static caching, the overhead of applying Bernoulli selection is negligible as the caching of $n_k\ \forall k$ items is fixed, whereas it has non-negligible overhead in dynamic caching because the content distribution changes.
2) Shortest-Queue Selection: From n_k nodes having content item k, the node which has the lowest number of peer retrieval requests waiting in the queue is selected. The implementation overhead depends on finding n_k in static and as well as dynamic caching. Therefore, it has the same order of implementation complexity as Bernoulli splitting in dynamic caching.

Note that existing analytical results regarding source selection are based on the assumption of homogenous peer content distribution. The analytical results of Shortest-Queue selections are based on approximation, especially for larger number of homogenous queues/peers. To analytically obtain the retrieval delay in the system considered here, the heterogeneous content distribution needs to factored into existing derivations of retrieval delay for both source selection strategies.

III. Retrieval Delay Modeling

As the retrieval delay from the local cache is assumed to be 0, the average content retrieval delay, D, is the weighted average of peer network retrieval delay, D^p, and the server retrieval delay, D^s. The corresponding weights, Ψ^s and Ψ^p, are the ratios of peer-retrieved and server-retrieved requests to total content requests:

$$D = \Psi^p D^p + \Psi^s D^s. \tag{1}$$

The ratio of locally-retrieved content items is denoted by Ψ^l, and $\Psi^l + \Psi^p + \Psi^s = 1$. The exact derivations of Ψ^p, Ψ^s, and the arrival rates at the server and peer network used for D_p and D_s in the following subsections are provided in the corresponding caching subsections.

A. Server Retrieval Delay, D^s

The server retrieval is modeled as an $M/M/C_s$ queueing system, where the number of server retrieval connections is C_s and there is one waiting queue. The total traffic arriving at the server is

$$\lambda^s = N\lambda\Psi^s,$$

which is caching policy dependent. The overall service rate is $C_s\mu^s$, and the utilization (load intensity) is $\rho^s = \frac{\lambda^s}{C_s\mu^s}$. The stability condition of the server system is

$$N\lambda\Psi^s \leq C_s\mu^s. \tag{2}$$

According to [3], the mean delay at the server is,

$$\begin{aligned} D^s &= \frac{Q^s}{\lambda^s}, \, where \qquad (3)\\ Q^s &= C_s\rho^s + \frac{\rho^s}{1-\rho^s}\frac{(C_s\rho^s)^{C_s}}{C_s!(1-\rho^s)} \cdot \pi_0 \\ \pi_0 &= \left\{ \sum_{k=0}^{C_s-1} \frac{(C_s\rho^s)^k}{k!} + \frac{(C_s\rho^s)^{C_s}}{C_s!}\frac{1}{1-\rho^s} \right\}^{-1}. \end{aligned}$$

B. Peer Retrieval Delay, D^p

The peer retrieval delay depends on dynamics of loads, which are controlled by the source selection strategies. Each peer retrieval queue can be analyzed independently when Bernoulli selection is used, whereas using Shortest-Queue selection requires all queues to be modeled simultaneously.

1) Bernoulli Selection (D^p_{Bnl}): The Bernoulli retrieval selection is load-oblivious, so the peer retrieval traffic received by each peer is independent from that received by other peers. Therefore, one can analyze the entire peer network as N independent $M/M/C_p$ queueing systems, node $i \in \{1 \dots N\}$ of which receives retrieval traffic at the rate λ^{p_i}, has a service rate of $C_p\mu^p$ and a corresponding retrieval delay of D^{p_i}. As we here consider $C_p = 1$, we can simply compute D^{p_i} by applying $M/M/1$ analysis [3] with $\rho^{p_i} = \frac{\lambda^{p_i}}{\mu^p}$ on all peer retrieval queues, then

$$D^p_{Bnl} = \mathbf{E}[D^{p_i}] = \sum_{i=1}^{N} \frac{1/(\mu^p - \lambda^{p_i})}{N}. \tag{4}$$

Note that as all content items are uniformly distributed under the proposed hybrid caching policy, all peer queues and their corresponding D^{p_i} are thus identical. D^p_{Bnl} can be computed by considering one queue as

$$D^p_{Bnl} = \frac{1}{\mu^p - \lambda^p_{Bnl}}, \, and \, \lambda^p_{Bnl} = \lambda\Psi^p. \tag{5}$$

The retrieval delay increases with N and ρ^p [11] under Bernoulli random splitting. This implies that given the fixed traffic intensity, the more distributed a system gets (bigger N), the higher the retrieval delay will be.

