Minimizing Retrieval Latency for Content Cloud

Mathias Björkqvist, Lydia Y. Chen
IBM Research Zurich Laboratory, Switzerland
{mbj,yic}@zurich.ibm.com

Marko Vukolić
Eurécom, France*
vukolic@eurecom.fr

Xi Zhang
Texas A&M University, USA
xizhang@ece.tamu.edu

***Abstract*—Content cloud systems, e.g. CloudFront [1] and CloudBurst [2], in which content items are retrieved by end-users from the edge nodes of the cloud, are becoming increasingly popular. The retrieval latency in content clouds depends on content availability in the edge nodes, which in turn depends on the caching policy at the edge nodes. In case of local content unavailability (i.e., a cache miss), edge nodes resort to source selection strategies to retrieve the content items either vertically from the central server, or horizontally from other edge nodes. Consequently, managing the latency in content clouds needs to take into account several interrelated issues: asymmetric bandwidth and caching capacity for both source types as well as edge node heterogeneity in terms of caching policies and source selection strategies applied.**

In this paper, we study the problem of minimizing the retrieval latency considering both caching and retrieval capacity of the edge nodes and server simultaneously. We derive analytical models to evaluate the content retrieval latency under two source selection strategies, i.e., Random and Shortest-Queue, and three caching policies: selfish, collective, and a novel caching policy that we call the *adaptive* caching policy. Our analysis allows the quantification of the interrelated performance impacts of caching and retrieval capacity and the exploration of the corresponding design space. In particular, we show that the adaptive caching policy combined with Shortest-Queue selection scales well with various network configurations and adapts to the load changes in our simulation and analytical results.

I. Introduction

Content distribution systems are becoming ever more popular in recent years, due to the evolution of Internet technology from web services and peer-to-peer networks to today's content clouds – the growing number of users of CloudFront [1] is merely one example. To provide end-users with ubiquitous content availability, a large amount of network and storage resources need to be deployed; this typically involves a central server and a network of edge nodes. The retrieval latency depends on the way content items are managed and on the distribution of retrieval loads across edge nodes and a central server [3], [13]. As a result, two interrelated factors: 1) the content caching policies and 2) the source selection strategies at edge nodes (used for retrieving data in case of a cache miss), are the main components in optimizing the retrieval latency in a modern content cloud.

The *hybrid* architecture [5], [6] using horizontal and vertical retrieval from the edge nodes and the server, respectively, has been shown effective for delivering content items to end-users. Fig. 1 illustrates the hybrid architecture of the content distribution system we consider in this paper. In principle, in the hybrid architecture, edge nodes contribute to better scalability with increasing total retrieval traffic [18], whereas the central server contributes with a greater bandwidth compared to any given edge node. Moreover, due to the typically smaller caching buffers, edge nodes can usually only cache a subset of the content items, whereas the central server typically stores all the relevant content items. The asymmetric properties of caching and retrieval capacity of the edge nodes and the server add another dimension of complexity to the designing of caching policies.

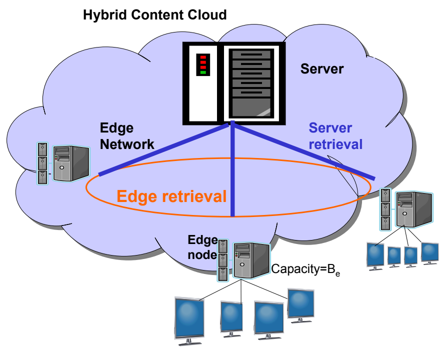


Fig. 1. System schematics of a hybrid content distribution system.

Caching policies have been widely used for improving content retrieval latency [15], [20], [23]. For example, the popularity-based caching policy is applied to optimize the hit ratio for a single node, so that the average latency of retrieving content items is minimized. Analytical studies of caching [8], [11] focus on the scalability of different policies regarding the caching capacity of a single node. On the other hand, practical caching policies [17], [19], [22] typically focus on maximizing the hit ratio in the edge nodes and minimizing the server retrieval traffic.

On the other hand, earlier studies on task assignment [9], [12] have shown that highly distributed systems can benefit from load-aware source selection strategies, e.g., sending a request to the source with the fewest number of outstanding requests. In contrast, load-oblivious source selection, e.g., sending requests to a random source, deteriorates system performance when the number of edge nodes increases. The analytical derivations of these results are based on homogeneous systems, in which all edge nodes cache all content items and end-user retrieval requests can be sent to any edge node.

*This work was done while the author was in the IBM Research Zurich Laboratory.

