Low-power Internet of Things with NDN & Cooperative Caching

Oliver Hahm, Emmanuel Baccelli, Thomas C. Schmidt, Matthias Wählisch, Cédric Adjih, Laurent Massoulié

Low-power Internet of Things with NDN & Cooperative Caching

Oliver Hahm
Inria
oleg@riot-os.org

Emmanuel Baccelli
Inria
emmanuel.baccelli@inria.fr

Thomas C. Schmidt
HAW Hamburg
t.schmidt@haw-hamburg.de

Matthias Wählisch
FU Berlin
m.waehlisch@fu-berlin.de

Cédric Adjih
Inria
cedric.adjih@inria.fr

Laurent Massoulié
Inria
laurent.massoulie@inria.fr

ABSTRACT

Energy efficiency is a major driving factor in the Internet of Things (IoT). In this context, an IoT approach based on Information-Centric Networking (ICN) offers prospects for low energy consumption. Indeed, ICN can provide local in-network content caching so that relevant IoT content remains available at any time while devices are in deep-sleep mode most of the time. In this paper, we evaluate NDN enhanced with CoCa, a simple side protocol we designed to exploit content names together with smart interplay between cooperative caching and power-save sleep capabilities on IoT devices. We perform extensive, large scale experiments on real hardware with IoT networks comprising of up to 240 nodes, and on an emulator with up to 1000 nodes. We show in practice that, with NDN+CoCa, devices can reduce energy consumption by an order of magnitude while maintaining recent IoT content availability above 90 %. We furthermore provide auto-configuration mechanisms enabling practical ICN deployments on IoT networks of arbitrary size with NDN+CoCa. With such mechanisms, each device can autonomously configure names and auto-tune parameters to reduce energy consumption as demonstrated in this paper.

CCS CONCEPTS

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

KEYWORDS

Information-centric networking, energy efficiency, cooperative caching

ACM Reference format:
Oliver Hahm, Emmanuel Baccelli, Thomas C. Schmidt, Matthias Wählisch, Cédric Adjih, and Laurent Massoulié. 2017. Low-power Internet of Things with NDN & Cooperative Caching. In *Proceedings of ACM ICN Conference, Berlin, Germany, Sept. 2017 (ICN'17),* 10 pages.
DOI:

1 INTRODUCTION

In the IoT, energy efficiency and memory efficiency play crucial roles. In particular, the amount of memory (RAM) is a key factor for both the price tag and the power draw of a low-end IoT device [6, 14]. Such devices need to be able to sleep a large part of the time to reduce battery drain, and thereby increase their life span—many of them are expected to last years on a small battery.

ICN'17, Berlin, Germany
2017. . . . $
DOI:

Common approach to energy efficiency in the IoT selectively combine the techniques below.

- **Energy efficient hardware** with micro-controller and radio consuming energy in mW range and ultra-efficient sleep modes in nW range. Energy harvesting techniques may also be applicable in some cases, but are not the focus of this paper.
- **Radio duty-cycling (RDC) at the MAC layer** like in TSCH [34] achieves low power by minimizing idle listening.
- **Less chatty network layer protocols** avoid communication in broadcast/multicast as the 6loWPAN protocols that adapt IPv6 to the IoT [31].
- **Centralized content caching** in the cloud or on a proxy, e.g. CoAP / HTTP caching [30].

With centralized content caching in place, content availability is preserved by a proxy or the cloud, while IoT devices sleep a large part of the time. Hence there is no trade-off between content availability and energy efficiency. Sensors can fully benefit from *coordinated* RDC mechanisms allowing less than 1% radio activity [34].

However, this standard approach suffers two fundamental limitations. First, when the local network gathers a large number of nodes, explicit synchronization and coordination of RDC with a MAC layer based on TSCH becomes impractical because user traffic suffers long delays or control traffic becomes large. Though alternatives have demonstrated explicit coordination of RDC at large scale [25], most existing solutions do not (yet) implement it and instead use *uncoordinated* RDC MAC layers such as ContikiMAC [11]. Second, in a variety of IoT use cases, connectivity with the designated gateway/proxy is *intermittent*, and centralized caching of IoT content fails. This happens when nodes or mobile and parts of the network temporarily fragment, but also when the gateway is a device deployed in the field along with the other IoT devices (e.g., the gateway is just an IoT device with dual radio). In such cases, the designated gateway/proxy is also unavailable a large part of the time.

In this context, in-network caching as introduced by Information-centric Networking (ICN) appears beneficial. It is our intuition that ICN could conveniently organize *distributed* caching of IoT content near each producer, so that fewer IoT devices would have to be active, while others can sleep and data would still be available locally, at any time. Recent work [5, 20, 21, 29] has shown potential for ICN in the Internet of Things. In addition, the convergence of ICN on TSCH promises further improvements of RDC [15]. Still, to

the best of our knowledge there is no prior experimental work on the interplay between power-save sleep techniques on IoT devices and in-network distributed caching strategies with ICN. In this paper we explore this idea, and make the following contributions:

(1) We design and implement CoCa, a side-protocol for NDN enabling distributed cooperative caching of IoT content.
(2) We carry out extensive experiments to evaluate and validate NDN+CoCa on several testbeds with IoT networks comprising of up to 300 of devices, as well as on an emulator with 1,000 IoT devices.
(3) We show that, by exploiting both content names and interplay between deep-sleep capabilities and content caching on IoT devices, NDN+CoCa can achieve 90% reduction in energy consumption compared to state-of-the-art, while availability of recent IoT content remains above 90 %.
(4) Using a theoretical model, we design and implement autoconfiguration mechanisms allowing each IoT device in a deployment of arbitrary size to autonomously configure NDN+CoCa and activate energy savings as demonstrated in this paper.

The remainder of this paper is structured as follows. Section 2 characterizes the targeted IoT scenarios, and outlines CoCa. In Section 3, we implement NDN+CoCa and evaluate it on real hardware to characterize the potential in terms of energy efficiency gains. Section 4 presents the design and implementation of autoconfiguration mechanisms which allow larger scale deployments of NDN+CoCa in practice. Finally, in Section 5, we evaluate NDN+CoCa operation at large scale on IoT-Lab testbeds, as well as on a emulated network with up to 1000 nodes.

