Scientific Research Publishing

Int. J. Communications, Network and System Sciences, 2017, 10, 76-87
http://www.scirp.org/journal/ijcns
ISSN Online: 1913-3723
ISSN Print: 1913-3715

Neighborhood-Based In-Network Caching for Information-Centric Networks

Shou-Chih Lo, Jhih-Sian Hu, Varsha A. Kshirsagar

Dept. of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan
Email: sclo@gms.ndhu.edu.tw

How to cite this paper: Lo, S.-C., Hu, J.-S. and Kshirsagar, V.A. (2017) Neighborhood-Based In-Network Caching for Information-Centric Networks. *Int. J. Communications, Network and System Sciences*, **10**, 76-87.
https://doi.org/10.4236/ijcns.2017.108B009

Received: May 13, 2017
Accepted: August 11, 2017
Published: August 14, 2017

Abstract

The current Internet is based on host-centric networking, and a user needs to know the host address before reaching a data target in the network. The new architecture of information-centric networking (ICN) facilitates users to locate data targets by giving their data names without any information about host addresses. In-network caching is one of the prominent features in ICN, which allows network routers to cache data contents. In this paper, we emphasize the management of in-network cache storage, and this includes the mechanisms of cache replacement and cache replication. A new cost function is then proposed to evaluate each cache content and the least valuable content is evicted when cache is full. To increase cache utilization, a cooperative caching policy among neighboring routers is proposed. The proper network locations to cache data contents are also discussed in the paper. Experimental results show the superiority of the proposed caching policy than some traditional caching polices.

Keywords

Information-Centric Networking, Named Data Networking, Cache Management, Cooperative Caching

1. Introduction

As the volume of data on the Internet becomes extremely huge, big data search techniques are necessary. A web search engine becomes such a solution to search information on the Internet. The search engine uses crawling software to browse web sites, which consumes network bandwidth and storage space for indexing web content. Content Delivery Networks (CDNs) [1] and Peer-to-Peer (P2P) networks [2] are then introduced to improve the efficiency of data sharing. These networks are constructed using overlay network technologies. Data contents are replicated and stored at multiple hosts in CDNs. When a user requests one data content, CDN routers direct the request to the closest storing host. In P2P, a data content is divided into chucks. A content request can be satisfied by downloading chucks from multiple hosts simultaneously.

Compared to overlay networking, the new information-centric networking (ICN) [3] changes the routing structure of the bare network to directly support content distribution. The current Internet is based on host-centric networking by which data exchanges are specified with source hosts and destination hosts. ICN enables users to locate data contents by giving their data names no matter where these data are stored. Data search is directly supported by the underlying routing protocol in the network rather than indirectly supported by a search engine. This revolution technology opens a potential direction on the future Internet.

In addition to name-based routing, one notable point of ICN is in-network caching by which each router has cache storage. A router can immediately respond a data request if there is one copy of the desired data content in its cache storage. This improves the delivery of popular data contents in time and bandwidth consumption. There are two types of caching schemes in ICN: off-path and on-path. In on-path caching, one data content is cached in all or part of the routers along the path from the data source node to the data request node. All routers in a path have the same cache content in the LCE (Leave Copy Everywhere) scheme [4], which incurs high content redundancy. Part of routers along a path can be properly selected according to a probability model [5] or graph-based metrics such as node degree and betweenness [6] [7]. In off-path caching, data contents are cached in a certain set of routers in the network and their locations are determined by a hash function [8]. A combination scheme which uses both on-path and off-path caching is also possible [9].

In this paper, we propose a new on-path caching with two distinguished features. First, all neighboring nodes along a path can share cache contents to increase the cache hit ratio. Second, the cache replacement policy is more practical, which always evicts the least valuable one in cache space according to a newly defined cost function. The remainder of this paper is organized as follows: Section 2 provides a brief survey on ICN; Section 3 describes the proposed in-network caching; Section 4 shows the performance evaluation; and Section 5 presents concluding remarks.

2. Background Knowledge

IP routing has an inherent defect that is host-centric in nature and hence there are no any hints in routing tables about where any data contents are in the network. Users need to discover data locations (*i.e.*, host addresses) by themselves and may rediscover the locations if data contents have been moved to another hosts. This raises the need of information-centric services that facilitate data searching. Some early ICN prototype systems [10] [11] provide name resolution services which are similar to the service of the domain name system (DNS). Users can look up data locations based on data names via these resolution services and then forward data requests to destination hosts by traditional IP routing.

