How to cache in ICN-based IoT environments?

Maroua Meddeb, Amine Dhraief, Abdelfettah Belghith, Thierry Monteil, Khalil Drira

How to cache in ICN-based IoT environments?

Maroua Meddeb*‡, Amine Dhraief*, Abdelfettah Belghith*†, Thierry Monteil‡§and Khalil Drira‡
‡CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
*HANA Lab, Univeristy of Manouba, Tunisia
†College of Computer and Information Sciences, King Saud University, Saudi Arabia
§Univ de Toulouse, INSA, F-31400 Toulouse, France

***Abstract*—Information-Centric Networking (ICN) is an emerging network paradigm based on name-identified data objects and in-network caching. Therefore, ICN contents are distributed in a scalable and cost-efficient manner. With the rapid growth of IoT traffic, ICN is intended to be a suitable architecture to support IoT networks. In fact, ICN provides unique persistent naming, in-network caching and multicast communications which reduce the data producer load and the response latency. Using ICN in an IoT environment requires a study of caching policies in terms of cache placement strategies and cache replacement policies. To this end, we address, in this paper, caching challenges with the aim to identify which caching policies are suitable for IoT networks. Simulation findings show that the combination of the consumer-cache caching strategy and the RR cache replacement policy is the most convenient in IoT environments in terms of hop reduction ratio, server hit reduction and response latency.**

I. Introduction

The Internet of Things (IoT) is a quiet revolution that is changing the world in which we live. The connected world is becoming a reality. It now appears in various domains; Smart Home, Wearable, Retail, Smart Cities, Healthcare, Agriculture, Automotive/Transportation, Industrial Automation and Energy Management. IoT could be defined as the network of heterogeneous resource constrained devices. The term constraints refers to battery, memory and computing power. Over the past six years, IoT has been the subject of numerous researches aiming to improve the data dissemination efficiency. This is due to the explosive growth of the number of "things" connected to the Internet and the advancement in wireless communication technologies. In fact, recent traffic statistics made by Cisco show that the annual global IP traffic has surpassed the 1 ZB (1 zettabyte = 1000 exabytes) threshold in 2016, and will reach 2.3 ZB by 2020 [11].

The Information-Centric Networking (ICN) [14] is a new paradigm which represents a content-based approach in which consumer's requests are satisfied regardless of the content's location or the nature of its producer. ICN is a promising candidate for IoT environment. In fact, this concept provides unique and location-independent content names, in-network caching and name-based routing. These features give ICN the potential to become a key technology for data dissemination in IoT. Indeed, it leverages easy data access and reduces both the retrieval delay and the load on the data producer. The ICN concept promises to eliminate significant communication overhead caused by the distribution of commonly accessed contents.

Thanks to the in-networking caching, consumer's requests are rarely satisfied by the producer since copies of the content are stored in different locations. Caching is an important component in ICN, it significantly reduces the amount traffic and avoids bottleneck caused by publishing data at a unique location. In addition, caches afford the reduction of the required distance for data retrieval. The cache efficiency depends on the adopted caching policy namely the caching strategy and the cache replacement policy. The caching strategy identifies the location of the cache nodes in the topology and the cache replacement selects the content to be evicted from the cache once this latter is full.

In this work, we have studied the impact of the different well-known caching policies. We have implemented different caching policies in an ICN-based IoT network using ccnSim simulator [9]. We note that different combinations of these approaches have implications on the performance of the network in terms of response latency, the publisher's load and the distance to a content. These metrics are adopted as the design objective for any ICN caching policy.

The rest of the paper is organized as follows. Section 2 gives an overview of ICN and its use under IoT networks. We detail, in section 3, different caching policies, notably, caching strategies and cache replacement policies. We evaluate the different combination of caching policies, in section 4, and we discuss the results in section 5. We finally conclude the paper in section 6.

II. Information-Centric Networking: An overview

In this section, we provide an overview of ICN and we focus in particular on Named Data Networking (NDN) architecture as a solution for IoT environment.

A. Why ICN for IoT?

In addition to the explosive traffic growth, users expectations have evolved. In fact, users seek to acquire data by connecting with different fixed or mobile equipment belonging to heterogeneous environments. The rapid traffic growth and users expectations have raised the need for a novel communication model. This issue inspired both research community and industrial so that a few solutions to match the new traffic pattern have been proposed such as Content Distribution Networks (CDNs) and Peer-to-Peer (P2P) overlays. Despite their advantages, CDN and P2P do not give a radical solution to deal with the fundamental issues caused by the current Internet

architecture. Indeed, these approaches are overlaid on top of the current networking architecture. With such fast growth of content and users simultaneously, the incremental changes or solutions to the current Internet architecture will hardly resist to the Internet evolution. Therefore, many efforts have been given in recent years to develop "Clean Slate" solutions for the architecture of the future Internet. The ICN paradigm is the cornerstone of all proposed solutions. In ICN, thanks to the naming scheme and in-network caching, requested contents are addressed by their unique name and can be satisfied by any cache holding it. As a consequence, caching content will impact the life time of the IoT devices' batteries: a request may be satisfied by an active node while the information producer remains in its sleep mode. ICN also addresses the security requirement and targets to secure the contents themselves rather than securing the channels connecting equipment with each other. ICN is a promising candidate for IoT environment. It can natively support IoT scenarios while improving data dissemination and reducing network complexity.

