Performance Comparison of Caching Strategies for Information-Centric IoT

Jakob Pfender
Sch of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
jakob.pfender@ecs.vuw.ac.nz

Alvin Valera
Sch of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
alvin.valera@ecs.vuw.ac.nz

Winston K.G. Seah
Sch of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
winston.seah@ecs.vuw.ac.nz

ABSTRACT

In-network caching is one of the most defining aspects of Information-Centric Networking (ICN). It ensures that relevant content is readily available across the network, even if the original producer is not reachable. However, in the Internet of Things (IoT), where memory is often severely limited, nodes cannot simply cache any and all content they receive, necessitating an increased reliance on caching strategies that offer heuristics on when to cache incoming content and which cached content to replace when the cache is full. In this paper, we discuss several existing ICN caching and cache replacement strategies as well as metrics suitable for evaluating them in an IoT context. We then evaluate multiple different strategies using IoT devices in a large testbed. Our experimental results show that simple stateless caching policies can perform equally well or sometimes even better than other, more complex schemes. This result is encouraging as it implies that it is indeed possible to employ effective ICN caching even in resource-constrained IoT nodes. To the best of our knowledge, this paper is the first to perform such an evaluation using actual IoT hardware in a realistic deployment.

CCS CONCEPTS

• **Networks**; • **Computer systems organization → Embedded systems**;

KEYWORDS

Information-centric networking, IoT, caching

ACM Reference Format:
Jakob Pfender, Alvin Valera, and Winston K.G. Seah. 2018. Performance Comparison of Caching Strategies for Information-Centric IoT. In *ICN '18: 5th ACM Conference on Information-Centric Networking (ICN '18), September 21–23, 2018, Boston, MA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3267955.3267966

ICN '18, September 21–23, 2018, Boston, MA, USA

ACM ISBN 978-1-4503-5959-7/18/09…$15.00
https://doi.org/10.1145/3267955.3267966

1 INTRODUCTION

Information-Centric Networking (ICN) has been shown to have a lot of promise for IoT applications, thanks to its content-centric nature and slim network stack. However, one of the core tenets of traditional ICN, the automatic and indiscriminate caching of any and all received content, does not translate well to the IoT, where resources such as memory are often severely limited. While in traditional networks, it is conceivable to have dedicated caches with virtually unlimited storage space, or even to have each participating node cache all content it receives, in IoT, this is neither possible nor productive. Instead, consideration needs to be given to the question of what content and how much of it should be cached, how long it should be cached, and what content should be replaced once the cache inevitably fills up. While traditional ICN has attempted to answer these questions, most of the proposed solutions still rely on somewhat large caches and have been shown to be insufficient when memory is severely limited.

Although transferring ICN caching to the IoT comes with these limitations, it is still a promising avenue of research. Especially in a multihop setup, where packets have to traverse multiple lossy wireless links, on-path caching capabilities can help reduce network load caused by retransmissions. If a data transmission fails due to a lossy link but the data was cached on an intermediate node, subsequent retransmissions will only need to fetch the data from the caching node instead of the original producer, thus reducing the total volume of traffic caused by retransmissions while increasing the likelihood of successful transmission. Gündoğan *et al.* [20] show that in a multi-hop scenario, ICN with on-path caching can outperform IP-based protocols in terms of flow balance and goodput thanks to this feature. Thus, it is worthwhile to pursue the feasibility of caching strategies in information-centric IoT.

In recent years, considerable amounts of research have been conducted into adapting traditional ICN caching strategies to the specific needs of IoT. Several different approaches were developed to counteract the limited resources as well as take advantage of the wireless, ad-hoc nature of IoT networks. Broadly speaking, a caching strategy can be applied on two different aspects: it can affect the *caching decision*, i.e. whether or not an incoming content chunk is to be stored in the cache, or it can affect the *cache replacement decision*, i.e. which piece of cached content to replace if a new content chunk is to be placed in a cache that has reached its capacity. Most proposed caching strategies only affect either the caching decision or the cache replacement decision, with a few holistic approaches that take both into consideration.

To the best of our knowledge, the most extensive comparisons so far of different caching strategies specifically for the IoT were published by Hail *et al.* [24] and Meddeb *et al.* [28]; however, both of these works use a network simulator to simulate their experiments, whereas we make use of a physical sensor testbed with real IoT hardware operating under realistic conditions. The more complex caching approaches have also not yet seen a comprehensive comparison; the existing publications that propose these new strategies generally only compare their solution to one of the rudimentary strategies.

The key contribution of this paper is a performance comparison of several different classes of caching strategies, evaluated with real IoT hardware using several metrics that are relevant to the performance of ICN in the IoT. This evaluation is structured as follows:

- We provide an overview of existing caching strategies, both for traditional ICN as well as for information-centric IoT (Section 2)
- We identify a set of metrics to evaluate the performance of caching strategies in an IoT context (Section 3).
- We evaluate and compare the caching strategies introduced in Section 2 using the performance metrics discussed in Section 3 (Section 4).
- Finally, we present potential future research directions (Section 5).

2 CACHING STRATEGIES

Ever since Jacobson *et al.* proposed Content-Centric Networking (CCN) [25], the Cache Everything Everywhere (CEE)/Least Recently Used (LRU) approach introduced there has been scrutinised for its suitability for CCN, Named Data Networking (NDN) [37], and their ICN relatives [12, 32]. The research community soon reached the consensus that CEE/LRU is not the most optimal caching strategy [13, 29, 30]. This resulted in a wealth of new caching approaches being developed.

As mentioned in Section 1, there are two aspects of the caching process that caching strategies can affect: The *caching decision* and the *cache replacement decision*. Since most existing approaches focus on only one of these aspects, we will discuss each aspect individually and introduce some strategies that change the behaviour of the respective aspect. Table 1 shows an overview of the different families of strategies for caching decision and cache replacement, with representatives for each.

Layer	Family	Representative Strategy
Caching decision	Traditional	CEE/LCE [25, 37]
	Static probability	$Prob(p)$ [23, 38]
	Dynamic probability	ProbCache [29]
		pCASTING [23]
	Topology-based	Betw/EgoBetw [14]
	Cooperative	CINC [26]
	Off-path	Dräxler & Karl [16]
		Saha *et al.* [33]
Cache replacement	Traditional	Random Replacement (RR)
		Least Recently Used (LRU)
		LFU [34]
	Content-based	MDMR [21]

Table 1: Families of caching strategies

2.1 Caching decision

The caching decision needs to be made whenever an ICN node receives a content chunk it does not yet have in its Content Store (CS). It determines whether the new content chunk will be stored in the CS or discarded.

2.1.1 Cache Everything Everywhere (CEE). The most straightforward caching decision strategy, CEE (also known as Leave Copy Everywhere (LCE)) is the strategy that is used in traditional ICN [25, 37]. Nodes will attempt to cache every incoming content chunk that is not already in their CS. Since in traditional ICN, caches can generally be assumed to be relatively large, this strategy is viable and causes little overhead. Its main advantage is that it offers the fastest possible propagation of content through the network — any node that receives (or, in the case of IoT, overhears) content will immediately store it, ensuring rapid replication of all available content. However, especially in a strongly connected network, it also leads to high redundancy: every node caching every piece of content means that every CS in the network will look roughly the same. This is where the crux of the problem lies when we consider a network with severely limited caching capabilities.