In this paper, we give the first analytical framework for characterizing (and minimizing) latency in hybrid content clouds that simultaneously considers the impact of different caching policies and source selection strategies. The system we consider is a general one that allows for heterogeneity in edge nodes with respect to local caching and source selection policies as well as asymmetric bandwidth and caching capacity across edge nodes and the central server. In principle, we characterize the average retrieval latency as the weighted average of the server retrieval latency and the edge network retrieval latency.

We put our analytical model to work by considering a) two source selection strategies, namely random selection and Shortest-Queue selection, and b) three caching policies: i) a selfish policy, ii) a collective policy which adopts the proportional replication policy from [19], and iii) a novel caching scheme that we call the *adaptive* caching scheme. Our adaptive caching scheme partitions content items into three popularity classes: gold, silver and bronze, with items in each class managed by different caching schemes. Concretely, items in the gold class are always stored at the edge node, whereas items in the bronze class as never stored. On the other hand, items in the silver class are managed locally either by a collaborative LRU (Least Recently Used) scheme, referred to as Adapt-L, or by a random discarding scheme, referred to as Adapt-R. In our experiments, we explore the scalability of the combinations of caching policies and source selection with various network configurations, i.e. the number of edge nodes, the buffer size, and content popularity changes.

Therefore, the contributions of this paper can be summarized as two-fold: i) we provide the analytical solution for evaluating content retrieval latency influenced by caching policies and source selection strategies in a hybrid content distribution system, and ii) we propose an adaptive caching policy that combined with a shortest-queue selection strategy, itself adaptable to varying buffer limits as well as different network loads and capacity, minimizes the latency of content clouds in the space we consider.

The remainder of this paper is organized as follows: The system specification and the description of the selfish and collective caching policies are given in Section II. We describe our retrieval latency model in Section III. We present our adaptive caching policy and its analysis in Section IV. Section V contains the experimental results. The related work is presented in Section VI. Section VII concludes the paper.

II. System, Caching Policies, and Source Selection Strategies

We consider a generic, hybrid, content cloud architecture, which has been studied in [5] and [3]. The architecture (Fig. 1) comprises a central server and N tier-2 nodes, referred to as the edge nodes (or, collectively, the edge network), connected to the server and each other. The server is assumed to have a sufficiently large buffer for storing all content items, whereas edge nodes have a limited caching capacity of B_e content items. Consequently, a content management policy, i.e., caching policy, is required to ensure a low retrieval latency for end-users, who send retrieval requests to the edge node they are connected to.

In the system there is a total of K unique content items, denoted by subscript $k = \{1 \ldots K\}$. Each node receives end-users' retrieval requests, following a Poisson distribution at rate λ. All content requests are assumed to be uniformly distributed among the edge nodes. Following previous work, we assume the popularity of content item k, r_k, follows a Zipf distribution, $r_k = \frac{1}{k^\alpha}$ [16]. 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, $\forall k$.

Upon receiving a request for a content item, an edge node first tries to satisfy the request from the local cache; if it is unable to do so, it queries the rest of the edge nodes to retrieve the requested content item. Among the edge nodes with the requested content item, one is chosen according to the source selection criteria described in the following subsection. A requested content can always be retrieved from the server if it is unavailable in the edge network. Every node can sustain C_e horizontal edge 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 $C_s > C_e$.

We model the retrieval time of a content item from an edge node and the server as exponential distributions with means $\frac{1}{\mu_e}$ and $\frac{1}{\mu_s}$, respectively. The retrieval requests join the waiting queues of the selected edge nodes or the server, both of which serve requests in a first-come, first-served manner. Delays arising during the process of querying network nodes to find out if they are storing a desired content item are out of scope and excluded from our calculations.

A. Content Caching Policies

We now summarize the content management policies and qualitatively discuss their optimality with respect to the system and network architecture.

1) Selfish Caching: We refer to the caching policy that optimizes for local requests only as *selfish*. Each edge node statically stores the B_e most popular content items with request rate $r_k, k = \{1 \ldots B_e\}$. As every node keeps the same set of content items, there is no horizontal retrieval under such a policy, and thus the corresponding retrieval capacity of the system is underutilized. Consequently, the requests for content items that are not cached locally will be retrieved from the server, whose capacity is constant, independent of the cloud size. Therefore, the performance using selfish caching degrades with an increasing number of edge nodes as the server becomes overloaded.

2) Collective Caching: In contrast to selfish caching, we refer to the caching policy that "centrally" optimizes requests for the entire system as *collective*. Utilizing all the buffer space in a centralized manner, the collective caching policy keeps the optimal number of copies of content item k in the edge network, n_k*, which in principle is proportional to the request rate r_k and which satisfies the following equations: (1) $0 \leq n_k* \leq N$ and (2) $\sum_k n_k* = NB$.

