Time-shifted TV in content centric networks: the case for cooperative in-network caching

Zhe Li, Gwendal Simon

Time-Shifted TV in Content Centric Networks: the Case for Cooperative In-Network Caching

Zhe Li*, Gwendal Simon*
*Institut Télécom - Télécom Bretagne, France
{firstname.lastname}@telecom-bretagne.eu

***Abstract*—Recent works on Content-Centric-Networking (CCN) enable the exploitation of the caching resources of the new generation of routers (Content Routers or CR). So far, only a basic Least Recently Used (LRU) strategy implemented on every CRs has been proposed. In this paper, we introduce a cooperative caching strategy that has been designed for the treatment of large video streams with on-demand access. This caching strategy addresses the need of Internet Service Provider by halving the cross-domain traffic. We present a first algorithm, which is a core element of this strategy, then we show the changes that have to be brought to the CCN protocol in order to implement this strategy.**

I. Introduction and Background

A. Context: Content Centric Networking

The deployment of Internet routers having caching capabilities (CR for Content or Caching Router [1]) represents an opportunity to revisit the techniques that are currently used to deliver content in the Internet. So far, the flaws of the Internet, in particular the poor performances of communication links traversing several Autonomous Systems (AS) [2], have been overcome by the deployment of large-scale Content Delivery Networks (CDN) such as the Akamai network [3]. The recent works toward *content centric networking* (CCN) [4] introduce new techniques, which allow to route queries and data based on content name. These protocols enables the exploitation of the storage resources of any machine in the network, in particular the CR. However, authors of CCN suggest to use a basic *Least Recently Used* (LRU) policy for the cache management of every CR. The current paper deals with a new caching policy for CR in order to build a *cooperative in-network cache*. This objective requires to take into account:

- *the distributed nature of this cooperative cache*. Contrarily to the centralized management of CDN, the envisioned network of CR is by nature distributed: every CR must decide by itself whether a content that it routes should be cached. Moreover, a claimed objective of CCN is to retain the simplicity and scalability of current Internet protocols. Therefore, CRs can only use local information in order to take their decision.
- *the peering relationships between ASes*. The equilibrium of the whole Internet depends on the selfish actions of every AS. In the CCN perspective, an operator of AS becomes a content provider through the CRs it manages. A rationale behaviour is to cache in priority the most expensive content, *i.e.* when the path to the server storing this content contains expensive transit inter-AS links.
- *the small caching capacity of CRs*. Studies show that video content will represent more than 90% of the whole Internet traffic in a few years [5]. High-definition video streams with bitrate in the order of megabits per seconds requires storage capacity in the order of gigabits. In comparison, the storage capacity of CR is expected to be small (for example, only 36 gigabits in [1]).

This work was supported by the ANR ViPeer project, grant ANR-09-VERSO-014 of the French Agence Nationale de la Recherche.

B. Our Focus: ISP-friendly Time-shifted Streaming

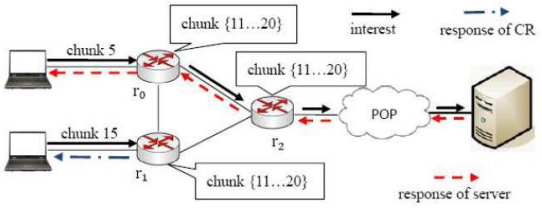
We consider an Internet Service Provider (ISP), which wants to minimize the cross-domain traffic related with *time-shifted TV*. This recent feature proposed by TV broadcasters consists of allowing viewers to watch their favorite broadcast TV programs within an expanded time window. A program broadcasted from a given time t is thus made available at any time from t to $t + \delta$ where δ can be potentially infinite. The popularity of TV services based on time-shifted streaming has dramatically rised [6]: *nPVR* (a Personal Video Recorder located in the network), *catch-up TV* (the broadcaster records a channel for a shifting number of days, and proposes the content on demand), *TV surfing* (using pause, forward or rewind commands), and *start-over* (the ability to jump to the beginning of a live TV program).

