Beyond Edge Caching: Freshness and Popularity Aware IoT Data Caching via NDN at Internet-Scale

Marica Amadeo, Claudia Campolo, Giuseppe Ruggeri, Antonella Molinaro

***Abstract*—In-network caching is one of the main pillars of the Named Data Networking (NDN) paradigm, where every Internet router, in the path between data sources and consumers, can cache incoming content packets. Multiple strategies have been designed for caching Internet of Things (IoT) data streamed by resource-constrained devices in edge domains and wireless sensor networks, while the benefits of IoT data caching at Internet-scale, including both edge and core network segments, have not been fully disclosed. In this work, we propose and analyse a novel probabilistic Internet-scale caching design for IoT data, which jointly accounts for the content popularity and lifetime. In the considered scenario, IoT contents are requested by remote consumers and delivered by crossing multiple edge and core network segments of the NDN-based future Internet. The proposal is composed of two distinct reactive caching strategies, a coordinated and an autonomous one, to be implemented in the edge and core domain, respectively. Achieved results show that the proposal outperforms state-of-the-art solutions by providing, among others, the highest cache hit ratio and the shortest number of hops. Such performances testify a lower pressure on energy-constrained devices and on the network infrastructure, overall contributing to the sustainability of the IoT ecosystem.**

***Index Terms*—Named Data Networking, Information Centric Networking, Caching, Edge computing, Internet of Things, Energy efficiency**

I. Introduction

The Internet of Things (IoT) aims at interconnecting billions of heterogeneous devices, including resource-constrained sensors that capture environmental and context parameters to feed multiple local and remote applications. Smart city, smart transportation, smart agriculture and smart building are among the most representative application domains that leverage IoT data streamed by devices massively deployed in the smart environments [1].

Compared to traditional Internet contents, such as multimedia files and web contents, IoT data show some distinctive features. First, many IoT sources like sensors implement low-power low-rate communication protocols and produce data with *very limited packet size*, e.g., a maximum payload of 104 bytes can be transferred by ZigBee devices [2]. Second, IoT data are usually short-lived and have a *transient* validity; they can be generated periodically and change over time like environmental parameters collected during the day. Last but not least, the same IoT data may be requested by multiple applications, hence, burdening the resource-constrained source devices and overwhelming the network with redundant data forwarding.

Notwithstanding the small IoT data size, still the massive data generation may largely affect the network performance in terms of congestion, latency and energy consumption, by mining the scalability and sustainability of the infrastructure. Today, IoT heavily relies on cloud computing facilities for long-term data storage and analytics [3]. Specifically, IoT sources upload their data in the cloud facilities, and IoT consumers, anywhere located, access the cloud to gather (processed) data. Therefore, a large portion of IoT data has to cross the Internet to be delivered from producers to consumers, generally located in different network access domains. Effectively and efficiently retrieving, distributing, storing (and possibly processing) the IoT data becomes mandatory across the heterogeneous network domains of the Internet, typically consisting of IoT sensor networks, edge and core domains.

Sustainability of the IoT ecosystem in presence of increasing data volumes and more demanding application requirements entails to re-engineer the network architecture.

In IP-based networks, data caching and processing is enabled in a limited number of specialized nodes, such as gateways, cloud and edge servers, while the Internet Protocol (IP) routers composing the Internet simply forward data packets, being unaware of the carried contents. On the contrary, by natively supporting name-based content delivery, Named Data Networking (NDN) [4] enables ubiquitous, Internet-scale, in-network caching by all routers, thus opening new caching opportunities for IoT applications. NDN has been recognized as a very effective information-centric network architecture for IoT [5] and Wireless Sensor Networks (WSNs) [6], and especially as an enabler of edge/fog computing [7], [8]. Available studies have demonstrated that caching IoT data via NDN in edge domain scenarios and multi-hop WSNs is extremely useful to reduce the retrieval latency. Moreover, by limiting the exchanged data traffic with IoT devices and reducing the amount of data needing to traverse the network, caching reduces energy consumption and carbon emissions, by contributing to green and sustainable networking [5], [9]–[13].

However, to cache or not to cache IoT data at Internet-scale, by using NDN in-network caching capability, is still an open question. Reserving caching resources for IoT data at the core NDN routers might seem useless, given the limited IoT data size and their transient validity. On the other hand, caching IoT data in the core network could effectively complement edge caching and further improve the network performance, while being fully aligned with the cloud-to-things continuum vision [14].

Marica Amadeo, Claudia Campolo, Giuseppe Ruggeri and Antonella Molinaro are with CNIT and the DIIES Department, University Mediterranea of Reggio Calabria, Via Graziella, Loc. Feo di Vito, 89100, Reggio Calabria, Italy (corresponding author's e-mail: claudia.campolo@unirc.it). Antonella Molinaro is also with CentraleSupélec/L2S, Université Paris-Saclay, France.

To the best of our knowledge, this is the first paper conducting a thorough analysis to answer the above research question, by providing the following main contributions:

- We discuss the peculiarities of IoT data generated within WSN domains and, by referring to current literature works, we identify the benefits and the open challenges of NDN caching.
- We propose a novel strategy for caching IoT data streamed by WSNs at Internet-scale, which is built upon two distinct policies applied, respectively, at the edge and the core network segments. They are light and reactive NDN-compliant schemes, both jointly accounting for content popularity and freshness.
- We evaluate the performance of our proposal through the official NDN simulator [15], when compared against several representative literature solutions, in terms of valuable metrics, such as cache hit ratio, content retrieval latency, number of hops, and under different realistic settings in terms of request patterns and loads. A storage and computational overhead analysis is also provided to assess the practicality of the proposal.

The remainder of the paper is organized as follows. In Section II, a short summary about NDN caching strategies conceived in the literature is provided. Section III examines the peculiarities of IoT data, by pinpointing the benefits brought by NDN to address the relevant challenges in their dissemination and caching. The main findings of the analysis are summarized in Section IV before describing the proposed caching strategy. Results of the conducted evaluation study are reported in Section V, before concluding in Section VI.

II. Caching in NDN

A. NDN: the Main Pillars

NDN is one of the most prominent Information-Centric Networking (ICN) instantiation. Communication in NDN is based on the exchange of two named packet types: the *Interest*, issued by *consumers* to request contents by name, and the *Data*, sent back by the original producers or any node caching a copy of them [4].

Three tables are maintained at the forwarding plane of NDN nodes: the Content Store (CS) caching incoming Data packets; the Pending Interest Table (PIT) tracking the forwarded Interests; and the Forwarding Information Base (FIB) storing the outgoing interface(s) per each known named prefix. At the Interest reception, the node first looks for a Data packet in the CS and, in case of success, it immediately sends it back. Otherwise, the node looks for a matching in the PIT and, in case of success, it updates the PIT entry with the identifier of the interface the Interest arrived from and discards the Interest. Otherwise, the node looks in the FIB to possibly forward the packet towards the data source.

At the reception of the Data packet, the NDN node performs a name-based lookup in the PIT table. If it finds the correspondent pending Interest, then it cancels the request and forwards the Data packet back to the consumer(s). Otherwise, the Data packet is assumed unsolicited and it is discarded. Therefore, by design, there is a strict one-to-one correspondence between Interests and Data packets: each incoming Data packet consumes exactly one pending Interest.

B. Caching Solutions for Internet Contents

NDN caching is reactive and performed on a per-packet basis. The reference caching and replacement schemes in NDN are the Cache Everything Everywhere (CEE) and the Least Recently Used (LRU) replacement strategy, respectively. However, a vast research activity has been carried out in recent years that proposes more effective caching strategies for Internet contents, ranging from autonomous probabilistic solutions to complex cooperative strategies with explicit signalling [16]. In Internet-scale scenarios with potentially heavy traffic load, autonomous caching schemes or solutions with implicit signalling are preferable, due to the line rate constraints [17], i.e., the caching decision must be performed very quickly when forwarding the Data packets.