Since an edge node keep at most one copy of content item k, the maximum number of content items is N. The total number of content items cached in the system is equal to the total system capacity, NB. Since server retrieval is not explicitly considered when using this policy, it is possible that the server retrieval capacity may be underutilized.

Optimal Caching in Content Clouds: In summary, to optimize the retrieval traffic and bandwidth consumption of the edge network as well as the server [5], [6], the caching decision needs to consider the trade-offs between the local hit ratio, the edge network hit ratio and the server hit ratio. However, the server and edge network retrieval capacities and loads are usually not caching criteria considered in optimizing retrieval latency. In Section IV, we propose a load-aware hybrid caching policy which combines the merits of the existing selfish and collective 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 latency. We list key notations and corresponding definitions in Table I.

TABLE I
NOTATIONS AND DEFINITIONS

Notation	Definition
K	number of content items
N	number of edge nodes
B	buffer space in edge nodes
r_k	request rate of content item k
λ	total request arrival rate per node
$\mu^{\{s,e\}}$	content retrieval rate from a connection of server/edge nodes
$C_{\{s,e\}}$	number of retrieval connections per server/edge node
π_k	steady state buffer occupancy of content item k
$\Psi^{l,s,e}$	local hit ratio, the retrieval ratios of the server and the edge network
$D^{\{s,e\}}$	average latency of the server and the edge network

B. Source Selection

Source selection strategies have been well studied in the context of optimizing response times in web-server and P2P systems [5], [14]. The selection can be load-oblivious or load-aware. Load-aware selection has been shown to be optimal in minimizing the response time in generic and homogeneous multi-queue systems. In this paper, we use random and Shortest-Queue selection among the edge nodes, which are actually heterogeneous because of caching different content items.

1) Random Selection: From n_k edge nodes having content item k, one is selected with the probability of $\frac{1}{n_k}$. In static caching, the overhead of applying random selection is negligible as n_k is fixed for all k, 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 edge retrieval requests waiting in the queue is selected. The implementation overhead depends on finding n_k using static as well as dynamic caching. Therefore, it has the same order of implementation complexity as random splitting in dynamic caching.

Note that existing analytical results regarding source selection are based on the assumption that all edge nodes are homogeneous, i.e., holding the same set of content items. The analytical results of Shortest-Queue selections are based on approximation, especially for larger number of homogeneous queues/peers. To analytically obtain the retrieval latency in the system considered here, the heterogeneous content distribution needs to be factored into existing derivations of retrieval delay for both source selection strategies.

III. Retrieval Latency Modeling

Since we assume negligible latency of retrievals from the local cache, the average content retrieval latency, D, is the weighted average of the edge network retrieval latency, D^e, and the server retrieval latency, D^s. The corresponding weights, Ψ^e and Ψ^s, are the ratios of edge node and server serviced requests to total content requests:

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

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

A. Server Retrieval Latency, 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 depends on the caching policy and equals

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

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

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

Furthermore, the mean latency at the server is [4]:

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

B. Edge Node Retrieval Latency, D^e

The edge network retrieval latency depends on the dynamics of the load, which is controlled by the source selection strategies. Each retrieval queue of an edge node can be analyzed independently when random selection is used, whereas Shortest-Queue selection requires all queues to be modeled simultaneously.

1) Random Selection (D^e_{rnd}): The random retrieval selection is load-oblivious, so the edge retrieval traffic received by each edge node is independent from that received by other edge nodes. Therefore, one can analyze the entire edge network as N independent $M/M/C_e$ queueing systems, node $i \in \{1 \dots N\}$ of which receives retrieval traffic at the rate λ^{e_i}, has a service rate of $C_e \mu^p$ and a corresponding retrieval latency of D^{e_i}. As we here consider $C_e = 1$, we can simply compute D^{e_i} by applying $M/M/1$ analysis [4] with $\rho^{e_i} = \frac{\lambda^{e_i}}{\mu^e}$ on all retrieval queues of edge nodes. Then:

$$D^e_{rnd} = \mathbf{E}[D^{e_i}] = \sum_{i=1}^{N} \frac{1/(\mu^e - \lambda^{e_i})}{N}. \quad (4)$$

It is important to notice that under the proposed hybrid caching policy, all content items are uniformly distributed and, therefore, all edge queues and their corresponding D^{e_i} are identical. D^e_{rnd} can be computed by considering one queue as

$$D^e_{rnd} = \frac{1}{\mu^e - \lambda^e_{rnd}}, \, and \, \lambda^e_{rnd} = \lambda \Psi^e. \quad (5)$$

The retrieval latency increases with N and ρ^e [12] under random splitting. This implies that given the fixed traffic intensity, the more distributed a system gets (bigger N), the higher the retrieval latency will be.