Time-shifted TV services are accessible today through connected Digital Video Recorders (DVR). A trend is for these services to be offered by TV broadcasters. However, a large-scale time-shifted streaming service can use neither classic IPTV protocols – contrarily to live streaming systems, time-shifted systems can not directly use group communication techniques like multicast protocols, for the reason that clients require distinct portions of the stream –, nor data-centers – the disk-based servers that are currently used in on-demand video services (VoD) have not been designed for concurrent read and write operations [7]. As a matter of facts, time-shifted channels are restricted to a time delay ranging from one to three hours, despite only 40% of viewers watch their program less than three hours after the live program [8].

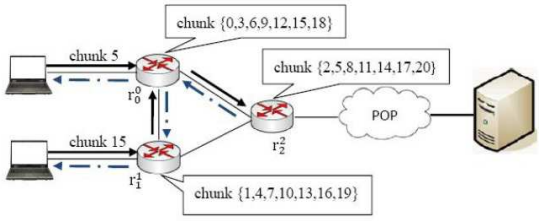
A series of recent works has explored CDN-based and peer-to-peer approaches for time-shifted TV [9]–[14]. However, none of these solutions takes ISPs' behalf into account.

In CDN-based systems, the quality of the distribution is a function of the location of CDN servers, and of the efficiency of the query redirection mechanism toward the appropriate server. An ISP that does not interact with a CDN provider is not able to manage the traffic for the end users located in its AS. In the context of time-shifted TV, this lack of interaction is expensive for the ISP because every request from end user is treated as one unique stream, resulting in larger incoming cross-domain traffic if the CDN is located outside of the AS.

Peer-to-peer and peer-assisted architectures present also some weaknesses. Despite recent efforts toward a better interaction between ISP and peer-to-peer applications [15], the proposals for time-shifted TV ignore the network location of peers. Hence, it may happen that the video is downloaded from one or several distant peers. In our previous works [14], we have addressed the problem of guarantying that all past chunks are correctly kept in a peer-to-peer system.



(a) Basic CCN function



(b) CCN with collaborative cache

Fig. 1. Comparison of basic CCN and CCN with collaborative cache

C. Our Proposal: Cooperative In-Network Caching

We aim at leveraging on a set of deployed CRs to minimize the amount of queries for time-shifted TV that are treated by servers outside the ISP network. Beyond time-shifted TV, our work addresses the problem of storing large-scale streams with on-demand access from end users in CCN.

In this paper, we propose to replace the LRU policy of CCN by a new cooperative policy, with respect to the simplicity of CCN protocols. Our proposal is illustrated in Figure 1. In this example, we assume that a stream is produced by a TV broadcaster. At a given time t, we consider that 21 *chunks* have been produced (from 0 to 20). Each CR has a cache capacity of 10 chunks. According to the LRU policy, the cache of the three CRs are filled by chunks $\{11 \dots 20\}$. At time t, two clients request a time-shifted part of the stream, respectively from chunk 5 and 15. With the CCN protocol, the latter request for chunk 15 is satisfied by the CR r_1, but the request for chunk 5 has to be forwarded to the server. The lack of coordination among CR results in an inefficient caching strategy with redundant data stored on adjacent CRs.

Our proposal is that a CR does not cache all the chunks that it routes, but only a part of them. Every CR is associated with a *label*, which is a positive integer smaller than a fixed integer k. Every CR uses the LRU policy only for chunks whose number modulo k is equal to its label. In our example, we assume that k is equal to 3, and every CR r_i is associated with label i. As can been seen in Figure 1(b), the CR r_0 stores the chunks $\{0, 3, 6, \dots, 18\}$, which correspond to the 10 last chunks routed by r_0 such that their chunk numbers modulo 3 are equal to 0. With this strategy, the request for chunk 5 is not forwarded to the server, but directly satisfied by r_2. In parallel, the request for chunk 15 is no longer treated by r_1, but r_1 forwards the request to r_0, which stores this requested chunk. With this cooperative in-network caching strategy, both requests are treated by machines in the AS of the end users.

D. Our Contributions: Algorithms and CCN Protocol

Due to page limitation, we can not describe all aspects of this proposal in the current paper. In particular, we do not detail how an ISP notifies all CRs that are under its control about the set of streams that requires to be stored for the purpose of a time-shifted service. This notification contains (i) the name of these streams, (ii) the amount of storage space devoted for these streams, and (iii) the number of different labels k. In this paper, we focus on three contributions.