The simplest autonomous caching scheme outperforming CEE is Random Caching (RC), where nodes cache incoming packets with a static probability p, with $0 \leq p < 1$ [18]. Other solutions take more judicious caching decisions by considering some node's attributes like the betweenness centrality [19], some content's attributes like the popularity [20], or a combination of them [21]. In particular, popularity is the most common parameter adopted in the caching literature, even outside the NDN scope. Indeed, caching the most requested contents reduces the load on the original producer, the network traffic and the retrieval delay [16]. A pioneering popularity-based caching scheme for NDN is the Most Popular Contents (MPC) strategy [20], which applies a popularity-based selection for caching the contents. It defines a fixed popularity threshold to distinguish between popular and non-popular contents and therefore, it caches only the packets belonging to the first category. The strategy is extended in [22] where the popularity threshold is adjusted every minute according to the frequency of received Interests and the available cache capacity. The proposal, called Fine-Grained Popularity based Caching (FGPC), further assumes that, if there is space in the content stores, also non popular contents are cached.

Conversely, the Betweenness and Edge Popularity based caching (BEP) scheme in [21] leverages three parameters for the caching decision: the betweenness centrality of nodes, the content popularity, and the cache size. In BEP, each Interest carries two additional fields: the content popularity of the requested content, included by the first edge node in the path between consumer and producer, and an array of betweenness centrality values, filled by all the on-path nodes. Based on this information, the strategy caches the most popular contents on the most central nodes, and the less popular packets in the less central nodes, by avoiding large amount of cache redundancy.

III. Caching IoT Data: What and Why

In principle, the NDN in-network caching strategies conceived for Internet contents and Internet-scale scenarios can be similarly applied to IoT contents and heterogeneous network environments, provided that each Data packet has a globally unique and recognisable name. For instance, the work in [23]

compares traditional schemes like CEE, RC, and the caching based on betweenness centrality against a consumer-cache strategy, which keeps content copies in the gateways directly connected to the consumers. The authors employ an IoT data set from a smart building and simulate a content catalog with 4000 items and 21 consumers requesting them according to a uniform distribution. Results show that consumer-cache outperforms the other schemes; however, the analysis does not focus on a peculiar feature of IoT data, that is their transiency, i.e., their limited time validity. Moreover, in the experimentation, the content popularity follows a uniform distribution, which could be not the case for some classes of IoT information.

Other works have started to extend existing caching approaches to the context of WSNs and IoT. For instance, the work in [24] proposes a caching strategy that takes into account the centrality of sensor nodes, like in [19], and their distance from the source of the contents, like in [25]. However, to effectively understand how to leverage NDN caching for IoT, it is crucial to identify the peculiarities and requirements of IoT data as streamed by WSNs, which are the main contributors of the IoT traffic. In the following, the key features of WSNs are first described, i.e., intermittent connectivity, energy constraints and limited capabilities asking for lightweight protocol stack and messages. Then, the two notable IoT content attributes, i.e., content transiency and popularity, are discussed.

A. Key Features of WSNs

Intermittent connectivity. The majority of IoT sources are wirelessly connected and, some of them, can be even mobile. This implies that devices can be intermittently reachable and transmissions are challenged by error-prone wireless links. In-network caching can largely cope with this issue.

In traditional IP-based design, purpose-built caching nodes like proxies and gateways are usually installed to manage the interactions between remote consumers and IoT data sources[1] [26]. Vice versa, any NDN node in the Internet, not only the gateway, natively implements in-network caching and may act as a caching proxy for other constrained devices.

In [27], a comprehensive comparison is presented between NDN and the main representative messaging protocols for IoT, i.e., Constrained Application Protocol (CoAP) and Message Queue Telemetry Transport (MQTT), by deploying an IoT testbed with wireless single-hop and multi-hop links. Results show that performance of the three protocols are similar in the single-hop scenario, while NDN outperforms the IP-based schemes in multi-hop scenarios, mainly thanks to the hop-by-hop caching strategy. Similar findings are provided by the work in [5], where experimental results are achieved in a campus testbed.

Energy constraints. Since many IoT sources are battery-powered devices, in-network caching can be a very useful mean to reduce the network energy consumption, even when transmitting small data. As discussed in [5], first, caching decreases the number of energy-challenged devices that are involved in the packet forwarding, and, second, it guarantees that contents are still available in the network even if their producers are in sleep mode. Therefore, resource-constrained nodes can potentially increase their sleeping period.

At the same time, it has been recognized that caching operations themselves incur a certain energy consumption and therefore, in presence of constrained devices, they should be either selectively performed or even completely inhibited [28]. Solutions that trade-off between the caching gain and the energy efficiency are reported in the literature. For instance, the PCASTING strategy in [10] leverages the battery level, among other parameters, as a metric for the probabilistic autonomous caching decision: nodes with a low battery level reduce the caching probability to save energy resources. Conversely, a cooperative strategy is reported in [11], where battery-powered devices periodically broadcast their data to the neighbours, which have to cache them. The target is to let only few IoT devices be active and provide the contents, while the others can sleep. By doing so, data are always available in the network but their producers are not forced to be awake all the time.

Lightweight protocol stack and messages. Besides energy consumption, wireless sensors generating IoT data have typically other limitations, e.g., in terms of computing and memory resources. Moreover, IoT access layer technologies are conceived to deliver small amounts of data in order to keep low the pressure on both network and device resources.

In NDN, caching is performed with a packet level granularity and each Data packet, even the small sized ones, must be uniquely named. Therefore, in principle, there are no technical challenges for caching atomic content units of a few bytes, as would be the case for data streamed by wireless sensor nodes. Of course, due to hardware limitation, not every IoT device will be able to cache Data. This is not an issue since caching in NDN is not mandatory: different nodes may apply different policies, according to their available resources, including not caching at all.

B. Peculiarities of IoT Contents

Popularity. The content popularity pattern can highly impact the benefits of caching. If the popularity is skewed, i.e., some contents are more requested than others, then caching the most popular contents can provide a high gain in terms of reduced load on the original producer, lower network traffic and shorter access delay. Vice versa, the caching gain diminishes if the content request distribution is uniform [29].

In [23], the authors argue that the request pattern of IoT contents likely follows a uniform distribution. Vice versa, by analyzing an IoT dataset and queries from search engines over a period of nine months, the authors in [30] find that the IoT popularity follows the Zipf's law [31] with a skewness parameter that can change on a regular basis. Such variations are due to the fact that requests for IoT data are related to the people's daily life and needs. The authors propose a distributed popularity-based caching algorithm, where caching nodes periodically leverage a Deep Neural Network (DNN) learning approach to predict the popularity of IoT contents

[1]In this paper we use the terms *source* and *producer* interchangeably, as well as *content* and *data*.

and prefetch the most popular ones. A Zipf distribution is also considered in [32], where the authors present an edge caching strategy for sensed vehicular traffic information.

We think that popularity should be taken into account as decision metric for IoT data caching, and expect that, as suggested in [33], the popularity pattern will likely fall between the uniform and the Zipf distributions.

Transient contents. Transiency is perhaps the most peculiar characteristic of many IoT data. On the one hand, simple sensors may generate data as a time series of readings, such as CO_2 emissions and temperature values, but also more complex devices may generate information that changes over time, e.g., cameras periodically capture and transmit pictures and videos about roads segments or building entrance. On the other hand, IoT applications are usually interested in fresh data, e.g., e-health services rely on highly fresh information to take proper decision in response to a change of vital parameters, such as blood pressure or blood oxygen level. Using stale measurements can lead to sub-optimal/wrong control decisions.

NDN recognizes transient contents thanks to the so-called *freshness period*, a field in the Data packet header, set by the source, indicating the lifetime of the content in seconds. When the freshness period expires, the packet is considered stale and the source may have produced a new data sample. The freshness period is treated like a timeout by the CS: when it expires, the data is considered invalid and dropped.

Several works have demonstrated that caching transient contents is still useful, even if their lifetime is of a few seconds [34], [35], [36]. They almost unanimously assume that the freshness must be considered in the caching decision in any network domain, ranging from WSNs to core segments. Of course, intuitively, reserving storage space for contents that are ready to expire could not be always convenient and some selective caching decisions should be implemented. In [10], the caching probability reduces with the freshness of the data, thus long-lasting contents have generally a higher caching probability than short-lasting ones. In [34] and [36], the caching strategy tries to trade off between data freshness and communication cost when fetching IoT data. The freshness cost increases as the Data packet moves away from the source towards the consumers, while the communication cost decreases in the same condition.