2) Shortest-Queue Selection (D^p_{SQ}): The peer network here is similar to an $M/M/C_pN/JSQ$ queueing model [3], where content items can be retrieved from all C_pN peer retrieval queues, and JSQ stands for Join Shortest Queue and denotes the source selection strategy used here. As $C_p = 1$, we simplify C_pN as N. The exact analysis of $M/M/N/JSQ$ is almost restricted to $C_pN = 2$. For $C_pN > 2$, the approximated analysis is based on an $M/M/C_pN$ queueing model, where there is only a single queue for all C_p nodes. In general, the retrieval delay of $M/M/C_pN/JSQ$ grows with traffic intensity and decreases with C_pN. The more widely distributed the content items are, the lower the retrieval delay is. As every node stores only a subset of content items due to caching limit and strategy, a requested content can be found and retrieved from a subset of N nodes, and the performance of this system differs very much from a system where content items are retrieved from all N nodes. Therefore, prior to obtaining the approximated retrieval delay using $M/M/C_pN/JSQ$ analysis, it is critical to know how many peers hold the requested content items, or, in other words, the distribution/dispersion of content items within the network. As peer retrieval occurs for content k with $0 < n_k < N$, we estimate N^*, the average number of peer nodes available for peer retrieval, by

$$N^* = \mathbf{E}_{k\in\{0<n_k<N\}}[n_k], \tag{6}$$

where n_k is number of copies of content item k. We are now ready to apply the approximated analysis of $M/M/C_pN^*/JSQ$ developed by Gupta et. al. [8] by using the arrival rate $\lambda^p_{SQ} = N\lambda\Psi^p$, the service rate $C_pN\mu^p$, and the traffic intensity $\rho^p_{SQ} = \frac{\lambda^p_{SQ}}{C_pN\mu^p}$ of the entire peer network.

C. Selfish Policy

In the selfish policy, all peer nodes are selfishly and statically storing the B most popular content items. As such, every peer node needs to retrieve content items $k > B$ from the server, $\Psi^s = \sum_{k=B+1}^{K} r_k$. The arrival rate of content retrieval requests to the server is $\lambda^s = N\lambda\Psi^s = N\lambda\sum_{k=B+1}^{K} r_k$, which is used to derive the server delay D^s in Eq. 3. Following the stability condition in Eq. 2, we know that the maximum system size is

$$N < \frac{C_s\mu^s}{\sum_{k=B+1}^{K} r_k}.$$

The content items which are not cached locally can only be found at and retrieved from the server, so $\Psi^p = 0$. Therefore, no peer retrieval traffic is generated when using this policy, and $D^p = 0$.

D. Altruistic Caching

Following the proportional replication principle stated in Section II, each node i is assigned a unique set of content items, F_i, whose cardinality is $|F_i| = B$, and there are n_k copies of content item k in the network. The server retrieval ratio is the sum of the request rates for content items of which there are no copies stored in the peer network, i.e., $n_k = 0$,

$$\Psi^s = \sum_{k\in\{n_k=0\}} r_k.$$

Therefore, the arrival rate at the server is $\lambda^s = N\lambda\sum_{k\in\{n_k=0\}} r_k$, on which Eq. 3 can be applied to obtain D^s. For content items k of which there are $1 \le n_k < N$ copies, peer retrieval requests are generated from the $N - n_k$ nodes not holding content item k with the probability $r_k\frac{N-n_k}{N}$. Therefore, the overall peer retrieval ratio is

$$\Psi^p = \sum_{k\in\{0<n_k<N\}} r_k\frac{N-n_k}{N}.$$

1) Bernoulli selection (D^p_Bnl): A peer node i assigned with one copy of content item k receives peer requests for this content item with the probability $1/n_k$, meaning peer retrieval requests for content item k here is $\lambda r_k\frac{N-n_k}{n_k}$. The total retrieval traffic received by node i is the sum of peer retrieval requests for content item $k \in F_i$,

$$\lambda^{p_i} = \lambda\sum_{k\in F_i} r_k\frac{N-n_k}{n_k}.$$

Therefore, the stability condition of the entire peer network is $\max_i \lambda\sum_{k\in F_i} r_k\frac{N-n_k}{n_k} \le \mu^p$. The retrieval delay of node i, D^{p_i}, of Eq. 4 can then be used to derive $D^p_{Bnl} = E[D^{p_i}]$.

2) Shortest-Queue selection (D^p_SQ): As n_k is deterministic in altruistic caching, N^* can be straightforwardly obtained by $N^* = \sum_k \in \{0 < n_k < N\}\frac{n_k}{K}$. Taking $\lambda^p = N\lambda\Psi^p$ in $M/M/C_pN^*/JSQ$ as described in Subsection III-B2, one can obtain D^p_{SQ}.

IV. Hybrid Caching

We believe that the optimal caching policy in a hybrid system is to adopt different degrees of selfishness and altruism depending on the system and network configuration. We propose the following caching policy, which partitions content items into 3 classes: (1) gold content; (2) silver content; and (3) bronze content, to each of which different caching strategies are applied. Thresholds, T_1 and T_2, are used to define each class. The gold content items, with $k \le T_1$, are the ones with the highest request rates, and these content items are kept selfishly in all peer nodes. The bronze content items, where $k > T_2$, are the ones with the lowest request rates and they are not kept in the peer network at all. Content items where $T_1 < k \le T_2$ are the silver class. These content items are always stored when received by a node. However, to make room for new content items, the silver items are also periodically discarded – the content item to discard is chosen either by a collaborative LRU (cLRU) policy, or randomly (Rnd). We name the two variants of the proposed hybrid caching policy *H-cLRU* and *H-Rnd*, respectively. The pseudo-code of the caching algorithm is illustrated in Algorithm IV.