We could interpret the totality of the many small caches in the network as one single, stratified cache. However, if every individual cache stores the same content, that means we are not using our caching capabilities to their full extent. Instead, it would be desirable to reduce overall redundancy by ensuring that different caches carry different contents, thus increasing the chance that content from any given producer will be replicated in at least one cache in the network. This is how we will define *diversity* in the remainder of this paper (see Section 3.5): The ratio of unique content objects in all caches to unique content producers. The more advanced caching strategies discussed below attempt to maximise this metric.

It should be noted that diversity is not necessarily desirable in every IoT scenario. In applications where the content request pattern is very skewed (e.g., using a Zipfian distribution with $\alpha > 1$) maximising diversity will actually hurt performance as cache resources are wasted on less popular content. However, the more uniform the distribution, the more beneficial cache diversity becomes, as a larger percentage of content objects are duplicated in the network. Since uniform request distributions are more common in IoT contexts [28, 31], we decided to emphasise diversity in our evaluation.

2.1.2 Probabilistic Caching, Static Probability. To increase cache diversity and decrease redundancy, the easiest solution is to simply introduce a certain probability that any given content chunk will not be cached. In the most straightforward version of this approach, we simply set an *a priori* probability p that a given node will store a given content chunk. Upon receipt of a new content chunk, the node generates a random number between 0 and 1. If the generated number is smaller than p, the content is stored in the cache; otherwise, it is forwarded without being cached. This has the obvious effect that not every content chunk is cached at every node, thus effectively increasing cache diversity across the network. An important implication is that more "popular" content — i.e., content that is requested and sent more frequently — has a

higher chance of being stored at more points in the network. More generally, if a node receives the same content chunk n times, that chunk's caching probability at that node is $1 - (1 - p)^n$ [38].

It is easy to see that probabilistic caching increases diversity and reduces redundancy across the network's caches. This means that on average, the number of unique content objects that have copies stored in the network is increased. It also inherently favours popular content, thus increasing the cache hit ratio (see Section 3.1). However, the complexity in getting the most out of probabilistic caching lies in choosing the best value for the caching probability p. It is intuitively clear that a lower p will result in lower cache redundancy; however, too low a value for p may result in underutilisation of available resources.

In approaches where p is set to a fixed value, the most commonly chosen value is $p = 0.5$ [23]; however, other probabilities, such as 0.7, 0.3, 0.1 and even 0.01 have also been considered [34]. It should be noted that CEE can be treated as a special case of probabilistic caching with $p = 1$. In previous studies, a lower p has been found to correlate with better performance [6, 23, 24, 29, 34, 38]; a definitive lower bound for p (i.e. a value for p at which the caching probability is too low to result in effective caching) has yet to be conclusively determined.

2.1.3 Probabilistic Caching, Dynamic Probability. Instead of defining an *a priori* caching probability that is the same for every caching decision at every node, we can design a technique that dynamically computes a caching probability for each individual node or even for each content chunk, based on available information, in order to adapt the caching behaviour to the state of the network. The strategies can be based purely on node-local information, such as the current contents of the cache or the node's battery levels; they can be based on properties of the incoming content chunk, such as its age, type, or producer; or they can be based on information from the wider network, such as the position of the caching node in the network topology or the cache contents of neighbouring nodes.

Hail *et al.* propose *pCASTING* [23], which uses local information for its probability calculation. It considers the freshness (age) of the content chunk as well as the node's battery level and its cache occupancy when calculating p. The calculated probability is directly proportional to the battery level, inversely proportional to the cache occupancy, and directly proportional to the content object's *residual freshness*, which is calculated from the object's time of creation, its age, and the time of reception. These three values are normalised to between 0 and 1 and weighted according to their desired relative importance. As a fully distributed approach, pCASTING has no communication overhead and requires only local information, at the expense of not being able to take larger-scale information about the network into account.

2.1.4 Other strategies. Many more caching decision strategies have been proposed, which can vary strongly in the type of information they consider and the way they arrive at a caching decision. The following section provides a brief overview of some of these strategies in order to shed light on the varied ways a caching decision can be made in ICN.

If we want the caching decision to be based on more than just local information, we can look at other factors, such as the network topology. Psaras *et al* propose *ProbCache* [29], which computes the caching probability of a given content chunk based on the distance between producer and consumer as well as the location of the caching node on the path. Chai *et al.* [14] propose *Betw* / *EgoBetw*, a caching scheme that also takes the topology of the network into account, but instead of using it to calculate a probability, simply caches content at the most central node (i.e. the node that is on the highest number of paths between pairs of nodes). Topology-based approaches have the advantage of offering a holistic approach to caching and — at least in theory — could allow us to cache data in the optimal location. However, in the IoT, topologies are often dynamic, which means that these approaches may incur too much overhead if the topological information needs to be updated constantly. Furthermore, topology-based strategies tend to result in an uneven utilisation of nodes, meaning that some nodes will have a significantly shorter battery life than others.

Other proposed strategies include those that use implicit instead of explicit coordination between nodes [26], as well as a whole class of strategies that utilise off-path caching — i.e. storing content not only in caches on the path but potentially also moving it to neighbouring nodes so as to achieve better cache spread and diversity [15, 16, 33, 35].

2.2 Cache replacement decision

The cache replacement decision needs to be made whenever a content chunk is to be stored in a CS that has reached its capacity. The cache replacement policy determines which cached content chunk is to be evicted in favour of the new content chunk. Regardless of whether the node uses the default CEE caching approach or one of the more sophisticated ones introduced above, it has to have some method of deciding which content objects to replace. It should also be noted that in general, the cache replacement policy cannot undo the decision to cache the new content chunk; in other words, once the caching decision has been made, the new content chunk is guaranteed to be stored and the evicted content chunk has to be one of the previously cached chunks.

It should be noted that considerably less research has gone into cache replacement policies as opposed to the caching decision. Some authors argue [36] that cache replacement in ICN should be performed as fast as possible and that it is thus more desirable to have a simple and fast cache replacement algorithm rather than an efficient but complex one. However, it remains to be seen whether this also holds true in IoT, where caches are more valuable resources and should be used as efficiently as possible.

2.2.1 LRU. The most widely used cache replacement policy and common in several other domains, LRU likely needs no further introduction. Each node keeps track of the Interests it serves and when it served them, and, when required to evict a content chunk, chooses the one that was least recently requested. The rationale behind this is that unpopular or outdated content will naturally be more likely to be removed, thus ensuring that only the freshest and most popular content is cached. In practice, this approach has been shown to be effective, although its performance can degrade if Interest patterns are cyclical and the number of popular content objects is greater than the size of the caches — an effect commonly referred to as *thrashing*.

As a variation on LRU, *Least Frequently Used (LFU)* keeps track of how often the objects in the cache are requested and evicts the least popular ones. In other words, where LRU records *when* content is requested, LFU records *how often* it is requested. While LFU mostly avoids thrashing and has been shown to result in good cache diversity [13], it performs poorly with variable access patterns, as it tends to overly favour items that experience spikes in Interest frequencies. This will eventually result in higher server load, as caches accumulate stale items instead of storing fresher content, forcing more Interests to be routed to the original producers [34].

2.2.2 Random Replacement (RR). The least complex of the basic cache replacement policies, RR simply evicts a random content objects every time it needs to replace a chunk. It tends to be used more as a benchmark for evaluating other replacement policies than as an actual policy.