B. ICN building blocks

Since 2007, many ICN architectures have been proposed. All of them provide a location-independent naming scheme and in-network caching, which are the staple key concepts of ICN. However, each one has its specific design and its own choice of features. Naming data objects is an important feature in ICN. Names should be unique, persistent and location-independent. According to the architecture, names can be flat or hierarchical, human-readable or not [27]. Once the contents are identified by their names, they may be stored in many locations in the topology and then retrieved to satisfy requests. To this end, ICN provides in-network caching and specific forwarding strategies. After naming data, the in-network caching is considered as the second important pillar in ICN [28]. Any node in an ICN architecture can be a cache node. The caching decision amounts to the adopted caching approach. In ICN, there is two routing approach; a name-based routing and a Name Resolution System (NRS) based routing [1]. The name-based routing uses the name aggregation to forward requests, that said, this approach uses hierarchical namespaces. By against, NRS-based routing has not a naming scheme restriction. This routing approach uses a third entity which is responsible for binding the name with its current location in the network.

C. NDN for IoT

The so-called NDN is considered as the most suitable ICN architecture for IoT systems [5][3]. NDN uses hierarchical human readable names and name-based routing approach. It defines a receiver-driven, pull-based, robust connection-less communication model. These features are beneficial for the IoT systems in terms of easy and scalable data access, energy efficiency, security and mobility support [4]. NDN architecture is composed of three structures: Content Store(CS), Pending Interest Table(PIT) and Forwarding Interest Base(FIB). Figure. 1 sketches the NDN architecture under IoT deployment. $Consumer1$ is interested in content $/Home/room1/Tmp$, so it sends an *Interest* packet towards $n1$. This latter, upon receiving the *Interest*, checks its CS to verify if there is a copy of the asked content in the cache. In the case of a cache hit, the *Data* is sent back to $Consumer1$. In the case of a cache miss, a PIT check is performed. When the content name is found as an entry in the PIT, this means that there is a node other than $n1$ that already ask for the same information. Then, the *Interest* packet is discarded and another entry is added in the PIT with $n1$ as the node that sends this request. If there is not a match in the PIT, the *Interest* is redirected to the next node as prescribed by its FIB. This latter stores a forwarding information to the next hop using a longest prefix match between the name and FIB entries. If the FIB does not give any information about the next hop, that said there is a routing problem then the *Interest* is deleted.

III. Caching policies

NDN approach natively supports the in-network caching using the CS structure. With NDN architecture, cache nodes belong to the request path and at the response time, nodes decide to cache or not a copy of the content according to the used caching strategy. We present, in the following subsection, different in-network caching strategies proposed in the literature.

A. In-network caching strategies

We explain in Figure. 2 the most cited caching strategies in the literature. In this figure, green gateways represent cache nodes.

Leave Copy Everywhere (LCE) strategy [22], as its name indicates, stores a copy of requested contents in all the nodes along the request path. By against, the Leave Copy Down (LCD) [22] strategy leaves a copy in just one node which is the gateway on one level down in the reverse path towards the consumer. The edge-caching strategy [13] also caches data in one node as with LCD, however, with this strategy, the cache position is totally the opposite, it is the last gateway in the reverse path towards the consumer. The consumer-cache strategy [16] is a variant of the edge-caching. In fact, since consumers are usually connected to edges, this strategy proposes to keep a copy of the content in gateways directly connected to a consumer. Another existing strategy proposed for ICN architectures is the betweenness-centrality strategy [7]. This latter depends on the betweenness-centrality parameter which is calculated for each node in the topology. It measures the number of times a node belongs to a path between any two nodes in the topology. The node with the highest betweenness-centrality parameter stores a copy of the data. Finally, the ProbCache caching strategy [20] decides to cache a copy with a probability inversely proportional to the distance between the consumer and the producer. As a consequence, this strategy privileges caching close to consumers.