The threshold values T_1 and T_2 are chosen with the following rationales: As the Zipf-distributed content request rates result in high popularity of a few content items (items with a small k value), we propose to use α percentage of a single buffer to keep stand-alone copies of those popular content items at each peer node. For example, when $K = 300$, $B = 30$, one can use $\alpha = 20\%$ of the buffer space to keep the most popular $T_1 = 6$ $(30\cdot 0.2)$ content items, which guarantees a 50% $(\sum_{k=1}^{k=6} r_k = 50\%)$ local hit ratio when $r_k = \frac{1}{k^{1.2}}$ and $\sum_k r_k = 1$. The value of T_2 is used to explicitly take advantage of server utilization at a certain level, say β, so that the demerits of low server utilization of altruistic caching can be avoided. To utilize at least β percentage of the server total retrieval capacity, we keep the total request rates of bronze content items at $N\lambda\sum_{k=T_2}^{K} = \beta(\mu_s C_s)$. The β value is kept low here as silver class content items generate additional server retrieval traffic. The optimal values of T_1 and T_2 can be numerically evaluated using the analysis provided in the following subsection.

To choose which silver class content items $(T_1 < k \le T_2)$ to discard from the local buffer, we apply either collaborative LRU or random discarding. Collaborative LRU discarding updates the LRU list upon receiving local requests and peer requests. In the light of the effectiveness of random discarding schemes in unstructured peer networks [5], [19], we propose using random discarding policy as a less selfish alternative to LRU. Note that the proposed hybrid caching policy is completely distributed and can be optimized according to the received request rate and the server and network retrieval and caching capacity.

A. Analysis

To obtain the retrieval delay D, we first show the derivation of Ψ^p, Ψ^s, based on steady state probability of content items being cached in a node, denoted as π_k. We refer to π_k

Algorithm 1 Hybrid Caching

```
if $k \le T_1$ (Gold) then
  Keep stand-alone content item $k$ locally.
else
  if $T1 < k \le T_2$ (Silver) then
    Always cache content $k$.
    When buffer is full,
    (1) Collaborative LRU discarding (H-cLRU) or
    (2) Random discarding (H-Rnd).
  end if
else
  if $k > T_2$ (Bronze) then
    Never cache.
  end if
end if
```

as content diffusion. The content retrieval requests can be fulfilled by peers when the requested content item is not available locally, which occurs with a probability $1-\pi_k$, and at least one of $N-1$ peer nodes has the content item in question, occurring with the probability $1-(1-\pi_k)^{N-1}$. Thus, we let the peer-retrieved probability of content item k be $\psi_j^p = (1-\pi_k)(1-(1-\pi_k)^{N-1})$. On the other hand, an uncached content item k is retrieved vertically from the central server with the probability $(1-\pi_k)^N$. Therefore, we can derive Ψ^p and Ψ^s in the following:

$$\Psi^s = \sum_k r_k(1-\pi_k)^N,$$
$$\Psi^p = \sum_k r_k\psi_k^p = \sum_k r_k(1-\pi_k)(1-(1-\pi_k)^{N-1}).$$

The arrival rates used in Eq. 3, 5, Subsection III-B2 for D^s, D^p_{Bnl}, and D^p_{SQ} are

$$\lambda^s = N\lambda\Psi_s,$$
$$\lambda^p_{Bnl} = \lambda\Psi^p,$$
$$\lambda^p_{SQ} = N\lambda\Psi^p.$$

The average content dispersion N^* for computing D^p_{SQ} is estimated by applying $n_k = N\pi_k$ in Eq. 6. We are now ready to derive π_k of the gold, silver and bronze classes, based on their respective caching policies. Gold content items, $k \le T_1$, are always cached, so $\pi_k = 1, k \le T_1$, whereas bronze content items, $k > T_2$, are not cached in any peer nodes, and so $\pi_k = 0, k > T_2$. The π_k of silver content item under collaborative LRU and random discarding can be computed through the iterative analysis shown in the following subsections. Note that there are only $B - T_1$ silver content items that can be cached.

1) Collaborative LRU discarding: The main idea behind the following derivation is based on the conservation law which states that, in steady state, the rate at which an item is brought into the buffer is the same as the rate at which it is taken out of the buffer [7]. Let $p_k(j)$ denote the probability that the content item k is at location j of the LRU list. As such, $p_k(1)$ is the probability that the latest request access is for content item k. As the local LRU list is updated by both local requests and peer requests, access to content k includes both of these request types. The local request rate of content item k is essentially the original request r_k, and peer request rate is $r_k\psi_k^p$. Thus, $p_k(1) = r_k(1+\psi_k^p)$.

Let $a_k(j-1)$ be the conditional probability that content item k is moved from location $j-1$ to location j in the LRU stack after a request, given that a content item is moved from location $j-1$ to j. The steady state of $p_k(j)$ can be effectively approximated as $p_k(j) = a_k(j-1)$ as proposed by [7]. Let $b_k(j)$ be the probability that content item k is contained in one of the first j positions.

$$b_k(j) = \sum_{l=1}^{j} p_k(l). \tag{7}$$

A content item is moved from location $j-1$ to j due to local requests and remote peer request for certain content items which are not cached in the first $j-1$ positions. For the requests generated locally, a content item k is pushed down from location $j-1$ at the same rate as the rate at which content item k is brought into the top $j-1$ positions, $r_k(1-b_k(j-1))$. To satisfy remote peer requests, a content item k needs to be stored between location j and $B-T_1$, so that the rate at which it is moved from location j to $j-1$ due to peer requests is $\psi_k^p(b_k(B-T_1)-b_k(j))$. We can then derive $p_k(j)$ in the following:

$$p_k(j) = a_k(j-1) = \frac{r_k(1-b_k(j-1)) + \psi_k^p(b_k(B-T_1)-b_k(j))}{e(j-1)}, \tag{8}$$
$$e(j-1) = \sum_{k\in\{Silver\}} \left\{r_k(1-b_k(j-1)) + \psi_k^p(b_k(B-T_1)-b_k(j))\right\}. \tag{9}$$