2) Shortest-Queue Selection (D^e_{SQ}): The edge network here is similar to an $M/M/C_eN/JSQ$ queueing model [4], where content items can be retrieved from all C_eN edge retrieval queues, and JSQ stands for Join Shortest Queue and denotes the source selection strategy used here. Since $C_e = 1$, we simply use N instead of C_eN. The exact analysis of $M/M/N/JSQ$ is almost restricted to $C_eN = 2$. For $C_eN > 2$, the approximated analysis is based on an $M/M/C_eN$ queueing model, where there is only a single queue for all C_e nodes. In general, the retrieval latency of $M/M/C_eN/JSQ$ grows with increasing traffic intensity and decreases with increasing C_eN. The wider the distribution of content items, the lower the retrieval latency. As every node stores only a subset of content items (due to buffer size and caching policy limitations), a requested content item can be found and retrieved from a subset of N nodes. Hence, the performance of this system differs significantly from a system in which content items are retrievable from all N nodes. Therefore, prior to obtaining the approximated retrieval latency using $M/M/C_eN/JSQ$ analysis, it is critical to know how many edge nodes hold the requested content items, or, in other words, the distribution/dispersion of the content items within the network. As edge network retrieval occurs for content k with $0 < n_k < N$, we estimate N^*, the average number of edge nodes available for edge network retrieval, by

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

where n_k is the number of copies of content item k. We are now ready to apply the approximated analysis of $M/M/C_eN^*/JSQ$ developed by Gupta et. al. [9] by using the arrival rate $\lambda^e_{SQ} = N\lambda\Psi^e$, the service rate $C_eN\mu^e$, and the traffic intensity $\rho^e_{SQ} = \frac{\lambda^e_{SQ}}{C_eN\mu^e}$ of the entire edge network.

C. Selfish Policy

In the selfish policy, all edge nodes are selfishly and statically storing the B most popular content items. As such, every edge 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 latency 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 $\Psi^e = 0$. Therefore, no edge retrieval traffic is generated when using this policy, and $D^e = 0$.

D. Collective Caching

Following the proportional replication principle described in Section II, each node i is assigned a unique set of content items, F_i, with cardinality $|F_i| = B$, 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 edge 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 $1 \leq n_k < N$ copies, edge 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 edge network retrieval ratio is

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

1) Random selection (D^e_{rnd}): An edge node i assigned with a copy of content item k receives edge requests for this content item with the probability $1/n_k$, meaning that the edge node retrieval request rate 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 all the edge node retrieval requests for content item $k \in F_i$,

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

Therefore, the stability condition of the entire edge network is $\max_i \lambda \sum_{k \in F_i} r_k \frac{N-n_k}{n_k} \leq \mu^e$. The retrieval latency of node i, D^{e_i}, of Eq. 4 can then be used to derive $D^e_{rnd} = E[D^{e_i}]$.

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

IV. Adaptive Caching

For a hybrid content distribution system, we propose an adaptive caching policy which adopts different degrees of selfishness and altruism depending on the system and network configuration. The adaptive caching policy partitions content items into three 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 $k \leq T_1$, are the ones with the highest request rates, and these content items are kept selfishly in all edge nodes. The bronze content items, where $k > T_2$, are the ones with the lowest request rates and they are not kept in the edge network at all. Content items where $T_1 < k \leq T_2$ 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 policy, or randomly. We name the two variants of the proposed adaptive caching policy *Adapt-L* and *Adapt-R*, 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 small k), we propose to use α percentage of a single buffer to keep stand-alone copies of those popular content items at each edge node. The specific α value should be subject to the total number of content items, the available buffer space and the retrieval rates from server/edges. 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 collective 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 ($T_1 < k \leq T_2$) 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 edge requests. In light of the effectiveness of random discarding schemes in unstructured edge networks [6], [21], we also use random discarding policy as a less selfish alternative to LRU. Note that the proposed adaptive caching policy is completely distributed and can be optimized according to the received request rate and the server and network retrieval and caching capacity.