Our previous work in [35] jointly considers the content lifetime and its popularity for designing a caching strategy at the edge. Most popular data with the highest residual lifetime are cached in order to maximize the cache hit ratio in the domain. A similar approach is followed in [37], with the main difference that the study provides more theoretical insights in network topologies with a very few caching nodes. The work in [38] leverages the content popularity together with a move copy down [16] caching policy: when the content popularity is higher than a predefined threshold the content is cached in the downstream node, towards the consumer(s). In [39] a proactive Edge Linked Caching (ELC) strategy is proposed, where edge leaf nodes, on the path towards the energy constrained producers (e.g., sensors) and directly connected to the IoT data consumers, are in charge of caching data to reduce the interactions between the two entities. If the leaf node has to evict a data item that is still fresh then it pushes another edge node for caching it. By doing so, a copy of the content is always available at the edge without the need of accessing the resource-constrained device. However, this mechanism requires major modifications in the NDN architecture, since push-based/proactive caching is not natively enabled.

The freshness may also drive the replacement policy. In [40], the Least Fresh First (LFF) policy evicts invalid cached contents based on time series forecasting of sensors-related future events. Basically, LFF predicts the time a sensor will produce a new sample in order to estimate the residual lifetime of the content. Contents that are supposed to be replaced soon by new ones from the producer are discarded with higher priority. Of course, the calculation is trivial if the source generates contents periodically and advertises the freshness period (e.g., to feed monitoring applications with a specific sampling time). Vice versa, in case of event-triggered data transmissions, an Autoregressive Moving Average (ARMA) model is used for time series forecasting.

IV. Designing a Caching Strategy for IoT Data at Internet-scale

A. Motivations and Reference Scenario

As shown in the previous Section, the majority of NDN-based IoT caching strategies typically consider network domains with a limited geographical scope, such as WSNs [6], [10], [11], [41], or edge networks [5], [30], [35], [39].

Less attention has been devoted to those cases where IoT consumers and producers are located in different access domains, and the IoT data may have to cross an Internet-scale network to reach the remote consumers. In such contexts, enabling in-network caching can offer further opportunities compared to the cases in which only edge and cloud servers can cache data. The work in [40] considers only the freshness information as a caching metric at Internet scale, while the work in [38] is centered around the popularity metric. The proposal in [34], instead, considers both freshness and communication metrics, without targeting the content popularity. In this paper, we design an Internet-scale caching strategy driven by *both* IoT content popularity and freshness parameters, and aimed to minimize the interactions with constrained data sources.

Caching IoT data at Internet-scale would be useful in several IoT scenarios, such as smart grid, smart city, factory and building automation, healthcare [42], usually characterized by a variety of client applications running in remote data centers or in distributed user devices like smartphones and tablets. In general, caching valid contents can avoid that multiple requests reach the IoT sources, which are typically constrained devices implementing low-power communication protocols. Moreover, caching in the right places, possibly close to both the consumer and/or the producer, may significantly reduce the content retrieval latency and the network traffic congestion. This makes caching advantageous also for data-hungry low-latency IoT applications, e.g., in the context of intelligent driving.

Without loss of generality, the reference scenario considered in our design is an Internet-scale network composed of edge and core domains. As shown in Fig. 1, different edge domains can provide network access to the end devices (both IoT data consumers and IoT data sources), while a core network interconnects the edge domains. IoT data sources are constrained devices, modelled as sensors in WSNs, generating contents that have to reach the (remote) consumers, generally located in different access domains, therefore crossing the Internet-scale network, unless data is cached in a network node in-between the source and the consumer. All the routers in the considered scenario, i.e., both core and edge nodes, are mains powered devices that implement the NDN architecture and perform in-network caching.

Edge domains cover small-to-medium sized area, like college campuses, airports or, at most, a city area, with tens of edge nodes [43], [44], which manage a moderate amount of incoming traffic. Vice versa, core nodes, being located in central points of the Internet, are supposed to receive a generally higher amount of incoming traffic than edge nodes. Moreover, they usually have more incoming/outgoing links than edge nodes.

Given the distinctive features of the two network domains, we present a reactive caching design for IoT contents that consists of two distinct strategies, namely Caching in the Core Strategy (CCS) and Caching at the Edge Strategy (CES). This is a further novelty compared to the existing Internet-scale caching schemes for IoT contents [34], [38], [40], which are all characterized by a *one-size-fits all* solution. In practical scenarios, it will be the network operator, in charge of a given domain, to configure the appropriate policy in the NDN nodes of its domain.

Both CCS and CES are centered on content freshness and popularity metrics. To the best of our knowledge, in the scientific literature a formal coupling relationship between the IoT content popularity and lifetime cannot be found, and in any case the two parameters are reasonably loosely related. In fact, IoT contents can be highly heterogeneous and vary in terms of popularity and lifetime, with one of these features that does not necessarily affect the other one. Short or long lifetime does not necessarily imply more or less popularity of a given content. Some short-lived contents may be highly requested during their brief lifetime, but the contrary can be true as well. Similarly, long-lived contents could be not requested at all during their long lifetime, and the contrary can also hold. Although intuitively the two parameters are not tightly coupled, they must be jointly considered when taking a caching decision, in order to ensure that the most popular data with the highest residual lifetime are cached with higher probability. This is one of the novelties of our work, in fact *unlike most of the existing literature solutions, we do not consider popularity and lifetime individually as decision criteria, but jointly*. Therefore, each node has to individually monitor both metrics.

According to the freshness period, stale contents are evicted from the content store by following the native NDN replacement routine. In addition, compared to the standard NDN node architecture including the CS, PIT and FIB data structures,

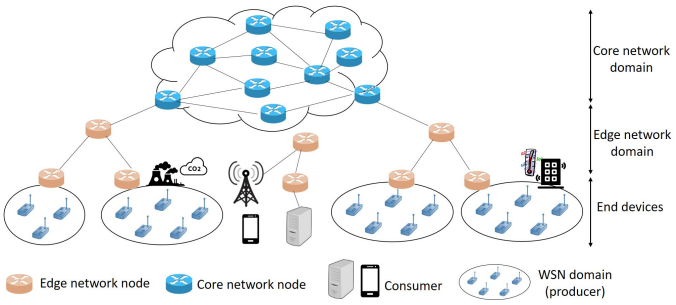


Fig. 1. Reference scenario.

both strategies require that NDN nodes maintain a Popularity Table. It tracks the requested contents and the related number of incoming Interest packets, together with additional variables taken as input for the caching decision, as it will be clarified in the following.

B. Caching in the Core Strategy

A low-complex and low-overhead caching strategy is to be implemented in the core nodes, which must take decision at line-speed in presence of a huge amount of data [17] generated by traditional Internet services and upcoming IoT applications. Therefore, we consider the implementation of an *autonomous probabilistic caching strategy* based on the popularity and freshness parameters, which can be locally estimated by each node from the received Interest and Data packets, without requiring any additional signalling exchange among nodes.

On the one hand, works like [45] demonstrated that online accurate popularity monitoring is possible even at line-speed. On the other hand, by accessing the freshness period field carried in the Data packet, each node may infer the content lifetime. Our target is caching at the core nodes *the most popular IoT contents which also have a long lifetime expectancy* (i.e., long-lived), in order to maximize the cache hit ratio. Indeed, given the high number of contents passing through the core nodes, there would be no gain in caching highly volatile contents that are rarely requested.

Popular contents are recognized according to a popularity threshold (P_{th}), which is periodically updated on a time window basis, T, e.g., set equal to a minute as in [22]. At each time interval T, the node stores in the Popularity Table the number of received Interests per each distinct content d_k, I_{d_k}, and computes the current average number of received Interests per content, $\bar{I}_{curr} = \frac{1}{N}\sum_{i=1}^{N} I_{d_i}$, with N being the number of distinct contents requested in the time interval. The popularity threshold P_{th} is defined as the average number of received Interests per content, and it is updated based on an Exponential Weighted Moving Average (EWMA), as follows:

$$P_{th} = (1-\gamma)P_{th,old} + \gamma\bar{I}_{curr}, \quad (1)$$

where $P_{th,old}$ is the previous value of the popularity threshold, and parameter $\gamma \in [0,1]$ is set to 0.125 to give more relevance to the historical values rather than to the instantaneous ones and to avoid large fluctuations in the result. If $I_{d_k} \geq P_{th}$ the content d_k is considered popular.