Some other ICN prototype systems [12] [13] [14] completely discard IP routing and deploy name-based routing. The name-based routing makes each data content directly addressable in the network. That is, each data content is viewed as a logical host with a data name as its host address. The routing table in each router can direct data requests to their destinations by giving data names. In comparison to the name resolution service where directory lookups are involved in certain servers, the name-based routing needs successive routing table lookups at intermediate nodes along paths to destinations. The request-response model is used in ICN. First, a user launches an interest packet containing the data name of the interested data content. Then, routers in the network will direct this packet to one (or more than one) host that can respond the request by sending back data packets.

ICN has some strengths against IP networking. First, data contents need not be distinct and there can have duplicated data contents with the same name in the network. This can increase data availability. Second, the length of a data name can be theoretically arbitrary, so there is no exhaustion problem on naming space. Third, some encryption information can be included into the data name for security and certification reasons. Fourth, user or data mobility has less influence on ICN due to the asynchronous nature between two communication endpoints.

New research problems are incurred by ICN as well. The scale of such a logical network by viewing each data content as a logical host is quite huge in the age of big data, and the space requirement for storing routing entries in each router is severe. The matching of routing rules based on data names in a router also becomes inefficient for several reasons such as character-based rules, various length rules, and a large amount of rules.

3. In-Network Caching

There are several projects aimed at future Internet architecture under the support of National Science Foundation (NSF) in US. One of the projects is the NDN (Named Data Networking) project [14], which is introduced in this paper to explain the proposed ideas. Note that our proposed mechanism actually needs not to rely on any specific type of ICN architecture.

As shown in **Figure 1**, each NDN router includes three components: CS (Content Storage), FIB (Forwarding Information Base), and PIT (Pending Interest Table). CS is in-network caching space. FIB stores routing rules as a typical router table but in name-based format, and these routing rules are maintained by a routing protocol. PIT also contains name-based routing rules, but these rules serve as data return paths and help to avoid redundant interest packet forwarding.

In the following, we briefly explain the routing process in NDN. When an interest packet arrives at a router. This router looks for a cache content in its CS whose data name exactly matches the interested data name specified in the interest

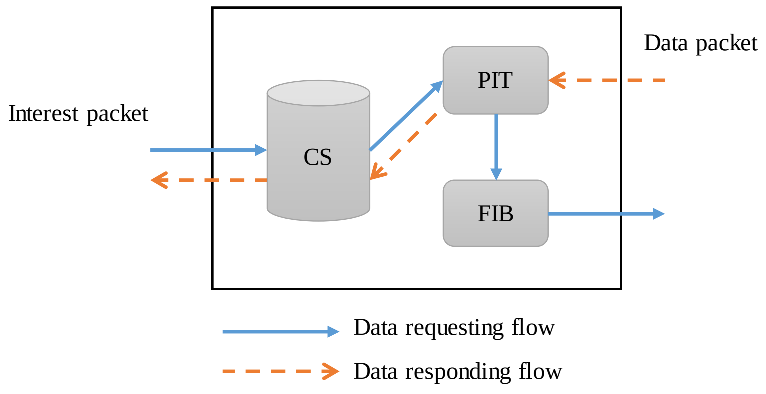


Figure 1. Basic packet forwarding in NDN.

packet. If one is matched, the cache content is immediately sent in a data packet back through the incoming interface of the interest packet. The interest packet is discarded then. Otherwise, the router matches the interested data name with each name entry in its PIT. If an exact match is found, this means that this router has already forwarded an interest packet with this same interested data name before. In this case, the incoming interface of the interest packet is appended to the matched entry in the PIT, and the interest packet is discarded then. Otherwise, the router adds a new name entry in its PIT by recording the interested data name and the incoming interface of the interest packet. The router then matches the interested data name with each name entry in its FIB. If a longest prefix match is found, the interest packet is sent out through the outgoing interface indicated in the matched entry. If there is no match in FIB, the interest packet is discarded then.

When a data packet arrives, the router matches the data name extracted from the data packet with each name entry in its PIT. If there is a match, the data packet is duplicated and forwarded to all interfaces listed in the matched entry, implying the possibility of multicasting. Based on a certain caching policy, this router may store one copy of this data packet into its CS.