Algorithm 1 Adaptive Caching

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

A. Analysis of Retrieval Latency

To obtain the retrieval delay D, we first show the derivation of Ψ^e and Ψ^s, based on the steady state probability of content items being cached in a node, denoted as π_k. We refer to π_k as content diffusion. The content retrieval requests can be fulfilled by edges when the requested content item is not available locally, which occurs with a probability $1 - \pi_k$, and at least one of $N - 1$ edge nodes has the content item in question, occurring with the probability $1 - (1 - \pi_k)^{N-1}$. Thus, we let the edge-retrieved probability of content item k be $\psi_j^e = (1 - \pi_k)(1 - (1 - \pi_k)^{N-1})$. 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^e and Ψ^s in the following:

$$\begin{aligned} \Psi^s &= \sum_k r_k (1 - \pi_k)^N, \\ \Psi^e &= \sum_k r_k \psi_k^e = \sum_k r_k (1 - \pi_k)(1 - (1 - \pi_k)^{N-1}). \end{aligned}$$

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

$$\begin{aligned} \lambda^s &= N\lambda\Psi_s, \\ \lambda^e_{rnd} &= \lambda\Psi^e, \\ \lambda^e_{SQ} &= N\lambda\Psi^e. \end{aligned}$$

The average content dispersion N^* for computing D^e_{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, $k \leq T_1$, are always cached, so $\pi_k = 1, k \leq T_1$, whereas bronze content items, $k > T_2$, are not cached in any edge nodes, and so $\pi_k = 0, k > T_2$. The π_k of silver content items 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 the 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 [8]. 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 edge 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 edge request rate is $r_k\psi_k^e$. Thus, $p_k(1) = r_k(1+\psi_k^e)$.

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 [8]. 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 edge request for certain content items which are not cached in the first $j-1$ positions. For the requests generated locally, 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 edge requests, content item k needs to be stored between the locations j and $B-T_1$, so that the rate at which it is moved from location j to $j-1$ due to edge requests is $\psi_k^e(b_k(B-T_1)-b_k(j))$. We can then derive $p_k(j)$ in the following:

$$\begin{aligned} p_k(j) &= a_k(j-1) \\ &= \frac{r_k(1-b_k(j-1)) + \psi_k^e(b_k(B-T_1)-b_k(j))}{e(j-1)}, \quad (8) \\ g(j-1) &= \sum_k \{r_k(1-b_k(j-1)) + \psi_k^e(b_k(B-T_1)-b_k(j))\} \quad (9) \end{aligned}$$

Finally,

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

One obtains π_k by solving Equations 7-9 for all silver content items and $B-T_1$ buffer positions by setting initial values of π_k as r_k. This solving procedure continues until the π_k values converge; 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