In addition, a freshness threshold (F_{th}) is defined to distinguish between long-lived and short-lived contents. F_{th} identifies the average freshness period of received IoT Data packets and it is dynamically updated, at every T interval, by the node through an EWMA, as follows:

$$F_{th} = (1-\gamma)F_{th,old} + \gamma \bar{F}_{curr}, \tag{2}$$

where $F_{th,old}$ is the previous value of the freshness threshold, and $\bar{F}_{curr}$ is the current average freshness period computed by considering the IoT Data packets received during T. A new received Data packet, d_k, whose freshness period, F_{d_k}, is higher than F_{th} is considered *long-lived*, vice versa it is *short-lived*.

Hence, according to content freshness and popularity, the caching probability of packet d_k, p_{c,d_k}, is derived as follows:

$$p_{c,d_k}(CCS) = \begin{cases} 1 & \text{if } I_{d_k} \geq P_{th} \quad \&\& \quad F_{d_k} \geq F_{th} \\ \dfrac{F_{d_k}}{F_{th}} & \text{if } I_{d_k} \geq P_{th} \quad \&\& \quad F_{d_k} < F_{th} \\ 0 & otherwise \end{cases} \tag{3}$$

Basically, popular long-lived Data packets are always cached, i.e., the caching probability is set equal to 1. Whereas, unpopular, either short-lived or long-lived, Data packets are never cached, i.e., the caching probability is set equal to 0. Popular short-lived contents are, instead, cached with a probability that is proportional to the ratio between their freshness period and the threshold. In particular, upon receipt of a new Data packet, d_k, the node generates a random number between 0 and 1. If the generated number is smaller than $p_{c,d_k}(CCS)$, the content is stored in the cache.

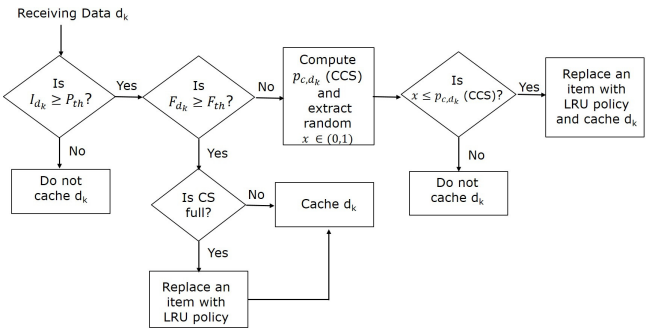
It is worth observing that, if contents are requested according to a uniform popularity distribution, the strategy is only driven by the freshness of contents, since the average number of received Interests, which drives the popularity threshold, would be almost the same for every content.

If the packet is to be cached but the CS if full, LRU is implemented as replacement policy in the core segment, given that it can guarantee good performance despite its extreme simplicity [18]. Fig. 2(a) summarizes the CCS workflow.

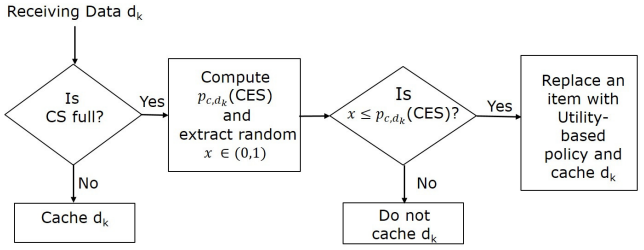
C. Caching at the Edge Strategy

Being closer to the IoT sources, edge nodes may have a crucial role in coping against intermittent connectivity and energy issues of resource-constrained devices. To reduce the number of requests reaching the WSNs, CES leverages the popularity metric to drive the caching decision strategy, while freshness and popularity are jointly used in the replacement policy, to compute an Utility Index estimating how useful would be keeping the packets in the content store during their lifetime. This is a novelty compared to other freshness-driven approaches, e.g., [10], [40], where the freshness is used only to identify and to possibly replace (or to avoid caching) the contents that are ready to expire.

In addition, CES aims at reducing the cache redundancy and therefore, at diversifying the contents cached at the edge in order to significantly limit the number of requests reaching the IoT data sources. The latter target is also achieved by



(a) CCS.



(b) CES.

Fig. 2. Workflow of the proposed strategies.

the edge caching strategy in [39], however, there, coordination is obtained with a pushing service that is not natively supported by NDN. Conversely, CES, is a light and reactive coordinated caching scheme that leverages an additional flag called CACHED, piggybacked in the Data packet header and initially set to *false* by the producer. The CACHED flag will be set to *true* by the first edge node that caches the packet.

Caching decision. At the reception of a Data packet d_k, the edge node connected to the IoT sources, in the following referred to as *leaf node*, checks if it has free storage space for it. If it is the case, it just caches the packet, otherwise it computes the caching probability as a function of the content popularity that is updated, like in CCS, on a time window basis.

A content d_k is cached with a probability defined as the ratio between the current number of received requests for d_k, I_{d_k}, and the maximum number of requests received for a content in the time window, both tracked in the Popularity Table:

$$p_{c,d_k}(CES) = \frac{I_{d_k}}{max_{\forall d_i}(I_{d_i})}. \tag{4}$$

Eq. (4) sorts contents according to their popularity as locally perceived by each node: $p_{c,d_k}(CES)$ equals 1 if d_k is the most requested content, whereas it is lower than 1 for contents receiving a lower number of requests.

Once the caching operation is performed by the leaf node, the CACHED flag is set to true, thus the following on-path nodes will simply forward the packet. Vice versa, if the leaf node has not cached the packet, then a parent node will try to cache it. In case of a positive outcome, the parent node will set the CACHED flag to true. Such a design choice, which attempts to achieve the caching of a given content into a single node only, has the aim of improving the utilization of precious

storage space, which can be leveraged to cache other contents, hence increasing content diversity.

Replacement. If the Data packet d_k is to be cached but the CS is full, then a stored packet is replaced according to a new policy based on the *Utility Index* of the cached items. For a cached packet d_j, the Utility Index, U_{d_j}, is obtained by estimating the number of cache hits for d_j, i.e., the number of received Interests for d_j that are locally satisfied by the node's CS, during its residual lifetime, L_{res,d_j}. Based on the information included in the Popularity Table, the node retrieves the number of Interests received for d_j and estimates the request frequency for the content, f_{d_j}. The Utility Index of d_j is then computed as:

$$U_{d_j} = L_{res,d_j} \cdot f_{d_j}. \quad (5)$$

Intuitively, the higher the cache hits for d_j the higher the utility of maintaining d_j in the CS. Vice versa, the replaced packet is the one which satisfies the following condition $min_{\forall d_i \in C_s}(U_{d_i})$, with C_s being the set of contents stored in the CS.

It is worth observing that the computation of the Utility Index includes an additional burden on the edge nodes, however this latter is limited for two main reasons. First, the Utility Index is computed only if a new item has to be included in the CS and this latter is full. Therefore, the calculation is not performed at the reception of each Data packet, but only if no other node has previously cached the same packet and if, based on the result from Eq. 4, it has to be cached. Second, the content residual lifetime, taken as input in the computation of the Utility Index, is natively calculated by NDN nodes in presence of transient contents. By design, NDN nodes, even those implementing the traditional CEE+LRU policy, periodically compute the residual lifetime of every cached item to clean up the CS and remove each stale Data packet (the default time interval between each check is one second [46]). As a result, CES simply leverages this available information to compute the Utility Index, when needed. The CES workflow is summarized in Fig. 2(b).