Finally,

$$\pi_k = b_k(B-T_1).$$

One obtains π_k by solving Equations 7-9 through all silver content items and $B-T_1$ buffer positions by setting initial values of π_k as r_k. Such a solving procedure continues until the π_k values converge, and our experiments show that this solving procedure can be very efficient. The pseudo-code of the iterative solving procedure is illustrated in Algorithm 2.

2) Random discarding: A content item k is brought into the local buffer at the rate of $r_k(1-\pi_k)$, when end users ask for content item k at rate of r_k and such a content item is not cached, which occurs with probability $1-\pi_k$. Let U be the total rate at which a content item is brought into the local buffer,

$$U = \sum_{k\in\{Silver\}} r_k(1-\pi_k). \tag{10}$$

According to the conservation law, a content item k is discarded at a rate of $\frac{\pi_k}{B-T_1}$, which is proportional to its occupancy in the buffer. A content item k is removed from the local cache at a rate of $\frac{\pi_k}{B-T_1}U$. Therefore, the content item k satisfies the conservation law in the following,

$$r_k(1-\pi_k) = \frac{\pi_k}{B-T_1}U.$$

Algorithm 2 Iterative solving procedure for π_k of silver content in collaborative LRU discarding

```
Initialize $\pi_k = r_k$
while $\sum_{k\in\{Silver\}} |\pi_k - \pi_k^{old}| \le \delta$ do
  $\pi_k^{old} = \pi_k$
  for j=1 to $K - T_1$ do
    for k= $T_1 + 1$ to $T_2$ do
      if $k = T_1$ then
        $p_k(j) = r_k(1 + \psi_k^p)$
      else
        Compute $p_k(j) = a_k(j)$ in Eq. 9.
      end if
      Compute $b_k(j)$ in Eq. 7.
    end for
    Compute $e(j)$ in Eq. 9.
  end for
  $\pi_k = b_k(B - T_1)$
end while
```

We can then obtain

$$\pi_k = \frac{r_k}{\frac{U}{B-T_1} + r_k}. \tag{11}$$

Equations 10 and 11 can be solved iteratively. We adopt the iterative algorithm in [7] by initializing $U = 1$. The iteration continues until the U value converges.

V. Numerical and Simulation Results

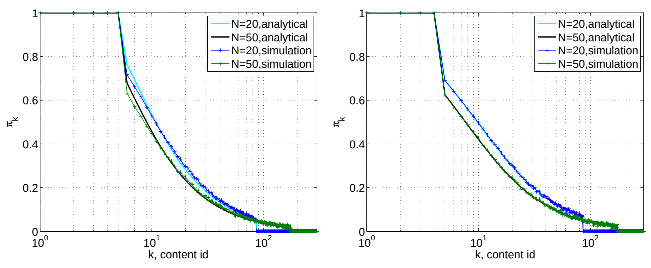
To validate the proposed framework for analyzing the retrieval delay, we built an event-driven simulator using the Java programming language. The focus here was on the scalability of the system, with respect to the content retrieval delay under the retrieval selection and caching policies considered. The proposed hybrid caching policy is compared to altruistic caching in peer networks of different sizes, and with varying buffer space. To manage transient popularity changes, we further develop an on-line implementation of H-cLRU and H-Rnd and present the simulation results in the last section.

In our experience, selfish caching can perform comparably to hybrid caching only when the buffer size is small and the number of peer nodes is kept low. However, these results are not presented here, as the selfish caching policy does not scale to the peer number and buffer sizes considered in the following experiments.

A hybrid system serving $K = 300$ content items is considered. The central server has $C_s = 10$ retrieval connections with a mean retrieval time of $\frac{1}{\mu_s} = \frac{1}{10}$. Each peer node has $C_p = 1$ connection with a mean retrieval time of $\frac{1}{\mu_p} = \frac{1}{8}$. The request arrival rate from end-users at a node is $\lambda = 0.22$. The values of the system size N and caching capacity B are varied in the following subsections.

A. Scalability of Retrieval Selections and Caching

We first present the retrieval delay with increasing N, under the constant caching capacity of a single node, i.e., $B = 20$. To decide the threshold values of T_1 and T_2 in the hybrid caching, we apply the following simple principles: (1) keep the server at least $\beta = 50\%$ utilized; and (2) use $\alpha = 25\%$ of the peer node buffer to keep static copies of popular content items. As a result, $T_1 = 5$ when $B = 20$ and T_2 dynamically increases as the number of peers grows, thus maintaining a constant degree of server utilization. Nonetheless, the optimal values of T_1 and T_2 are not necessarily achieved by the aforementioned principles.