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 $g(j)$ in Eq. 9.
    end for
    $\pi_k = b_k(B - T_1)$
end while
```

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,

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

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 [8] by initializing $U = 1$. The iteration continues until the U value converges.

V. Numerical and Simulation Results

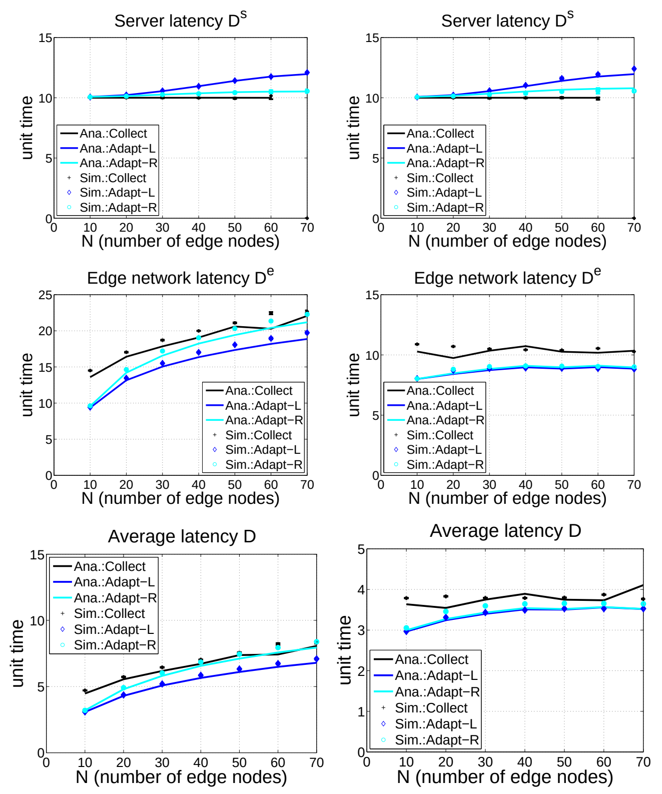
To validate the proposed framework for analyzing the retrieval latency, we built an event-driven simulator in Java. The focus here was on the scalability of the system, with respect to the content retrieval latency under the retrieval selection and caching policies considered. To manage transient popularity changes, we further develop an online implementation of Adapt-L and Adapt-R and present the simulation results in the last section.

Our experience shows that selfish caching performs comparably to adaptive caching only when the buffer size is small and the number of edge nodes is kept low. However, we omit these results, given that the selfish caching policy does not scale to the number of nodes 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 $\frac{1}{\mu_s} = \frac{1}{10}$ unit time. Each edge node has $C_e = 1$ connection with a mean retrieval time of $\frac{1}{\mu_e} = \frac{1}{8}$ unit time. The request arrival rate from end-users at a node is $\lambda = 0.22$ per unit time. The values of the system size N and caching capacity B are varied in the following subsections.

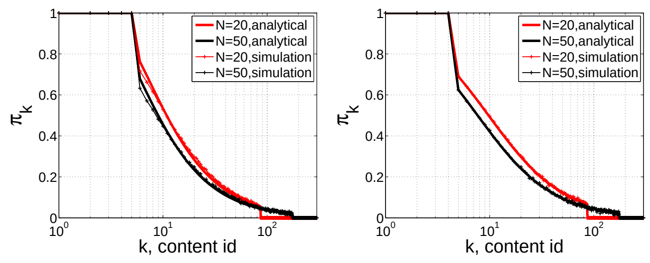
A. Analytical v.s. Simulation

We first present the retrieval latency 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



(a) Random selection (b) Shortest-Queue selection

Fig. 2. Latency: analytical derivation vs. simulation results with $B = 20$.



(a) Adapt-L (b) Adapt-R

Fig. 3. π_k: analytical derivation vs. simulation of Adapt-L and Adapt-R.

adaptive caching, we apply the following simple principles: (1) keep the server at least $\beta = 50\%$ utilized; and (2) use $\alpha = 25\%$ of the edge node buffer to keep static copies of popular content items. As a result, $T_1 = 5$ when $B = 20$ and T_2 dynamically increases with the number of edge nodes, thus maintaining a constant degree of server utilization. Nevertheless, the optimal values of T_1 and T_2 are not necessarily achieved by the aforementioned principles.

Fig. 2 summarizes analytical and simulations results, based on five independent runs. We can observe that the average and server retrieval latencies obtained from the analytical derivations match well with the simulation results. We further validate the analysis of the proposed adaptive caching policies with the steady state probabilities of buffer occupancy by content items, π_k, shown in Fig. 3 for system sizes $N = 20$ and $N = 50$. We conclude that proposed analysis is accurate and can enable a more efficient design space exploration of hybrid system deployment.

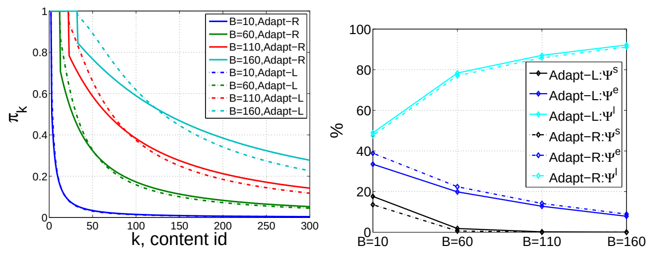
Comparing Fig. 2 (a) and (b), one can easily observe that D^e increases as the number of edge nodes increases when using random selection for all three caching policies, whereas D^e is relatively stable when using Shortest-Queue selection. Consequently, when $N = 70$, the retrieval latency with random selection is around twice as high as with Shortest-Queue selection, around 20 compared to around 20. In fact, the traffic intensity of the entire edge network ρ^e increases as the number of edge nodes increases (from roughly $\rho_e = 0.3$ to $\rho_e = 0.6$) for all caching policies. Shortest-Queue selection is extremely effective in distributing the edge retrieval traffic, especially in a larger distributed system. Moreover, when the number of edge nodes increases, the difference in edge retrieval latency between Adapt-L and Adapt-R increases under random selection and decreases under Shortest-Queue selection. This observation leads us to that: (1) when the retrieval selection is load-oblivious, Adapt-L is more desirable due to its higher percentage of local hit and lower edge retrieval ratios; and (2) Adapt-R is more recommended under load-aware selection, i.e., Shortest-Queue selection, especially in a larger system.