D. Understanding the Utility Index

To theoretically figure out how the conceived Utility Index affects the content replacement, we performed a preliminary numerical study in Matlab, when considering a catalog of 5000 contents requested according to a Zipf-based popularity distribution [31]. Fig. 3 reports the Utility Index when varying the content residual lifetime in the range [1-500]s and the frequency of requests, for the Zipf's skewness parameter, α, equal to 0.4, 0.8, 1.2. It can be clearly observed that the higher the residual lifetime the higher the Utility Index. The higher the frequency of requests (which is much more true for higher α values) the higher the Utility Index. As a further remark, let us consider the impact of the parameter α. It can be observed that for higher α values, there are a few very highly requested contents; i.e., a few caching probability points are shown corresponding to high frequency of requests values. With α passing from 1.2 to 0.4, the Utility Index points span different frequency of requests values. In other words, the number of requests is more uniformly distributed. Moreover, the number of points increases because a higher number of distinct contents from the catalog is requested.

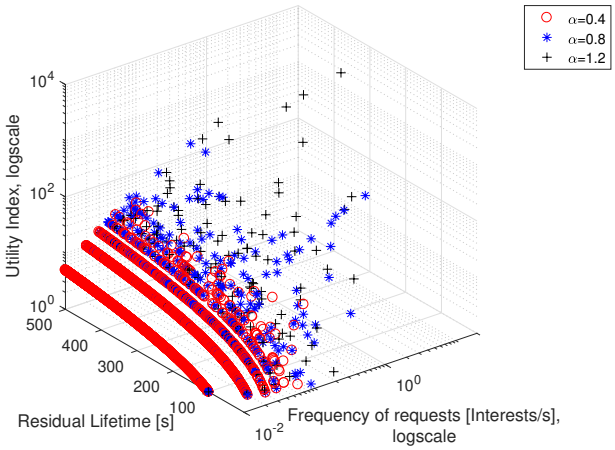


Fig. 3. Utility index when varying content residual lifetime and popularity distribution.

V. Performance Evaluation

A. Reference Scenario and Settings

The performance evaluation aims at reproducing a large-scale network scenario and it is not meant to assess the viability of the proposal for a single IoT application, but to provide general findings in presence of a varying traffic pattern with realistic settings from related literature, i.e., [30], [33], [47], [48].

More specifically, we consider a realistic hybrid network scenario that includes a backbone network connecting multiple edge domains which WSNs are attached to. To model the backbone network, we consider the GEANT topology, a high-bandwidth, high-speed and highly resilient pan-European network. The topology, accessed from the Internet Topology Zoo database (www.topology-zoo.org/), consists of 39 nodes. Among them, 25 GEANT nodes are randomly selected to be access routers towards the edge domains, which are simulated as 2-level fat tree topologies. 50 leaf nodes of the edge domains are randomly selected to be attached to access points acting as gateways towards the WSNs.

In particular, we consider 50 distinct WSNs, each one including 100 IoT sensors[2] that leverage ZigBee to connect to the gateway. Each source produces periodically a single transient content and, therefore, the total catalog is of 5000 contents. Each IoT content, e.g., a sensor reading, is assumed to be small enough in terms of memory to fit within a single Data packet and a single radio transmission. Moreover, each source includes in the Data packet a freshness period, which is uniformly varying in the range $[1 - 500]s$. Such range is selected to reflect the typical temporal scope of different real application domains, ranging from smart mobility to smart grid

[2]A number of sensor nodes in the order of thousands is representative of a typical smart city deployment, see for instance the SmartSantander case [49].

TABLE I
MAIN SIMULATION SETTINGS.

Parameter	Value
Number of core nodes	39
Number of edge nodes	175
Number of WSNs domains	50
Content catalog size	5000
Content lifetime	[1,500]s
Content request rate	Poisson, λ=5 requests/s
Content popularity	• Zipf, α=1 • uniform distribution
Node caching capacity	5 Data packets

[47], [48], under the overall smart city umbrella. For the sake of simplicity, IoT sources never sleep during the simulation.

We assume the content request rate follows a Poisson distribution with parameter λ equal to 5 Interests/s. Requests are generated in the network and assigned to consumer nodes that are randomly attached to the leaf nodes of the edge domains. Therefore, in principle, the majority of requests for IoT contents come from remote consumers w.r.t. the IoT sources. The request pattern follows two distinct popularity distributions, uniform and Zipfian with skewness parameter α=1, to cover both the patterns identified in the literature [30], [33], also according to the analysis of realistic datasets. The main simulation settings are summarized in Table I.

B. Benchmark Schemes and Metrics

We compare the proposed CCS/CES against the following reactive benchmark schemes:

- CEE, representing the simplest caching solution deployed in vanilla NDN, where each node caches all the contents crossing it, until it has space in the cache and then it uses LRU as replacement policy.
- RC, where contents are randomly cached with a fixed probability $p = 0.5$, which limits the data redundancy without underusing the available resources [33]. It is also coupled with LRU replacement.
- Least Fresh First (LLF) [40], representing a benchmark freshness-aware policy where short-lived contents are replaced in favour of long-lived ones.
- Fine Grained Popularity-based Caching (FGPC) [22], representing a traditional popularity-aware scheme oblivious of the freshness parameter. It is coupled with LRU replacement.
- Betweenness and Edge Popularity Caching (BEP) [21], where contents are cached based on their popularity and the betweenness of the nodes. It is coupled with LRU replacement.

All the selected benchmark schemes were specifically designed to work in an Internet-scale scenario, being this latter the focus of our study. In particular, CEE and RC are selected as baselines for our performance comparison, while LLF and FGPC allow to show the benefits of the metrics leveraged in our proposal, respectively freshness and popularity, when they are individually considered. Finally, BEP allows to assess the joint effect of popularity and topological centrality of the nodes, being this latter another commonly used caching decision parameter. We recall that, in any of the aforementioned cases, a lifetime is associated to each Data packet: whenever the corresponding timeout expires, the content is removed from the CS.

The following performance metrics are measured:

- **Cache hit ratio.** It is defined as the average ratio between the number of requests satisfied at the intermediate nodes and the total number of received requests. Therefore, it assesses the effectiveness of the caching strategy and it also reflects the reduction of the number of requests reaching the original IoT source in the WSN domain, which translates in a lower energy consumption of battery-powered devices.
- **Packet delay.** It is the time elapsed since the Interest is sent from the consumer until the IoT Data packet is received. This metric allows to assess the effectiveness of the caching strategy from a consumer-centric perspective.
- **Hop count.** The number of hops needed to retrieve the content. This metric allows to assess the efficiency of the caching strategy from an operator-centric perspective: the smaller the hop count the lower the probability of network congestion and the amount of data traversing the network, translating in a reduction of energy consumption.

All simulations include a warm-up period of 5 minutes, which is useful to make popularity-based schemes reach a stationary regime. During the warm-up, the schemes simply cache incoming packets according to the CEE+LRU policy and update the caching parameters; after the warm-up, we start to compute the performance metrics over the newly generated content requests. The simulation ends when all the content requests generated after the warm-up period are satisfied (i.e., the consumers receive the requested contents). The targeted number of content requests over which metrics are computed is varying from 200 to 1600.

Results are averaged over ten independent runs and reported with 95% confidence intervals.

C. Results

Fig. 4 compares the proposal against the considered benchmark schemes, when varying the number of targeted content requests in presence of the Zipf distribution, in terms of the chosen evaluation metrics.

It can be clearly observed that, as expected, the CEE scheme (black curves) exhibits the poorest performance for all the considered metrics. By allowing all packets to be cached indiscriminately, the strategy tends to cache the same contents in the CSs of involved nodes. This translates in the lowest cache hit ratio (see Fig. 4(a)), the highest data retrieval delay (see Fig. 4(b)), and the highest number of hops (see Fig. 4(c)). Better performances are achieved by the RC scheme (blue curves), since it natively increases the cache diversity by probabilistically caching contents. RC is further outperformed by the LFF scheme (cyan curves). The latter one, by leveraging the freshness information in the replacement strategy, erases the contents with the shortest residual lifetime and leaves space for long-lived contents. By giving caching priority to popular contents, FGPC scheme (green curves)

outperforms LFF, RC and CEE. In turn, FGPC is slightly outperformed by BEP (orange curves) because, in addition to the content popularity, this latter takes into account the betweenness of the nodes and caches the contents according to the relationship between the two parameters, thus limiting the cache redundancy. However, by combining the benefits of a freshness- and popularity-aware strategy, our proposal outperforms all the considered benchmark schemes: it takes more judicious caching decisions aimed at selectively caching contents and improving the storage resources utilization.