First, we give a theoretical focus on the *initialization stage*, the phase during which each CR determines its label. A trivial implementation consists in a random choice. In previous works, we have shown that significant gains can be obtained from a label assignment that takes into account the network linkage among CRs [16]. However, the optimal assignment has been shown to be NP-complete. We present in the current paper a distributed algorithm that allows each CR to determine its label, this assignment of labels being not worse than $(\frac{3}{2}k - \frac{5}{2})$ of the optimal assignment.

Second, we describe an augmented version of the CCN protocol that implements our cooperative caching strategy. We show in particular that the protocol keeps the simplicity of the originel CCN protocol. We present the refinements that are necessary to implement the cooperative caching.

Finally, we show some simulation results. We have used the synthetic traces generated in [14] to emulate the behavior of users of time-shifted services. We have also implemented the ISP topology measured by *Rocketfuel* [17]. Our results are impressive. When the ISP reserves 1 giga-bytes of cache for five channels, the cooperative caching strategy performs 60% better than the LRU policy.

II. Network Model

We consider a network $\mathcal{N}$ consisting of a set of routers, and a set of bidirectional links between these routers. We note by V the subset of routers that are CR (*i.e.* having caching capacity).

We assume that the ISP is able to compute a static *distance* d_{ij} between two CRs r_i and r_j. This distance reflects the connectivity of two CRs. This function is generic: for example the length of the shortest path joining r_i to r_j in $\mathcal{N}$, the inverse of the capacity of routers on this path, or the average latency measured between these two routers.

The $k-1$ CRs in V that are the *nearest* from the CR r_i are expected to cooperate with r_i. Here, nearest means having the smallest distance. Our goal is to avoid that these CRs store the same chunks. We note by $N(i)$ this subset of CRs in V, and, by extension, $N[i]$ is the set $N(i) \cup \{r_i\}$.

In the following, we assume that non-CR routers are able to transmit the messages from one CR to another without troubles. The CRs do not experience failures.

III. Initialization Stage

Each CR should initially determine its label. Our goal is to ensure that every CR is as close as possible from all the labels that are different that its own label. We note by $L(i)$ the $k-1$ CRs having the $k-1$ other labels and that are collectively the closest from r_i. The sum of distances from a given CR r_i to the CRs in $L(i)$ is called the *rainbow distance* of r_i, and it is noted d_i. Formally, $d_i = \sum_{r_j \in L(i)} d_{ij}$. Determining the optimal assignment of labels, *i.e.* the assignment such that the sum of all rainbow distances is minimal, is NP-hard [16].

To prove the performance ratio of our algorithm, we begin with the definition of lower bound. Given an instance of the problem, it is possible to determine a lower bound solution by setting that every CR with its $k-1$ closest CRs store collectively the k labels, formally $L(i) = N(i)$ for every CR r_i. This obvious optimal assignment is impossible in many cases, but it gives a lower bound. We call *fractional distance*, denoted by $\bar{d}_i$, the sum of the distances between a CR r_i and its $k-1$ nearest neighbors, so $\bar{d}_i = \sum_{r_j \in N[i]} d_{ij}$

A. Distributed Algorithm

There are two rounds. First, each CR exchanges information with its 2-hop neighbors. Then, each CR allocates labels on its neighbors and itself.

For each CR r_i, the first round goes as follows: 1) it collects from its $k-1$ nearest neighbors their $k-1$ nearest neighbors, thus, every CR knows all CRs that are at 2 hops in the $k-1$-nearest neighbor graph. 2) it sends to this 2-hops neighborhood the fractional distance $\bar{d}_i$. 3), then it enters *waiting mode*. 4) it waits until all two-hop neighbors having a fractional distance that is lower than $\bar{d}_i$ emit a *release* message. 5) it executes a *Label Allocation Process* (LAP), then broadcasts a *release* message. 6) when all two-hop neighbors have sent a *release* message, if r_i is both marked as `saved` and not assigned label, then it chooses the farthest label for itself.