We note that strategies presented above are not complex and do not need costly calculations. Some other studies have proposed more complex in-network caching strategies for ICN

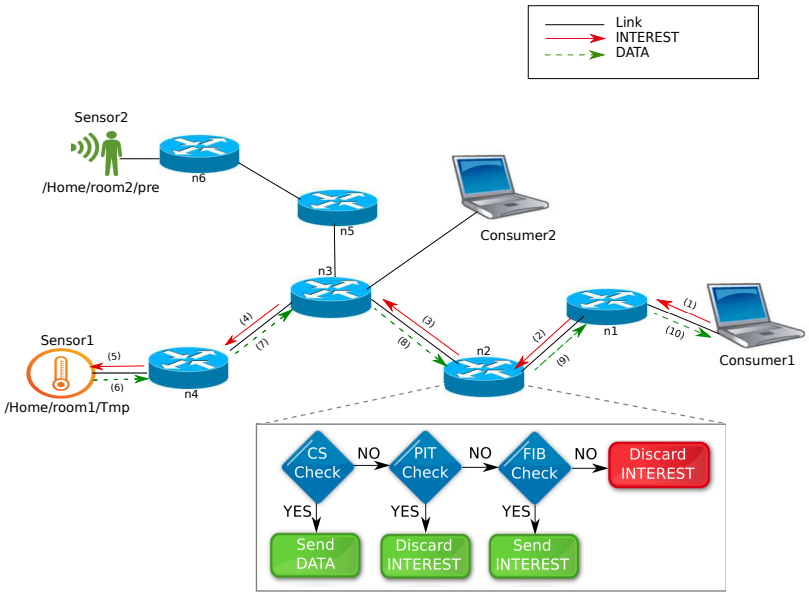


Fig. 1. Forwarding operations in NDN

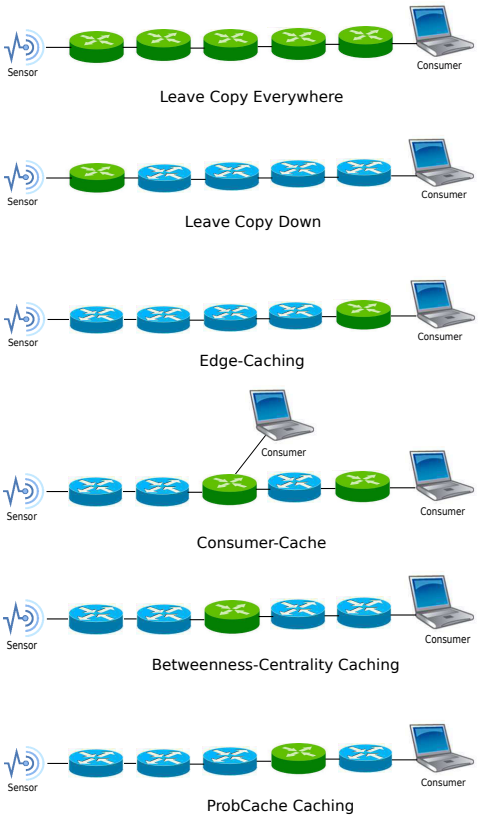


Fig. 2. In-network caching strategies

with the aim to increase the system performances. These advanced researches target more outcomes such caching redundancy [25][19], network congestion [6], supporting diverse traffic types [24], routing of requests to the nearest cache [12]. It is worth-noticing that in-network caching efficiency can be strictly dependent on the context in which it is used. In fact, each context has its specificity in terms of traffic volume and type, the frequency of content generation, consumers' expectations and the ability of machines to support heavy computation.

The cache placement is one of the pillars of ICN which can significantly improve the performances of IoT networks. It increases the data availability in the network. However, it is essential to ensure that the in-network caching does not increase the content redundancy in the network as with LCE and that there are not unusable caching nodes. On the other hand, since the cache size is limited, once the cache is full, some stored content must be deleted to allow caching new items. To this end, cache replacement policies are used to choose the data object to be evicted. We present, in the following subsection, proposed cache replacement policies.

B. Cache replacement policies

Limited cache sizes require the use of content replacement policies. These policies affect the cache efficiency. In fact, one of the objectives of the in-network caching was the requirement for low content delivery delays. So, if finally cached contents are evicted without satisfying any request during their lifetime in a cache, this cache is inefficient. As a consequence, in-network caching efficiency depends not only on the caching strategies but the replacement policies as well.

Traditional cache replacement policies were designed for computer architectures and databases [8]. The dominant cache replacement policy is First In First Out (FIFO). This latter is the default policy in NDN [9]. With FIFO, the oldest data object is replaced by the newly arriving content. This policy has a slightly simpler implementation in comparison to the other strategies because it is the same used policy to store contents in the cache.

TABLE I
CACHE REPLACEMENT POLICIES

Category	Description	Existing Policies	Representative Policy
Recency-based	Keeps the recently referenced objects	LRU, LRU-threshold, LRU*, LRU-hot, SLRU, HLRU, LRU-LSC, SB-LRU	LRU
Frequency-based	Keeps popular contents	LFU, LFU-Aging, LFU-DA, swLFU, Window-LFU	LFU
Randomized policy	A simple random choice to avoid high computation overhead	RR, RAND, HARMONIC	RR
Size-based	Evicts large contents	SIZE, PSS, CSS	SIZE

With the appearance of P2P and CDN, cache replacement policies have begun to be used in web caches [18][26]. Existing policies in the web can be classified into four categories; the recency-based policies, the frequency-based policies, the size-based policies and the randomized policies. The rationale behind the *recency-based* category is that recently requested contents are more likely to be requested again in the near future. These policies are more efficient in the case of high temporal locality of request streams. This means that there is an important number of consumers interested in a common set of data objects. The most known policy in this category is the Least Recently Used (LRU). LRU has been adopted in a considerable number of caching policies [20][10][23][16]. This eviction policy replaces the data object that is not being used for the longest time. LRU is efficient for line speed operations because both search and replacement tasks can be performed in constant time (O(1)).