Regardless of the specific caching scheme, metrics improve as the number of requests increases. Such a trend is due to the fact that as the number of requests increases, with a Zipf distribution, it is more likely that they concentrate on a few popular contents, i.e., a subset of the contents in the catalog. Hence, it is more likely to cache contents that can serve multiple requests. This proves the scalability achieved by the in-network caching solution, which is particularly helpful to reduce the interactions with energy-constrained wireless sensor nodes acting as producers.

Fig. 5 reports the same metrics discussed above, when considering the content popularity to be uniformly distributed. Compared to previous results in Fig. 4, it can be observed that the content retrieval latency is higher and tends to increase with the number of requests (Fig. 5(b)), the cache hit ratio is lower (Fig. 5(a)) and the number of hops significantly higher (Fig. 5(c)). Indeed, with content requests following a uniform distribution, there is less chance to serve multiple requests through cached contents. In such a scenario, freshness-aware strategies, like LFF and CCS/CES, perform better than the other schemes, since they privilege caching contents with a longer lifetime, which can potentially serve multiple requests. In addition, thanks to the cooperative caching scheme at the edge, coupled with freshness-aware replacement, our proposal is able to outperform LFF. The poor performance of BEP is due to the fact that it ranks the contents based on their popularity level. However, in presence of a uniform popularity distribution, all the contents have basically the same ranking and therefore, they are cached at the same central nodes, which become overwhelmed by multiple caching and replacements operations, while the storage space distributed in the other nodes remains underutilized.

To gain insight into the performance boost individually provided by CCS and CES, we focus on the cache hit ratio metric and distinguish: *(i)* the percentage of cache hits cumulatively obtained by the core nodes implementing CCS and *(ii)* the percentage of cache hits cumulatively obtained by the edge nodes implementing CES. Fig. 6 shows this metric in the simulated network scenario, when content requests follow the Zipf and the uniform popularity distribution. In the Zipf case, it can be observed that CCS accounts for about the 60% of the cache hits in presence of 200 requests, while the contribution grows to 78% in presence of 1600 requests. The higher gain of CCS w.r.t. CES is mainly due to the fact that, in the considered scenario, the majority of requests *(i)* concentrate on few popular contents and *(ii)* are generated by remote consumers and cross the core network before reaching the edge domain where the IoT sources are located. As a result, there is a high chance to find the popular contents cached by core routers implementing CCS. Vice versa, the gain of CCS reduces when the popularity pattern of requests is uniformly distributed. In this case, the benefits of caching are generally less remarkable, since consumers tend to ask for distinct contents and a higher number of requests reach the edge domains, where the IoT sources are attached to. As a result, there is a higher chance to obtain a cache hit at the edge nodes implementing CES.

D. Storage Overhead Analysis

To assess the practicality of the proposal, in this section, we discuss the storage overhead of the proposed solution w.r.t. the benchmark schemes. Table II presents the storage overhead incurred to drive the caching decision. It can be observed that all the popularity-based schemes, i.e., FGPC as well as BEP (but at the edge nodes only) and the proposed CCS and CES, store the Popularity Table tracking the number of requests for each distinct named content. A 4 bytes-long (float) counter of requests is supposed to be kept for each received content. Hence, the overhead is upper bounded by $(4 + 40) \cdot |C_r|$ bytes, being $|C_r|$ the number of requested contents, which is typically lower than the content catalog size [50], [51], and 40 the average content name length (in bytes) [52], [53]. As a result, under the considered assumptions, the storage overhead per content is 44 bytes for the proposed strategies[3]. This number passes to 48 bytes in case of BEP due to the historical popularity field kept in the Popularity Table.

The CCS scheme also stores the popularity and the freshness thresholds computed through a simple moving average, while BEP leverages the betweenness centrality parameter of the nodes, each resulting in additional 4 bytes of storage footprint. However, it is worth to highlight that BEP introduces a non negligible signalling overhead w.r.t. CCS/CES. In particular, each Interest carries in piggybacking: *(i)* the current popularity of the requested contents (i.e., a float variable) and *(ii)* the betweenness centrality of all the nodes traversed by the request (i.e., an array of float variables, whose length depends on the number of on-path nodes). The Data packet instead carries a field identifying the betweenness centrality of the selected cacher (i.e., a float variable). Vice versa, the only signalling overhead needed in our proposal is the CACHED bit, i.e., a boolean variable foreseen by CES in the Data packet.

E. Computational Analysis

When receiving a Data packet, the basic benchmark schemes CEE+LRU and RC+LRU simply evict the least recently used item from the cache. The computational complexity of the two schemes is well known in literature and equal to $O(1)$ [54]. The LFF scheme, instead, performs a replacement based on the least fresh cached content. Therefore, LFF has to find the minimum lifetime among the set of cached items and the complexity of this operation is bound by $O(S_{CS})$, being

[3]According to [22], in order to significantly reduce the overhead of the Popularity Table, specific techniques such as the message-digest (MD5) hash algorithm could be effective, but this is out of scope of the current paper.

TABLE II
STORAGE OVERHEAD: OUR PROPOSAL VS. THE BENCHMARK SCHEMES

Strategy	Storage	Footprint
LFF/CEE/RC	-	-
FGPC	Popularity Table • Content name • Counter tracking the number of requests	$44 \cdot \lvert C_r \rvert$ bytes
BEP	Popularity Table (Edge nodes only) • Content name • Counter tracking the number of requests • Historical Popularity Betweeness Centrality (All nodes)	$48 \cdot \lvert C_r \rvert$ bytes 4 bytes
CCS	Popularity Table • Content name • Counter tracking the number of requests Popularity Threshold Freshness Threshold	$44 \cdot \lvert C_r \rvert$ bytes 4 bytes 4 bytes
CES	Popularity Table • Content name • Counter tracking the number of requests	$44 \cdot \lvert C_r \rvert$ bytes

S_{CS} the size of the CS, which - in the NDN context - can be expressed in terms of number of cached Data packets.

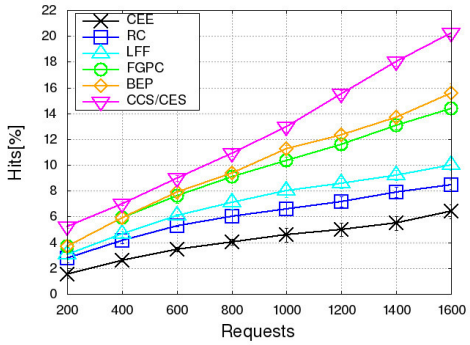
It is worth noticing that the estimation of the content residual lifetime is a feature natively provided by NDN and implemented by all the schemes. Both CCS and FGPC are threshold-based schemes and replace contents by using LRU. Therefore, the additional overhead, w.r.t. the previous schemes, consists only in the computation of the threshold(s), which however is not performed at each packet reception, but periodically, e.g., every minute.

Compared to the previous schemes, CES and BEP incur a slightly higher computational overhead since, to derive the caching probability they have to identify the most popular content(s). The computation overhead of this operation depends on the number of contents $|C_r|$ tracked in the Popularity Table and, therefore, the complexity is in the order of $O(|C_r|)$. However, similarly to CCS, this operation is performed periodically, on a time window basis. In BEP, the popularity is computed only at the edges nodes and then, shared with the rest of the nodes. The replacement operation in CES requires the computation of the minimum of the Utility Indexes of the contents cached in the CS. This operation, similarly to LFF, is bound by $O(S_{CS})$ but, as previously discussed, it is performed only in presence of a positive outcome of the caching decision strategy.

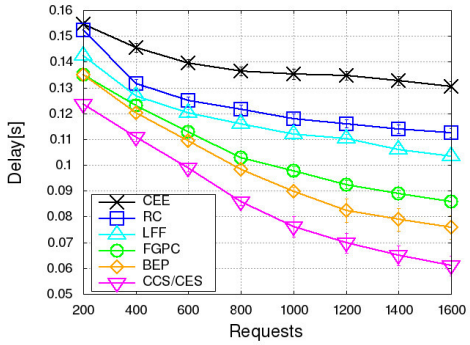
VI. MAIN FINDINGS AND CONCLUSION

In this paper, an Internet-scale caching strategy for IoT contents generated by WSNs nodes is designed that jointly considers popularity and freshness parameters as decision metrics. The strategy has different targets, depending on the core and edge network segments where it is implemented.