2.2.3 Max Diversity Most Recent (MDMR). MDMR, developed by Hahm *et al.* [21] specifically for information-centric IoT, is a cache replacement strategy that, as the name implies, aims to maximise the content diversity of individual caches while also keeping them up to date. It uses the original producers of content chunks to achieve this (it is assumed that chunk names include their original producers). When a new content chunk is to replace an old one, MDMR first attempts to remove an older cached chunk from the same producer. If the new chunk is from a previously unknown producer, MDMR attempts to remove the oldest chunk from a producer with more than one chunk in the cache. If the cache contains exactly one chunk per producer, it simply replaces the oldest chunk. Hahm *et al.* show that MDMR achieves better cache diversity on average than RR.

2.3 Caching in information-centric IoT

The usefulness of ICN paradigms in the domain of wireless sensor networks (WSNs) and IoT has been widely agreed upon [2–4, 9, 19, 27, 39, 40]. However, the idiosyncrasies of IoT — wireless communication, limited resources, transient contents — call into question whether the approaches that have been shown to be successful in traditional ICN can simply be transferred over to this new domain or whether new solutions are necessary. While caching is not the only aspect of ICN affected by this, it is one of the most prominent since limited memory is one of the most obvious hurdles for any IoT application. Other IoT-specific challenges include mobility, dynamic topologies, unreliable communications, and energy efficiency [2, 5, 22, 39, 40].

Zhang *et al.* [36], Fang *et al.* [17], and Zhang *et al.* [38] provide comprehensive surveys and taxonomies for the major classes of caching strategies in traditional ICN, as well as presenting challenges and potential future research directions. Tarnoi *et al.* [34] and Zhang *et al.* [38] present comparative evaluations of several different caching approaches for traditional ICN. While these contributions are extremely valuable, it is not a given that the results can be applied directly to IoT environments. The most comprehensive survey of caching schemes for information-centric IoT is presented by Arshad *et al.* [6], which provided the inspiration for this paper. Their paper, however, is a pure survey, with no experimental evaluation or comparison of the presented strategies.

So far the only comparative studies on ICN caching strategies specifically in the IoT were carried out by Hail *et al.* [24] and Meddeb *et al.* [28], both of whom use simulated environments for their evaluations. Hail *et al.* compared all four permutations of the CEE and probabilistic (with $p = 0.5$) cache decision policies and the RR and LRU cache replacement policies. The performance metrics measured in their experiments were cache hit ratio, data retrieval delay, and Interest retransmissions. Meddeb *et al.* compared CEE, Leave Copy Down (LCD), ProbCache, Btw [14], Edge Caching [18], and their own Consumer-Cache policy in combination with the RR, LRU, LFU, and First In First Out (FIFO) cache replacement strategies and evaluated them in regards to cache hit ratio, number of evictions, hop reduction ratio, and data retrieval delay.

While our paper is inspired by this previous research, we performed all of our evaluations on physical hardware as used in typical IoT deployments. Although this reduces reproducibility somewhat, it makes the results more meaningful in regards to applicability in the real world — and as we will see in Section 4, the results obtained through real-world experiments do in fact differ from those obtained through simulation in several aspects.

3 PERFORMANCE METRICS

In the existing literature, there is a wide range of metrics that have been used to evaluate the performance of caching strategies. In this section, we will discuss the ones we have selected for our comparison of caching strategies.

3.1 Server load and cache hit ratio

In ICN, a *cache hit* occurs whenever an Interest is served by a cache in the network instead of the requested content's original producer. Conversely, a *server hit* is when the Interest needs to travel all the way to the original producer. The dual metrics of *server load* and *cache hit ratio* [14, 23, 34, 36, 38] measure the percentage of Interests that result in server hits or cache hits respectively. Thus, these two metrics give an indication of how well popular content is distributed across the network — the higher the cache hit ratio, the lower the server load and thus the lower the strain on content producers.

3.2 Data retrieval delay

The *data retrieval delay* [23] measures the average time it takes for an Interest to be satisfied, including possible retransmissions. Retrieval delay may be affected by several caching-independent factors, such as network congestion and density, but it is also affected by cache diversity — the better the average availability of content chunks across the network, the lower the retrieval delay. If we assume that congestion and density stay roughly the same, we can use the retrieval delay to evaluate a caching strategy's effectiveness in making content quickly available.

3.3 Interest retransmission ratio

The *Interest retransmission ratio* [6] measures the percentage of Interests that are retransmitted due to having timed out. This metric is related to data retrieval delay in that it is affected by congestion and density, but also the efficiency of the caching strategy. An ideal caching strategy would result in cache diversity that is high enough for Interests to always be satisfied upon initial transmission.

It should be noted that in ICN with on-path caching, as explained in Section 1, a retransmission does not necessarily imply that the data needs to be re-fetched from the same node it was originally sent from. If the data was transmitted part of the way before being lost, it may be cached in an intermediate node, thus decreasing the time required to obtain it. In other words, the amount of additional load and delay caused by a given retransmission varies depending on the state of the caches on the path, and subsequent retransmissions are likely to be less costly.

3.4 Total cache evictions

The number of *total cache evictions* [29, 33] is an indicator for how well the caching strategy is able to adapt the cached contents to the popularity and propagation of content in the network. If a caching strategy results in thrashing (see Section 2.2.1), this will be evident in an increased number of cache evictions. Conversely, if a caching strategy is able to effectively distribute contents within the network such that they can satisfy as many Interests as possible, cache evictions will be rarer.

3.5 Diversity Metric

The *Diversity Metric (DM)* [21] is a measure that indicates the diversity of cache contents across the network. Denoted by D, it is calculated as follows:

$$D = \frac{|C_{disj}|}{|S|},$$

where $|S|$ is the number of content producers and $|C_{disj}|$ is the number of disjoint name prefixes in all caches — the prefix, in this case, being an identifier for the producer of a content chunk. In other words, DM measures the percentage of content producers that are represented in caches in the network at any given time. Ideally, if cache diversity is a desired goal in a given application, we want DM to approach 1, meaning that every content producer has at least one content object stored somewhere in the network.

While DM can be a useful metric in scenarios where diversity is a priority, it only measures the diversity in terms of *producers*, not in terms of actual content. In a real network, it is possible for some content producers to be much more prolific than others, producing significantly more unique and distinct content objects than others. In this case, DM would not be able to capture how well this producer's contents are distributed across the network as opposed to the remaining content. In other words, we also need a content-level diversity metric.

3.6 Cache Retention Ratio

The *Cache Retention Ratio (CRR)* [33] is a diversity metric that works on the content level. As such, it complements DM by measuring the ratio of distinct objects that are stored in caches at a given time to all generated objects; denoted by C, CRR is given by:

$$C = \frac{D_q}{D_p},$$

where D_q is the number of unique objects currently stored in at least one cache and D_p is the total number of unique objects generated during the network's lifetime. It should be noted here that since cache size is limited, CRR will obviously decrease over time as new content objects are constantly being created while cache capacity stays the same, meaning objects will eventually vanish from the network entirely. This is expected; however, in terms of comparing caching strategies, it is interesting to examine how fast CRR deteriorates — in other words, how effective the caching strategy is at keeping content objects available in the network for as long as possible.