The second round, namely LAP, is label allocation. The algorithm tests the condition that *no two CR r_j and $r'_j \in N[i]$ can hold the same label*. If $N[i]$ satisfies the condition, i allocates labels on every CR both in $N[i]$ and holding no label, such that no j and j' hold the same label. Then i marks itself as `optimized`. If $N[i]$ does not satisfy the condition, i marks itself as `saved`. Note that some of the saved CRs are labeled but others not.

B. Correctness and Analysis

Provided that the algorithm runs in a *correct environment*, i.e., there is neither faulty links nor faulty nodes, it returns a solution satisfying the following conditions. First, it runs in finite time. Second, each CR eventually holds a label. Third, there is no missing label in the system.

Theorem 1 *The algorithm gives a valid solution in a correct environment.*

Proof. The last condition is easily satisfied when the first CR (the CR possess the local minimum rainbow cost) executes LAP. To show that the first and second conditions are also tenable, we just need to prove that i will receive all *release* messages from its two-hop neighbors in a finite time. If the algorithm does not terminate, it must be some nodes i and j such that i never receives a *release* message from j, so i never executes LAP, and broadcasts the *release* message. Yet, the fractional distance being a unique real number, there is always a CR with a smallest distance, which can enter LAP and broadcast the release message. This also leads to the fact that each CR will execute LAP. Together with the fact that the distance of each CR is broadcasted only once, we conclude that no CR will be in waiting mode for infinite time. Since the number of nodes is finite, the algorithm terminates in finite time, thereafter each CR holds a label. □

Each CR executes LAP, and, as the distance function d gives a total order on nodes, no two nodes within two hops are local minimal at the same time, so no two nodes within two hops execute LAP at the same time.

Theorem 2 *For any $k \geq 3$, the distributed algorithm gives a solution no more than $\frac{3}{2}k - \frac{5}{2}$ times the lower bound.*

Proof. For an optimized CR $r_{i'}$, we know that $d_{i'} = \bar{d}_{i'}$. For a saved CR r_i, there are two cases: 1) the label on r_i has been assigned by another CR, and this label coincides with the label held by one of its $k-1$ nearest neighbors, 2) two nodes in $N(i)$ hold the same label.

In the first case, the label on r_i has been assigned by an optimized CR $r_{i'}$. It means that $r_i \in N(i')$, and that $\bar{d}_{i'} < \bar{d}_i$ (because $r_{i'}$ executed LAP before r_i). Assume that the label of r_i is 1, and the neighbor of $r_{i'}$ hosting label l is noted $r_{j'l}$. Then the rainbow cost of r_i can be calculated as follows. Since $r_j \in L(i)$ is the nearest neighbor of r_i, we have $\sum_{r_j \in L(i)} d_{ij} \leq \sum_{r_{j'} \in N[i']} d_{ij'}$.

$$\sum_{r_j \in L(i)} d_{ij} \leq \sum_{r_{j'} \in N[i']} d_{ij'} = \sum_{l=2}^{k} d_{ij'l} \leq \sum_{l=2}^{k} (d_{ii'} + d_{i'j'l}) =$$

$$(k-1)d_{ii'} + \sum_{l=2}^{k} d_{i'j'l} = (k-2)d_{ii'} + \sum_{l=1}^{k} d_{i'j'l} \leq (k-1)\bar{d}_i$$

In the second case, there must be an optimized CR $r_{i'}$ within two hops from r_i, such that $\bar{d_{i'}} < \bar{d_i}$. Assume that r_{j_1} and r_{j_2} are the two nodes that prevent r_i from entering the optimized state, and $d_{ij_1} < d_{ij_2}$. Without loss of generality, we can assume label 1 at j_1. If r_i chooses label h in the second phase, then $h \neq 1$, as r_{j_1} is among the nearest neighbor of r_i. According to the algorithm, we have $r_{j_1} \in N(i) \cap N(i')$. After labels allocation is finished, r_i and r_{j_1} hold different labels. Thus $rj_1 \in L(i)$. Then the rainbow distance of r_i can be calculated as follows:

$$\sum_{r_j \in L(i)} d_{ij} \leq \sum_{r_{j'} \in N[i']} d_{ij'} = \sum_{l=1}^{k-1} d_{ij'^l} \leq$$

$$d_{ij_1} + \sum_{l=2}^{k-1} (d_{ij_1} + d_{j_1 i'} + d_{i'j'^l}) =$$

$$\sum_{l=1}^{k-1} d_{i'j'^l} + (k-3) d_{i'j_1} + (k-1) d_{ij_1} \leq$$

$$\sum_{l=1}^{k-1} d_{i'j'^l} + (k-3) d_{i'j_1} + \frac{k-1}{2}(d_{ij_1} + d_{ij_2}) \leq$$

$$(k-2) \sum_{l=1}^{k-1} d_{i'j'^l} + \frac{k-1}{2} \sum_{l=1}^{k-1} d_{ij^l} \leq (\frac{3}{2}k - \frac{5}{2})\bar{d_i}$$

As $k - 1 \leq \frac{3}{2}k - \frac{5}{2}$ for any k greater than 3, our algorithm gives a solution no more than $\frac{3}{2}k - \frac{5}{2}$ times the lower bound. □

IV. Augmented CCR Protocol

We start by a quick summary of the main principles of CCN. Please refer to [4] for more details. Then, we present the changes that we propose in order to implement our cooperative caching strategy.

A. CCN in a Nutshell

In CCN, every content is identified by a hierarchical name like URL and divided into multiple chunks. Each chunk is indicated by the content name plus a sequence number. When a content is published by a provider, the CR connected with that provider floods an advertisement of the content to adjacent CRs. A *Forwarding Information Base* (FIB) is established to redirect any incoming *interest* (*a.k.a.* request) toward content provider. When an interest is forwarded according to the FIB, an entry into the *Pending Interest Table* (PIT) is created to trace the requesting interface, so that the content can be sent back along the reverse path of interest. The content is then cached by the CRs on its forwarding path. If the content is requested again, the replica in the *Content Store* (CS), or cache, is directly delivered by the CR.

B. New Tables in CCN

In order to implement our cooperative caching strategy, we require two new tables. First, every CR r_i maintains the information of its $k-1$ closest CRs in $L(i)$ in a new table, namely *Collaborative Router Table* (CRT). There are three fields in CRT of a CR: the label, the identifier of the collaborative router and the interface. Thus, every CR knows where to redirect an interest or forward a chunk. The second table added on the basic CR is the *Collaborative Content Store* (CCS). In CCS, a CR keeps the names and the sequence numbers of all the chunks that may be found in its collaborative cache. When an interest arrives, the preference of the four prefix matches is CS match to CCS match to PIT match to FIB match.

C. Distribute Chunks in the Cooperative Cache

When a chunk c is sent back to consume an interest, a CR r_i with label l_i, which receives c, should take a decision (whether to cache it or not) based on l_i, on the identifier c of this chunk, and the match result. We describe the action as follows:

- this chunk is handled by r_i, that is $c \mod k = l_i$. The CR r_i adds c into its cache, and removes the least recently used chunk. Then r_i calculates a PIT match. If a PIT match is found, it forwards the data to the interfaces indicated by the PIT, otherwise, the process is finished.
- this chunk is not handled by r_i, that is $c \mod k \neq l_i$. The CR r_i first finds in its CRT the router r_j having the label l_j that matches with the chunk c. Then r_i sends the chunk c to r_j. Moreover, if r_i finds a match in its PIT for this chunk, it also forwards c to the requesters. Finally, r_i adds c in the CCS Table, so that later interests requiring the same chunk will be forwarded to r_j, but no longer according to the FIB.

In this scheme, each data packet should carry a random *nonce* to prevent broadcast storm. When a duplicated packet with the same nonce is received, it should be immediately discarded.