3.1. Cache Management

Cache management includes two basic functions. One is cache replacement by which one victim is selected and replaced by a new cache content when CS becomes full. The other is cache replication by which the same cache content is stored into several CS nodes.

There are four common cache replacement polices as follows:

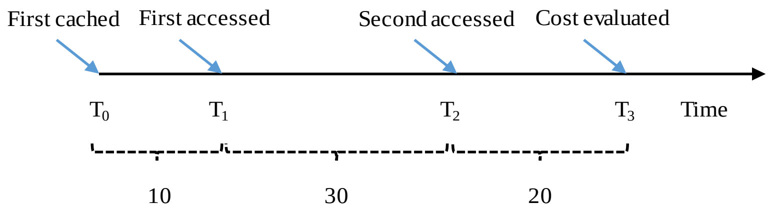
- FIFO (First-in-First-Out): The victim is always the cache content that was the earliest cached in CS.
- LRU (Least Recently Used): The victim is always the least recently used cache content in CS.
- LFU (Least Frequently Used): The victim is always the least frequently used cache content in CS.
- RAND (Random): The victim is a cache content randomly selected from CS.

These policies are simple but are not enough to select the least valuable cache content from CS. We propose a new cost function to evaluate each cache content by combining several cost metrics. The first cost metric is the access frequency (AF) as in LFU. A cache content that is frequently accessed is valuable. The second cost metric is the frozen time (FT) during which a cache content is not accessed by any users. A cache content that has a short frozen time in total is valuable. The third cost metric is the hop count (HC) of the distance between the CS node and the source node where a cache content is from. A cache content that is far away from its source node is valuable, because the re-copy cost is high if this cache content is deleted.

The proposed cost function is shown in (1). An example cost evaluation is depicted in **Figure 2**. In this example, the cache content is away from the source node with 4 hops and is cached in CS at time T_0. This cache content is accessed at time T_1 and T_2. T_3 is the point of time to evaluate the cost value. The cache replacement works as follows. When CS is full and cannot accommodate a new cache content, all cache contents are evaluated using (1). The cache content that has the smallest cost value is evicted.

$$\text{Cost value} = \frac{AF \times HC}{\sum FT^2}. \tag{1}$$

To facilitate the computing of each cost value, the CS node can store cache contents based on the table structure in **Figure 3**. The timestamp filed records the time of each event, namely initialization, data access, or value evaluation. When a cache content is accessed, the current *AF* is increased by 1, the cumulative *FT* square is increased by the recent *FT* square, and the timestamp is updated. The recent *FT* square is computed by referring to the original timestamp value in the table and the current time. When a cache content is evaluated, the cumulative *FT* square is increased by the recent *FT* square, the timestamp is updated, and the newly computed cost value is recorded. To prevent a large cumulative *FT* square to dilute the total cost value, the time unit should be large enough.



$$Cost\ value = \frac{2 \times 4}{10^2 + 30^2 + 20^2}$$

Figure 2. An example to evaluate the cost value of a cache content.

Name	Data Content	HC	Current AF	Cumulative FT Square	Timestamp	Cost Value

Figure 3. The data format used in CS.

The core of cache replication is the selection of CS nodes to replicate cache contents. In on-path caching, this selection is restricted to those CS nodes along the path from the data provider (or server) to the data consumer (or client). As the example shown in **Figure 4**, where the data consumer and the data provider are connected through a routing path along NDN routers R1 to R5. Without any cache contents on these routers, the data consumer needs to directly retrieve the interested data content from the data provider, which may wait for a long data response time. One extreme policy is all-copy replication by which all these five routers in the figure have a copy of this data content. Although this policy can largely reduce the data response time, redundant cache contents make CS easily overwhelmed. Another extreme policy is one-copy replication by which one of the routers is selected to cache the data content. This policy has the least replication overhead, but the question is which node we should select.

There have three possible locations for the cached node in one-copy policy as shown in the above figure. First, the cached node can be the closest router (*i.e.*, R5) to the data provider, and this policy is called server-side caching. This caching policy can increase the utilization of the cache content, because all interested consumers which may attach to different routers will eventually reach this cached node. The drawback is the data response time may be still high. Second, the cached node can be the closest router (*i.e.*, R1) to the data consumer, and this policy is called client-side caching. This caching policy can reduce the data response time. The drawback is the utilization of the cache content may be low, because other consumers with the same interests may attach to different routers. To leverage the tradeoff between server-side and client-side caching, the middle node (*i.e.*, R3) in the routing path becomes the cached node, and this policy is called middle-side caching. The middle node can be located by recording the hop count of the routing path. When an interest packet is forwarded to the data provider, the hop count is increased by one for each router forwarding. When the data provider is reached, the half of the final hop count is computed. This half value is decreased by one for each data forwarding to the data consumer. The middle node is located when this half value becomes zero.