2 IoT SCENARIO CHARACTERIZATION

In this section we describe IoT device hardware and IoT network characteristics from the physical and logical point of view.

2.1 IoT Network Characteristics

We first consider a single wireless broadcast domain that gathers a set of sensors of various types as shown in Figure 1. This domain is connected to the Internet via an intermittent uplink, available when the gateway is not sleeping. When the uplink is on, it can send an interest for available content, whereupon nodes (if not sleeping) can reply with chunks of data stored in their cache. We consider that wireless links between nodes to be similar to IEEE 802.15.4 capacity, i.e. very low, measured in *kb/s*. Such low capacity is due both to the limited processing power [6] of micro-controllers typically used on IoT devices (easily overwhelmed), to the low-power and lossy nature of IoT link layers which yield low data rates (congestion appears fast), and to omnidirectional radio communication (unable to avoid interferences).

2.2 IoT Device Hardware Characteristics

We consider typical low-end IoT devices [14], with a few kBytes of RAM and a low-power CPU to which are connected peripherals including a low-power radio interface, and one or more sensors. Such an IoT device is hereafter called a *node*.

A node can be in either of two states: *active* or *sleeping*. In the sleeping state, a node has both its radio switched off, and its CPU

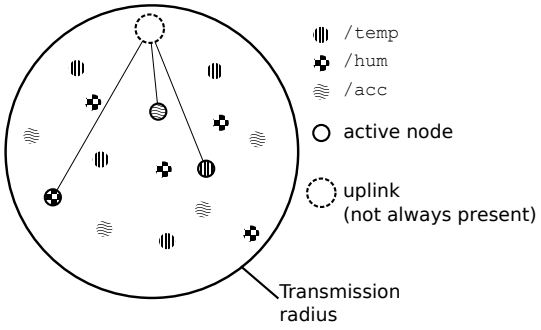


Figure 1: IoT scenario, single-broadcast domain.

in sleep mode. A node in sleeping state transitions to the active state triggered by an external interrupt, either generated by a timer or a sensor, e.g., by a temperature sensor if temperature is above a certain threshold.

In the active state, a node's CPU is running, its radio transceiver is listening or transmitting. We consider a scenario where, when a sensor has new data, it can wake up the node if it has been in sleeping state. The node can process the data accordingly, in active state, upon which it may consider going back to sleeping state. This assumption is in line with the capabilities of typical IoT hardware and available IoT software platform (e.g. RIOT [4]).

Note that, while in active state, idle listening optimizations such as radio duty cycling (RDC) at the MAC layer [13] may be used in practice to further reduce energy consumption. Irrespective of using RDC or not, the mechanisms described in this paper are applicable. We define p as the average ratio of the time a node spends in sleep mode. To save energy, we are interested in cases where p is big, for instance $p \geq 0.9$.

2.3 ICN-IoT Logical Architecture

We use NDN [19] on low-end IoT devices, as depicted in Fig. 2. We consider that each sensor is a source of IoT data and thus a content *producer*, while only the (intermittent) uplink is a consumer. Each sensor generates data as a time series of sensor readings. A sensor reading is assumed to be small enough in terms of memory to fit within a single chunk, and a single radio transmission. A sensor produces content using a specific name. In particular, sensors hosted on the same node produce content using distinct names.

CoCa: Cooperative Caching Side-Protocol.
On one hand, in face of intermittent connectivity and sleeping nodes, NDN will incur delays and many retransmissions of Interests. On the other hand, due to extreme memory contraints on IoT devices, some nodes may not have any memory at all to cache IoT content, while other nodes can devote a part of their memory to a content store, as demonstrated in [5]. We thus consider NDN operation together with *CoCa*, a simple side protocol letting nodes cooperate locally to share their caching capacity.

We implemented CoCa by reusing as much as possible mechanisms already present on each node: link-local broadcast and NDN basic primitives. With CoCa, upon generating a new chunk of IoT content, a producer shares it via link-local broadcast. Upon receiving a new chunk of IoT content via link-local broadcast, an active node does not forward it, but may decide to cache it in its Content

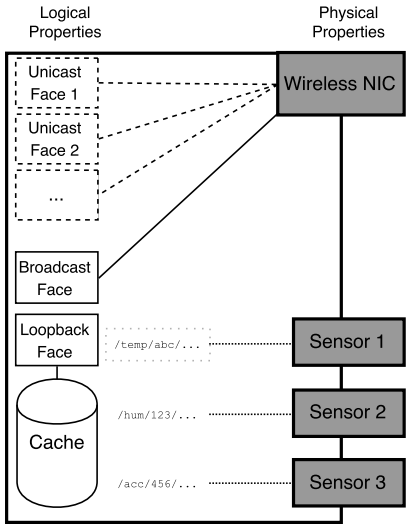


Figure 2: IoT NDN device.

Store, depending on a caching and replacement strategy (see the detail of such strategies in Section 3.3 and beyond). Note that additional memory requirements to run NDN+CoCa are negligible, compared to basic NDN requirements. Using NDN with CoCa, not only is data available immediately (sparing Interest retransmissions) but we can also expect energy efficiency gains – which are the focus in the following.

3 AVAILABILITY *vs* ENERGY TRADEOFFS

In this section we implement and evaluate on real IoT hardware NDN+CoCa. We characterize potential in terms of energy efficiency gains.

3.1 Performance Metrics

We focus on scenarios where sensors monitor a phenomenon whereby (i) data relevance strictly decreases with time, and (ii) a more complete view of what the sensors are monitoring is achieved if available data comes from a larger number of distinct sources (i.e. sensors).

We then evaluate the availability of IoT content considering the below metrics:

Diversity corresponds to the requirement to have a complete view of what the sensors are monitoring. Maximum diversity is achieved if content from all possible sources is retrieved by the uplink.

Freshness corresponds to the requirement to have an up-to-date view of what the sensors are monitoring. Maximum freshness is achieved if the newest data is retrieved by the uplink. We consider the tolerated lifetime of the data L, where $L = k$ denotes that only the k newest values of each sensor are useful.