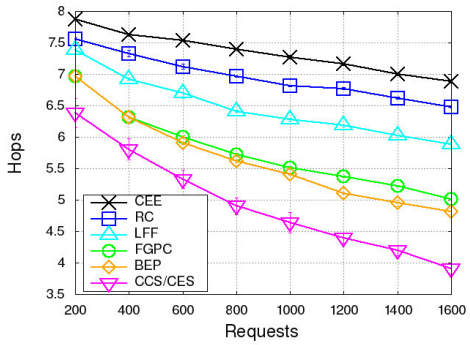
Via extensive simulations conducted under realistic settings, we evaluated the performance of the conceived solution both when compared against baseline benchmarks as well as other representative freshness-aware or popularity-aware



(a) Cache hits.



(b) IoT data retrieval delay.



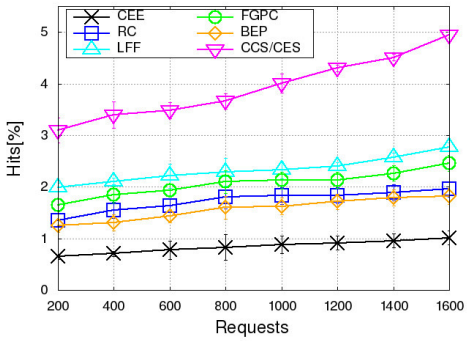
(c) Hop Count.

Fig. 4. Metrics when varying the number of IoT Data requests (Zipf distribution).

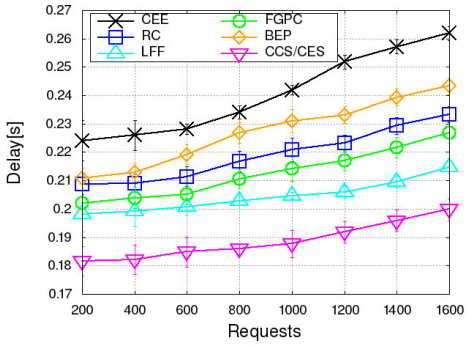
caching schemes available in the literature. We observed the superiority of the proposal under all the considered settings. Improvements are gradually achieved when moving from completely blind caching solutions to those accounting for popularity, freshness, betweenness, and a combination of them, confirming the need for more judicious caching decisions.

As a further finding, the simulation study shows that the content popularity distribution also affects the results. In case of Zipf distribution, the content popularity is crucial as a caching decision metric, since the majority of requests are centered around few popular contents. Conversely, when the uniform distribution is considered, the popularity metric is irrelevant since contents are equally requested.

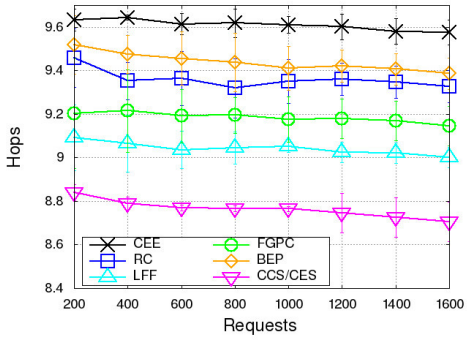
Moreover, although overall contributing, at an end-to-end level, to the improvements of the proposal compared to



(a) Cache hits.



(b) IoT data retrieval delay.



(c) Hop Count.

Fig. 5. Metrics when varying the number of IoT Data requests (Uniform distribution).

the other benchmarks, the two proposed schemes, CES and CCS, applied in the edge and the core segments respectively, differently contribute to the performance and in a different way for different content distributions. Under the Zipf distribution, there is a high chance to find the popular contents cached by core routers implementing CCS. Whereas, there is a higher chance to obtain a cache hit at the edge nodes implementing CES when the popularity pattern of requests is uniformly distributed. Such a finding confirms our intuition about the need for a differentiated caching decision into different network segments. Although specifically shown to be suited to deal with the caching of IoT data, such an approach looks promising also to treat other kinds of contents traveling over the Internet.

The measured higher cache hit ratio and lower number

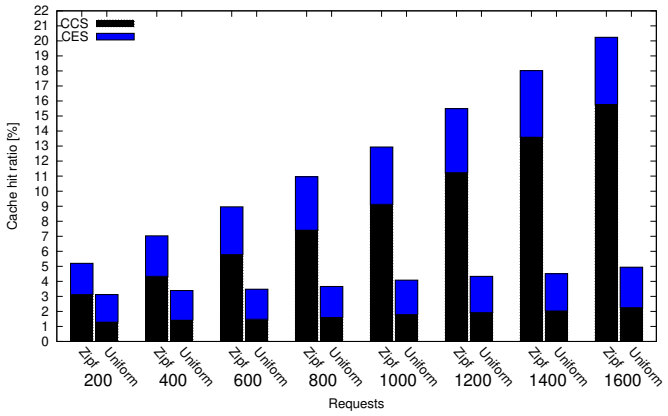


Fig. 6. Cache hit ratio individually provided by CCS and CES (Zipf and uniform distribution).

of hops traversed by data, coupled with the negligible incurred signalling and lower footprint per node, compared to the benchmarks, overall ensure a lower pressure on energy-constrained devices and less burden on the network nodes for data forwarding. This further translates into a higher sustainability of the network infrastructure.

Future work will focus on further improving the performance of the caching strategy for IoT contents, for example by adding other decision parameters like topological centrality, and by integrating NDN with other future Internet networking paradigms like Software Defined Networking (SDN) to support more judicious caching as well as energy-efficient decisions.

ACKNOWLEDGMENTS

This work has been partially supported by the “A COGnItive dynamic sysTem to allOw buildings to learn and adapt’ (COGITO) project, funded by the Italian Government (PON ARS01_00836).

REFERENCES

[1] F. Cicirelli, A. Guerrieri, G. Spezzano, A. Vinci, O. Briante, and G. Ruggeri, “iSapiens: a platform for social and pervasive smart environments,” in *IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 365–370.

[2] I. Poole, “What exactly is ZigBee?” *Communications Engineer*, vol. 2, no. 4, pp. 44–45, 2004.

[3] S. Sarkar, S. Chatterjee, and S. Misra, “Assessment of the Suitability of Fog Computing in the Context of Internet of Things,” *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2015.

[4] L. Zhang *et al.*, “Named data networking,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[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 *ACM Conference on Information-Centric Networking*, 2014, pp. 77–86.

[6] M. Amadeo, C. Campolo, A. Molinaro, and N. Mitton, “Named data networking: A natural design for data collection in wireless sensor networks,” in *2013 IFIP wireless days (WD)*. IEEE, 2013, pp. 1–6.

[7] A. Mtibaa, R. Tourani, S. Misra, J. Burke, and L. Zhang, “Towards edge computing over named data networking,” in *IEEE International Conference on Edge Computing (EDGE)*, 2018, pp. 117–120.

[8] M. Amadeo, G. Ruggeri, C. Campolo, and A. Molinaro, "IoT services allocation at the edge via named data networking: From optimal bounds to practical design," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 661–674, 2019.

[9] J. Takemasa, Y. Koizumi, T. Hasegawa, and I. Psaras, "On energy reduction and green networking enhancement due to in-network caching," in *IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, 2015, pp. 513–518.

[10] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in Named Data Networking for the Wireless Internet of Things," in *IEEE RIoT*, 2015, pp. 1–6.

[11] O. Hahm *et al.*, "Low-power internet of things with NDN & cooperative caching," in *ACM Conference on Information-Centric Networking*, 2017, pp. 98–108.

[12] M. A. Naeem, R. Ali, M. Alazab, Y. Meng, and Y. B. Zikria, "Enabling the content dissemination through caching in the state-of-the-art sustainable information and communication technologies," *Sustainable Cities and Society*, vol. 61, p. 102291, 2020.

[13] J. Xu, K. Ota, and M. Dong, "Energy efficient hybrid edge caching scheme for tactile internet in 5g," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 2, pp. 483–493, 2019.