Frequency-based policies are based on objects popularity. The rationale behind this category is to keep popular objects in the cache to satisfy a high number of requests. For instance, this category is efficient with web pages that provide news or new films. The least frequently used (LFU) policy is the simple variant in this category. With LFU, the cache node keeps track of the number of times a data object satisfies a request. LFU purges the item with the lowest content frequency. However, unlike LRU, the implementation of LFU is computationally expensive since it cannot be implemented in such a way that both search and replacement tasks can be executed in constant time. The drawback of LFU policy is that popular contents which become unpopular remain in the cache for a long time.

The *randomized* policies are specifically designed for a cache with complex data structures. The RR policy is the simplest one. It is a *Random Replacement* policy which replaces a randomly chosen cached item. Obviously, this strategy is not complex in terms of search and replacement tasks.

Finally, the *size-based* category depends on object size as the primary factor. This category comes with the logic to evict large object in order to provide room for many small objects. This category works well with web sites that maintain more text file than multimedia. The basic policy in this category is SIZE.

We conclude that FIFO, LRU, RR and Size replacement policies can perform both operations in constant time, the complexity of LFU replacement operation is O(c) and grows linearly with the cache size c. We summarize in Table. I different categories of cache replacement. LRU, LFU, FIFO, RR and SIZE are already used on the web since a long time. Some other strategies are recently proposed in the context of ICN. In [17], authors proposed a Universal Caching (UC) strategy designed for ICN. UC makes the replacement decision depending on a parameter assigned to any incoming content. This parameter includes the distance from the source, reachability of the router and the frequency of the content access. Al-Turjman et al. in [2] introduced the Least-Value First (LVF) cache replacement policy which takes into account the delay for retrieving content as well as the popularity and age of content experienced by a node.

It is worth noting that both content replacement policies and caching strategies are important to evaluate the in-network caching performance and may complement each other. For this reason, we evaluate in the next section the performance of some combination of caching policies. In this work, we implemented and tested the existing caching policies in an NDN-based IoT environment.

IV. PERFORMANCE EVALUATION

This section details a performance evaluation of our different caching policies for information-centric IoT networks. For this purpose, we use the ccnSim simulator [9]. It is a C++ framework under the OMNeT++ discrete-event simulator which implements the routine to simulate an NDN network. Every NDN node implements the three system elements mentioned in section I, in the form of layers. A core layer which is responsible for PIT management and the communication with other layers. A cache layer since CS acts according to a caching strategy and a replacement strategy. And finally, a strategy layer which takes the decision about interest forwarding. CcnSim uses by default the shortest path forwarding strategy. In the following, we describe adopted metrics, the simulation scenario and the obtained results.

A. Simulation scenario

Analyzing the request popularity distribution in different geographical locations, S. K. Fayazbakhsh et al. deduced, in [13], that the web distribution, used by all works on ICN, behaves as a Zipfian distribution. Virtually, all the ICN studies use the Zipf distribution. However, under IoT, we do not refer to this distribution since it is designed for web-based contents and Internet applications. In our case, INTEREST packets are uniformly distributed.

TABLE II
SYSTEM PARAMETERS

Parameter	Meaning	Values
C	Cache size	4 chunks
$\mid F \mid$	Producers	4000 sensors
F	File size	1 chunk
$Cons$	Consumers	25 consumers
λ	Arrival rate	1
R	Replicas	1
FS	Forwarding strategy	SPR
transmission _delay	Transmission Delay	$[2; 78]\, ms$
simulation _time	Simulation Time	$200s$

In [21], D. Rossi and G. Rossini show that in existing studies the ratio of the cache size C over the catalogue size $F \mid F \mid$, $\frac{C}{F|F|} \in \left[10^{-5}; 10^{-1}\right]$. In our simulation, we set $\frac{C}{F|F|} = 10^{-3}$. We choose the cache size $C = 4$ chunks, the file number $\mid F \mid = 4000$ files and the file size $F = 1$ chunk. We set $\lambda = 1$. We consider a topology following the Transit-Stub model with $N = 260\ nodes$. The topology consists of 2 transit domains with on average 10 transit nodes connected each one to 2 stub domains with on average 6 stub nodes. Figure. 3 presents our TS topology. The 4000 sensors are connected to 40 Gateways. We consider 25 consumers and we suppose that all consumers are already connected at the beginning of the simulation. We generate the topology with GT-ITM[1] (Georgia Tech Internetwork Topology Models). Concerning the transmission delay, they are set by GT-ITM. Values are in the range of $[2; 78]\, ms$.

Our simulations were carried out with a real IoT data extracted from ADREAM [15] building in LAAS-CNRS laboratory which is a smart building.