Energy consumption measured based on the duration that a node spent in active and sleeping state respectively, plus the number of unicast and broadcast transmissions. In practice, we use a common energy model as proposed in [28]: $E = \sum_{state} P_{state} \cdot t_{state}$, where P_{state} is the power consumed for a given *state* and t_{state} is the time to spent in this state. Values for power consumption per state are taken from the datasheets of the MCU and radio transceiver. We consider the states: *sleeping*, *active* (listening and receiving), *sending unicast*, and *sending broadcast* packets. Finally, we assume a typical value for the idle radio duty cycle: 0.6% as described in [13].

3.2 Implementation & Experiments

We implemented our approach on top of RIOT [4], extending the CCN-lite [8] library. To implement the cooperative caching side protocol CoCa described in Section 2.3, we reused NDN mechanisms with minor additions: **(i)** a callback to specify CoCa caching and replacement strategy, **(ii)** a callback to preprocess an Interest packet (e.g., in order to handle wildcard prefixes), and **(iii)** *netapi* [22] options to manipulate the PIT before a node goes into sleep mode. These callbacks—as provided by CCN-lite—are called whenever an Interest is received *before* it gets further processed. This allows to perform a local wildcard matching on the names in the content store.

The experiments were conducted on several testbeds: the Lille site and the Grenoble site of the FIT IoT-Lab testbed [1]. Our experiments took place on networks comprised of up to 300 sensor nodes, communicating with IEEE 802.15.4 2.4 GHz radio transceivers. Further experimental results were obtained via emulating a network of 1,000 nodes running NDN+CoCa on RIOT.

Each run lasted for 30 minutes. Sources produced data periodically (period between 1s and 35s) and every 1s each node decided whether to sleep for the next 1s (with probability p) or to be active for the next 1s (with probability $1 - p$). As soon as a new sensor value was measured, the sensor attempted to cache it using CoCa. In our experiments, the uplink periodically became active, upon which it sent an Interest for content with a wildcard name prefix. All nodes currently in active mode then replied with chunks of content available in their content store. Data availability was computed as the ratio of data received by the uplink with data produced overall.

3.3 Results on Availability & Energy

We first consider a small network of 50 nodes on IoT-Lab with our implementation as described in Section 3.2. In the following, unless specified otherwise, nodes' sleep/activity cycles are *uncoordinated* and thus not necessarily synchronized.

Availability:

We evaluate data availability and energy consumption for different values of sleep ratio p, with NDN+CoCa using a simple caching strategy called *Random Caching*. With *Random Caching*, a new chunk received via CoCa at a content store is cached with probability $q = 0.5$ and cache replacement strategy is a simple LRU, similarly to the approach employed in [17].

In Fig. 3 we observe that the *Random Caching* strategy achieves encouraging performance in terms of availability (near 100%) for low values of p which confirms initial simulation results from prior work [17]. Even for small values of L, i.e., considering only very *fresh* data, the availability is still about 90% for these low values of p. However, we note availability drops sharply for high values of p, for example with $p = 0.95$.

In order to achieve substantial energy efficiency gains, we nevertheless seek to maintain data availability at a higher level for

larger values of p, which corresponds to longer periods in sleeping state, and thus less energy drain. For that purpose, we also evaluate NDN+CoCa with a different caching strategy called *Max Diversity Most Recent* (MDMR), which exploits the name of content to decide when to cache and what to replace in the cache. We assume that content names include the ID of the producer and a timestamp (this assumption matches IoT scenarios with time-series of sensor values). MDMR then operates as follows: first, the cache tries to replace older chunks from the same producer. Next, the cache tries to replace the oldest chunk of a producer from which several chunks are present in the cache. Finally, if there is only one entry per source, the oldest entry in the cache is replaced. In Fig. 3 we observe that MDMR consistently achieves better availability than Random Caching, especially for high values of $p \geq 0.9$, where availability remains around 90% around that mark.

Energy:
To evaluate energy gains, we compare NDN+CoCa with two extreme cases: on one hand NDN with only radio duty cycling (based on ContikiMAC), and on the other hand NDN+CoCa with *coordinated* sleep/activity cycles, whereby only one node is awake at any time: the *deputy*. In details: the deputy stays active and caches available content: this role is periodically assumed by each node, in a round-robin fashion. Cache hand-over is performed between outgoing deputy and incoming deputy (using the same Interest-based mechanism as the uplink to request available content from active nodes).

In order to evaluate the energy consumption for these experiments, we measured the duration each node spends in active and sleeping state, and the number of unicast and broadcast transmissions. We then fed these values to the energy consumption model described in Section 3.2. In Fig. 4 we compare performance in terms of energy consumption. We observe that compared to the baseline radio-duty cycling approach, an approach using MDMR saves about 90% of energy consumption while most of the relevant data remains available at any time. In Fig. 4 we also compare MDMR energy consumption to the baseline with coordinated sleeping. We notice that compared to coordinated sleeping, for high values such as $p = 0.95$, uncoordinated sleeping with MDMR can achieve roughly similar energy-efficiency gains, while most of the data remains available (as shown in Fig. 3).

3.4 Discussion

Generality of the approach:
The results we have obtained so far focus on deployments that fit a single broadcast domain. However, NDN+CoCa with uncoordinated sleeping can also be applied on deployments that do not fit a single broadcast domain, assuming that FIB entries are populated accordingly. Mimicking this scenario, we ran another experiment setup where nodes scattered over a large office building in a manner such that nodes did not have direct radio connectivity with all other nodes (the network diameter was 5 hops, with 300 nodes). Results in Fig. 5 indicate that NDN+CoCa achieves comparably good content availability of about 80% for $L > 1$ in such scenarios. These values are similar to the achieved availability for the single-hop case.

Comparing with IP:
Cooperative distributed caching with CoCa (or a similar protocol)

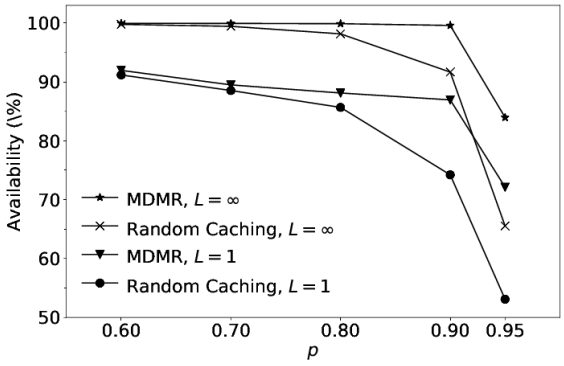


Figure 3: Availability with NDN+CoCa, for different values of p. Experiments conducted with 50 nodes and a cache size that can hold up to 80 entries.

