Content Delivery Mechanism for ISP

Qiao Li, Hui He, Bin-Xing Fang, Hong-Li Zhang and Wei-Zhe Zhang

Department of Computer Science and Technology, Harbin Institute of Technology, Heilongjiang, P.R.China, 150001
liqiao84@hit.edu.cn; {hehui, bxfang, zhl, zwz}@pact518.hit.edu.cn

Abstract

Most CDNs pay more attention to CSP (Content Service Provider) to gain more profit from faster access rate and lower server load. However they ignore that the CDNs pay the ISPs (Internet Service Provider) for using the underlying networks based on the traffic volume generated by CDN servers. ISPs expect to save their operational cost from inter-ISP traffic and improve the QoS (Quality of Service) as well. In this paper, we propose novel decentralized content delivery architecture, ISP Manage Content Delivery Network (IMCDN). For more effectively improving the performance of IMCDN, we propose a cooperative dynamic caching strategy based on the dleft Record Counters Bloom Filter (dl-RCBF). We use the dl-RCBF to improve the remote hit rate and adopt optimistic synchronization to restrict the broadcast traffic. The experiments show that our dynamic caching strategy outperforms state-of-the-art models in hit rate for multiple cache sizes and inter-ISP traffic reduction.

Keywords: *distributed systems, content delivery, cooperative cache, counter filter*

1. Introduction

As the Internet technology upgrades, the performance of server devices, network forwarding devices and terminal devices have been greatly enhanced, *e.g.*, processing capacity, throughput, forwarding rate and bandwidth. High-bandwidth real-time online applications result in Internet traffic doubling year by year. More efficient content delivery strategy over the Web has become an important factor of improving Web performance [1, 14]. To provide content delivery service efficiently within the current Internet architecture, there have been two representative solutions: peer-to-peer (P2P) systems and content delivery networks (CDNs). CDNs have been a successful business model, which provide stable and efficient content delivery services leveraging distributed data centers (often worldwide). CDNs improve network performance and offer fast and reliable applications and services by delivering content to cache servers located close to users. The earlier CDN provider distributes the copies of contents to its servers deployed worldwide upon the requirements of content providers. If a user requests for a content file, the request will be redirected to the CDN server in the closest proximity in order to lower the access delay.

The surrogate (we use "cache server" in this paper) location indicates the aim of CDNs: 1) the surrogates deployed in the edge of network (close to user or origin server) can reduce the load on origin servers and replicas transmission cost, *e.g.*, Akamai[2], PPlive[3]; 2) the surrogates deployed in the middle of network are aimed at saving the inter-carrier payment, *e.g.*, iCODE[4]. For example, With 25,000 servers in 650 cities and 70 countries, Akamai is within one network hop of 90% of Internet users and can support the capacity sufficient to accommodate the high-definition(HD) Web and future IP needs[5]. Different from Akamai, Limelight deploys more than 20 data centers over the fiber-optic backbones to obtain 2.5Tbps aggregate bandwidth, which support to speed up the HD video transmission [6].

CSPs (Content Service Provider) deploy their CDN in order to increase the access rate of end users that reside near the edge of network. Many commercial CDN are CSP-CDN, which improve performance of accessing the content from user perspective because they deliver the content from a nearby location.

Though the surrogates placed in the edge of network lower the user access delay, CDNs add network complexity even impact the ISP cost and traffic engineering. Actually ISPs have the economic incentives to resolve the adverse impacts produced by CDNs. The motivation of ISP for building its own CDN is to reduce the inter-traffic among ISPs and save the operational cost. Compared with CSP-CDN, ISP has inherent advantages in surrogate servers and replicas placement due to the full topology knowledge. The motivation for this paper lies in saving the ISP operational costs by reducing the inter-ISP traffic. We devise a novel non-centric CDN architecture managed by ISP, called ISP Manage CDN (IMCDN), to save the inter-ISP traffic efficiently and avoid single point of failure. In IMCDN, the surrogates are deployed close to the boundary routers to minimize the inter-domain traffic.

In this paper, we firstly analyze the access delay of the users residing the ISP in order to find which content are more valuable for ISPs. Then we propose a cooperative cache strategy based on dleft Record Counter Bloom Filter (*dl-RCBF*), which deals with two important issues in the cooperative cache management: locating content and synchronization among cache servers. We propose *dlRCBF-search* algorithm, which achieves good cache hit rate by decreasing the false positive rate using feedback policy. For the second issue, we design *dlRCBF-syn* algorithm to reduce the intra-ISP bandwidth cost through dynamic replication and optimistic synchronization mechanism. Finally, the simulation experiments show that the IMCDN provides good performance for both the lowering access delay and inter-ISP traffic reduction for an ISP network.

The paper is organized as follows: In Section 2, we review the related work. Then, Section 3 describes the IMCDN architecture. In Section 4, we describe the dl-RCBF data structure and the cooperative cache mechanism of IMCDN. We report our experiments with it in Section 5. Finally, Section 6 concludes the paper.

2. Related Work

CDNs play a key role in the Internet infrastructure due to their high end-user performance and cost savings [8]. A typical CDN is an overlay network across Internet, which consists of a set of surrogate servers distributed around the world, routers and network elements. Choosing the best location for each surrogate is important for each CDN infrastructure because the location of surrogates is an important factor during the content delivery process. ISP has the inherent advantage in the topology information and routing. Hence, unlike in traditional CDN or P2P-based delivery systems, ISPs can optimally design server location, deploy content, and select the best server for each request [9]. There has been also significant effort on ISP merged CDN, a new technical trend in content delivery mechanism. Kideok Cho *et al.*, proposed iCODE system using routers or network entities as storage modules for content caching and content delivery [4]. They redirect the user requests by a center server named iTracker, which would result in single point failure. Noriaki Kamiyama *et al.*, devised I-CDN to serve users rich content economically and efficiently through optimizing the content deployment, the content delivery process, and the server allocation [9, 20]. However they do not consider the cache mechanism for multiple boundaries in the ISP network, which we focus on in this paper.