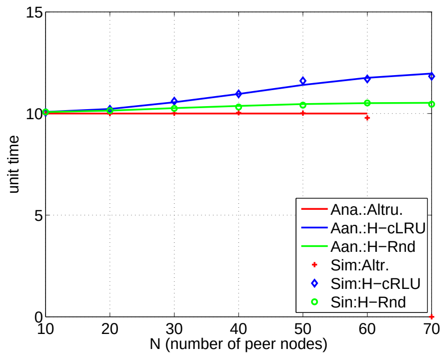
(a) H-cLRU (b) H-Rnd

Fig. 2. π_k: analytical derivation vs. simulation results of H-cLRU and H-Rnd.

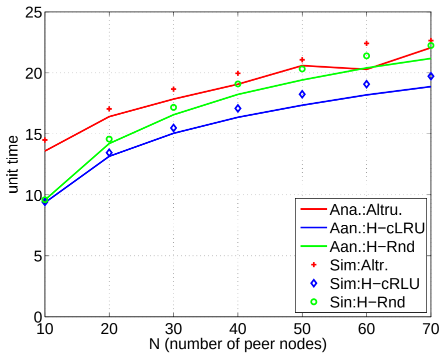
The server, peer and average retrieval delays obtained from the simulations and the derived analysis are depicted in Fig. 3 for the Bernoulli selection and in Fig. 4 for the Shortest-Queue selection. The corresponding statistics of the retrieval traffic – the local hit ratio, Ψ^l, the peer hit ratio, Ψ^p, and the server retrieval ratio, Ψ^s – are summarized in Table II.

From Fig. 3 and Fig. 4, we can observe that the average and server retrieval delays obtained from the analytical derivations match well with the simulation results. Another validation to the analysis of the proposed hybrid caching policies are the steady state probabilities of buffer occupancy by content items, π_k, shown in Fig. 2 for system sizes $N = 20$ and $N = 50$. The analytical results for both H-cLRU and H-Rnd match well with the simulation results. We believe that the proposed analysis is accurate and can enable a more efficient design space exploration of hybrid system deployment.

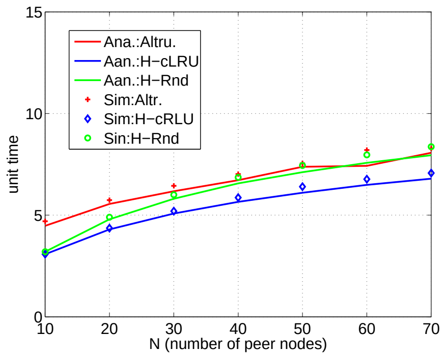
Comparing Fig. 3, and Fig. 4, one can easily observe that for all three caching policies D^p increases with the number of peer nodes when using Bernoulli selection, whereas D^p is relatively stable when using Shortest-Queue selection. Consequently, when $N = 70$, the difference in retrieval delays between Bernoulli and Shortest-Queue is more than a magnitude of 2. In fact, the traffic intensity of the entire peer network ρ^p increases with increasing peer nodes (from roughly $\rho_p = 0.3$ to $\rho_p = 0.6$) for all caching policies. Shortest-Queue selection is extremely effective in distributing the peer retrieval traffic, especially in a larger distributed system. Moreover, when the number of peer nodes increases, the difference in peer retrieval delay between H-cLRU and H-Rnd increases under Bernoulli selection and decreases under Shortest-Queue selection. This observation leads us to believe that: (1) when the retrieval selection is load-oblivious, H-cLRU is more desirable due to its higher percentage of local hit and lower peer retrieval ratios; and (2) H-Rnd is more recommended under load-aware



(a) Server retrieval delay, D^s

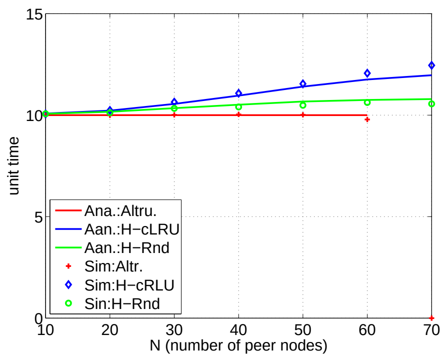


(b) Peer retrieval delay, D^p

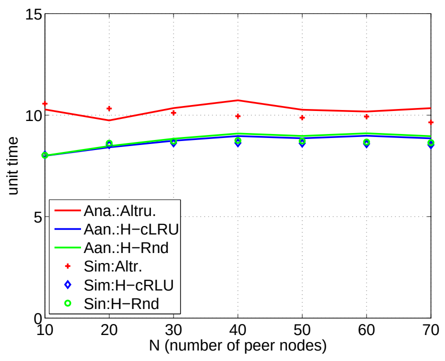


(c) Average retrieval delay, D

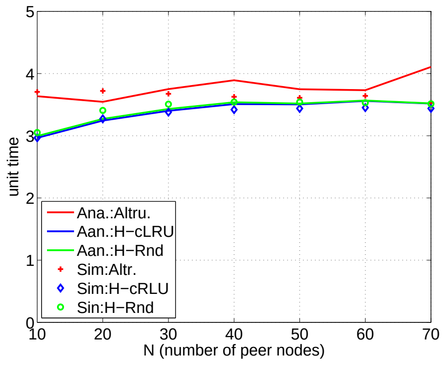
Fig. 3. **Bernoulli Selection**: analytical derivation vs. simulation results with $B = 20$.