B. Performance metrics

In our performance evaluation, we measure the *hop reduction ratio*, the *server hit reduction ratio* and the *response latency*.

The *Hop Reduction Ratio* α represents how faster, in term of hops, a content is fetched from a cache than from the server. It is analytically represented by equation Eq. 1. Each client i sends R requests. For each request r from i the hop reduction ratio is calculated. It is the ratio of the path length from the client to the cache that satisfies the request h_{ir} over the path length from the client to the server H_{ir}. The hop reduction ratio of a simulation is the average over N clients of averages over R requests per client of $\frac{h_{ir}}{H_{ir}}$.

$$\alpha = 1 - \frac{\sum_{i=1}^{N} \frac{\sum_{r=1}^{R} \frac{h_{ir}}{H_{ir}}}{R}}{N} \tag{1}$$

The *Server hit Reduction Ratio* β expresses the alleviation of the server load. It is the ratio of the number of requests satisfied by the server $serverhit$ over the totality of requests, which represent requests satisfied by the server and by caches (see Eq. 2).

[1]http://www.cc.gatech.edu/projects/gtitm/

$$\beta = 1 - \frac{\sum_{i=1}^{N} serverhit_i}{\sum_{i=1}^{N} totalReq_i} \tag{2}$$

The *Response Latency* T_{ir} is the duration between the delivery of a content request r from a client i and the response, knowing that the transmission delays between nodes are randomly set. In equation Eq. 3, we calculate γ the average of the response latency over N clients that send R requests.

$$\gamma = \frac{\sum_{i=1}^{N} \frac{\sum_{r=1}^{R} T_{ir}}{R}}{N} \quad (ms) \tag{3}$$

C. Simulation Results

Experiments have been run with different caching policies. We use different combinations of caching strategies and cache replacement policies. In our scenario, all contents have the same size. In addition, in IoT networks, contents are generally small since they represent sensors' values. So, we will not consider the SIZE cache replacement policy. CcnSim is distributed with native support of LRU policy. We have enhanced it to support RR, LFU and FIFO. As caching strategies, it already provides LCE, LCD and ProbCache. We have implemented Btw, edge-caching and consumer-caching strategies. We remind that these simulations have been carried out to evaluate the server hit reduction ratio, the hop reduction ratio and the response latency.

Figure. 4 plots the system performances. We respectively plot in Figure. 4a, Figure. 4c and Figure. 4d, the server hit reduction ratio, the hop reduction ratio and the response latency. We remark that different caching strategies have the same behavior vis-a-vis the cache replacement policies. We first analyze caching strategies independently of used cache replacement policies.

The LCE strategy stores copies everywhere, which make content available at every node. However, caches fill up quickly and consequently, contents are rapidly evicted from the cache which increases the number of evictions and leads to cache misses. This explains the fact that this strategy performs the worst results in this scenario. To better understand the server hit reduction ratio results, we plot in Figure. 4b the average number of evictions for each caching strategy. We notice that the increase in the number of evictions diminishes the cache efficiency. The LCE strategy has the highest number of evictions as it was expected. Figure. 4b shows that the closer the cache nodes are to the producer the higher the number of evictions is. This is because nodes close to the producers belong to many request paths by against nodes close to consumers belongs only to request paths starting from this consumer. Since the LCD strategy selects the cache nodes at one level down from the producer, it has a high number of evictions. With ProbCache, contents can be cached on more than one node within the request path probably near to the consumer. However, with Btw, there is one cache node in each request path and generally in the middle of the path. ProbCache and Btw have a very close number of evictions.

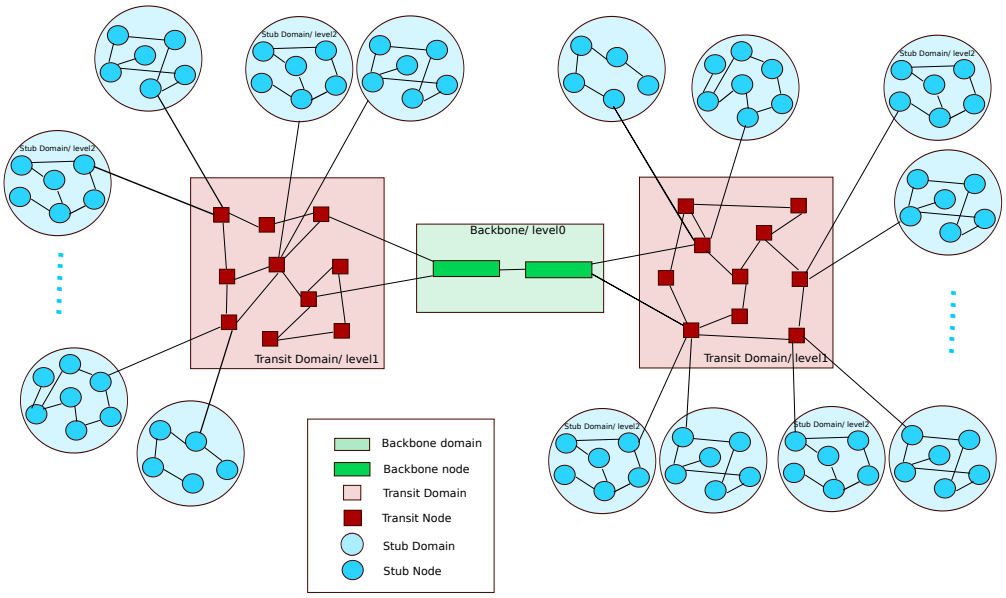


Fig. 3. Transit-Stub topology

This latter is slightly lower under Btw. Finally, the edge-caching and Consumer-cache have almost the same number of evictions.