D. CCS Consistency

At every time, the CS of a given CR r_i should be consistent with all the CCS tables of all CRs that consider r_i among its closest CR. In particular, when a entry of the CS r_i is discarded by the caching policy, the corresponding entry in the CCS of a CR r_j with $r_i \in L(j)$ should also be deleted, otherwise interests for the eliminated content may be lost in the forwarding process. For example, if r_j receives an interest requiring chunk c, it finds the CCS match point to r_i. Assume that chunk c in r_i has been discarded. The CR r_i forwards the interest following the FIB entry. If r_j is an intermediate CR between r_i and the corresponding server, the interest will be regarded as a duplicated one, and discarded by r_j. Therefore, the interest for chunk c is lost. We should remind that the lost interest can be recognized as a duplicated one because every interest is given a random nonce when it is generated.

To both maintain consistency and avoid increasing control messages, we use piggyback interest (*p-interest*) to carry the

control information. A CR r_i with label l_i acts as follows when an interest for chunk c is received:

- the requested chunk c is handled by r_i, that is $c \mod k = l_i$. The CR r_i first calculates the CS match. If a CS match is found, it sends back the data directly. Otherwise, if a PIT entry is found, it adds the requiring face into the pending list. If neither CS match nor PIT match is found, r_i changes the interest into a p-interest, it generates a new nonce for the p-interest, and forwards this p-interest according to FIB entry.
- the requested chunk c is not handled by r_i, that is $c \mod k \neq l_i$, and the interest is a p-interest. The CR r_i needs to determine whether the CR r_j indicated in the p-interest is in the CRT of r_i. When r_j is not the relative collaborative router, r_i executes normal process. Otherwise, r_i should eliminate the CCS for the chunk required in the interest, then adds the requiring face in its PIT. Finally, r_i forwards the interest according to the FIB, even if PIT already existed. The final step ensures that the interest arrives at a provider.
- the requested chunk c is not handled by r_i, that is $c \mod k \neq l_i$, and the interest is not a p-interest. The CR r_i just executes the normal CCN process (collaborative CS match is preferred than PIT match, and PIT match is preferred than FIB match).

V. Simulations

The goal of these simulations is to evaluate the benefits one can expect from the cooperative in-network caching strategy. We develop our simulator over OMNET++, a simulation framework for communication networks.

A. Simulation Setup

To build a typical ISP network, we use the real backbone topology measured by *Rocketfuel* [17]. We choose 87 routers, 5 point of presences (POPs) and 161 bidirectional links with latencies from the AS of European Backbone (Ebone). Every POP is connected with one server, which stores all the produced chunks. Chunks are pushed into servers from 6 TV providers with different popularities. We deploy 200 clients uniformly on the access routers locating at the edge of the topology. We reserve 1 giga-Bytes in each CR to cache time-shifted TV streaming. The basic data unit of the TV streaming is a chunk, which contains the streaming for 1 minute playback. One new chunk is produced every simulation minute by each TV provider. We assume the streaming playback rate is 1 megabits per second, so that the size of one chunk is 7.5 mega-Bytes. Therefore the cache of a CR can store 130 chunks, approximately two hours of video.

We use the same synthetic model as [14] for modeling the behavior of users of time-shifted services. This model is based on two measurement studies conducted in 2008 and 2009 [8, 18]. This model includes that a TV stream is divided into *programs*, associated with a *genre*. The popularity of programs decreases with time. Moreover, the number of clients varies following a given distribution. In our case, according to different hours in a day, the number of activated clients ranges from 20 to 180. Every client get assigned a role: half of the clients are *surfers* (watch a same program during 1 or 2 chunks before to switch to another program), 40% of them are *viewers* (switch after a duration uniformly chosen between 2 and 60 minutes), and only 10% are *leavers* (stay on a program during a time comprised between 60 and 1000 minutes).

We run our simulation for $9{,}000$ minutes, *i.e.* about one week. Since six TV streams are in the system, $54{,}000$ chunks are produced during the simulation. We measure in particular:

- the *caching diversity* of the policy by counting the number of distinct chunks that are stored in the network. The more distinct chunks are stored in the system, the better is the cooperative caching system. With 87 CRs having each a maximum caching capacity of 130 chunks, the maximal caching diversity is $11{,}310$ chunks.
- the *ISP-friendliness* of the policy by measuring the number of requests that are treated by servers outside the network. The lesser is the number of requests, the friendlier is the caching policy.