[14] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: A survey," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–41, 2019.

[15] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2.0: A new version of the NDN simulator for NS-3," *NDN, Technical Report NDN-0028*, 2015.

[16] M. Zhang, H. Luo, and H. Zhang, "A Survey of Caching Mechanisms in Information-Centric Networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1473–1499, 2015.

[17] G. Rossini and D. Rossi, "Coupling caching and forwarding: Benefits, analysis, and implementation," in *ACM Conference on Information-Centric Networking*, 2014, pp. 127–136.

[18] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, and Y. Ji, "Performance of Probabilistic Caching and Cache Replacement Policies for Content-Centric Networks," in *IEEE LCN*, 2014, pp. 99–106.

[19] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache "less for more" in Information-Centric Networks," in *International Conference on Research in Networking*. Springer, 2012, pp. 27–40.

[20] C. Bernardini, T. Silverston, and F. Olivier, "MPC: Popularity-based Caching Strategy for Content Centric Networks," in *IEEE International Conference on Communications (ICC)*, 2013, pp. 3619–3623.

[21] Q. Zheng, Y. Kan, J. Chen, S. Wang, and H. Tian, "A cache replication strategy based on betweenness and edge popularity in named data networking," in *IEEE ICC*, 2019, pp. 1–7.

[22] M. D. Ong, M. Chen, T. Taleb, X. Wang, and V. Leung, "FGPC: Fine-Grained Popularity-based Caching Design for Content Centric Networking," in *ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, 2014, pp. 295–302.

[23] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, and K. Drira, "How to cache in ICN-based IoT environments?" in *IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 2017, pp. 1117–1124.

[24] G. Jaber and R. Kacimi, "A collaborative caching strategy for content-centric enabled wireless sensor networks," *Computer Communications*, vol. 159, pp. 60–70, 2020.

[25] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. of the 2nd edition of the ICN workshop on Information-centric networking*, 2012, pp. 55–60.

[26] J. Mišić and V. B. Mišić, "Proxy cache maintenance using multicasting in coap iot domains," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1967–1976, 2018.

[27] C. Gündoğran, P. Kietzmann, M. Lenders, H. Petersen, T. C. Schmidt, and M. Wählisch, "NDN, CoAP, and MQTT: a comparative measurement study in the iot," in *ACM Conference on Information-Centric Networking*, 2018, pp. 159–171.

[28] Y. An and X. Luo, "An in-network caching scheme based on energy efficiency for content-centric networks," *IEEE Access*, vol. 6, pp. 20 184–20 194, 2018.

[29] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.

[30] B. Chen, L. Liu, M. Sun, and H. Ma, "IoTCache: Toward data-driven network caching for internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 064–10 076, 2019.

[31] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *IEEE INFOCOM'99*, vol. 1, 1999, pp. 126–134.

[32] J. Yao and N. Ansari, "Joint content placement and storage allocation in C-RANs for IoT sensing service," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1060–1067, 2018.

[33] J. Pfender, A. Valera, and W. K. Seah, "Performance Comparison of Caching Strategies for Information-Centric IoT," in *ACM Conference on Information-Centric Networking*, 2018, pp. 43–53.

[34] S. Vural, N. Wang, P. Navaratnam, and R. Tafazolli, "Caching Transient Data in Internet Content Routers," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1048–1061, 2016.

[35] M. Amadeo, G. Ruggeri, C. Campolo, A. Molinaro, and G. Mangiullo, "Caching popular and fresh IoT contents at the edge via named data networking," in *IEEE INFOCOM WKSHPS*, 2020, pp. 610–615.

[36] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin, "Caching transient data for Internet of Things: a deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2074–2083, 2018.

[37] S. Fatale, R. S. Prakash, and S. Moharir, "Caching policies for transient data," *IEEE Transactions on Communications*, vol. 68, no. 7, pp. 4411–4422, 2020.

[38] B. Nour, H. Khelifi, H. Moungla, R. Hussain, and N. Guizani, "A distributed cache placement scheme for large-scale information-centric networking," *IEEE Network*, vol. 34, no. 6, pp. 126–132, 2020.

[39] H. Asmat, I. U. Din, F. Ullah, M. Talha, M. Khan, and M. Guizani, "ELC: Edge linked caching for content updating in information-centric internet of things," *Computer Communications*, vol. 156, pp. 174–182, 2020.

[40] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and H. Mathkour, "Least fresh first cache replacement policy for NDN-based IoT networks," *Pervasive and Mobile Computing*, vol. 52, pp. 60–70, 2019.

[41] M. A. M. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "On the Performance of Caching and Forwarding in Information-Centric Networking for the IoT," in *International Conference on Wired/Wireless Internet Communication*. Springer, 2015, pp. 313–326.

[42] P. Schulz *et al.*, "Latency critical iot applications in 5G: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, 2017.

[43] J. Augé, G. Carofiglio, G. Grassi, L. Muscariello, G. Pau, and X. Zeng, "Map-me: Managing anchor-less producer mobility in content-centric networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 596–610, 2018.

[44] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Scalable mobile backhauling via information-centric networking," in *IEEE Local and Metropolitan Area Networks (LANMAN)*, 2015, pp. 1–6.

[45] H. Dai, Y. Wang, H. Wu, J. Lu, and B. Liu, "Towards line-speed and accurate on-line popularity monitoring on NDN routers," in *IEEE IWQoS*, 2014, pp. 178–187.

[46] A. Afanasyev *et al.*, "Nfd developer's guide," 2014.

[47] Y. Atif, S. Kharrazi, D. Jianguo, and S. F. Andler, "Internet of Things Data Analytics for Parking Availability Prediction and Guidance," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 5, p. e3862, 2020.

[48] M. Sharaf *et al.*, "Modeling and code generation framework for IoT," in *Int. Conf. on System Analysis and Modeling*. Springer, 2019, pp. 99–115.

[49] L. Sanchez *et al.*, "SmartSantander: IoT experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[50] M. Z. Shafiq, A. X. Liu, and A. R. Khakpour, "Revisiting caching in content delivery networks," in *ACM international conference on Measurement and modeling of computer systems*, 2014, pp. 567–568.

[51] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," *Relatório técnico, Telecom ParisTech*, pp. 1–6, 2011.

[52] A. V. Ventrella, G. Piro, and L. A. Grieco, "Publish-subscribe in mobile information centric networks: Modeling and performance evaluation," *Computer Networks*, vol. 127, pp. 317–339, 2017.

[53] Y. Wang, B. Xu, D. Tai, J. Lu, T. Zhang, H. Dai, B. Zhang, and B. Liu, "Fast name lookup for named data networking," in *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*. IEEE, 2014, pp. 198–207.

[54] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 374–398, 2003.

Marica Amadeo is an assistant professor at University Mediterranea of Reggio Calabria. She received a master degree (2008) in telecommunications engineering from the University Mediterranea of Reggio Calabria, and a Ph.D. degree in 2013 from the same University. Her major research interests are in the field of Information-Centric Networking, Internet of Things and edge computing.

Claudia Campolo is an associate professor of telecommunications at the University Mediterranea of Reggio Calabria. She received an master degree (2007) and a Ph.D. degree (2011) in telecommunications engineering from the same university. Her main research interests are in the field of vehicular networking, 5G and future Internet architectures.

Giuseppe Ruggeri received the master degree in electronics engineering in 1998 from the University of Catania, Italy and, in 2002, he received the Ph.D. in electronics, computer science and telecommunications engineering from the University of Palermo, Italy. He is currently assistant professor at the University Mediterranea of Reggio Calabria. His current interests include self organizing networks, Internet of Things, Social Internet of Things.

Antonella Molinaro is an associate professor of telecommunications at the University Mediterranea of Reggio Calabria, Italy, and has a double affiliation with CentraleSupélec/L2S, Université Paris-Saclay, France. Previously, she was an assistant professor with the University of Messina (1998-2001) and the University of Calabria (2001-2004), and a research fellow at the Politecnico di Milano (1997-1998). She was with Siemens, Munich (1994-1995). Her current research focuses on 5G, vehicular networking and future Internet architectures.