With LCE caching strategy, the server hit reduction ratio (Figure. 4a) is from 0.32 to 0.56. This value means that only 56% of requests are satisfied from cache nodes. The hop reduction ratio (Figure. 4c) for this strategy is between 0.44 and 0.67. Which means that paths are reduced by 67% in term of number of hops. Concerning the third metric, which is the response latency (Figure. 4d), it is about 120ms to 221ms.

The caching strategy LCD decides to cache contents at the node on one level down from the response source (Producer/cache node). After a certain number of requests, it tends to LCE and all path nodes become caches. For this reason, LCD results, are not so good as LCE. Figure. 4c shows that LCD records from 53% to 69% of hop reduction, and requests take about 144ms to 187ms of response latency (Figure. 4d). Its server hit reduction is about 0.50 to 0.68.

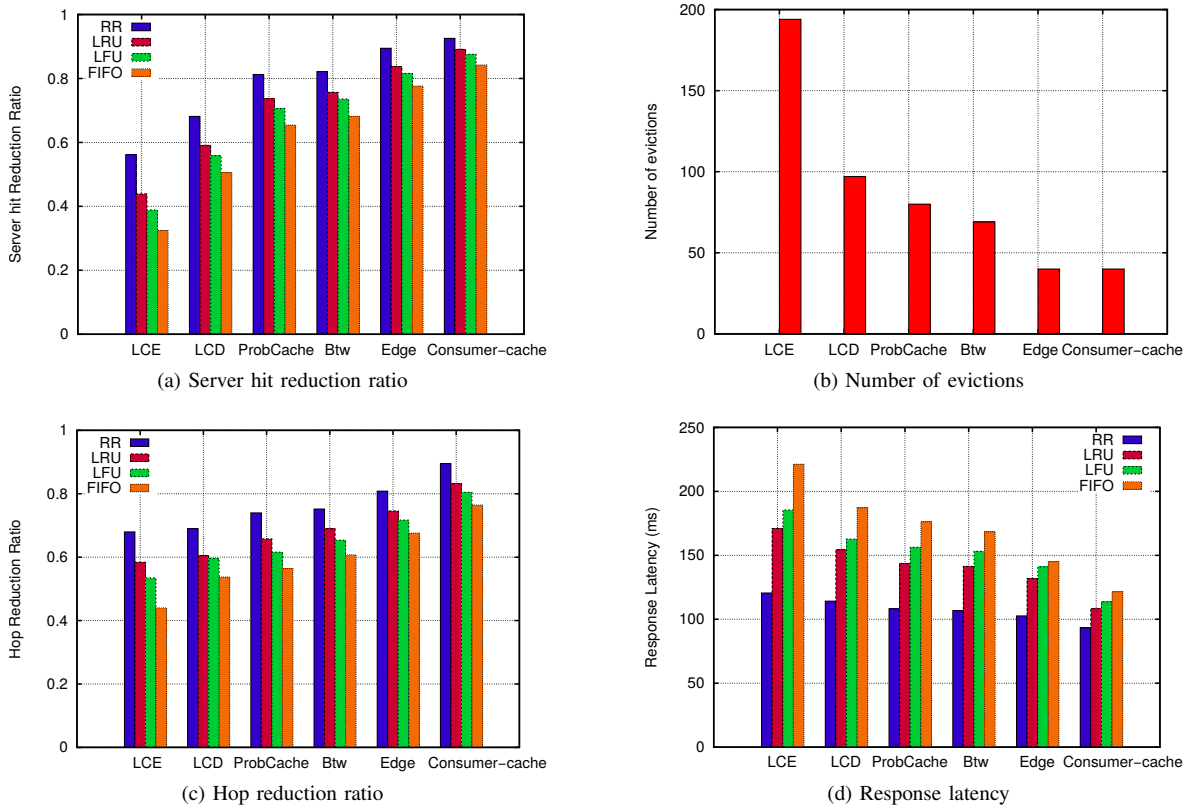
Concerning ProbCache and Btw, cache nodes are selected in the middle of the request path and probably more close to consumers, in the case of ProbCache. Simulation results of these two strategies are medium comparing to other caching strategies. The hop reduction ratio using ProbCache and Btw is respectively about 0.56 to 0.73 and 0.60 to 0.75. The same for response latency, ProbCache and Btw reports respectively about 108ms to 176ms and 106ms to 168ms. Figure. 4a shows between 0.65 and 0.81 of server hit reduction under ProbCache and from 0.68 to 0.82 for Btw.