There have been many studies for data placement and location in cooperative cache mechanism. Dominguez-Sal *et al.*, proposed Broadcast Petition Recently (BPR) in [10], which uses a variable time window to control the number of duplicates in the network but does not use global system information. Some placement solutions are based on hash functions [11]. But one drawback of Distributed Hash Tables (DHTs) is that it needs to contact several nodes sequentially during the location procedure, which introduces latency in the data inquiry. David Dominguez-Sal *et al.*, propose a distributed cooperative caching strategy based on the Evolutive Summary Counters (ESC), a new data structure that stores an approximated record of the times of accessing [12]. However, when local miss occurs, they forward the request to other nodes in a certain probability that would increase the miss ratio. Our proposal overcomes this problem because we use optimistic synchronization to obtain higher remote hit ratio.

Besides, many cache mechanisms have been proposed to increase the performance of CDN including replicas placement [15] and cooperative push-based or pull-based scheme. However, the purpose of ISP-CDN is to design a content delivery mechanism loosely coupled with CSP. Considering the particularly of ISP-CDN, our approach focuses on decreasing the user access delay and inter-domain traffic reduction for ISP, which the former studies do not take into account together.

3. The Architecture of IMCDN

In CDN, the user requests are redirected to the "nearby"(means lower latency) surrogates in order to decrease the origin server load and improve user experience. The unnecessary long distance transmission means higher bandwidth cost and more inter-domain traffic, which will result in a negative impact on ISP [18]. Moreover, the competition between the ISP seriously affects the QoS (Quality of Service). Meanwhile, CDNs may choose the server that has a shorter end-to-end delay to the user for better service performance, but ISPs may prefer choosing the servers in the domain to save on the intercarrier payment. Based on the above analysis, we design IMCDN system to alleviate the conflict between CDNs and ISPs, which will be detailed description in this section.

The IMCDN architecture is illustrated in Figure 1. As shown in Figure 1, when users want to get the object from the Internet, the request path can be divided into two parts: 1) the in-domain path of the request, which is noted as (1); 2) the out-domain path of the request, which is noted as (2). Similarly, the reply path can be split into two parts: 1) the out-domain path, which is noted as (3); 2) the in-domain path, which is noted as (4) (In this paper, we only focus on the objects outside the ISP). In order to reduce the ISP operational cost generated by inter-domain traffic and improve the user experience efficiently and flexibly, IMCDN deploys the cache servers (noted as *CacheS*) connected with the inter-domain routers directly, *e.g.*, *CacheS1* and *R1*. When the user request arrives on *R1*, *R1* will forward them to cacheS1. Then the CacheS1 analyzes the user requests (*e.g.*, http, p2p) and find the required content in which cache server. Our cache servers compose a distributed cache system and the cache objects in the *CacheS_i* compose a DHT, see the cycle among *CacheS* in Figure 1. There will be three situations when the CacheS1 receives a request:

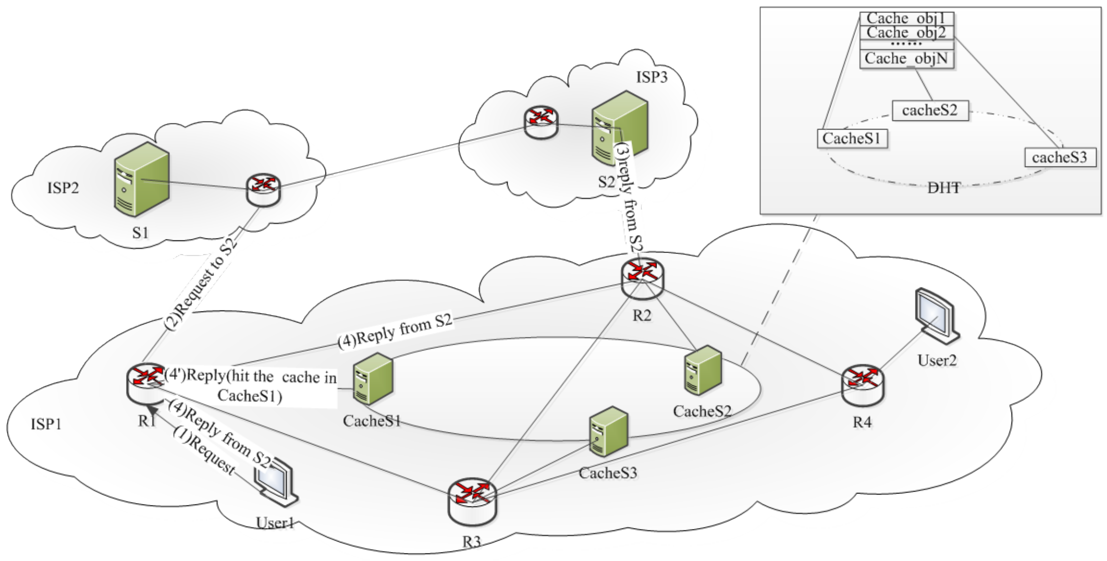


Figure 1. The Architecture of IMCDN

1) the required content is not in IMCDN system, which means each cache servers do not have it. The *CacheS1* will send the request to the origin server outside the domain;

2) the required content is in *CacheS1*. The CacheS1 will send it to the user, see the path (4') in Fig. 1.

3) the required content is in other cache servers, *e.g.*, *CacheS2*. The *CacheS1* will forward the request to *CacheS2*.

Because the *CacheS* communicate with each other in the p2p manner, the broadcasting traffic will increase the link load. For minimize the broadcasting traffic, we propose a novel data structure as data digest, which we will describe in Section 4.