(a) Server retrieval delay, D^s



(b) Peer retrieval delay, D^p



(c) Average retrieval delay, D

Fig. 4. **Shortest-Queue Selection**: analytical derivation vs. simulation results with $B = 20$.

TABLE II
TRAFFIC STATISTICS: CONTENT LOCAL HIT RATIO, PEER HIT RATIO, AND SERVER RETRIEVAL RATIO, WHEN $B = 20$ AND $\lambda = .22$.

	N=10			N=20			N=30			N=40			N=50			N=60		
Policy	Altru.	H-cLRU	H-Rnd	Altru.	H-cLRU	H-Rnd	Altru.	H-cLRU	H-Rnd	Altru.	H-cLRU	H-Rnd	Altru.	H-cLRU	H-Rnd	Altru.	H-cLRU	H-Rnd
local Ψ^l	.64	.69	.68	.64	.64	.63	.64	.63	.61	.64	.63	.61	.63	.62	.61	.63	.62	.60
peer Ψ^p	.25	.08	.09	.30	.22	.25	.32	.27	.29	.34	.29	.32	.35	.31	.33	0.36	.32	.34
server Ψ^s	.10	.23	.23	.06	.13	.13	.04	.10	.09	.03	.08	.07	.01	.07	.06	.002	.06	.05

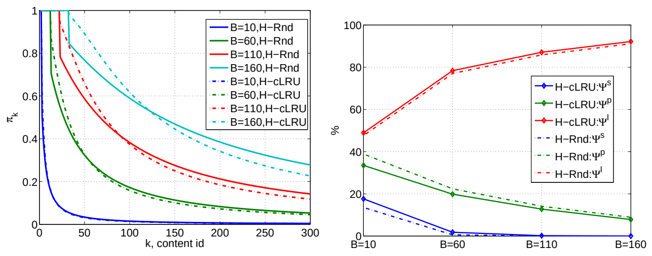
selection, i.e., Shortest-Queue selection, especially in a larger system.

Moreover, the altruistic caching policy results in a constant local hit ratio, an increasing peer hit ratio and a decreasing server retrieval ratio with an increasing number of peers N. The trend of increasing peer hit ratio and decreasing server retrieval ratio can also be observed in H-cLRU and H-Rnd; however, their local hit ratios decrease slightly due to the increasing T_2. One can also observe that given the same threshold values of T_1 and T_2, H-cLRU consistently generates more server retrieval traffic and lower peer retrieval traffic than H-Rnd. As H-cLRU and H-Rnd explicitly take advantage of the server retrieval capacity, the peer delay with the altruistic caching policy is higher than with H-cLRU and H-Rnd, especially when the size of the peer network is small, as shown in Fig. 3 (a) and 4 (a). On the other hand, the peer retrieval capacity becomes significantly larger than the server retrieval capacity in a larger peer network, i.e., $N > 50$, and here altruistic caching can perform very well by solely relying on peer retrievals, which are faster than the server retrieval. Nevertheless, the proposed hybrid caching policies perform well by dynamically adopting to the different network configurations.

B. Collaborative LRU vs. Random Discarding

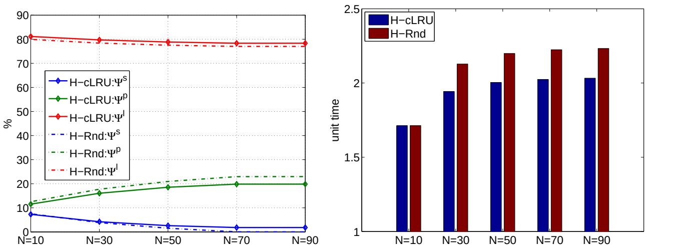
For a fair comparison between the two proposed policies H-Rnd and H-cRLU, we only apply Shortest-Queue selection in this subsection.

1) Increasing Buffer: In this scenario, we have a fixed number of peer nodes, $N = 20$, and increase the buffer size B from 10 to 70 in increments of 10. We use $\alpha = 20\%$, so that $T_1 = 0.2B$. Fig. 5 (a) depicts π for H-cLRU and H-Rnd. One can see that H-cLRU has a higher π_k for more popular content compared to H-Rnd. This difference also increases with B. The corresponding local hit ratio Ψ^l, the peer hit ratio Ψ^p and the server retrieval ratio Ψ^s with increasing values of B are summarized in Fig. 5 (b) – the bigger the buffer, the higher the local hit ratio. The peer hit ratio and server retrieval ratio



(a) π_k (b) Ψ^l, Ψ^p and Ψ^s

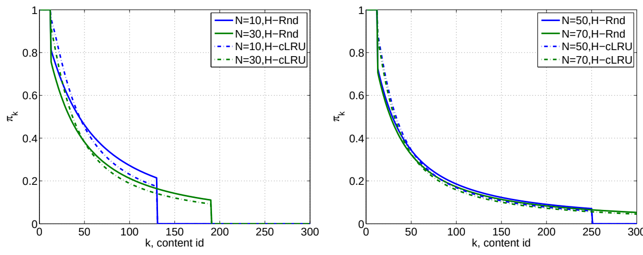
Fig. 5. Comparisons of H-cLRU and H-Rnd with increasing B when $N = 20$.