4 EVALUATION

In this section, we present the main contribution of the paper — a comprehensive comparison and evaluation of several classes of caching strategies for information-centric ICN. For this comparison, we ran a series of experiments on the **FIT IoT-LAB** [1] open testbed (using the *Strasbourg* site). We used IoT-LAB's specially developed **M3 node**[1], which has an STM32 (ARM Cortex M3) microcontroller with 512 kB ROM and 64 kB RAM and an Atmel AT86RF231 [7] 2.4 GHz transceiver operating on IEEE 802.15.4, as our IoT hardware. As firmware for the nodes, we use a simple **RIOT-OS** [8] application using **CCN-lite**[2] as the ICN implementation, modified to support the different caching strategies. ICN cache sizes are set to hold up to 20 objects, with all content objects produced in the experiment having uniform size.

The experiment setup consists of 60 M3 nodes distributed evenly across a single building. The transmission range of individual nodes is not large enough to reach all other nodes in the network, meaning that we have a multihop setup, with paths being two to three hops in length on average. The experiment is managed by a control script using the IoT-LAB API, which has access to all node caches, outputs, and inputs. All content a given node produces is prefixed with that node's unique ID. In an initial setup phase, all nodes broadcast their own prefixes. Nodes record the prefix announcements they receive from their neighbours along with those neighbours' hardware addresses and the hop count value (which is initially 0) in their Fowarding Information Bases (FIBs). All prefixes that resulted in new FIB entries are then rebroadcast with an increased hop count value. When forwarding Interests, nodes choose the corresponding FIB entry with the lowest hop count, with broadcast as a fallback.

After setup is completed, all nodes begin producing random content chunks in a random interval of 1 to 5 seconds. The names of all produced chunks are collected by the experiment controller. The controller instructs each network participant to request a random piece of existing content every second (with up to ±500 ms of random jitter). The frequencies for publishing and requesting data were chosen as sensible maxima, as at higher frequencies,

[1] https://github.com/iot-lab/iot-lab/wiki/Hardware_M3-node
[2] https://github.com/cn-uofbasel/ccn-lite

performance tended to degrade in our experiment setup and results became inconclusive. A more in-depth study of the effects of high-frequency content creation and consumption on the various performance metrics may prove useful in future research.

The distribution of the content requests can have two modes: They can be uniformly random or follow a Zipf-like [11] pattern (with an α parameter of 1.0), which results in a skewed distribution. The Zipf distribution is commonly used in traditional ICN research because web content generally follows this pattern [18]. However, IoT content requests are very different from web content; they depend on the application but generally tend to approximate uniform distribution [28, 31]. Therefore, we regard these two modes as the edge cases of request distribution, with real-world patterns likely to fall somewhere between the two. Thus, by examining both modes we cover the range of possible content distributions in our experiments.

Each experiment runs for a total of 20 minutes, during which content is continuously produced and requested at the frequencies specified above. We repeat each experiment 100 times.

The controller takes snapshots of cache contents every 5 seconds and logs all cache and server hits, cache evictions, and Interest packet retransmissions. We use this data to evaluate the caching strategies according to the performance metrics introduced in Section 3.

IoT is a large space, and the number of potential deployment scenarios is much too high to be reflected in a single experiment. Therefore, instead of focusing on a specific application, we designed a generic experiment setup that includes common elements likely to be found in many IoT deployments — class 2 constrained devices [10], a low-power and lossy network, and multihop machine-to-machine communication. We do not claim universal applicability of the results to any IoT use case; rather, the evaluation presented here is intended to serve as a baseline performance comparison using a variety of metrics we believe to be useful in estimating the viability of a given caching strategy in a given scenario.

In the following sections, we will evaluate the two aspects of caching strategies — the caching decision and the cache replacement decision — independently. In Section 4.1, we compare caching decision policies by fixing the cache replacement policy to LRU, while in Section 4.2, we compare cache replacement policies by fixing the caching decision policy to CEE.

4.1 Caching decision policies

We evaluated the three caching decision policies CEE, *Prob*(p), and pCASTING, using the most common cache replacement policy, LRU. The results are presented in Fig. 1.

Recall that in Section 2.1.3, we explained that pCASTING can take into account three factors — the freshness of the incoming content, the node's cache occupancy, and the node's battery level — when calculating the caching probability. However, since the nodes in the IoT-LAB testbed have access to a constant power source, we decided to ignore the battery level and considered only cache occupancy and content freshness.

4.1.1 Server load. Fig. 1a shows the average server load — i.e. the percentage of Interests that were transmitted all the way to the original producer instead of being satisfied by an in-network cache — for the different caching decision strategies, separated by request distribution.

It is relatively intuitive that a probabilistic caching policy would result in a higher server load, since the probability that a given content chunk can be found in a content cache is lower if content is not guaranteed to be cached. pCASTING causes less server load than the static probability approach, implying that on average, the dynamic caching probability calculated by its heuristic is higher than $p = 0.5$.

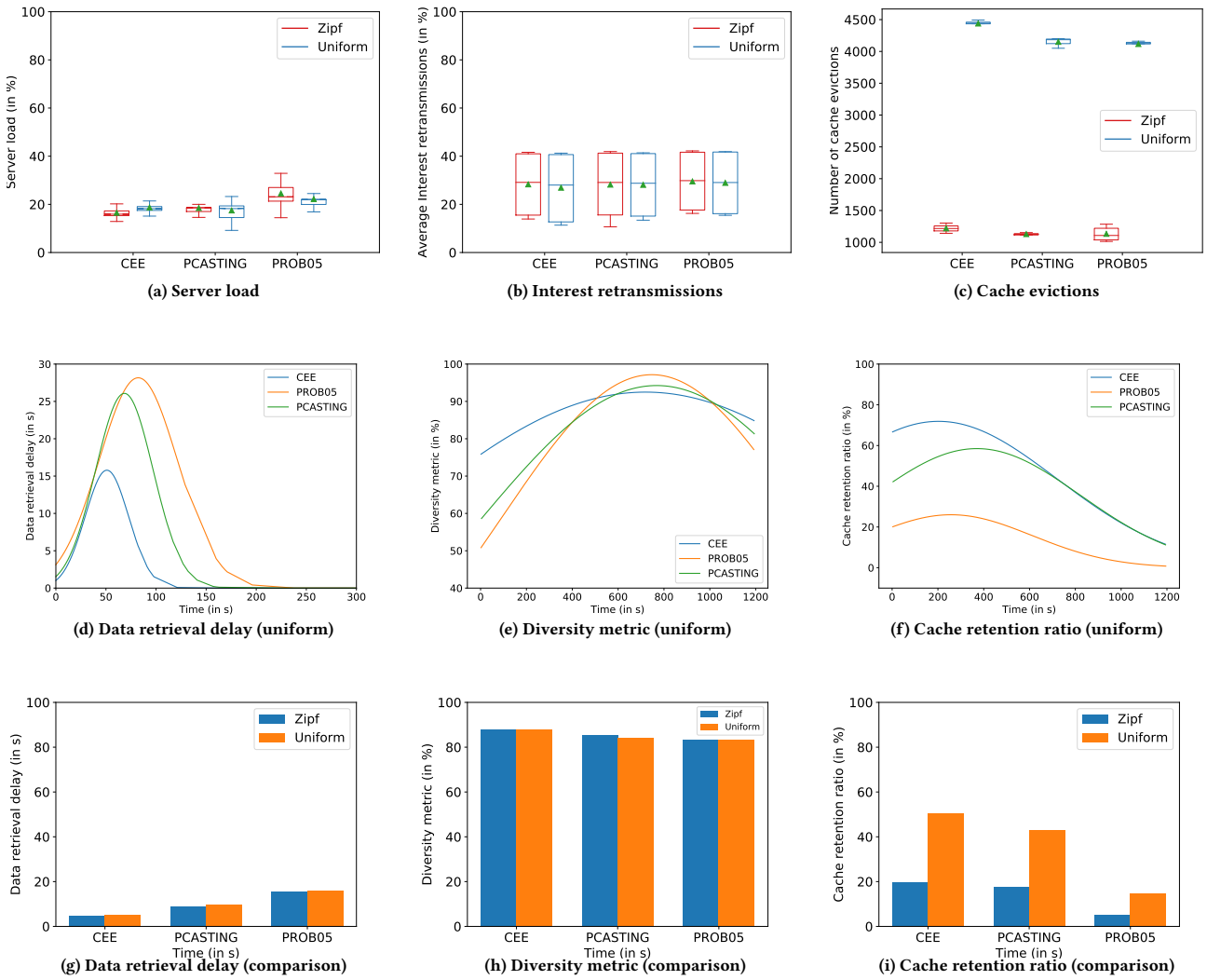
Differences between Zipfian and uniformly random request distributions are minor; when using the uniform distribution, we can see that server load is slightly higher for CEE and slightly lower for the other strategies. The reason CEE performs better for the Zipfian distribution is due to the skew in requests; since every satisfied request results in the corresponding data being stored along the request path, a distribution in which some content is much more likely to be requested than other content makes it more likely for that content to be found in a cache. Due to cache sizes being limited, a uniform request distribution reduces this likelihood. On the other hand, probabilistic caching, while decreasing the overall likelihood of a given content chunk being stored, evens out the caching probabilities between less and more popular content by increasing cache diversity, which has an averse effect on server load given a Zipfian distribution.