4. Dynamic Cache Mechanism in IMCDN

In general, ISPs prefer the traffic to stay within their own network, *i.e.*, so-called “on-net” traffic. Thereafter ISPs regulate their “off-net” traffic, which leaves their networks, through inter-domain routing policies. ISP can reduce the inter-domain traffic by sharing the cache index. The cache synchronization mechanism is one of the important factors that impact the ISP network performance. There has been also significant effort on the synchronization of documents in a distributed environment. Fan *et al.*, use a bloom filter-based data structure as content summarize broadcasted to other nodes periodically [16]. However their data structure only represents the existence of the content, the trends of the content could not be depicted. Dominguez-Sal *et al.*, implement a novel cooperative caching strategy based on evolution summarize counter [12]. They use ESC-summaries to obtain approximate statistics of the document entries accessed by each user. The access delay could be increased by forwarding the request to remote server at a certain probability. Considering the different contents in each cache server and the locality of user requests, we improve quality of service by following three aspects:

(1) We synch to the content in each cache server by data summary. Different from [12, 16], we avoid the announcement among the cache servers and reduce the cost of querying by using dl-RCBF as content summary;

(2) Using non-center distributed architecture to prevent the single point of failure and performance bottleneck, different from [4];

(3) In consideration of traffic generated by synchronization, we use a dynamic content replication strategy.

4.1. dl-RCBF Data Summary

We briefly review *dleft* Count Bloom Filter; for further details, see [17]. Suppose that we have a set $U = \{key_0, key_1, ..., key_n\}$ and split the hashtable into *d* subtables. Each subtable has *B* buckets with *m* cells. Simply, we use a single hash function: $U \xrightarrow{hash} [B]^d \times remainder$, giving *d* choices for each element, and store the *remainder* in the least loaded bucket of the *d* choices. The *remainder* includes finger and counter. A false positive would occur if and only if for a query $y \notin U$, there existed $x \in U$ with $hash(x) = hash(y)$.

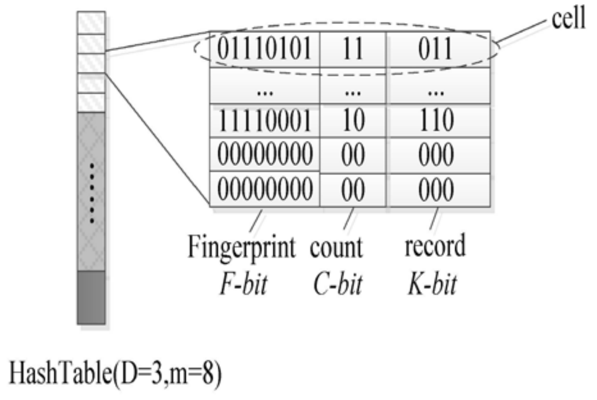


Figure 2. An Example of *dl-RCBF* Structure

Now we introduce our *dl-RCBF* structure. Suppose that we have a set $S = \{x_0, x_1, ..., x_n\}, |S| = n$ that is changing by item insertion and deletion over time and the number of cache servers is *K*. The *dl-RCBF* is a hashtable with *n* buckets noted as *BK*, which is divided into *d* subtables of *n/d* buckets (For convenience we assume *n/d* is an integer). In Figure 3, it schematically depicts the construction of *dl-RCBF*. Each *BK* contains *m* cells of fingerprint (*F-bit*), counter(*C-bit*) and record (*K-bit*, *K* is the number of cache servers) identifying the location of the element. For a situation when a false positive occurred in local cache, the count would be updated. Obviously, the false positive of *dl-RCBF* is equal to *dl-CBF*. As Figure 2 shown, the record 011 in cell indicates the element which the fingerprint of is 01110101 is stored in cache server 1 and 2. The *dl-RCBF* gives a memory size bounded by *n+n*m(F+C+K)* bits.

To insert an element *x*, as shown in Algorithm 1, if *x* does not exist in *dl-RCBF*, we store *x* in the least loaded subtable of d choices as *dl-CBF* and the local node bit of record is set as 1. If the value of *hash(x)* exsited in the *dl-RCBF*, the corresponding cell counter would be increased.

Algorithm 1 Pseudo-code for insertion , search and deleting in dl-RCBF

```
Function RCBF-INSERT(key,localnodeid)
1: $key \xrightarrow{hash} value[d] \times fingerprint$
2: for each i in [0, m-1] do
3:   if BK[i]: fingerprint == 0 then
4:       BK[i].fingerprint = fingerprint
5:       BK[i].record[localnodeid] = 1 // record[] is a bit array
6:       BK[i].count = 1
7:       return 1
8:   else
9:       BK[i].record[localnodeid] = 1
10:      BK[i].count + +
11:      BK← the least load bucket among the subtable[value[d]]
12:      return 0
13:  end if
14: end for
Function RCBF-SEARCH(key)
1: $key \xrightarrow{hash} value[d] \times fingerprint$
2: for each j in [0, d-1] do
3:     BK← subtable[value[j]]
4:     for each i in [0, m-1] do
5:         if BK[i].fingerprint == fingerprint then
6:             return 1
7:         end if
8:     end for
9:     return 0
10: end for
Function RCBF-DELETE(key,localnodeid)
1: $key \xrightarrow{hash} value[d] \times fingerprint$
2: for each j in [0, d-1] do
3:     BK← subtable[value[j]]
4:     for each i in [0, m-1] do
5:         if BK[i].fingerprint == fingerprint then
6:             if BK [i].count == 0 then
7:                 BK[i].record[localnodeid] = 0
8:             else
9:                 BK[i].count++
10:            end if
11:        end if
12:    end for
13: end for
```

When the inserted item is found through *RCBF-SEARCH(key)*, the cell counter is just decremented. The local node bit of corresponding cell record is set as 0 when the cell counter decreases to 0.

Broadcasting the cache summary to each server periodically could result in burst traffic and high server load. In order to decrease the cost of network and servers, the cache server sends the content index to other nodes when the number of contents changed to the threshold depending on the size of cache. As algorithm 2 shows, when the synchronizer trigger is active, the local cache server sends the syn-hashtable to the other nodes. We put the local content record into syn-hashtable, as *RCBF-SYNCHRO-SEND()*.

Sometimes the users in the ISP want to request the same content, *e.g.*, online video streams of popular political events and live sports events. In this situation, one of the cache servers stored the resources firstly; the others would forward the requests and lead to high traffic in

ISP. To reduce the traffic pressure of ISP and the concurrency load of servers, we deliver the popular resources to those nodes requesting for them frequently.