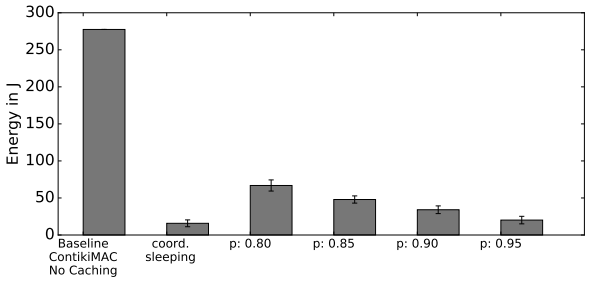


Figure 4: Average energy consumption per node for different values of p, compared with baseline RDC approach (ContikiMAC). Active MCU (Cortex M3) consumes 70 mA, listening consumes 12.8 mA and a broadcasts costs approx. 1.43 mJ. The error bars in this and all following figures depict the standard deviation.

could also be implemented on top of IP. However, NDN+CoCa offers the key advantage of being able to reuse caching capabilities built in the (NDN) network stack. In comparison, an IP stack does not provide built-in caching capabilities, but still uses a large part of the RAM on low-end IoT devices (almost all of it, e.g. on IoT devices with 16kB of RAM or less). Hence, on low-end IoT devices, NDN+CoCa can dedicate significantly more RAM to content caching, compared to CoCa (or an equivalent) on top of IP. Results in Fig. 6 indicate that even a small amount of additional cache size (e.g., 5 kB) can significantly increase the content availability. Thus, NDN+CoCa can provide a significant advantage over a similar approach on IP in terms of energy efficiency.

3.5 Intermediate Conclusion

Based on the above evaluation, we can conclude that (i) substantial energy gains seem achievable with NDN in IoT using uncoordinated sleeping and cooperative caching with CoCa and (ii) by exploiting names and time-stamps for the caching and replacement strategy in CoCa, we can achieve better performance compared to random caching.

In the following, we seek to validate these conclusions: we carry out further experiments with larger ICN deployments and

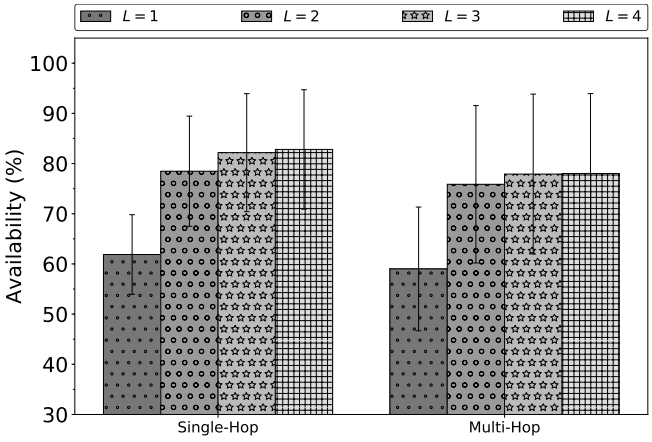


Figure 5: Comparing availability using NDN+CoCa in a single-broadcast *vs* a multi-hop scenario. Results for $p = 0.95$, *uncoordinated sleeping* and varying data lifetimes L. The cache can store up to 80 entries per node.

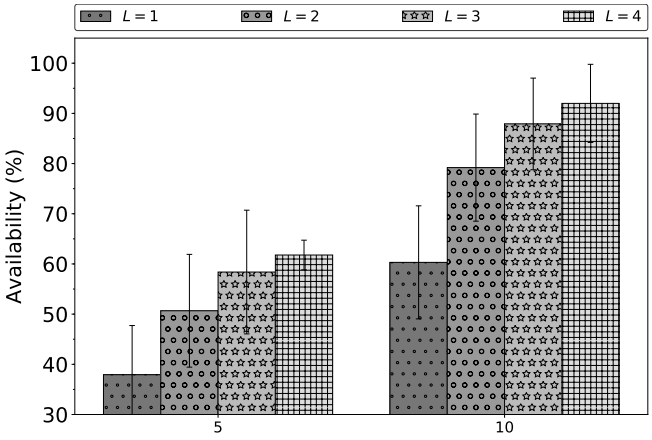


Figure 6: Comparing availability with NDN+CoCa for different cache sizes (CS). Results $p = 0.8$, *uncoordinated sleeping*, and different tolerated lifetimes L.

we design practical mechanisms to auto-configure arbitrary-sized deployments, so that they achieve the projected energy efficiency using uncoordinated sleeping and NDN+CoCa.

4 AUTOCONFIGURATION SCHEMES

One of the biggest hurdles with IoT is device auto-configuration. The need for automatic self-configuration stems from both to the (large) scale of IoT deployments, and to the inherent lack of human in the loop, at the device level. In this section, we thus design and implement the necessary mechanisms so that, subject to the desired content availability ratio A and tolerated data lifetime L, each device in an arbitrary-sized IoT deployment can completely and autonomously configure itself to achieve the energy efficiency shown in Section 3.

4.1 Autoconfiguration of Names

To bootstrap an IoT deployment with ICN, content naming first has to be configured. To allow for automatic self-configuration on each device, each name must be derivable locally and must satisfy the requirements of (i) meaningfulness, and (ii) uniqueness. In order to satisfy the first requirement, we can use a prefix that is derived from sensor type identifier and a unique identifier of the node, e.g. a vendor ID. A modern IoT operating system provides the necessary interfaces for this purpose. For instance, we use RIOT, which provides **(i)** a device driver API offering functions to read the unique CPU ID and/or the hardware address of the network interface, and **(ii)** a high-level API called Sensor Actuator Uber Layer (SAUL), enabling upper layers to request information about all connected sensor devices (e.g., type and name of the sensor). To fulfill the second requirement, we extend the prefix of the name by a suffix, the timestamp, which can also serve as a version number. The name could be enhanced with further information, e.g., based on geographical or organizational properties. A name generated by our autoconfiguration mechanism then looks like */hum/DEADBEEF/1466250645*. Such names have the advantage to require no prior manual configuration: they are generated on the fly, on each node, and they are exploitable in practice to achieve the energy savings shown in section 3. For the experiments and the results described in the rest of the paper, we have implemented and used this scheme to auto-configure names.