In addition to all-copy and one-copy policies, the partial-copy policy is another possible way. However, the selection of cached nodes becomes more complex. We get the following conclusions through our performance evaluation. The one-copy policy is better than the all-copy policy. The middle-side policy is better than server-side and client-side policies.

3.2. Cooperative Caching

Because the one-copy replication policy is used in our approach, there have less

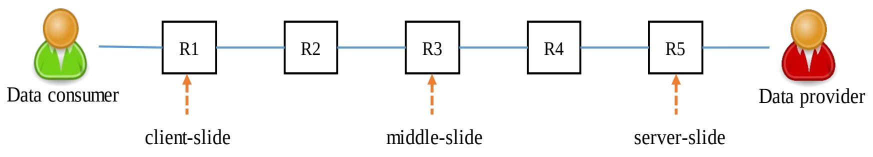


Figure 4. Different cache locations for one-copy replication.

cache contents in the network as compared to all-copy and partial-copy polices. To increase the utilization of any cache contents, we propose a simple cooperative caching mechanism by which those routers in the neighborhood can share their cache contents. When a router cannot find any matched cache content to an interest packet in its CS, this router will overlook the CS in each surrounding router. If there is a possible match, this interest packet is forwarded to another router.

The overlooking is achieved by exchanging a summary of cache contents between two neighboring nodes. Here we use a Bloom filter to do CS summary. All names of cache contents in CS are added into a bit array (called a Bloom filter and denoted as BF_{total}) using a set of hash functions. If we want to know whether a target name exists in this CS, this target name will be represented as another bit array (denoted as BF_{target}) using the same set of hash functions. The target name is definitely not in this CS if the bitwise and operation between these two bit arrays (*i.e.*, BF_{total} and BF_{target}) is not equal to BF_{target}. Otherwise, the target name may or may not exist in this CS, and this would incure a false positive.

The Bloom filter technique has one restriction, which does not support deletion operations. This implies cache contents cannot be removed from CS. Fortunately, this problem can be solved by using a counting Bloom filter [15].

Under cooperative caching, each router in the network will maintain its Bloom filter as any changes of cache contents in CS and flood this Bloom filter to all its one-hop neighboring routers. The new packet forwarding flow in NDN is changed below. Besides checking the local CS, a router checks each neighboring CS (except for the CS of the next-hop router) by examining the corresponding Bloom filter. **Figure 5** gives a scenario example to explain this new packet forwarding. The dotted line depicted in the figure indicates the routing path between the data consumer and the data provider. When an interest packet arrives at R1, R1 decides to forward this packet to the next-hop router R3, because there

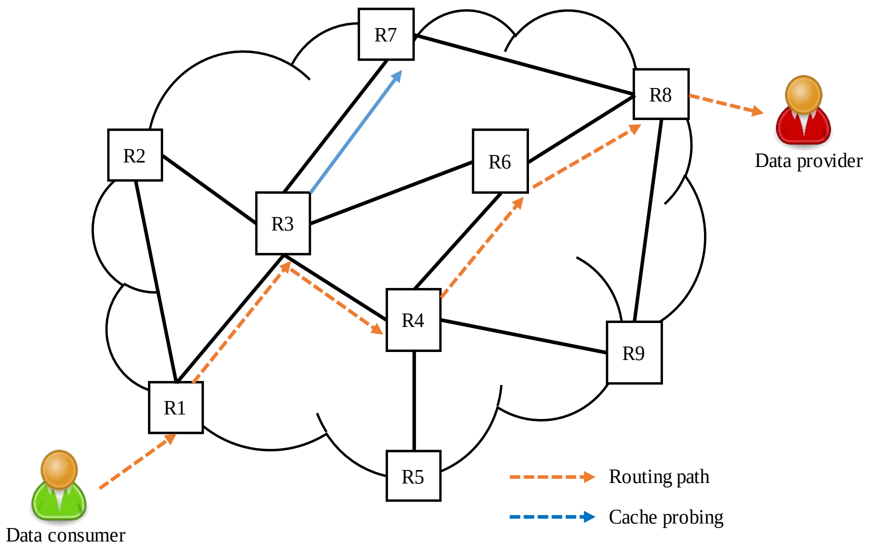


Figure 5. A forwarding example using cooperative caching.