As algorithm 2 shows, even in the worst case, the time complexity of *RCBF-SYNCHRO-RECV()* is $O(dm^2)$,which d is the length of a subtable and m is the number of cells in a bucket. When cache server merges the subtables receiving from other nodes into local subtable, the buckets may overflow. In order to deal with this trouble, we delete the cell which has the minimum counter value.

Algorithm 2 Pseudo-code for updating and synchrony in dl-RCBF

```
Function RCBF-UPDATE(key, nodeid)
1: $key \xrightarrow{hash} value[d] \times fingerprint$
2: for each j in [0, d-1] do
3:     BK← subtable[value[j]]
4:     for each i in [0;m-1] do
5:         if BK[i].fingerprint == fingerprint then
6:             BK[i].record [nodeid] = 1
7:         end if
8:     end for
9: end for
Function RCBF-SYNCHRO-RECV(remotesubtable , remotenodeid)
1: for each m in [0,length of remotesubtable] do
2:     tempbucket←remotesubtable [m]
3:     BK←subtable[m]
4:     for each i in [0, m-1] do
5:         tempfingerprint = tempbucket[i].fingerprint
6:         for each j in [0, m-1] do
7:             if tempfingerprint == BK[j].fingerprint then    //local node have the key or false positive
8:                 BK[j].record[remotenodeid] = 1
9:             end if
10:            if BK[j].fingerprint == 0 then //cell not full
11:                BK[j].fingerprint = tempfingerprint
12:                BK[j].record[remotenodeid] = 1
13:            end if
14:            if j == m - 1 then    //cell is full
15:                delete the min counter cell
16:                n←the position of min counter cell
17:                BK[n].fingerprint = tempfingerprint
18:                BK[n].record[remotenodeid] = 1
19:            end if
20:        end for
21:    end for
22: end for
Function RCBF-SYNCHRO-SEND(sendsubtable , remotenodeid)
1: for each j in [0, length of remotesubtable] do
2:     BK←subtable[j]
3:     for each i in [0, m-1] do
4:         if BK [i] :record [remotenodeid] == 0 then
5:             if BK[i].record [localnodeid] == 1 then
6:                 insert the BK to sendsubtable
7:             end if
8:         end if
9:     end for
10: end for
11: return sendsubtable
```

4.2. Dynamic Cache Mechanism

The inter-ISP links are expensive because an ISP either pays another ISP for carrying its traffic (in a customer-provider relationship) or it needs to mutually carry the same amount of traffic from the other ISP(in a peer-to-peer relationship) [19]. In addition, the locality of reference of request streams indicates caching the information also benefit the clients in ISP. We deploy the cache servers near the inter-ISP routers for caching the content which are requested frequently and reducing the inter-ISP traffic.

In this section, we propose a novel dynamic cache mechanism using dl-RCBF. The cooperation in the cache group works as follows: When a cache receives a request for an object, it first searches the record in dl-RCBF by hash function and finds out where the object was stored. If false positive occurs or any of them does not have the object, the request is forward to external network.

As algorithm 3 shows, once a local miss occurs, the cache server forwards the request to other servers according the record in dl-RCBF. When the remote node request the same content more than a given threshold, the replica will be placed in it for alleviating the network traffic pressure. In this case, especially popular contents are stored in each cache server. As we know, consistency is the basic problem in distributed environment. The content index is shared among the servers through optimistic synchronization mechanism, so when the consistency problem happens, the receiver will inform the sender independent of index synchronization. In practice, because of the access locality, the above case occurs rarely.

Algorithm 3 Pseudo-code for dynamic caching mechanism

```
Function dynamic-caching-send(key)
1: if RCBF-SEARCH(key)! = 0 then
2:     nodeid←get the nodeid from dl-RCBF
3:     return nodeid
4: else
5:     return 0
6: end if
Function dynamic-caching-recv(key,remotenodeid)
1: if  RCBF-SEARCH(key)! = 0 then
2:     if the key existes in the cache then
3:        sendto(obj,client)
4:        frequent[key]++
5:        if frequent[key]≥threshold then
6:            send the replica to the remotenode
7:        end if
8:     else   //false positive
9:        forward the key to external network
10:    end if
11: else
12:     notice the remotenode this key is invalid
13: end if
14: return 1
```

5. Experiments

In this section we evaluate our IMCDN system at two aspects: 1) evaluate our *dl-RCBF* on the remote hit rate, comparing with *ESC*, *BPR*; 2) evaluate our IMCDN system through OMNET++ simulator. We will compare our method with the following algorithms:

1. A local policy (local) that is a non-cooperative manner.

2. Broadcast Petition Recently (BPR), which means only the elements that have been accessed multiple times in a fixed window of time can be stored in local [10].

3. Evolutive Summary Counters placement (ESC), which finds the node with the largest number of recent accesses in their corresponding ESC-summary [12].

Table 1. Notation for Hit Rate Description

Symbol	Description
$ToRq_i$	The number of total requests in $CacheS_i$
$FwRq_i$	The number of forwarding requests from $CacheS_i$ to another
$LoHit_i$	The number of hit requests which $CacheS_i$ receives from other $CacheS$
$RoHit_i$	The number of hit requests which $CacheS_i$ receives from users

To describe the analysis more explicitly, we define global hit rate and remote hit rate as follows:

Definition 1: Global hit rate (*GHR*): the required content is stored in one of the cache servers at least. It can be computed as formula (1), *n* is the number of cache servers.

$$GHR = \frac{\sum_{i=1}^{n} LoHit_i}{\sum_{i=1}^{n} ToRq_i} \tag{1}$$

Definition 2: Remote hit rate (RHR): the required content is not stored in the cache server closest to the user, but in others. It can be computed as formula (2).