4.2 Sleep Ratio Auto-Configuration

To benefit from the energy savings achievable with cooperative caching shown in section 3, the sleep ratio p must be (auto)configured such that it provides the desired content availability ratio A. For this purpose, we design a simple theoretical model predicting the availability ratio A, based on the sleep ratio p.

In our model, we consider n equal nodes that act as data sources and simultaneously provide some caching capacity. There are $|S| \geq n$ data sources (as one node may host several sensors). Each source produces new data periodically with period T, and we assume data has a common lifetime $L \geq T$. Cached data is replaced whenever new data from the source arrives or the lifetime of the data is exceeded. Whenever a new sensor value is observed, the host node performs a link-local broadcast attempting to cache the new chunk in all caches of currently active nodes (including its own on-board cache). Neighbors are likely to sleep, but are awake with a common probability $1 - p$.

4.2.1 *Analysis of Random Caching.*

With this approach, new chunks received at an active node's content store via CoCa are cached with probability $p_c < 1$. To simplify our model, we neglect radio interferences. Since nodes are uncoordinated, we can assume independence of nodes and caches. Data replication initiated by sources can thus be modeled as a Bernoulli experiment with success probability $p_s = (1 - p) \cdot p_c$ at each replicator.

Let R_i be the actual number of replicas for the most recent sensor value of source i. Then R_i follows a binomial distribution with

$$\mathbb{P}[R_i = r] = \binom{n-1}{r} {p_s}^r (1 - p_s)^{n-1-r}. \quad (1)$$

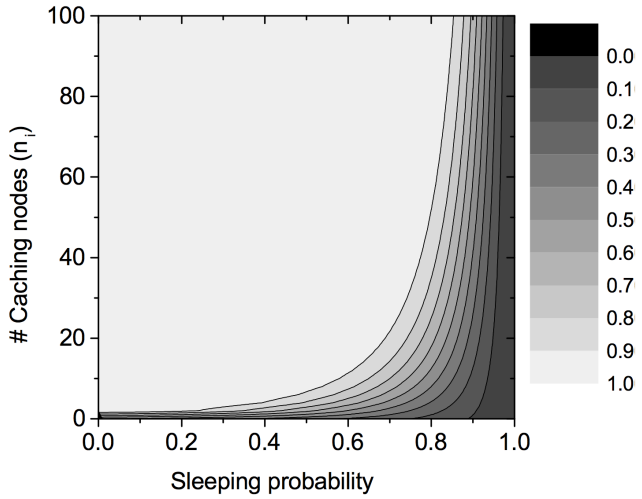


Figure 7: Content availability A, with MDMR, for $L = 1$.

Hence, on average each content item is stored at the source and in caches throughout the network, i.e.,

$$\mathbb{E}[content\ multiplicity] = 1 + \mathbb{E}[R_i] = 1 + (n-1)p_s. \quad (2)$$

Let $R_i(L)$ be the actual number caches which contain *any* of the L most recent sensor value of source i. We can derive this distribution from Equation 1, modified such that it captures L 'losses' within r trials, i.e.,

$$\mathbb{P}[R_i(L) = r] = \left(\begin{array}{c} n-1 \\ r \end{array} \right) {p_s}^r (1-p_s)^{(n-1)+(L-2)r}. \quad (3)$$

Whenever content is requested from the uplink, our network carries data replicated according to Equation 3, but nodes are likely to sleep. A content item is only available, when at least one caching node is awake, i.e., with probability

$$A = \mathbb{P}[content\ availability] = 1 - \sum_{r=0}^{n-1} p^{(r+1)}\, \mathbb{P}[R_i(L) = r]$$

Which simplifies to:

$$A = 1 - p\left(1 - p_s + p \cdot p_s\,(1-p_s)^{L-1}\right)^{n-1}. \quad (4)$$

4.2.2 Analysis of MDMR.
With this approach, we assume that a fixed number n_i of designated caching nodes is selected for each content source i. In detail, if a node is active and receives via CoCa a new content chunk with a name matching its designated content, the node caches it in its Content Store ($p_c = 1$). The problem then decomposes into node groups of sizes n_i, within which caching decisions depends only on the probability $1 - p$ that a node is active. Equation 4 can then be simplified as:

$$A = 1 - p\left(p + (1-p)p^L\right)^{n_i - 1}. \quad (5)$$

Based on Equation 5, the predicted content availability for $L = 1$ is shown Figure 7.

4.2.3 Autoconfiguration of Sleep-Activity Ratio.
We verify experimentally in Section 5 that data availability predicted by the above model does match real data availability.

We estimate n (respectively n_i) locally, on each IoT device by reusing the NDN Interest-response scheme (thus requiring no additional RAM). Based on first autoconfigured name(s), a node initially sets the prefixe(s) it will prefer caching: for instance, if the name is */hum/DEADBEEF/1466250645*, the node will set its preferred prefix to */hum*. During a bootstrap phase, the uplink first broadcasts Interests to a well-known name */bootsrc* to which each source answers with an empty content chunk using the name(s) autoconfigured as described in Section 4.1. Nodes overhear available content names and may (optionally) modify their prefix caching preference configuration. The uplink then broadcasts Interests to another well-known name */boot*, to which each node with caching capacity (i.e. with a content store) answers with a content chunk listing the names (prefixes) it is dedicated to cache. By counting the number of matches with its own designated content caching names (or prefix) in bootstrap chunks advertised by other caches, a node computes n (respectively n_i). Note that we use Bloom filters to detect potential duplicate advertisements. The rate of false positive duplicate detection with Bloom filters can be tuned such that the error is negligible, while keeping additional RAM requirements to a few hundred Bytes, even for large networks.