4.1.2 Interest retransmission ratio. The results shown in Fig. 1b make it fairly clear that different caching decision strategies have no statistically significant effect on the percentage of Interests that have to be retransmitted. It is likely that the Interest retransmission ratio is influenced more strongly by factors in the network itself, such as topology and congestion. Interestingly, this means that another possible conclusion to be drawn from this is that caching policies have no measurable effect on network congestion.

It should be noted that these results conflict with the results reported by Hail *et al.* [24], who found more significant differences in average Interest retransmissions between different caching strategies. However, these differences may also be attributed to differences in experiment setup — their experiment features a less dense network with fewer nodes, where not all nodes act as consumers or producers, some nodes being pure relay nodes instead. It is possible that with increased reliance on multi-hop communications, differences in Interest retransmission rates become more noticeable. Hail *et al.*'s experiments were also carried out in a network simulator instead of a physical testbed, which might indicate that the influence of physical effects such as signals interference were not adequately modelled in the simulation.

4.1.3 Cache evictions. The number of cache evictions is correlated with the rate at which content objects fill the cache. This can be observed when considering the results shown in Fig. 1c, although the differences are minor. The CEE strategy, which on average caches more packets than the probabilistic approaches, also exhibits a higher number of cache evictions.

Unsurprisingly, this metric exhibits a clear distinction between the two request patterns. Due to the skew in the Zipfian distribution, caches will accumulate the most popular content, reducing the likelihood of thrashing when compared to the uniform distribution.



(a) Server load (b) Interest retransmissions (c) Cache evictions

(d) Data retrieval delay (uniform) (e) Diversity metric (uniform) (f) Cache retention ratio (uniform)

(g) Data retrieval delay (comparison) (h) Diversity metric (comparison) (i) Cache retention ratio (comparison)

Figure 1: Comparison of caching decision policies using the LRU cache replacement strategy

The number of cache evictions should also be examined in the context of total traffic produced in the network. In an average experiment run, the number of packets, including retransmissions, that are produced by all nodes varies between 40,000 and 75,000, while cache evictions peak at around 1250 or 4500 depending on request pattern. In other words, for the uniform distribution, between 5% and 12% of packets result in cache evictions, while for the Zipfian distribution, this number is between 1% and 4%. It should also be noted that the number of evictions is heavily dependant on cache size. The caches used in our experiments hold up to 20 content objects; if memory is more constrained and/or content objects are larger, the rate of cache evictions will increase.

4.1.4 Data retrieval delay. Figs. 1d and 1g show the average data retrieval delay — i.e. the average round trip time between the original transmission of an Interest and its satisfaction by the corresponding content, including any retransmissions — as a time series and averaged over the first quarter of the experiment duration respectively.

At the beginning of the experiment, the latency is obviously orders of magnitude larger than after the caches have filled up, simply because any new content object can only ever be found at its original source until it has been requested at least once.

Unsurprisingly, the probabilistic approaches have much higher peaks than the CEE strategy, since they make it less likely, especially at the beginning of an experiment, for a content chunk to be found at a given cache in the network, thus necessitating more direct requests to sources. For the same reason, the average delay in the probabilistic approaches also declines at a slower rate, since caches take longer to fill.

Note that the graph in Fig. 1d is scaled to show only the first quarter of the experiment's duration, since there is no significant change in the average retrieval delay after about 200 seconds.

To cut down on visual noise, the time series in Figs. 1d to 1f show only the measurements taken with the uniformly random request distribution, as this distribution is closer to expected request patterns in a real IoT scenario. Figs. 1g to 1i respectively show the comparisons between the uniform and the Zipfian distributions. For the data retrieval delay, there is no significant difference between the distributions.

4.1.5 Diversity metric. Figs. 1e and 1h show the Diversity Metric (DM) as a time series and as an average for the different caching decision strategies. As explained in Section 3.5, DM indicates the diversity of cache contents in all caches across the network by calculating the ratio of disjoint name prefixes in all caches to the number of distinct content producers. It therefore shows the percentage of content producers whose content is cached somewhere in the network at a given time.

Overall, we can see that the majority of content producers are represented in the network at any given time. However, the caching strategies differ in both the rate of distribution and the value at which they peak. CEE starts off at a higher diversity than the probabilistic strategies, since it caches all content objects from the start, thus filling its caches more quickly. However, as the caches fill up, the probabilistic strategies are actually able to surpass the diversity of CEE. This is due to the fact that CEE, while ensuring caches are used to their full capacity, leads to high redundancy — all caches in the network will have very similar contents since they cache every object they encounter. The probabilistic methods, on the other hand, result in more diverse caches, thus increasing the probability of all content producers being represented in the network to almost 100%, a peak that is not reached by CEE.

4.1.6 Cache retention ratio. As noted in Section 3.6, CRR naturally decreases over time as new content objects are constantly being created while cache size stays the same, meaning content chunks will inevitably fade out of the network entirely after a finite time. However, as we can see in Fig. 1f, the rate at which CRR decays as well as its starting value vary depending on the caching strategy.

If we use a probabilistic caching policy, caches take longer to fill up and peak at a lower value than the CEE approach; this is simply due to the fact that the rate of content creation is the same in all experiment runs, but the probabilistic approaches take longer to accumulate data, meaning that by the time they have reached cache saturation, the total number of content objects created is already greater than what the caches can hold. The static probabilistic approach exhibits a smoother rate of decline; since fewer cache evictions take place (see Section 4.1.3), content has a longer average lifetime.