$$RHR = \frac{\sum_{i=1}^{n} RoHit_i}{\sum_{i=1}^{n} FwRq_i} \tag{2}$$

5.1. *dl-RCBF* Evaluation

The purpose of constructing the *dl-RCBF* is decreasing the synchrony traffic and increasing the remote hit rate in a cooperative cache system. Because the structure of *dl-RCBF* is similar to *dl-CBF*, the false positive rate of *dl-RCBF* is equal to *dl-CBF*. However, the remote hit rate is an important metric to represent the advantage of a data summary structure. We validate the *dl-RCBF* by trace-based simulation on url access log as Table 3 shows. In the evaluation, we change the node number in order to test the remote hit rate. We set the parameters as Table 2. In this test, we do not consider the cache size and cache replacement policy, which means the cache servers store each requests. However we consider the false positive in *dl-CBF* and *ESC* in this experiment. It is a pure evaluation on the remote hit rate in an ideal environment. As Figure 3(a) shows, the *BPR* only gets no more than 60% remote hit rate through its broadcasting manner. Meanwhile, the *BPR* do not know which cache server has the content. The *dl-RCBF* maintains the remote hit rate more than 90% as the cache server number increased. An important factor of the advantage is our *dl-RCBF* records the content location and have feedback mechanism. Moreover, these requests missed are caused by false positive. In contrast, we observe a significant decline in the remote hit rate of *ESC* because it forwards the request to other nodes in a certain probability. We also notice the *ESC* and *dl-RCBF* have approximate *RHR* when the cache numbers is 2. This phenomenon is due to the *ESC* only has one choice to forward the request. Then we test the remote hit rate in the condition of considering the cache size and local cache policy (we use LRU), as Figure 3(b) shows. We observe that the *RHR* is decline in each approach because many urls are replaced by LRU. Our approach performs better than other two on account of the record structure. Moreover, we see that *dl-RCBF* declines 10%-15% *RHR* than previous experiment. The optimistic synchrony algorithm lead to the decrease, which means the record may be invalid in the synchronization interval.

Table 2. The Parameters in Evaluation on Remote Hit Rate

Symbol	value	description
D	*4*	The hashtable is divided into 4 subtables
F	*16bit*	The size of finger
C	*4bit*	The size of count
m	*8*	The number of cells in a bucket
K	*8bit*	The size of record
BucketNum	*100k bit*	The number of bucket in a hashtable
URLS	*100k*	Total urls follow $zipf_{\alpha=0.8}$
RQS	*URLS/cachenum*	The number of requests that each cache server receives

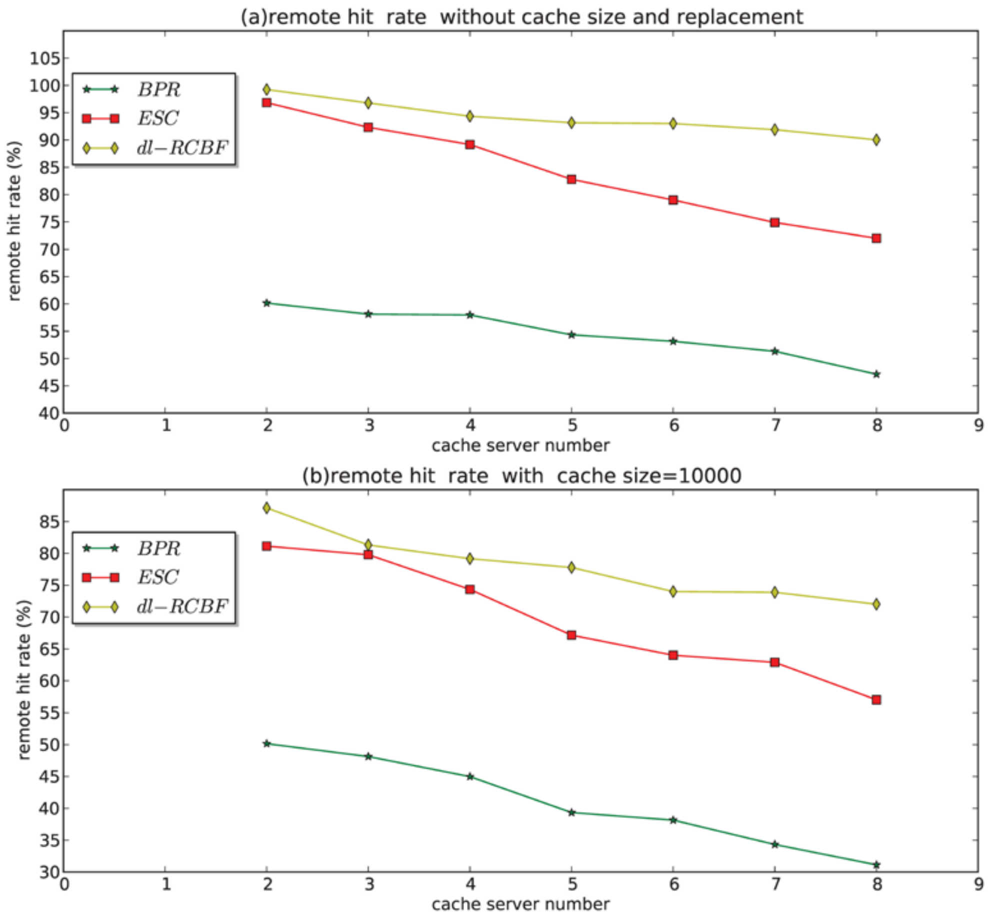


Figure 3. The Remote Hit Rate Comparison

A key factor that impacts the performance of *dl-RCBF* is the traffic generated by forwarding and synchronization. The parameters of this experiment are same as above, see Table 2. Moreover, the traffic generated by broadcasting or ESC forwarding is set as 100 byte, which is the average length of a url. In this experiment, we do not consider the traffic of content moved from one *CacheS* to another. We set the threshold of the synchronization as one percent of *RQS* in *dl-RCBF*. Table 3 shows the overhead of each method we mentioned above. We notice that when the cache server number is 2, the *dl-RCBF* traffic is nearly 10 times than other approaches because the size of *dl-RCBF* subhashtable is larger than broadcast traffic. However, the traffic is only 1.9MB. As the number of servers increasing, the overhead among these methods becomes smaller. When the cache server number increased to 7, the traffic of *dl-RCBF* is smaller than others. We notice that the *dl-RCBF* traffic grows linearly, because the threshold of synchronization declines as the number of servers increasing.