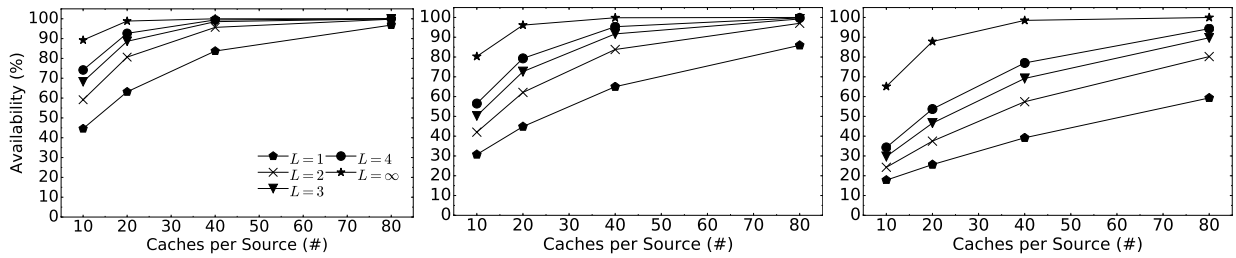
We can now predict (locally on each device) data availability A with NDN+CoCa using the identity provided by Equation 5 if the caching strategy is MDMR (or Equation 4 with Random Caching). We use a bisection method on the value of p to match the required data availability ratio A, within a given error margin. In our implementation we perform 7 iterations, which ensure a value of p within an error of less than 1%. These iterations are computed during a short bootstrap phase, and are completed in a matter of milliseconds on a typical MCU used in IoT (e.g. the ARM Cortex M3 on IoT-Lab testbed devices).

4.2.4 Discussion. The mechanism to autoconfigure p is applicable for stable IoT deployments. In particular, it is not applicable if the number of nodes in the network varies quickly in time. However, if the number of nodes varies slowly, a straightforward approach would be to repeat periodically the bootstrap phase described in 4.2.3.

Assuming each IoT device is (trivially) pre-configured with the desired content availability ratio A and tolerated data lifetime L, the NDN+CoCa network can thus autoconfigure content names and automatically tune p appropriately. All in all, these autoconfiguration mechanisms fit because **(i)** they run locally on low-end IoT devices, **(ii)** the memory, computation and communication overhead they incur is non-significant, and **(iii)** their precision is tunable.

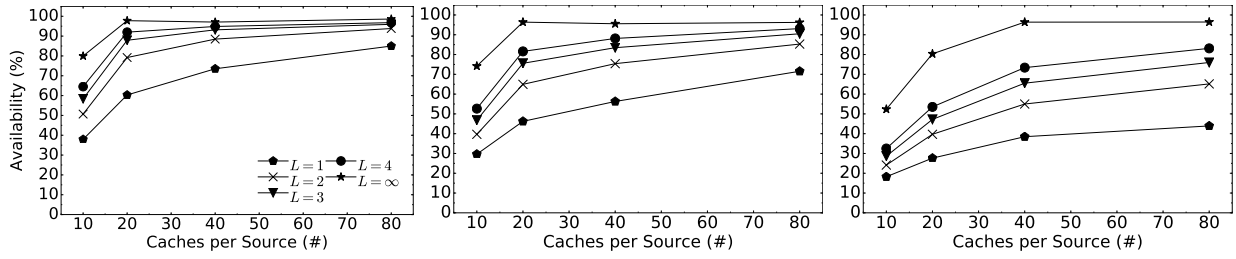
5 LARGE SCALE OPERATION

Combining NDN operation with cooperative caching implemented in Section 2.3 and the autoconfiguration mechanisms implemented in Section 4 we were able to deploy and operate in practice energy-efficient IoT with information-centric networking at larger scale. Based on this approach, we carried out further experiments on ICN networks of 240 up to 1000 nodes.



(a) $p = 0.80$ (b) $p = 0.85$ (c) $p = 0.90$

Figure 8: Theoretical results: Availability as a function of cache numbers n_i for various sleep probabilities p. The x-axis indicates the maximum number of possible cache entries that can store a particular content.



(a) $p = 0.80$ (b) $p = 0.85$ (c) $p = 0.90$

Figure 9: Experimental results: Availability as a function of cache numbers n_i for various sleep probabilities p.

5.1 Model Validation

First, we validate the model we employed in Section 4.2.2 to design our autoconfiguration mechanism. As a derivation from our model, we vary two parameters: (i) the sleep probability p and (ii) the number of caching nodes per source. In practice, the upper limit for this second parameter is given by the memory constraint of the node. We evaluate the availability of content items with respect to different lifetimes (L). Comparing the results from the experiments in Figure 9 to the values derived from the model as depicted in Figure 8, we see that both show very similar trends. However, we see that the availability in the testbed results is slightly below the results from the model, in particular for a high sleeping probability. These small deviations may be caused by link layer effects, such as interference or packet loss, which are not part of the theoretical model.

5.2 Impact of Data Lifetime

Next, we compare the results of NDN+CoCa with *uncoord. sleeping* against the baseline using *coord. sleeping*. Figure 10 reveals that the *uncoord. sleeping* approach can achieve a similar high availability compared to the *coord. sleeping* for values of $L \geq 3$, but gets outperformed by the *coord. sleeping* approach for smaller values of L.

However, tolerating larger values of L means tolerating decreased freshness of data on average, which may not be desired. A way to mitigate this unwanted side-effect is to enhance CoCa with *Source-Based Replication* (SBR). With this strategy, a source periodically rebroadcasts *middle-aged* content. In detail, a source that has just produced and broadcasted a new content item sets a timer to be woken up again after some time t at which point it rebroadcasts this content (somewhat aged already, but still its freshest) via CoCa. t is ideally chosen in a way so that the timer fires before the next sensor value is produced. This procedure could be performed multiple times with a smaller t.

In experiments on the testbed, it could be observed that replicating the latest content item only once already improves availability significantly. The results depicted in Figure 11 were gathered from an experiment with the same settings as the one depicted in Figure 9*a* ($p = 0.80$) and reveal an improvement of availability by up to 20%.

It is possible to extend our model and our autoconfiguration mechanism for the sleep-activity ratio to capture the effect of SBR, as SBR essentially doubles the number of caching attempts per new chunk of data.

5.3 Impact of Hardware

Next, we studied the impact of heterogeneous IoT hardware on the energy-efficiency of our approach. We thus compared average energy consumption with NDN+CoCa on (newer) M3 IoT-LAB nodes shown Figure 4, with energy consumption on (older) WSN430 IoT-LAB nodes, shown in Figure 12.