Results in [13] showed that the edge nodes are the best placement for cache nodes. Our findings confirm this conclusion. In fact, edge-caching reports good results. Under this strategy, we measured from 0.77 to 0.89 in server hit reduction, between 0.67 and 0.80 of hop reduction ratio, and 102ms to 145ms as response latency.

We detail now the results of consumer-cache strategy. This latter stores copies in nodes attached to consumers which allow these consumers to easily reach requested contents. This strategy is a variant of the edge-caching strategy. Consumer-cache has the best simulation results because requests are, in most cases, satisfied by the first hop node. We report for this strategy from 0.84 to 0.92 of server hit reduction. The hop reduction ratio is about 0.76 to 0.89, this implies that requests only cross 11% of hops on the path towards the producer. Finally, with this strategy, the response latency varies from 93ms to 121ms.

We analyze now the cache replacement policies results. Figure. 4 shows that RR outperforms other policies. To understand this point, it is worth noticing that these findings are strictly dependent on the request distribution. In fact, in the web, with a Zipfian distribution, requests privilege popular contents. As a consequence, it is more advantageous to keep in cache nodes the most requested contents. In this sense LRU and LFU were designed. These policies may perform better results in the web environment. By against, in an IoT environment, requests are uniformly distributed and all sensors have close probabilities of being solicited. In other words, contents are randomly requested. This explains why, in our scenario, the RR policy outperforms LRU and LFU. The FIFO policy aims to keep each content as long as possible in the cache node regardless of the frequency with which each content is requested and the evicted item is not uniformly selected. This policy may be suitable for closed queue-based request distribution. In our scenario, FIFO presents the worst results.

Figure. 4a reports from 0.56 to 0.92 of server hit reduction with RR policy. Under LRU policy, from 43% to 89% of requests are satisfied by cache nodes. This figure portrays between 0.38 and 0.87 of server hit reduction using LFU. Finally, with FIFO policy, results are almost from 0.32 to



(a) Server hit reduction ratio

(b) Number of evictions

(c) Hop reduction ratio

(d) Response latency

Fig. 4. System performances

0.84. Figure. 4c gives the same performance results. RR policy outperforms other replacement policies with 0.67 to 0.89 for hop reduction ratio. This ratio is about 0.58 to 0.83 and between 0.53 and 0.80 under LRU and LFU respectively. With FIFO policy, requests only cross from 24% to 56% of the path towards the producer. The response latency, depicted in Figure. 4d, is the lowest with RR policy. It is about 93ms to 120ms. With LRU and LFU, it respectively varies from 108ms to 170ms and from 113ms to 185ms. The FIFO policy reports the longest response latency with 121ms to 221ms.

V. Discussion

In addition to caching policies, the caching efficiency also depends on the application and consumers expectations. In fact, the notion of content's popularity makes sense in the web but not in an IoT network. This comes down to the nature of the content; some contents in the web as videos, pictures, weather, news, etc. may be more popular than others. For instance, a new film or news that everyone is talking about. Popular contents are more solicited by consumers, as a consequence, many copies of this data will be available in the network and it will be easier to reach this information. In this case, caches are not quickly filled because if a popular content already exists in the cache, just an update of this version is performed. By against, in an IoT environment, all sensors have the same probability of being solicited, so cache nodes store copies of different contents and become full faster. In addition, the request frequency is more important in IoT than in the web since in IoT there are many monitoring applications that periodically request for sensor's values. We conclude that caching policies in the IoT scenario is more complicated.

Based on our findings, we conclude that, in an IoT scenario, the couple consumer-cache caching strategy and RR cache replacement policy performs the best results. We can also remark that when the number of evictions, under a specific caching strategy is not very important, the choice of the used cache replacement policy does not significantly impact the results. In fact, with the consumer-cache caching strategy and the edge-caching strategy, the number of evictions is very low (Figure. 4b), as a consequence, the cache replacement policy is not frequently used. We then report a minor difference using different cache replacement policies with these two strategies. However, under LCE strategy, which has a very high number of evictions, we can obviously notice the influence of the cache replacement policy choice on the system performance results.

To recapitulate, in an IoT environment:

- the choice of caching strategies is a compromise between data availability and the caching cost. For instance, with LCE we have a very high availability, nevertheless, it performs the worst results. However, with consumer-cache strategy, although the number of cache nodes is much fewer than LCE, it has best results. This is due to the fact that consumer-cache makes content more close to consumers.

- the cache replacement policies are highly dependent on the content distribution. The RR policy is recognized for uniform distribution, while for Zipfian distribution, LRU outperforms other cache replacement policies.
- an ideal caching policy should minimize the number of cache evictions. For that, it is necessary to avoid caching in nodes with high traffic as the backbone or in transit domains, otherwise, caches will fill up very quickly. In addition, it is important to keep in the cache only useful copies and this depends, as we said in the previous item, on the content distribution.