Table 3. The Traffic (KB) Generated by each Approach

cache server numbers	BPR	ESC	dl-RCBF
2	191.32	201.45	1912.50
3	856.78	979.19	3921.18
4	1701.64	2071.24	5427.51
5	4067.35	4388.55	7234.25
6	9167.35	8290.29	9024.64
7	21837.35	14104.55	13389.53

The *dl-RCBF* obtains good performance in remote hit rate though it generates more traffic in the above experiments. Moreover, the traffic is very small in the real network.

5.2. Evaluate the Performance of IMCDN

There are two important aspects impacting the performance of IMCDN: 1) global hit rate(GHR), which indicates the effectiveness of our cache mechanism; 2) the traffic including the inter-domain and intra-domain, which presents the bandwidth cost and the performance of IMCDN system.

Experiment 1(Global Hit Rate): In this experiment, we test the *GHR* of the IMCDN system for a variety of request distributions. We generated three request sets following different *Zipf* distributions with varying parameterization: $zipf_{\alpha=0.5}$, $zipf_{\alpha=0.8}$, and $zipf_{\alpha=1.2}$. The parameters used in this experiment are same as those in Table 2. Moreover, we set the number of cache as 3. Different from the experiments in Section 5.1, the purpose of this experiment is to evaluate the performance of our cache mechanism from a global perspective.

Figure 4 shows the hit ratio for the different algorithms tested. We observe that the query distribution affects significantly the global hit rate for all the algorithms. For the less skewed distribution the global hit rate is smaller because the system accesses a wider diversity of requests. The primary conclusion from this result is that increasing coordination can improve performance. All of the distributed caching algorithms generally outperform local policy because the available memory in one node is small with respect to the number of resources. In Figure 4(a), we notice that local policy only gets 20% *GHR* and the increase of cache size affects GHR little, because the percentage of distinct requests in $zipf_{\alpha=0.5}$ is much larger than cache size. However, in Figure 4(c), local policy is in a similar situation because the popular requests are small than the cache size in one cache server.

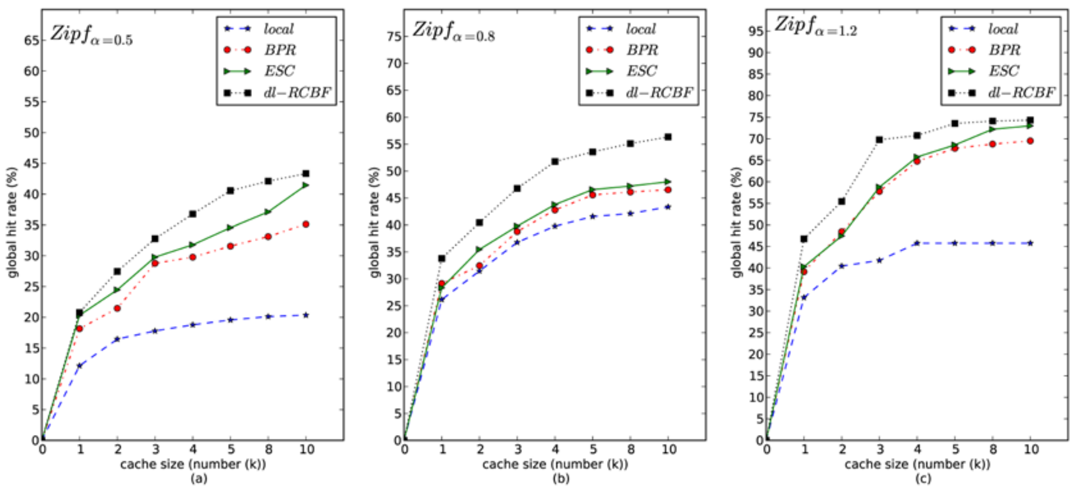


Figure 4. The Global Hit Rate with Different *Zipf* Distribution

Though the remote hit rate of *BPR* is lower than *RCBF* and *ESC*, the difference between it and other algorithms is not so large in global hit rate. Because the *BPR* stores the content accessed many times in local, this manner can improve the local hit rate effectively. We also notice that our method outperforms others in each datasets. Meanwhile, as the cache size increasing, the difference among the three cooperative algorithms becomes little. It indicates when the cache size is close to 10% of the number of requests, all cooperative mechanisms achieves good performance. We notice that the *BPR* outperforms the *ESC* in some situations. This phenomenon indicates that the *ESC* is not a steady algorithm which uses a certain probability to forward the requests. In our system, we overcome it through feedback operation and synchronization. In a word, our approach is a good cooperative mechanism for distributed cache system. We will evaluate the IMCDN system by OMNET++ in the following part.

Experiment 2(Simulation): In this experiment, we test the performance of the IMCDN system on two important metric: intra-traffic and intra-domain traffic. We use real network logs as the dataset including 1,414,287 requests, collected by three gateways of our campus network, as shown in Table 4. We use a typically synthetic architecture which was described in Section 3, as shown in Figure 1. The purpose of this experiment is to evaluate the performance of IMCDN on traffic cost and average access delay by simulation.

Our dataset follows $zipf_{\alpha=0.645}$ distribution. We consider two important factors in the evaluation. The first one is traffic cost including inter-domain traffic and intra-domain traffic, which is the main goal of IMCDN. Fig. 5 shows the traffic of all the policies. We compare our method with other four schemes: 1) no-cache, which means there is no cache system in ISP; 2) local, which means the ISP use non-cooperative cache system; 3) *BPR* and 4) *ESC*. We observe that each cooperative cache scheme reduces 50% inter-domain traffic than non-cache at least, even the local cache strategy reduce 32% as well. We see that *dl-RCBF* outperforms others on inter-traffic, because it obtains higher *GHR* and *RHR*. We notice that the *BPR* and local policies reduce the intra-traffic due to hot-potato algorithm. Moreover, the *BPR* get obvious inter-domain traffic reduction than local. Though the *BPR* do not have extremely high *GHR* than local, its cache server moves the content to another when the content is required several times recently. This manner also decrease more inter-traffic in our simulation because the reply path is different from request path in some case. When some popular replicas were moved to other cache servers, it would reduce the intra-traffic.