We observe significant differences. On newer IoT hardware, we can reduce the energy consumption by about 90% compared to the baseline without affecting the data availability (compare Figure 10). Comparably, on older IoT hardware, shown in Figure 12, we see that RDC alone has a much bigger impact. Furthermore, we observe

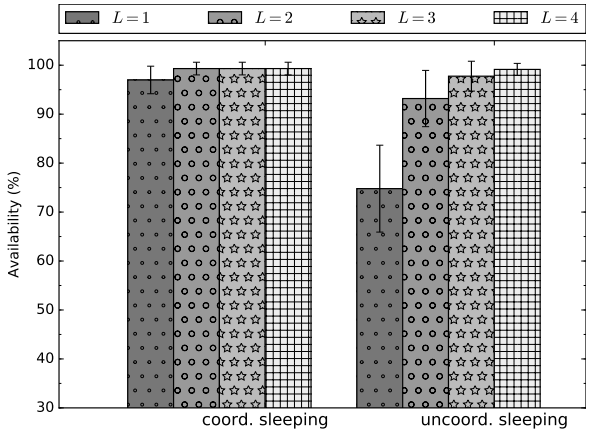


Figure 10: Comparing availability with NDN+CoCa using *coord.* or *uncoord. sleeping*, for different values of L. The interval of the active cycle was set to 30 s in the *coord. sleeping* approach and $p = 0.8$ for the *uncoord.* one.

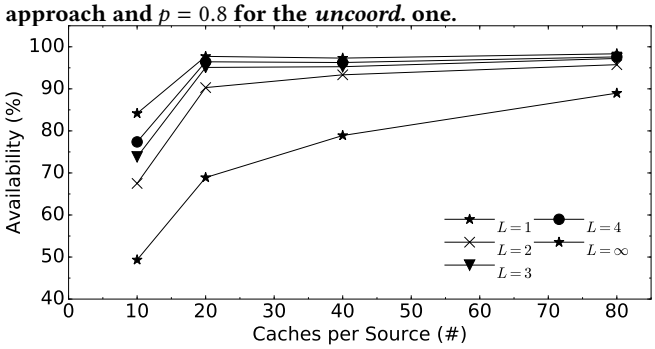


Figure 11: Availability as a function of n_i with source based replication.

that *coord. sleeping* approach perform relatively much worse. A reason for this difference is that newer MCUs consume comparably more energy when active, while being very efficient when sleeping. Another reason is that sending with newer transceivers has become cheaper energy-wise. Hence, our proposed approach whereby a node does not only perform RDC, but can also power down the CPU for most of the time, is much very energy efficient—even taking the additional traffic into account.

5.4 Very Large Networks

Using NDN+CoCa with a caching strategy such as MDMR, each cache entry is *hardwired* to a particular source identified by its name. This has two drawbacks in large networks (i) each cache needs to be pre-configured with a large number of names, or needs to somehow gather this information during bootstrap, and (ii) a node needs to perform full name matching per received content chunk. The former is tedious at small scale and impossible at large scale. The latter wastes (scarce) energy on low-end IoT devices.

Hence we explored alternatives exploiting name structure.Using the name autoconfiguration mechanism we developed in Section 4, content names are naturally split into batches matching *prefixes* which distinguish sensor type. A node can thus autoconfigure itself to cache content that have a name prefix matching its preferred

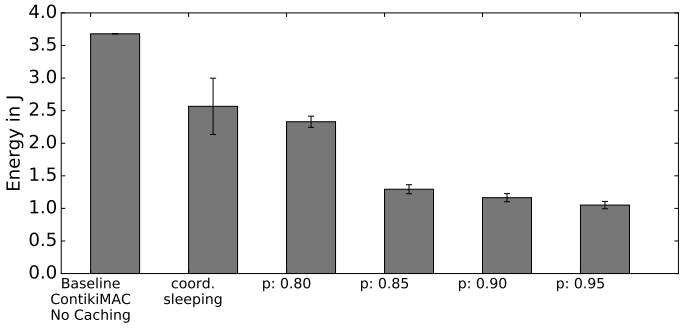


Figure 12: Older IoT hardware: Average energy consumption per node for *coord.* and *uncoord. sleeping* approaches on an MSP430 based node. ContikiMAC is used for RDC. Active MCU consumes 0.5 mA, listening consumes 19.9 mA and a broadcasts costs approx. 2.8 mJ.

sensor type (computationally much less expensive than matching the full name). In the following, we call this caching strategy P-MDMR.

In Figure 13 we compare content availability in a similar setting as in the previous section, looking only at the case for $p = 0.9$ and $L = 4$. The nodes are equipped with three different types of sensors, i.e. they can be grouped in three different prefix classes (/temp, /hum, /light). We observe that NDN+CoCa with the simplified caching strategy P-MDMR achieves even better availability than with MDMR.

Finally, we conducted experiments on RIOT *native* emulating 1,000 nodes (this time deploying five different sensor types). We first compared the results for MDMR in a network with 240 nodes in the testbed to results on the emulated network with 1,000 nodes, using the same setup. From Figure 14 we observe that results on the testbed match results on the emulator. Then we compared MDMR against P-MDMR for the same 1,000 nodes network on the emulator. In Figure 15 we see that availability significantly improves with P-MDMR, close 100% while $p = 0.9$, which means that nodes sleep 90% of the time.

6 RELATED WORK

Caching in Sensor Networks. Early work in sensor networks proposed on-path caching [12, 32] to reduce the need for end-to-end retransmissions at the transport layer in multi-hop wireless sensor networks (WSN). More recently, a content-aware diffusion mechanism was proposed for WSN leveraging on-path caching [18]. A similar approach is recently the focus of a growing community: the information-centric networking paradigm [2] proposes communication that is not host-centric and conversational such as with TCP/IP, but content-centric and completely connectionless.