is no any match in CS and Bloom filters. R3 finds a match to the Bloom filter from R7, so the interest packet is forwarded to R7 for cache probing. If this cache probe is successful (*i.e.*, there is a match in the CS of R7), a data packet is retuned then. Otherwise, the interest packet is returned to R3 and then is forwarded to the next-hop router R4. Therefore, a failed cache probe incurs the waste of one round-trip time to a neighboring router. Note that cache probes only happen on one-hop neighboring routers except for the next-hop router. For example, if R3 finds a match to the Bloom filter from R4, it is unnecessary to do this cache probe. The reason is that the interest packet will be eventually forwarded to R4 according to the normal NDN routing.

4. Performance Evaluation

Our proposed caching scheme combines middle-side caching, cooperative caching, and value-based cache replacement. To evaluate the performance of this scheme, we write a simulation program. A real network topology is constructed by using one dataset from the web site of the Internet topology zoo (topology-zoo.org/dataset.html). The original network map contains 113 nodes. We remove 13 nodes of degree 2 and leave 100 nodes as NDN routers.

Data names are generated as follows. 1000 words are selected from an English dictionary and are divided into 200 words, 300 words, and 500 words as three groups. One word is chosen from each of these groups according to a Zip's distribution probability that can model data skewness. These three chosen words constitute a hierarchical data name. Based on this method, 1000 data names are generated and treated as source data contents. In other works, each data content contains nothing but a data name for simplify. The size of CS in each router is specified by the percentage of the total number of data contents. For example, the CS size is denoted as 10% and this means the actual size is 1000 × 10% = 100 cached contents. All generated data contents are then randomly placed on routers with the constraint that each CS cannot be overwhelmed. Finally, 200 interest packets are generated and randomly fed into our routers. The data name specified in an interest packet is generated using the same method mentioned above. The size of an interest packet or a data packet is also dominated by the data name included without considering any header overhead. The size of one Bloom filter is two bytes in the simulation.

Three cost metrics are introduced in performance evaluation:

- Average cache hit ratio: the cache hit ratio of one router is defined as the percentage of CS lookups that lead to successful matches. The average cache hit ratio is the average value over all routers.
- Average hop count: this indicates the average distance in hops between the router where an interest packet is originated and the router where the interested data content is found.
- Average link overhead: this indicates the average number of data bytes that are transmitted over a network link. These data bytes are contributed from interest packets, data packets, and Bloom filters.

Experimental Results

First, we compare the different policies on cache replication under the same value-based cache replacement policy. The cooperative caching is always enabled in this experiment. **Figure 6(a)** shows the comparison of the average cache hitratio and the result is middle-side > all-copy > client-side > server-side. **Figure 6(b)** shows the comparison of the average hop count and the result is middle-side < client-side < all-copy < server-side. **Figure 6(c)** shows the comparison of the