Figure 6 shows the average access delay per content, which is the second metric we focus on. The average access delay is computed as formula (3). The totalurls means the number of urls, the *lastpkttime* means the arrival time of the last packet of content and the *firstreqtime* means the request time. Therefore, the large content will take more time to get. Though the cooperative caching algorithms performs better than local policy, our dynamic caching strategy performs best because of the highest global hit rate. Compared with the local policy, our approach has a 37% reduction in average access delay. All in all, our IMCDN system obtains the best performance of all cache algorithms analyzed due to the good hit rate obtained with minimal inter-domain network traffic. This is evident especially for real-world setups, where the request distribution is not too skewed and the local cache is not sufficient.

$$avedelay = \frac{\sum_{i=1}^{totalurls} lastpkttime_i - firstreqtime_i}{totalurls} \tag{3}$$

Table 4. The Parameters of Simulation

parameter	value	description
Total requests	1,414,287	417,523 in Gateway1
		586,772 in Gateway2
		409,992 in Gateway3
Distinct URLs	612,947	158,575 in Gateway1
		318,641 in Gateway2
		135,731 in Gateway3
number of inter-domain routers	3	
number of cache servers	3	as same as the number of routers
number of clients	120	Each router serves 40 clients
link delay	10ms	the threshold of replica moving 20 If remote access times up to 20,this resource would be copied to relative cache server
the threshold of synchrony	2,000	If the change in cache come up to 10%,local server will broadcast its dl-RCBF to each server
cache size	20,000	Each server can cache 20,000 objects
the size of content follows normal		distribution between 100KB and 1MB
Routing algorithm in ISP		Shortest path algorithm
Routing algorithm among ISPs		Hot-potato algorithm

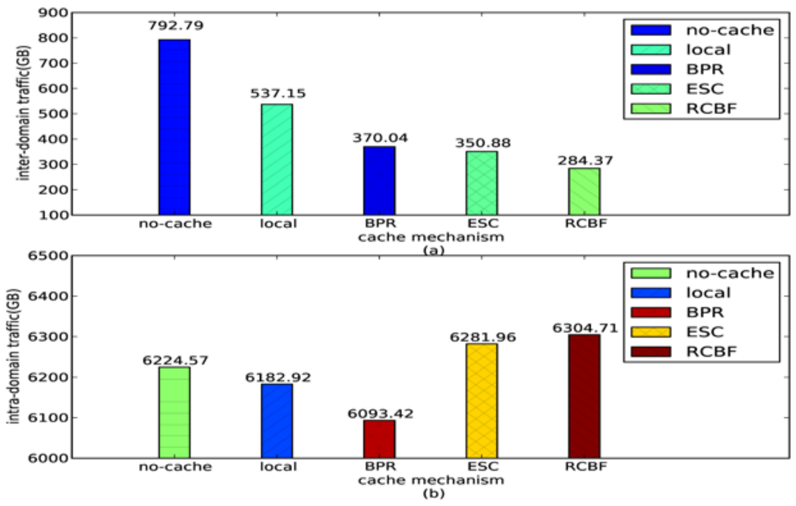


Figure 5. Traffic Cost on Different Cache Strategies

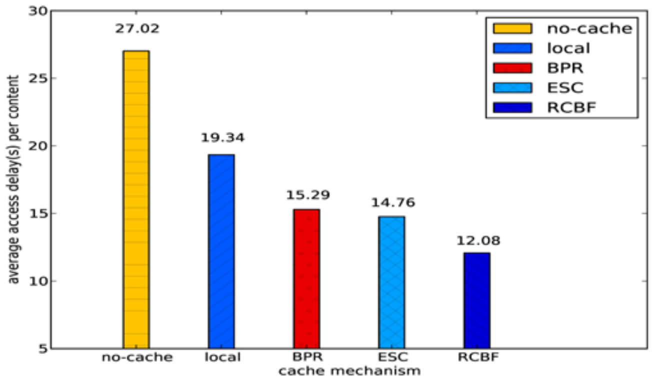


Figure 6. The Average Access Delay of Content

6. Conclusions

IMCDN is a new content delivery mechanism. Through deploying it, ISP obtains more benefit not only for business also for technology: 1) incremental deployment, which means an ISP can deploy the IMCDN independently of other ISPs for providing better services to the in-ISP and out-ISP end users; 2) decreasing inter-domain payment through reducing inter-ISP traffic; 3) improving their own competitiveness through improving user experience (reducing access delay); 4) cache management, which means ISP can choose more valuable content to store in cache servers with the changes in the external environment. This paper introduces a novel ISP-CDN architecture that focuses on inter-ISP traffic and end user access delay. To our knowledge, IMCDN is the first proposal for an ISP which has multiple inter-ISP routers.

The basis of our distributed cache environment is *dl-RCBF*, which is an efficient data structure that records where the content is stored. In order to improve the overall performance, we design an optimistic synchronization in the distributed environment. Our tests have also shown that *dl-RCBF* obtains the higher global hit rate than *ESC*.

Additionally, our IMCDN system achieves lower inter-traffic than other state-of-the-art approaches such as conservative caching mechanism. Our proposal obtains inter-domain traffic reduction more than 47% and 19% comparing with *local* policy and *ESC* respectively.

Though we consume more bandwidth than others, the overhead is far small comparing the overall traffic. In future, we will investigate the details of content caching policy considering the size of content in real network. Needless to say, network caching mechanism remains an interesting open problem.