Moreover, the collective caching policy results in a constant local hit ratio, an increasing edge network retrieval ratio and a decreasing server retrieval ratio with an increasing number of edge nodes N. The trend of increasing edge network retrieval ratio and decreasing server retrieval ratio can also be observed in Adapt-L and Adapt-R; 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, Adapt-L consistently generates more server traffic and less edge network traffic than Adapt-R. Since Adapt-L and Adapt-R explicitly take advantage of the server retrieval capacity, the edge network latency with the collective caching policy is higher than with Adapt-L and Adapt-R, especially when the size of the edge network is small. On the other hand, the edge network retrieval capacity becomes significantly larger than the server retrieval capacity in a larger edge network, i.e., $N > 50$, and here collective caching can perform very well by solely relying on edge network retrievals (which are faster than the server retrieval). In summary, the performance impact of source selection is much more prominent in a larger edge network, whereas the performance impact of caching is much more relevant in a smaller edge network.

B. *Collaborative LRU vs. Random Discarding*

For a fair comparison between the two proposed policies Adapt-R and Adapt-L, we only apply Shortest-Queue selection in this subsection.

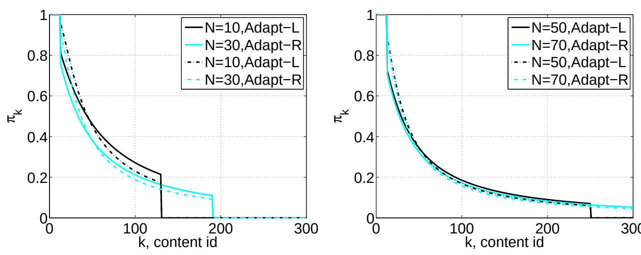
1) Increasing Buffer: In this scenario, we have a fixed number of edge nodes, $N = 20$, and increase the buffer size B from 10 to 70 in increments of 10. We use $\alpha = 20\%$, so that $T_1 = 0.2B$. Fig. 4 (a) depicts π for Adapt-L and Adapt-R. Clearly, Adapt-L has a higher π_k for more popular content compared to Adapt-R. This difference also increases with B. The corresponding local hit ratio Ψ^l, the edge network retrieval



(a) π_k (b) Ψ^l, Ψ^e and Ψ^s

Fig. 4. Comparisons of Adapt-L and Adapt-R when $N = 20$.

ratio Ψ^e and the server retrieval ratio Ψ^s with increasing values of B are summarized in Fig. 4 (b) – the bigger the buffer, the higher the local hit ratio. The edge network retrieval ratio and server retrieval ratio decrease with increasing B. Moreover, the difference in the edge network retrieval ratio and the server retrieval ratio between Adapt-L and Adapt-R decreases with increasing B. When $B = 160$, Adapt-L has only a slightly higher local hit ratio at a trade-off of a slightly lower edge network retrieval ratio, compared to Adapt-R. Overall, the retrieval latency significantly decreases with larger buffer sizes.



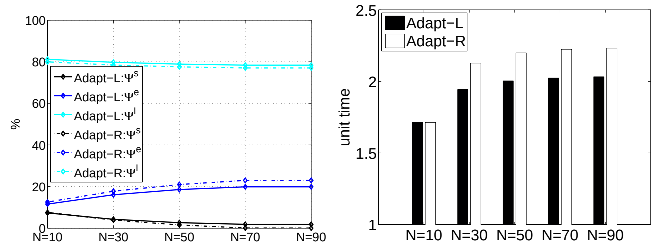
(a) $N = 10, 30$ (b) $N = 50, 70$

Fig. 5. Comparing π_k of Adapt-L and Adapt-R 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. 5 depicts how π_k changes with increasing N under Adapt-L and Adapt-R with $B = 50$. The curves of π_k are shown to converge with increasing N. Meanwhile, the difference between the π_k curves of Adapt-L and Adapt-R decreases as N increases. However, the difference between the edge network and the server retrieval ratio of the two policies increases as shown in Fig. 6 (a). This is because the larger N magnifies the difference of π_k. Adapt-R has visibly higher edge network retrieval traffic at a trade off of negligible server retrieval traffic compared to Adapt-L when $N = 90$. We also summarize the retrieval latency in Fig. 6 (b), and Adapt-L performs slightly better than Adapt-R for all N values compared. Similar to the edge network retrieval ratio, the marginal gain in retrieval latency for Adapt-L over Adapt-R decreases as N increases. From our experiments, we observe that Adapt-R is simple yet very robust, especially when the edge network is large.

C. Online Adaptive Caching

To adapt to changes in popularity among the content items over time, we developed online algorithms for Adapt-L and



(a) Ψ^l, Ψ^s, and Ψ^e (b) latency (D)

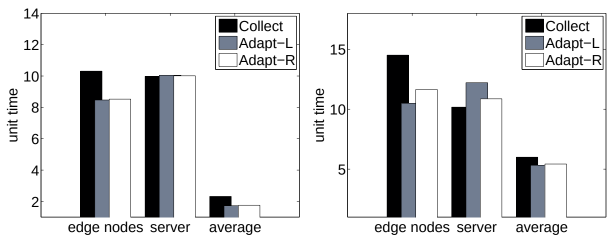
Fig. 6. Comparisons for Adapt-L and Adapt-R when $B = 50$.

Adapt-R. 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 decreasing it 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 increasing it 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 latencies obtained from online Adapt-L and Adapt-R are depicted in Figure 7. The predefined values for gold and bronze thresholds are $\alpha = 20\%$ and $\beta = 0.5$. For comparison purposes, we also study the system performance where the edge nodes employ a collective caching policy where the fixed number of content items in the network is based on an exponent value of 1.2. In both cases, Adapt-L and Adapt-R can achieve lower average retrieval latency by having lower edge network retrieval latency and slightly higher server retrieval latency compared to the collective policy. Another general observation is that collective caching can be very robust with respect to popularity changes when the total system buffer space is sufficiently large. Since the proposed adaptive caching is fully distributed, an online algorithm can be implemented to adopt to popularity changes with negligible overhead. From the results presented here, we believe that Adapt-L and Adapt-R also have good potential to be used in highly dynamic hybrid content distribution systems. Nevertheless, the performance of Adapt-L and Adapt-R can be improved by tuning the control parameters, such as α and β.