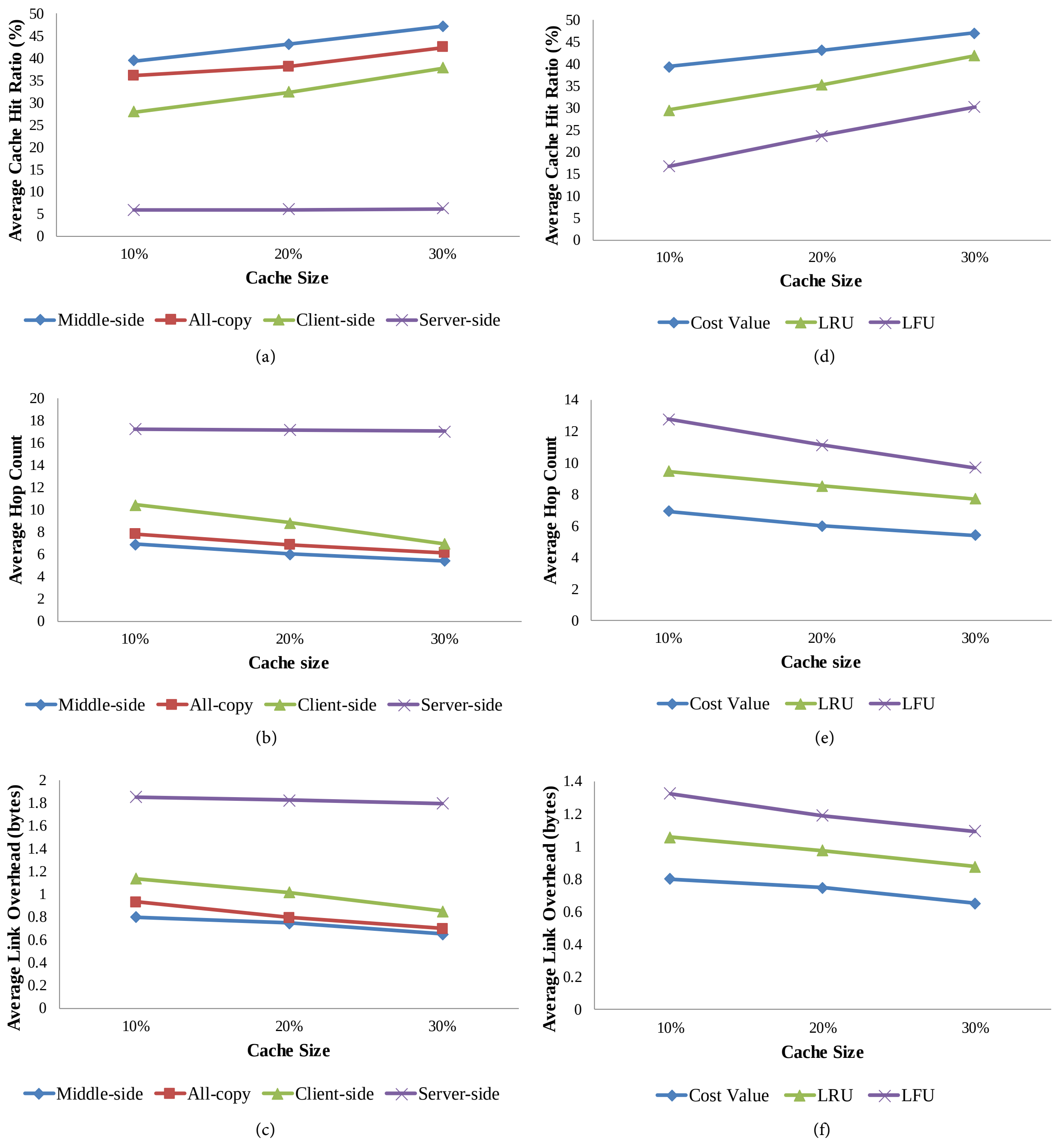


Figure 6. Comparisons of different cache replication policies ((a), (b), (c)) and different cache replacement policies ((d), (e), (f)).

average link overhead and the result is middle-side < client-side < all-copy < server-side. From the above results, we know that the middle-side policy is the best one among all. The server-side policy performs worse because the cache content is very close to the data source and there are many unmatched events for in-between CS. As a result, the average hop count and the average link overhead are both very high. It is supposed that the average hop count should be small in the client-side policy but the fact is not. This is because that the average cache hit ratio is low. A latter data consumer does not get the benefit from the cache content made by a former data consumer, if they attach to different routers. The all-copy policy is the second best approach among all. Actually, redundant cache contents make CS tend to be full and the comparison overhead in each CS is very high. It is reasonable to see that the general benefit from data caching becomes more significant when the cache size is getting large.

Second, we compare the different policies on cache replacement under the same middle-side cache replication policy. The cooperative caching is always enabled in this experiment too. **Figure 6(d)** shows the comparison of the average cache hitratio and the result is Cost Value > LRU > LFU. **Figure 6(e)** shows the comparison of the average hop count and the result is Cost Value < LRU < LFU. **Figure 6(f)** shows the comparison of the average link overhead and the result is Cost Value < LRU < LFU. The above results indicate that our proposed value-based approach performs best among all. The performance improvement to LRU and LFU becomes more significant when the cache size is small. The reason is as follows. The actions to do cache replacement become frequent when the cache size is small, and our proposed approach does evict the least valuable cache content from CS.