It is here in particular that we can see a significant difference between the probabilistic caching with static $p = 0.5$ and pCASTING. At the beginning of the experiment, pCASTING starts at a higher peak than $p = 0.5$ and its CRR convergence is more similar to that of CEE than $p = 0.5$. This is an effect of the different factors that go into the decision made by pCASTING. Since it takes into account both the freshness of the incoming content chunk as well as the cache occupancy (see Section 4.1), it acts differently when the cache is empty than when it is full. With an empty cache, the caching probability is much higher, as the corresponding part of the decision is inversely proportional to cache occupancy. This means that at the beginning of the experiment, pCASTING will act more like CEE or $p = 0.9$ until the caches fill up, at which point the probability will drop.

CRR also exhibits a clear distinction between Zipfian and uniform request patterns. The strong skew of the Zipfian distribution means that less popular content will fade much quicker while popular content is retained, making caches more homogeneous over time. A uniform distribution ensures similar lifespans for all content chunks, thus increasing the average CRR.

4.2 Cache replacement policies

We evaluated the three cache replacement policies LRU, MDMR, and RR using the most common caching decision policy, CEE. The results are presented in Fig. 2.

4.2.1 Server load. Fig. 2a shows the average server load for the different cache replacement strategies. It is immediately obvious that all three cache replacement strategies result in very similar server load rates. We can thus conclude that the cache replacement strategy has a negligible effect on this metric. We can compare these results to those discussed in Section 4.1.1, where we found that the caching decision strategy did have a measurable effect on server load. Therefore, while it might not be the most conclusive metric, it is still valuable for distinguishing between caching decision strategies. It may also be worth investigating how much of an impact other factors, like network topology, congestion, traffic shape, etc., have on the server load rate.

Perhaps most importantly, we can see that RR, the simplest of the cache replacement strategies, does not perform any worse than the other approaches, which come with a much larger overhead. This supports the argument put forward by Zhang *et al.* [36] that complex heuristics in the cache replacement policy are not worth the effort and that RR meets all requirements with the added benefit of being simple and fast.

4.2.2 Interest retransmission ratio. The results shown in Fig. 2b confirm what we established in Section 4.1.2 — the caching policy appears to have no measurable effect on the average percentage of Interests that have to retransmitted. This is still a valuable result, however, since it shows that any variations in retransmission rates one might observe between different experiments most likely have causes independent from the chosen caching strategy.

4.2.3 Number of cache evictions. As noted in Section 4.1.3, the request distribution has a significant effect on the number of cache evictions. Furthermore, this metric is also affected by the cache replacement strategy. Interestingly, we found that the relative performances of the replacement strategies differed depending on the distribution pattern: Given a Zipfian distribution, RR outperforms the other approaches, whereas with a uniform distribution, MDMR produces the best results. MDMR was designed to maximise diversity (see Section 2.2.3) and was originally evaluated in a scenario with uniform request patterns [21]. As such, it is unsurprising to see it perform well in this scenario. Given a skewed request distribution, however, it is not able to outperform LRU. This is because caches in this scenario are already much more likely to hold more popular content, reducing the impact of cache-shaping strategies. The fact that RR performs better than the other approaches in the Zipfian

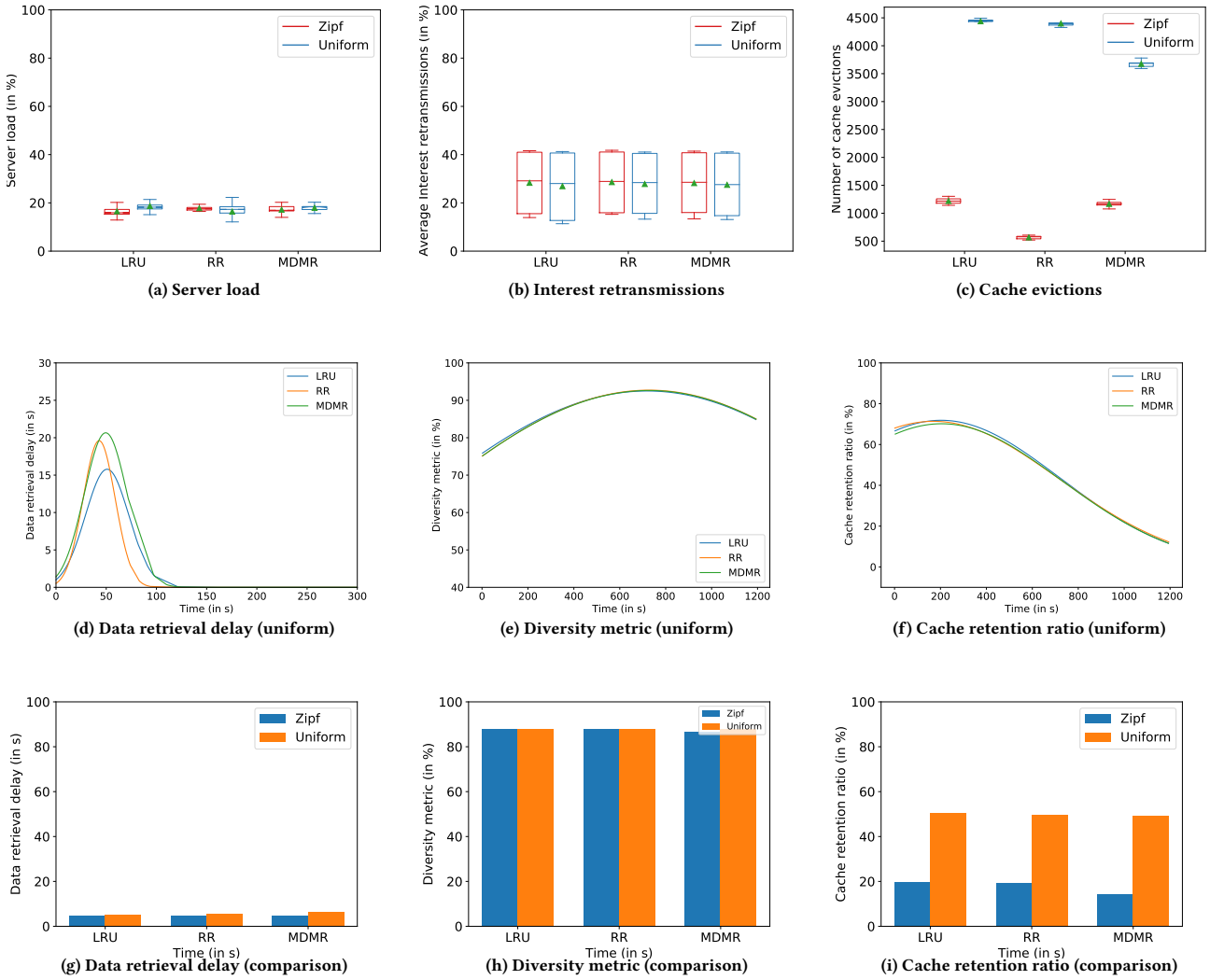


Figure 2: Comparison of cache replacement policies using the CEE caching decision strategy

scenario is surprising; it may be that there is still some thrashing at the tail of the Zipf distribution, where popularities even out, and that random replacement counteracts this effect.