(a) Ψ^l, Ψ^s, and Ψ^p (b) Delay (D)

Fig. 7. Comparisons for H-cLRU and H-Rnd with increasing N when $B = 50$.

decrease with increasing B. Moreover, the difference in the peer hit ratio and the server retrieval ratio between H-cLRU and H-Rnd decreases with increasing B. When $B = 160$, H-cLRU has only a slightly higher local hit ratio at a trade off of slightly lower peer hit ratio, compared to H-Rnd. Overall, the retrieval delay significantly decreases with larger buffer sizes.



(a) $N = 10, 30$ (b) $N = 50, 70$

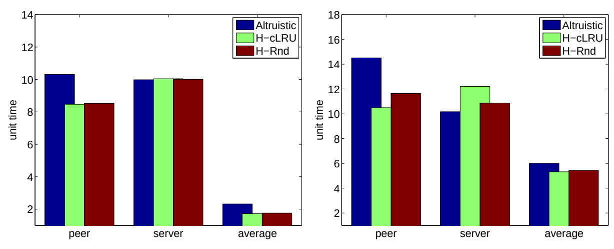
Fig. 6. Comparing π_k of H-cLRU and H-Rnd with increasing N.

2) Increasing N: Here the caching capacity is $B = 50$, and the system size is $N = \{10, 30, 50, 70, 90\}$. Fig. 6 depicts how π_k changes with increasing N under H-cLRU and H-Rnd with $B = 50$. The curves of π_k are shown to converge with increasing N. Meanwhile, the difference between the π_k curves of H-cLRU and H-Rnd decreases as N increases. However, the difference between the peer hit ratio and the server retrieval ratio of the two policies increases as shown in Fig. 7 (a). This is because the larger N magnifies the difference of π_k. H-Rnd has visibly higher peer retrieval traffic at a trade off of negligible server retrieval traffic compared to H-LRU when $N = 90$. We also summarize the retrieval delay in Fig. 7 (b), and H-cLRU performs slightly better than H-Rnd for all N values compared. Similar to the peer hit ratio, the marginal gain in retrieval delay for H-cLRU over H-Rnd decreases with increasing N. From our experiments, we observe that H-Rnd is simple yet very robust, especially when the peer network is large.

C. Online Hybrid Caching

To adapt to changes in popularity among the content items over time, we developed online algorithms for H-cLRU and H-Rnd. Two steps are required: (1) a frequency estimation of r_k; and (2) an adaptive threshold setting. We use a moving-window average to periodically estimate the content popularity. If the popularity has changed, we recalculate the gold (T_1) and bronze (T_2) thresholds using the principles described in Section IV.

1) Simulation Results: We simulate two types of content popularity changes: (1) increasing the exponent of the Zipf distribution from 1.2, 1.4, 1.5 and then back to 1.3 in a system where $B = 20$, $N = 10$ and $\lambda = 0.22$; and (2) decreasing the exponent of the Zipf distribution from 1.2, 1.0, 0.9 and finally back to 1.1 in a system where $B = 50$, $N = 10$. Each of the exponent values are simulated for an equal amount of time. The Shortest-Queue selection is applied. The resulting retrieval delays obtained from online H-cLRU and H-Rnd are depicted in Figure 8. The predefined values for gold and bronze thresholds are $\alpha = 0.5$ and $\beta = 0.5$. For comparison purposes, we also study the system performance where the peer nodes employ an altruistic caching policy where the fixed number of content items in the network is based on an exponent value of 1.2. In both cases, H-cRU and H-Rnd can achieve lower average retrieval delay by having lower peer retrieval delay and slightly higher server retrieval delay compared to the altruistic policy. Another general observation is that altruistic caching can be very robust with respect to popularity changes when the total system buffer space is sufficiently large. As the proposed hybrid caching is fully distributed, an on-line algorithm can be implemented to adopt to popularity changes with negligible overhead. From the results presented here, we believe that H-cLRU and H-Rnd also have good potential to be used in highly dynamic hybrid content distribution systems. Nevertheless, the performances of H-cLRU and H-Rnd can be improved by tuning the control parameters, such as α and β.



(a) $\frac{1}{k^{1.2}} \to \frac{1}{k^{1.4}} \to \frac{1}{k^{1.5}} \to \frac{1}{k^{1.3}}$ (b) $\frac{1}{k^{1.2}} \to \frac{1}{k^{1.0}} \to \frac{1}{k^{0.9}} \to \frac{1}{k^{1.1}}$

Fig. 8. Comparisons of the average retrieval delays when request popularity changes.

VI. Related Study

Previous related work mainly addresses the searching delay under various content caching/replication policies and P2P network topologies. Cohen and Shenker [6], [12] compared and evaluated different replication strategies, namely uniform, proportional and square root. The square root replication strategy is proved to optimally minimize the expected search size and can be carried out by their proposed path replication algorithm. In particular, LRU and LFU (Least Frequently Used) strategies are not considered in their content deleting process. Tewari and Kleinrock [16] showed that replicating content in proportion to the request rate can minimize the average retrieval time and also ensures fairness in the workload distribution. They further showed that LRU can automatically achieve near-optimal proportional replication in a distributed manner and minimize the average network bandwidth used per download [17]. However, the server retrieval capacity and the impact of retrieval selection are not considered. There are few analytical studies in the design of hybrid system with a central server and one or more peer nodes. Ioannidis and Marbach [9] developed a mathematical model to analyze the scalability of the query searching time of a hybrid P2P network under two query propagation mechanisms, the random walk and the expanding ring. Borst et al. [4] have developed a collaborative caching strategy in the hybrid system to optimize bandwidth consumption. We focus on the retrieval delay of a hybrid system from the perspective of the dynamics of loads, which are influenced by both caching policies and retrieval selection strategies.