5. Conclusions

In this paper, we introduce the advantage of information-centric networking as compared to the traditional host-centric networking like the Internet. There remain several research challenges in such a totally new network architecture. The major contribution of this paper is from the proposed mechanisms on in-network caching. We compare two basic types of cache replication policies: all-copy and one-copy. The experimental results show that one-copy policy is sufficiently better than all-copy policy if the location to cache the single copy is properly selected. We conclude that the middle node along the routing path between a data consumer and a data provider is a good candidate to cache data. A cooperative caching is also introduced by sharing cache contents between neighboring nodes in the network. Instead of exchanging whole cache contents, a summary of cache storage is used by the Bloom filter technique.

For cache replacement, a new cost function is proposed to evaluate each cache content by considering multiple factors such as access time, access frequency, and access distance. The experimental results show that the multiple-factor approach performs better than the single-factor approach such as LRU and LFU. This research work is just a beginning to ICN and more research efforts will be made in the future.

Acknowledgements

The authors would like to thank the anonymous referees for their helpful suggestions. This research was partially supported by Ministry of Science and Technology of the Republic of China under Contract No. MOST 105-2221-E-259-019.

References

[1] Pathan, M. and Buyya, R. (2007) A Taxonomy and Survey of Content Delivery Networks. Technical Report GRIDS-TR-2007-4, The Univ. of Melbourne.

[2] Lua, E.K., Crowcroft, J., Pias, M., Sharma, R. and Lim, S. (2005) A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys & Tutorials*, **7**, 72-93. https://doi.org/10.1109/COMST.2005.1610546

[3] Xylomenos, G., Ververidis, C.N., Siris, V.A., Fotious, N., *et al.* (2014) A Survey of Information-Centric Networking Research. *Communications Survey and Tutorials*, **16**, 1024-1049. https://doi.org/10.1109/SURV.2013.070813.00063

[4] Laoutaris, N., Che, H. and Stavrakakis, I. (2006) The LCD Interconnection of LRU Caches and Its Analysis. *Performance Evaluation*, **63**, 609-634. https://doi.org/10.1016/j.peva.2005.05.003

[5] Psaras, I., Chai, W. and Pavlou, G. (2012) Probabilistic in-Network Caching for Information-Centric Networks. *Proc. Workshop on Information-Centric Networking*, 55-60.

[6] Chai, W.K., He, D., Psaras, I. and Pavlou, G. (2012) Cache "Less for More" in Information-Centric Networks. *Proc. Intl. Conf. on Networking*, 27-40.

[7] Rossi, G. and Rossini, G. (2012) On Sizing CCN Content Stores by Exploiting Topological Information. *Proc. Workshop on Emerging Design Choice in Name-Oriented Networking*, 280-285.

[8] Sourlas, V., Psaras, I., Saino, L. and Pavlou, G. (2016) Efficient Hash-Routing and Domain Clustering Techniques for Information Centric Networks. *Computer Networks*, **103**, 67-83. https://doi.org/10.1016/j.comnet.2016.04.001

[9] Rath, H.K., Panigrahi, B. and Simha, A. (2016) On Cooperative on-Path and off-Path Caching Policy for Information Centric Networks (ICN). *Proc. IEEE Intl. Conf. on Advanced Information Networking and Applications*, Crans-Montana, 23-25 March 2016, 842-849. https://doi.org/10.1109/AINA.2016.131

[10] Tarkoma, S., Ain, M. and Visala, K. (2009) The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture. *Towards the Future Internet—A European Research Perspective*, 102-11.

[11] Ahlgren, B., D'Ambrosio, M., Dannewitz, C., Eriksson, A., *et al.* (2010) Second NetInf Architecture Description. Tech. Rep. D-6.2 v2.0, 4WARD EU FP7 Project.

[12] Chawla, M., Chun, B.G., Ermolinskiy, A., Kim, K.H., Shenker, S. and Stoica, I. (2007) A Data-Oriented (and beyond) Network Architecture. *Proc. ACM SIGCO MM Conf.*, 181-192.

[13] Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H. and Braynard, R.L. (2009) Networking Named Content. *Proc. Int. Conf. Networking Experiments and Technologies*, Rome, 1-4 December 2009, 1-12. https://doi.org/10.1145/1658939.1658941

[14] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., *et al.* (2014) Named Data Net-

working. *ACM SiGCOMM Computing Communication Review*, **44**, 66-73. https://doi.org/10.1145/2656877.2656887

[15] Fan, L., Cao, P., Almeida, J. and Broder, A.Z. (2000) Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE/ACM Transactions on Networking*, **8**, 281-293. https://doi.org/10.1109/90.851975