4.2.4 Data retrieval delay. Figs. 2d and 2g show the average data retrieval delay as a time series using a uniform request pattern and averaged over the first quarter of the experiment using both request distributions. While the overall expectation for the average retrieval delay is the same for all strategies — a peak at the beginning followed by a gradual decline — the strategies differ in both the height of the peak and the rate of decline. LRU exhibits a lower initial peak in the retrieval delay, but converges as slowly as MDMR. RR's delay peaks similarly to MDMR but declines much faster.

Since the caching decision policy used is CEE, we can assume that caches fill up at the same rate on average. The reason MDMR exhibits a higher initial peak is most likely that content freshness, which is prioritised by MDMR, only becomes a significant factor after the experiment has run for a certain time and the ages of content objects begin to diverge. In other words, MDMR needs a minimum amount of time to become effective at making content available.

After a few minutes, the data retrieval delay stabilises at a very low value, ensuring timely delivery during the majority of the network's lifespan. The significance of the early spikes in retrieval delay is more in the strain that is put on individual devices, which becomes particularly important if the devices are battery-powered, putting data producers in greater danger of early failure.

4.2.5 Diversity Metric. It is immediately obvious from Figs. 2e and 2h that the cache replacement policy seems to have no measurable impact on the diversity metric. All policies follow roughly the same curve described in Section 4.1.5, which exhibits consistently high diversity, but never reaches the peak diversity the probabilistic approaches do.

It is not necessarily intuitive that the cache replacement policy would have no impact on the diversity metric. After all, the replacement policy directly affects cache contents, and some policies, such as MDMR, make it their intended goal to maximise diversity. These results are thus slightly surprising. However, it needs to be noted that MDMR was explicitly intended to maximise diversity in a scenario in which nodes would periodically sleep [21], so it may not be best suited for our experiment setup.

4.2.6 Cache retention ratio. From Fig. 2f, we can see that the CRRs for the cache replacement policies all follow the same pattern; what cache contents are replaced does not seem to have an impact on how long contents are retained in the network. As observed in Section 4.1.6, Fig. 2i shows that content fades significantly quicker given a Zipfian distribution.

5 CONCLUSIONS AND FUTURE WORK

We have presented a comparison and evaluation of different caching strategies for information-centric IoT. Our results show that it is likely not worth implementing overly complex cache replacement strategies, as simple stateless policies can perform equally well or perhaps even better. This confirms that the findings published by Zhang *et al.* [36] for traditional ICN also hold true for information-centric IoT, which is very encouraging as it means that effective caching can be achieved even when using resource-constrained IoT devices.

Identifying the ideal caching decision strategy for an IoT application depends on the requirements of that particular application as well as the available resources. Some applications may produce highly homogeneous data that needs to be disseminated as rapidly as possible, while others may prioritise maximising the diversity of cached content over fast response times. Depending on whether the devices are battery-driven or have access to a constant power source, alleviating strain on single nodes may be more or less important. Therefore, the contribution of this paper is not to offer a universal solution to the question of which caching approach is ideal for information-centric ICN, but rather to serve as a comprehensive overview of the benefits and shortcomings of the different approaches in relation to different metrics, not all of which may be of equal importance for any given application.

The results presented in this paper only scratch the surface of recent developments in caching in ICN in general and information-centric IoT in particular. Many more strategies than the ones evaluated here have been proposed, and research in this area is very active at the time of writing. There are also more potential performance metrics that may be used to evaluate these caching strategies, specifically ones that take into account topological factors such as hop reduction or other aspects of ICN, such as content popularity. A more comprehensive survey and evaluation of these techniques and performance metrics would be highly desirable, but would have far exceeded the scope of this paper.

Another potential research direction for further evaluation of caching strategies would be a comparison using multiple different experiment setups. IoT, being an extremely diverse field with many different applications varying in scale, hardware, available resources, mobility, and priorities, cannot realistically be represented by a single experiment configuration. Thus, it might be fruitful to assess the performance of promising caching strategies in several experiments representing common IoT use cases.

REFERENCES

[1] Cédric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noel, Roger Pissard-Gibollet, Frédéric Saint-Marcel, Guillaume Schreiner, Julien Vandaele, and others. 2015. FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed. In *Proceedings of the 2nd IEEE World Forum on Internet of Things (WF-IoT)*. Milan, Italy.

[2] Marica Amadeo, Claudia Campolo, Antonio Iera, and Antonella Molinaro. 2014. Named Data Networking for IoT: An Architectural Perspective. In *Proceedings of the European Conference on Networks and Communications (EuCNC)*. Bologna, Italy, 1–5.

[3] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. 2014. Multi-Source Data Retrieval in IoT Via Named Data Networking. In *Proceedings of the 1st International Conference on Information-Centric Networking (ICN)*. 24-26 Sep, Paris, France, 67–76.

[4] Marica Amadeo, Claudia Campolo, Antonella Molinaro, and Nathalie Mitton. 2013. Named Data Networking: A Natural Design for Data Collection in Wireless Sensor Networks. In *Proceedings of the IFIP Wireless Days (WD)*. Valencia, Spain, 1–6. http://ieeexplore.ieee.org/abstract/document/6686486/

[5] Marica Amadeo, Claudia Campolo, Antonella Molinaro, and Giuseppe Ruggeri. 2014. Content-Centric Wireless Networking: A Survey. *Computer Networks* 72 (2014), 1–13.

[6] Sobia Arshad, Muhammad Awais Azam, Mubashir Husain Rehmani, and Jonathan Loo. 2017. Information-Centric Networking Based Caching and Naming Schemes for Internet of Things: A Survey and Future Research Directions. *arXiv preprint arXiv:1710.03473* (2017).

[7] Atmel. 2009. AT86RF231 Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE, SP100, WirelessHART, and ISM Applications. (2009). http://www.atmel.com/images/doc8111.pdf

[8] Emmanuel Baccelli, Oliver Hahm, Mesut Günes, Matthias Wählisch, and Thomas C. Schmidt. 2013. RIOT OS: Towards an OS for the Internet of Things. In *Proceedings of the IEEE INFOCOM Workshops*. Turin, Italy, 79–80. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6970748

[9] Emmanuel Baccelli, Christian Mehlis, Oliver Hahm, Thomas C. Schmidt, and Matthias Wählisch. 2014. Information Centric Networking in the IoT: Experiments with NDN in the Wild. In *Proceedings of the 1st International Conference on Information-Centric Networking (ICN)*. Paris, France, 77–86. http://dl.acm.org/citation.cfm?id=2660144

[10] Carsten Bormann, Mehmet Ersue, and Ari Keranen. 2014. *Terminology for Constrained-Node Networks*. Technical Report.

[11] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. 1999. Web Caching and Zipf-Like Distributions: Evidence and Implications. In *Proceedings of the IEEEINFOCOM*. 21-25 Mar, 126–134.

[12] Giovanna Carofiglio, Massimo Gallo, Luca Muscariello, and Diego Perino. 2011. Modeling Data Transfer in Content-Centric Networking. In *Proceedings of the 23rd International Teletraffic Congress*. San Francisco, CA, USA, 111–118.

[13] Giovanna Carofiglio, Vinicius Gehlen, and Diego Perino. 2011. Experimental Evaluation of Memory Management in Content-Centric Networking. In *Proceedings of the IEEE International Conference on Communications (ICC)*. Kyoto, Japan, 1–6.