B. Results Analysis

We first investigate the impact of k on the performance of the system. We change k from 1 to 6, where $k = 1$ is exactly the basic LRU policy. In Figure 2, we show the caching diversity at the end of the simulation. For any $k \geq 3$, the system using collaborative cache can keep at least 700 distinct chunks more than the system using basic LRU. The number of distinct chunks keeps increasing although it grows slower after $k = 4$. When $k = 6$, the caching diversity reaches $4{,}500$ chunks, that is, the collaborative cache with $k = 6$ outperforms the basic LRU by almost 60%. As can be expected, the cooperative caching policy increases the caching diversity by avoiding redundant chunk caching.

We demonstrate the efficiency of our proposal in Figure 3, where we compare the ISP-friendliness of the basic LRU policy implemented in CCN to our cooperative caching strategy with $k = 6$. In average, every server should upload 20.56 chunk by minute with the basic LRU system, and only 8.92 in our proposal. In other words, the ISP can expect a reduction of around 60% of the cross-domain traffic.

Moreover, we observe that the workload in basic LRU system is not well balanced , with servers 3 and 5 exhibiting two times more traffic than server 4. The workload depends on the network topology: less CRs locate around the POP which is connected with server 4, so less requests for the old chunks, which no longer exist in the cache, arrive at server 4. The reverse situation, which happens on server 3 and 5 causes the unbalance of the workload between servers. However, in collaborative cache system, every server sustains approximately the same number of requests. Since most of the chunks for shifted streaming are kept in the collaborative cache, a majority of the requests redirected to servers are the requests for live streaming.

To further study the popularity of chunks stored in the system, we investigate the time interval between last two

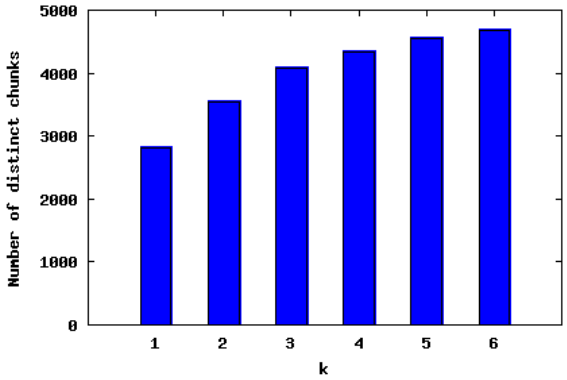


Fig. 2. *Caching diversity*: the number of distinct chunks stored in the set of CRs when the number of labels k varies

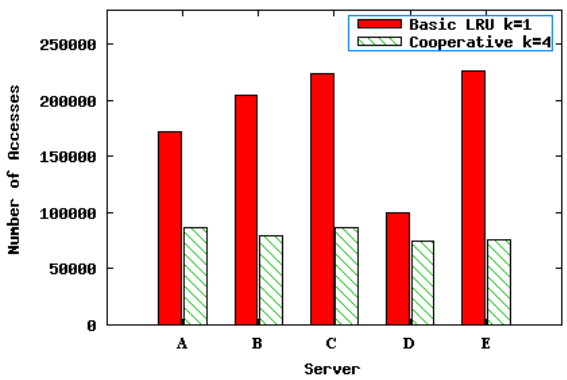


Fig. 3. *ISP-friendliness*: the number of times each server located is accessed. The smaller is the bar, the more ISP-friendly is the caching strategy.

requests for each cached chunk. This indicates the volatility of content in the cache: the smaller are the time intervals, the more frequent are the read-write operations on the cache. In average, the basic LRU policy has a more intensive usage of the cache. We show the Cumulative Distribution Function of the number of chunks with regard to their time interval in Figure 4. A point at $(40, 0.85)$ means that 85% of the chunks have been accessed at most 40 minutes ago. As can be expected, our cooperative caching policy produces a less intensive caching strategy. On one hand, it means that operations on the disks are less frequent. On the other hand, the content would have higher probability to be removed if ISP were unable to reserve a certain storage space in the cache because unpopular chunks should be replaced by other data.