VII. Concluding Remarks

In this paper we provide analytical modeling to investigate the scalability of retrieval delay in a hybrid system under different source selection strategies and caching policies. The steady-state analysis of content items derived from caching policies is used to compute the retrieval loads on the server and network nodes. The retrieval delay is then derived according to the loads and source selection among nodes with heterogeneous cached content distribution. The retrieval delay of the Shortest-Queue selection strategy is shown to strengthen the scalability of caching policies in large systems, whereas retrieval delay using Bernoulli selection decreases with increasing system size. In our experiment results, our two proposed distributed hybrid caching policies, H-cLRU and H-Rnd, outperform selfish caching and altruistic caching in a variety of network sizes. As the hybrid caching policies partition the content into gold, silver, and bronze classes based on the retrieval capacity, caching capacity and popularity, H-cLRU and H-Rnd can effectively attain a good trade-off between local cache retrieval, peer retrieval and server retrieval with low complexity. H-cLRU is observed to have a higher local hit ratio and a lower peer hit ratio compared to H-Rnd. In a larger system, we recommend H-Rnd combined with Shortest-Queue selection due to the very low complexity and robust retrieval delay. For future work, we would like to further explore optimal threshold values of the hybrid caching policies and develop asymptotic results.

References

[1] http://aws.amazon.com/cloudfront/.

[2] http://www.akamai.com/.

[3] G. Bolch, S. Greiner, H. Meer, and K. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*. Wiley, 2006.

[4] S.C. Borst, V. Gupta, and A. Walid. Distributed Caching Algorithms for Content Distribution Networks. In *Proceedings of IEEE INFOCOM*, 2010.

[5] Y. Chen, M. Meo, and A. Scicchitano. Caching Video Content in IPTV Systems with Hierarchical Architecture. In *Proceedings of International Conference on Communications (ICC)*, 2009.

[6] E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.

[7] A. Dan and D. Towsley. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. *SIGMETRICS Perform. Eval. Rev.*, 18(1):143–152, 1990.

[8] V. Gupta, M. Harchol-Balter, K. Sigman, and W. Whitt. Analysis of Join-the-Shortest-Queue Routing for Web Server Farms. *Perform. Eval.*, 64(9-12):1062–1081, 2007.

[9] S. Ioannidis and P. Marbach. On the Design of Hybrid Peer-to-Peer Systems. In *Proceesings of SIGMETRICS*, pages 157–168, 2008.

[10] P. R. Jelenkovic. Asymptotic Approximation of the Move-to-Front Search Cost Distribution and Least-Recently Used Caching Fault Probabilities. *Ann. Appl. Probab.*, 9(2):430–464, 1999.

[11] H-C Lin and C.S. Raghavendra. An Approximate Analysis of the Join the Shortest Queue (JSQ) Policy. *IEEE Trans. Parallel Distrib. Syst.*, 7(3):301–307, 1996.

[12] Q. Lv, P. Cao, E. Cohen, K. Lai, and S. Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of International Conference on Supercomputing (ICS)*, pages 84–95, 2002.

[13] S. Podlipnig and L. Böszörmenyi. A Survey of Web Cache Replacement Strategies. *ACM Comput. Surv.*, 35(4), 2003.

[14] T. Qiu, Z. Ge, S. Lee, J. Wang, Q. Zhao, and J. Xu. Modeling Channel Popularity Dynamics in a Large IPTV System. In *Proceedings of SIGMETRICS*, pages 275–286, 2009.

[15] F. Simatos, P. Robert, and F. Guillemin. A Queueing System for Modeling a File Sharing Principle. *SIGMETRICS Perform. Eval. Rev.*, 36(1):181–192, 2008.

[16] S. Tewari and L. Kleinrock. On Fairness, Optimal Download Performance and Proportional Replication in Peer-to-Peer Networks. In *IFIP Networking*, pages 709–717, 2005.

[17] S. Tewari and L. Kleinrock. Proportional Replication in Peer-to-Peer Networks. In *Proceedings of IEEE INFOCOM*, 2006.

[18] S. Vanderwiel and D. Lilja. Data Prefetch Mechanisms. *ACM Comput. Surv.*, 32(2), 2000.

[19] V. Vishnumurthy and P. Francis. A Comparison of Structured and Unstructured P2P Approaches to Heterogeneous Random Peer Selection. In *Proceedings of the USENIX Annual Technical Conference*, pages 1–14, 2007.

[20] D. Wu, Y. Liu, and K. Ross. Queuing Network Models for Multi-Channel P2P Live Streaming Systems. In *Proceedings of IEEE INFOCOM*, 2009.

[21] P. Yu and E. MacNair. Performance Study of a Collaborative Method for Hierarchical Caching in Proxy Servers. *Comput. Netw. ISDN Syst.*, 30(1-7):215–224, 1998.