[14] Wei Koong Chai, Diliang He, Ioannis Psaras, and George Pavlou. 2013. Cache "Less for More" in Information-Centric Networks (Extended Version). *Computer Communications* 36, 7 (2013), 758–770.

[15] Raffaele Chiocchetti, Dario Rossi, Giuseppe Rossini, Giovanna Carofiglio, and Diego Perino. 2012. Exploit the Known or Explore the Unknown?: Hamlet-Like Doubts in ICN. In *Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking*. Helsinki, Finland, 7–12.

[16] Martin Dräxler and Holger Karl. 2012. Efficiency of on-Path and Off-Path Caching Strategies in Information Centric Networks. In *Proceedings of the IEEE International Conference on Green Computing and Communications (GreenCom)*. Besancon, France, 581–587.

[17] Chao Fang, F. Richard Yu, Tao Huang, Jiang Liu, and Yunjie Liu. 2014. A Survey of Energy-Efficient Caching in Information-Centric Networking. *IEEE Communications Magazine* 52, 11 (2014), 122–129.

[18] Seyed Kaveh Fayazbakhsh, Yin Lin, Amin Tootoonchian, Ali Ghodsi, Teemu Koponen, Bruce Maggs, K. C. Ng, Vyas Sekar, and Scott Shenker. 2013. Less Pain, Most of the Gain: Incrementally Deployable ICN. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 147–158.

[19] Nikos Fotiou and George C. Polyzos. 2014. Realizing the Internet of Things Using Information-Centric Networking. In *Proceedings of the 10th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine)*. Rhodes, Greece, 193–194.

[20] Cenk Gündoğan, Peter Kietzmann, Martine Lenders, Hauke Petersen, Thomas C. Schmidt, and Matthias Wählisch. 2018. NDN, CoAP, and MQTT: A Comparative

Measurement Study in the IoT. *arXiv:1806.01444 [cs]* (June 2018). http://arxiv.org/abs/1806.01444 arXiv: 1806.01444.

[21] Oliver Hahm, Emmanuel Baccelli, Thomas Schmidt, Matthias Wählisch, Cédric Adjih, and Laurent Massoulié. 2017. Low-Power Internet of Things with NDN & Cooperative Caching. In *Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN)*. Berlin, Germany.

[22] Oliver Hahm, Emmanuel Baccelli, Thomas C. Schmidt, Matthias Wahlisch, and Cédric Adjih. 2016. A Named Data Network Approach to Energy Efficiency in IoT. In *Proceedings of the IEEE Globecom Workshops (GC Wkshps)*. Washington, DC, USA, 1–6.

[23] Mohamed Ahmed Hail, Marica Amadeo, Antonella Molinaro, and Stefan Fischer. 2015. Caching in Named Data Networking for the Wireless Internet of Things. In *Proceedings of the International Conference on Recent Advances in Internet of Things (RIoT)*. Singapore, 1–6.

[24] Mohamed Ahmed M. Hail, Marica Amadeo, Antonella Molinaro, and Stefan Fischer. 2015. On the Performance of Caching and Forwarding in Information-Centric Networking for the IoT. In *Proceedings of the International Conference on Wired/Wireless Internet Communication (WWIC)*. Springer, Malaga, Spain, 313–326.

[25] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. 2009. Networking Named Content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. Rome, Italy, 1–12. http://dl.acm.org/citation.cfm?id=1658941

[26] Zhe Li and Gwendal Simon. 2011. Time-Shifted TV in Content Centric Networks: The Case for Cooperative In-Network Caching. In *Proceedings of the IEEE International Conference on Communications (ICC)*. Kyoto, Japan, 1–6.

[27] Anders Lindgren, Fehmi Ben Abdesslem, Bengt Ahlgren, Olov Schelén, and Adeel Mohammad Malik. 2016. Design Choices for the IoT in Information-Centric Networks. In *Proceedings of the 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. Las Vegas, NV, USA, 882–888.

[28] Maroua Meddeb, Amine Dhraief, Abdelfettah Belghith, Thierry Monteil, and Khalil Drira. 2017. How to Cache in ICN-Based IoT Environments?. In *IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*. IEEE, 1117–1124.

[29] Ioannis Psaras, Wei Koong Chai, and George Pavlou. 2012. Probabilistic In-Network Caching for Information-Centric Networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking*. Helsinki, Finland, 55–60.

[30] Ioannis Psaras, Richard G. Clegg, Raul Landa, Wei Koong Chai, and George Pavlou. 2011. Modelling and Evaluation of CCN-caching Trees. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking (NETWORKING)*. Valencia, Spain, 78–91. http://dl.acm.org/citation.cfm?id=2008780.2008789

[31] Akhila Rao, Olov Schelén, and Anders Lindgren. 2016. Performance Implications for IoT Over Information Centric Networks. In *Proceedings of the Eleventh ACM Workshop on Challenged Networks*. ACM, 57–62.

[32] Elisha J. Rosensweig, Jim Kurose, and Don Towsley. 2010. Approximate Models for General Cache Networks. In *Proceedings of the IEEE INFOCOM*. San Diego, CA, USA, 1–9.

[33] Sumanta Saha, Andrey Lukyanenko, and Antti Ylä-Jääski. 2013. Cooperative Caching Through Routing Control in Information-Centric Networks. In *Proceedings of the IEEE INFOCOM*. Turin, Italy, 100–104.

[34] Saran Tarnoi, Kalika Suksomboon, Wuttipong Kumwilaisak, and Yusheng Ji. 2014. Performance of Probabilistic Caching and Cache Replacement Policies for Content-Centric Networks. In *Proceedings of the IEEE 39th Conference on Local Computer Networks (LCN)*. Edmonton, AB, Canada, 99–106.

[35] Xenofon Vasilakos, Vasilios A. Siris, George C. Polyzos, and Marios Pomonis. 2012. Proactive Selective Neighbor Caching for Enhancing Mobility Support in Information-Centric Networks. In *Proceedings of the Second Edition of the Icn Workshop on Information-Centric Networking*. Helsinki, Finland, 61–66.

[36] Guoqiang Zhang, Yang Li, and Tao Lin. 2013. Caching in Information Centric Networking: A Survey. *Computer Networks* 57, 16 (2013), 3128–3141.

[37] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, and others. 2014. Named Data Networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 66–73. http://dl.acm.org/citation.cfm?id=2656887

[38] Meng Zhang, Hongbin Luo, and Hongke Zhang. 2015. A Survey of Caching Mechanisms in Information-Centric Networking. *IEEE Communications Surveys & Tutorials* 17, 3 (2015), 1473–1499.

[39] Yanyong Zhang, Dipankar Raychadhuri, Luigi Alfredo Grieco, Emmanuel Baccelli, Jeff Burke, Ravishankar Ravindran, Guoqiang Wang, Anders Lindgren, Bengt Ahlgren, and Olov Schelén. 2015. Requirements and Challenges for IoT over ICN. (2015). http://www.diva-portal.org/smash/record.jsf?pid=diva2:1043578

[40] Yanyong Zhang, Dipankar Raychadhuri, Ravi Ravindran, and G. Wang. 2013. ICN Based Architecture for IoT. *IRTF contribution, October* (2013).