VI. Related Studies

Previous related work mainly addresses the lookup latency under various content caching/replication policies and P2P network topologies. Cohen and Shenker [7], [14] 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



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

Fig. 7. Comparison of the average retrieval latencies when request popularity changes.

algorithm. In particular, [7], [14] do not consider LRU and LFU (Least Frequently Used) strategies in their content deleting process. Tewari and Kleinrock [18] 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 [19]. 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 [10] 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. [5] have developed a collaborative caching strategy in a 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 the analytical analysis for numerical evaluation of the data retrieval latency in a hybrid content distribution system under interrelated caching policies and source selection strategies. Our analysis captures important system parameters, i.e., the size of edge network and asymmetric retrieval and caching capacities of the central server and the edge network. Our analysis is composed of the two main components: 1) the retrieval loads derived from the steady-state content distribution, and 2) the retrieval latency derived based on the retrieval loads. Overall, the Shortest-Queue selection strategy can greatly strengthen the scalability of a content distribution system, whereas the retrieval latency increases with the number of edge nodes under random selection. Moreover, our proposed distributed adaptive caching policies, Adapt-L and Adapt-R, which are retrieval capacity aware, can attain a good trade-off between local cache retrieval, edge network retrieval and server retrieval, compared to selfish and collective caching. In principle, we recommend Adapt-R combined with Shortest-Queue selection and Adapt-L combined with random selection for low complexity and robust retrieval latency. For future work, we would like to further explore optimal threshold values of the adaptive caching policies and develop asymptotic results.

References

[1] http://aws.amazon.com/cloudfront/.
[2] http://www-01.ibm.com/software/webservers/cloudburst/.
[3] M. Björkqvist and L. Y. Chen. Content Retrieval Delay Driven by Caching Policy and Source Selection. In *Proceedings of IEEE MASCOTS*, 2010.
[4] G. Bolch, S. Greiner, H. Meer, and K. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*. Wiley, 2006.
[5] S.C. Borst, V. Gupta, and A. Walid. Distributed Caching Algorithms for Content Distribution Networks. In *Proceedings of IEEE INFOCOM*, 2010.
[6] 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.
[7] E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.
[8] 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.
[9] 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.
[10] S. Ioannidis and P. Marbach. On the Design of Hybrid Peer-to-Peer Systems. In *Proceesings of SIGMETRICS*, pages 157–168, 2008.
[11] 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.
[12] 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.
[13] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang. Performance Bounds for Peer-Assisted Live Streaming. *SIGMETRICS Perform. Eval. Rev.*, 36(1):313–324, 2008.
[14] 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.
[15] S. Podlipnig and L. Böszörmenyi. A Survey of Web Cache Replacement Strategies. *ACM Comput. Surv.*, 35(4), 2003.
[16] 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.
[17] 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.
[18] 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.
[19] S. Tewari and L. Kleinrock. Proportional Replication in Peer-to-Peer Networks. In *Proceedings of IEEE INFOCOM*, 2006.
[20] S. Vanderwiel and D. Lilja. Data Prefetch Mechanisms. *ACM Comput. Surv.*, 32(2), 2000.
[21] 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.
[22] D. Wu, Y. Liu, and K. Ross. Queuing Network Models for Multi-Channel P2P Live Streaming Systems. In *Proceedings of IEEE INFOCOM*, 2009.
[23] 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.