VI. Conclusion

In this paper, we focused on the advantage of ICN paradigm in IoT scenarios. Since the number of connected things increases significantly year after year, it is important to pay particular attention to its evolution. As a few other researches, we admit that ICN has the potential to become a key technology for data dissemination in IoT networks. The in-network caching assumes an important role in the ICN efficiency. To this end, we have proposed in this work, a comparative study of different caching policies, with the purpose to appoint the most suitable combination of caching strategy and cache replacement policy for IoT networks. To evaluate the performance of each caching policy, we calculated the server hit reduction ratio, the hop reduction ratio and the response latency. Results showed that the combination of the consumer-cache strategy and the RR policy exhibits superior performance.

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, July 2012.

[2] F. M. Al-Turjman, A. E. Al-Fagih, and H. S. Hassanein, "A value-based cache replacement approach for information-centric networks," in *38th Annual IEEE Conference on Local Computer Networks - Workshops*, Oct 2013, pp. 874–881.

[3] M. Amadeo, C. Campolo, A. Iera, and A. Molinaro, "Named data networking for iot: An architectural perspective," in *Networks and Communications (EuCNC), 2014 European Conference on*, June 2014, pp. 1–5.

[4] M. Amadeo, C. Campolo, and A. Molinaro, "Internet of things via named data networking: The support of push traffic," in *Network of the Future (NOF), 2014 International Conference and Workshop on the*, Dec 2014, pp. 1–5.

[5] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, "Information centric networking in the iot: Experiments with ndn in the wild," in *Proceedings of the 1st International Conference on Information-centric Networking*, ser. ICN '14. New York, NY, USA: ACM, 2014, pp. 77–86.

[6] M. Badov, A. Seetharam, J. Kurose, V. Firoiu, and S. Nanda, "Congestion-aware caching and search in information-centric networks," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, ser. ACM-ICN '14. New York, NY, USA: ACM, 2014, pp. 37–46.

[7] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in information-centric networks," in *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I*, ser. IFIP'12. Springer-Verlag, 2012, pp. 27–40.

[8] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical internet object cache," in *Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '96. Berkeley, CA, USA: USENIX Association, 1996, pp. 13–13.

[9] R. Chiocchetti, D. Rossi, and G. Rossini, "ccnsim: An highly scalable ccn simulator," in *Communications (ICC), 2013 IEEE International Conference on*, June 2013, pp. 2309–2314.

[10] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *2012 Proceedings IEEE INFOCOM Workshops*, March 2012, pp. 316–321.

[11] Cisco. (1999) Cisco vni forecast and methodology, 2015-2020. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html.

[12] S. Eum, K. Nakauchi, M. Murata, Y. Shoji, and N. Nishinaga, "Catt: Potential based routing with content caching for icn," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 49–54.

[13] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 147–158, Aug. 2013.

[14] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '09. ACM, 2009, pp. 1–12.

[15] LAAS-CNRS, "Adream," http://www.laas.fr/1-32329-Le-batiment-intelligent-Adream-instrumente-et-econome-en-energie.php, 2013.

[16] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, and K. Drira, "Cache coherence in machine-to-machine information centric networks," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*, Oct 2015, pp. 430–433.

[17] B. Panigrahi, S. Shailendra, H. K. Rath, and A. Simha, "Universal caching model and markov-based cache analysis for information centric networks," in *2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, Dec 2014, pp. 1–6.

[18] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374–398, Dec. 2003.

[19] I. Psaras, W. K. Chai, and G. Pavlou, "In-network cache management and resource allocation for information-centric networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2920–2931, Nov 2014.

[20] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ser. ICN '12. New York, NY, USA: ACM, 2012, pp. 55–60.

[21] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing," Telecom ParisTech, Tech. Rep., 2011.

[22] L. Saino, I. Psaras, and G. Pavlou, "Icarus: a caching simulator for information centric networking (icn)," in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS '14. ICST, Brussels, Belgium, Belgium: ICST, 2014.

[23] V. Sourlas, P. Flegkas, and L. Tassiulas, "A novel cache aware routing scheme for information-centric networks," *Computer Networks*, vol. 59, pp. 44 – 61, 2014.

[24] C. Tsilopoulos and G. Xylomenos, "Supporting diverse traffic types in information centric networks," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '11. New York, NY, USA: ACM, 2011, pp. 13–18.

[25] J. M. Wang, J. Zhang, and B. Bensaou, "Intra-as cooperative caching for content-centric networks," in *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ser. ICN '13. New York, NY, USA: ACM, 2013, pp. 61–66.

[26] K.-Y. Wong, "Web cache replacement policies: a pragmatic approach," *IEEE Network*, vol. 20, no. 1, pp. 28–34, Jan 2006.

[27] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1024–1049, Second 2014.

[28] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128 – 3141, 2013, information Centric Networking.