Finally, in Table I, we compare the average response time of each request, that is, the round trip time between the sending of a request and the receiving of the corresponding chunk. The response time in collaborative cache is just 40ms more than that in the basic LRU. Thus, our collaborative cache does not cause any significant degradation of the Quality of Experience.

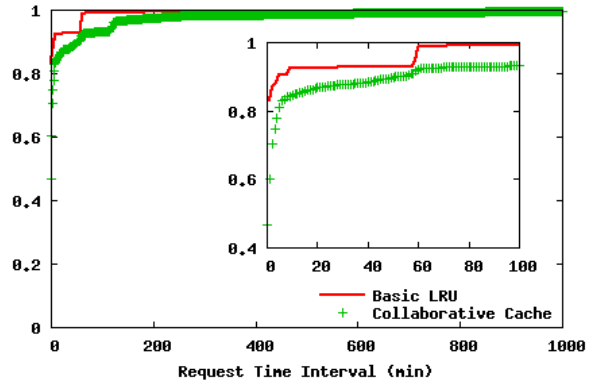


Fig. 4. Cumulative Distribution Function. The y-axis is the ratio of chunks, the x-axis is the time elapsed between two consecutive access on a CR.

Cache scheme	*Average response (ms)*
Basic LRU	243.38
Cooperative cache	288.25

TABLE I
COMPARISON OF RESPONSE TIME AND REQUESTED TIME INTERVAL

REFERENCES

[1] U. Lee, I. Rimac, and V. Hilt, "Greening the internet with content-centric networking," in *International Conference on Energy-Efficient Computing and Networking*, 2010.

[2] T. Leighton, "Improving performance on the internet," *Communications of the ACM*, vol. 52, no. 2, pp. 44–51, 2009.

[3] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: a platform for high-performance internet applications," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. of the Int'l Conf on emerging networking expe. and tech. (CoNEXT)*, 2009.

[5] *The Cisco Visual Networking (VNI) Forecast 2009-2014*, Cisco, June 2010.

[6] *Three Screen Report Q1*, Nielsen Company, June 2010.

[7] *Evolving Requirements for On Demand Networks*, Motorola, Inc., March 2009.

[8] Nielsen, "How DVRs Are Changing the Television Landscape," Nielsen Company, Tech. Rep., April 2009.

[9] J. Zhuo, J. Li, G. Wu, and S. Xu, "Efficient cache placement scheme for clustered time-shifted TV servers," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1947–1955, November 2008.

[10] T.Wauters, W. de Meerssche, F. Turch, B. Dhoedt, P.Demeester, T. Caenegem, and E.Six, "Co-operative proxy caching algorithms for time-shifted iptv services," in *IEEE Computer Society Washington*, 2006.

[11] J.-C. Zhuo, J. Li, G. Wu, and L.-Y. Zhu, "A novel data replication and placement scheme for time-shifted tv cluster," in *International Conference on Computer Science and Software Engineering*, 2008.

[12] F. V. Hecht, T. Bocek, C. Morariu, D. Hausheer, and B. Stiller, "LiveShift: Peer-to-Peer Live Streaming with Distributed Time-Shifting," in *Proc. of 8th Int. P2P Conf.*, 2008, pp. 187–188.

[13] D. Gallo, C. Miers, V. Coroama, T. Carvalho, V. Souza, and P. Karlsson, "A Multimedia Delivery Architecture for IPTV with P2P-Based Time-Shift Support," in *Proc. of 6th IEEE CCNC*, 2009, pp. 1–2.

[14] Y. Liu and G. Simon, "Distributed Delivery System for Time-Shifted Streaming System," in *35th IEEE Conf. on Local Computer Networks (LCN)*, 2010.

[15] H. Xie, A. Krishnamurthy, A. Silberschatz, and Y. Yang, "P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers," *P4PWG Whitepaper, May*, 2008.

[16] Y. Chen, J. Leblet, and G. Simon, "On reducing the cross-domain traffic of box-powered cdn," in *Proc. of IEEE ICCCN*, 2009.

[17] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *SIGCOMM*, 2002.

[18] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatrianin, "Watching television over an ip network," in *Proc. of Usenix/ACM SIGCOMM Internet Measurement Conference (IMC)*, 2008.