Acknowledgements

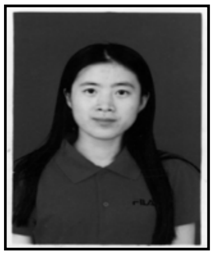
This work is partially supported by the High-Tech Research and Development Plan of China (Grant No. 2011AA010705, 2012AA012506), National Grand Fundamental Research 973 Program of China (Grant No. 2011CB302605); the National Natural Science Foundation of China (Grant No. 61173145, 61202457); The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers.

References

[1] G. Pallis and A. Vakali, “Insight and Perspectives for Content Delivery Networks”, Comm. ACM, vol. 49, no. 1, **(2006)**, pp. 101-106.
[2] Akamai, Powering a better Internet. http://www.akamai.com/.
[3] PPTV. http://www.pptv.com/.
[4] H. Jung, M. Lee, D. Ko, T. T. Kwon and Y. Choi, “How can an ISP merge with a CDN?”, IEEE, Communications Magazine, vol. 49, no. 10, **(2011)**, pp. 156-162.
[5] Akamai Briefing: Highly Distributed Computing is Key to Quality on the HD Web. http://www.akamai.com/hdwp.
[6] C. Huang , A. Wang, J. Li and K. W. Ross, “Measuring and evaluating large-scale CDNs”, Proc. 8th ACM SIGCOMM conference on Internet measurement, Vouliagmeni, Greece, **(2008)** October, pp. 15-29.
[7] A. Varga, “OMNeT++ Discrete Event Simulation System”, http://www.omnetpp.org/.
[8] Market, “Content Delivery Networks, Market Strategies and Forecasts (2001-2006)”, Technical report, AccuStream iMedia Research, **(2006)**.
[9] N. Kamiyama, T. Mori, R. Kawahara, S. Harada and H. Hasegawa, “ISP-Operated CDN”, IEEE INFOCOM Workshops, Rio de Janeiro, Brazil, **(2009)** April, pp. 1-6.
[10] D. Dominguez-Sal, J. Larriba-Pey and M. Surdeanu, “A Multi-Layer Collaborative Cache for Question Answering”, Proc. Euro-Par Conf., **(2007)**, pp. 295-306.
[11] T. Cortes, S. Girona and J. Labarta, “Design Issues of a Cooperative Cache with no Coherence Problems”, Proc. Fifth Workshop IO in Parallel and Distributed Systems (IOPADS), **(1997)**, pp. 37-46.
[12] D. Dominguez-Sal, J. Aguilar-Saborit, M. Surdeanu and J. L. Larriba-Pey, “Using Evolutive Summary Counters for Efficient Cooperative Caching in Search Engines”, IEEE Trans. Parallel and Distributed Systems, vol. 23 , no. 4, **(2012)**, pp. 776-784.
[13] H. Jiang, J. Li, Z. C. Li and X. Y. Bai, “Efficient Large-scale Content Distribution with Combination of CDN and P2P Networks”, International Journal of Hybrid Information Technology, vol. 2, no. 2, **(2009)**, pp. 13-24.
[14] J. Kangasharju, J. Roberts and K. W. Ross, “Object replication strategies in Content Distribution Networks”, Computer Communications, vol. 25, no. 4, **(2002)**, pp. 376-383.
[15] R. Buyya, A. K. Pathan, J. Broberg and A. Tari, “A Case for Peering of Content Delivery Networks”, IEEE Distributed Systems Online, vol. 7, no. 10, **(2006)**.
[16] L. Fan, P. Cao, J. Almeida and A. Z. Broder, “Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol”, IEEE/ACM Transactions on Networking, vol. 8, **(2000)**, pp. 281-293.
[17] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh and G. Varghese, “An improved construction for counting bloom filters”, LNCS 4168, 14th Annual European Symposium on Algorithms, **(2006)**, pp. 684-695.
[18] G. Hasslinger, F. Hartleb and T. Beckhaus, “User access to popular data on the Internet and approaches for IP traffic flow optimization”, Proc. Analytical and Stochastic Modeling Techniques and Applications Conf., Madrid, Spain, **(2009)**, pp. 42-55.
[19] X. Dmitiropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. Clay and G. Riley, “As Relationships: Inference and Validation”, ACM SIGCOMM Computer Comm. Rev., vol. 37, no. 1, **(2007)** January, pp. 31-40.
[20] J. E. Lim, O. H. Choi, H. S. Na and D. K. Baik, “An efficient content distribution method using segment metadata annotation in CDN”, International Journal of Advanced Science and Technology, vol. 1, no. 12, **(2008)**, pp. 85-90.

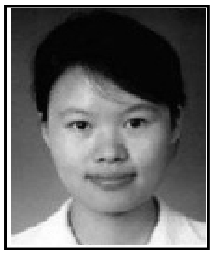
Authors

Qiao Li received his B.S. and M.S. degrees in computer science from Harbin Institute of Technology (HIT) in 2007 and 2009. He is now working towards his Ph.D. degree in computer science at Harbin Institute of Technology. His research interests include computer network, distributed computing.



Hui He received the B.S., M.S. and Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China. Since September 1999, she has been with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China, where she became an Associate Professor in October 2007. Her research interests include network computing, network security.

Bin-Xing Fang received his M.S. and Ph.D degrees in computer science from the Tsinghua University and Harbin Institute of Technology of China in 1984 and 1989 respectively. He is currently a member of Chinese Academy of Engineering. His current research interests include information security, information retrieval, and distributed systems



Hong-Li Zhang received her MS and Ph.D in Computer Architecture from the Harbin Institute of Technology on July 1996 and December 1999, respectively. Her research interests are focused in the area of network security, Internet measurement and network computing. She was awarded 3 Ministry Science and Technology Progress awards and published over 50 papers in journals and international conferences.

Wei-Zhe Zhang received his B.S., M.S. and Ph.D. degrees in computer science from Harbin Institute of Technology (HIT) in 1999, 2001 and 2006 respectively. He has been working in HIT from 2002 and been an Associate Professor at computer science of HIT since 2007. His research interests include computer network, theory of computation and parallel computing.