Cooperative Caching. Cooperative caching has been discussed in several network scenarios, including ICN [7, 9, 23, 26, 27, 33]. Those approaches assume caches under multiple administrative authorities, which is different from common wireless IoT scenarios as we envision in this paper. We thus neither ask the question how to force nodes to cooperate, nor do we need to solve a global optimum heuristically [7, 9, 23, 26, 27]. Furthermore, our approach

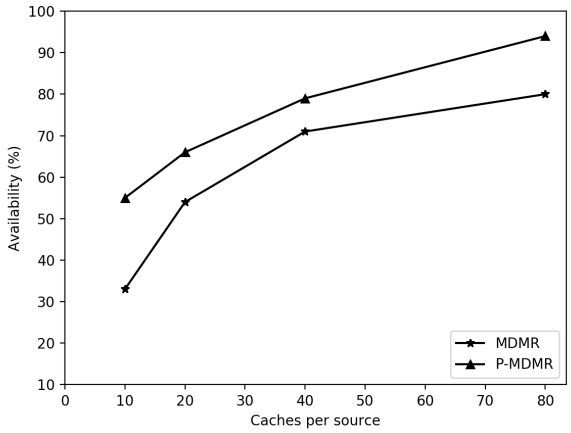


Figure 13: Availability with NDN+CoCa as a function of n_i, for the caching strategies MDMR, and P-MDMR. $p = 0.90$ and $L = 4$.

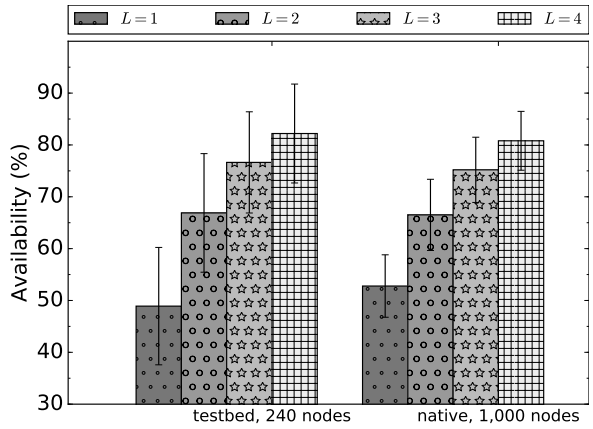


Figure 14: RIOT Emulator with 1,000 nodes: Comparing availability on a network with 240 nodes in the testbed to the *native* emulator with 1,000 nodes for various values for L. $p = 0.90$.

does not introduce additional signaling overhead for coordination as it is common for distributed algorithms in this context [33].

Coordinated vs *Coordinated Sleeping.* Radio duty cycling mechanisms were proposed by prior work to reduce power drain due to idle radio listening. TSCH [34] combines frequency hoping and synchronized sleeping based on TDMA, while ContikiMAC [11] proposes uncoordinated sleeping combined with CSMA. Other approaches use (shared) pseudo-random number generators [10] [25] to synchronize on-off radio schedules.

ICN & IoT. First experiments with NDN on an IoT testbed hinted at potential memory- and energy-efficiency gains compared to the traditional 6LoWPAN approach, but stopped short of studying caching and replacement strategies [5]. An NDN optimisation was proposed to exploit the wireless broadcast nature of IoT networks to retrieve content from multiple producers with a single interest,

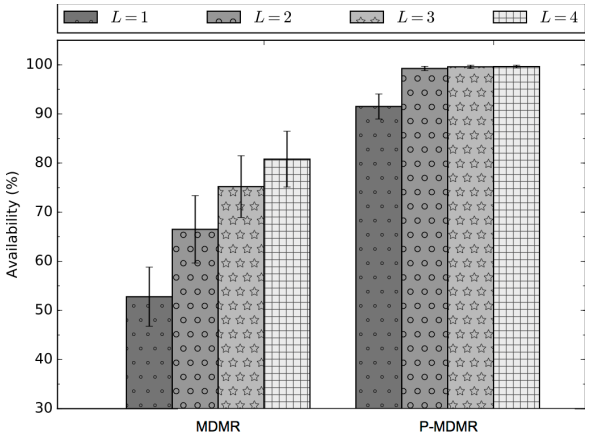


Figure 15: RIOT Emulator with 1,000 nodes: Availability as a function of n_i for MDMR and P-MDMR. $p = 0.90$.

using persistent PIT entries [3] . Complementary mechanisms to adapt NDN to information freshness requirements specific to IoT sensor data were studied in [24]. A high-level overview of advantages, trade-offs and challenges of information-centric networking for IoT was published in [20].

The closest related work is [17], which includes a study of a basic random caching strategy with LRU and observe performance gains in content delivery, via simulations on a grid topology. In [16], we had shown preliminary experimental results for a distributed caching strategy in small networks. In contrast, in this paper we present results for large networks and more varied strategies, as well as a theoretical model and autoconfiguration mechanisms. To the best of our knowledge, there is no prior work on advanced distributed caching strategies in IoT on real hardware, that addresses the trade-off of data availability and energy efficiency on large scale networks. In this paper, we focused on this problem.

7 CONCLUSION

In this paper, we have proposed and studied NDN+CoCa, a variety of mechanisms for content-centric, decentralized, cooperative caching replication of IoT content leveraging NDN. These mechanisms allow to capture most of the phenomenons observed by IoT devices' sensors in common IoT scenarios, while draining drastically less energy compared to prior art. We have developed and analyzed a theoretical model capturing such mechanisms. Our analysis derives simple identities relating sleep/activity ratio with IoT content availability. These identities allowed us to design autoconfiguration mechanisms to tune automatically, and autonomously each device, in order to achieve the projected energy efficiency with NDN+CoCa. We have implemented these caching replication mechanisms as extensions of the NDN protocol supported on RIOT, a popular software platform for low-end IoT devices. We carried out extensive experiments with this implementation, both on real hardware with hundreds of IoT devices, and on an emulator with up to 1,000 emulated IoT devices. We show that content-centric, cooperative caching mechanisms can achieve an order of magnitude reduction in energy consumption, while maintaining tolerably recent content availability above 90%.

REFERENCES

[1] Cedric Adjih and others. 2015. FIT IoT-LAB: A large scale open experimental IoT testbed. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*. IEEE, 459–464.

[2] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Börje Ohlman. 2012. A Survey of Information-Centric Networking. *IEEE Communications Magazine* 50, 7 (July 2012), 26–36.

[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*. ACM, 67–76.

[4] Emmanuel Baccelli, Oliver Hahm, M Gunes, M Wahlisch, and Thomas C Schmidt. 2013. RIOT OS: Towards an OS for the Internet of Things. In *Computer Communications Workshops (INFOCOM), 2013 IEEE Conference on*. IEEE, 79–80.

[5] 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 *Proc. of 1st ACM Conf. on ICN*. ACM, New York, 77–86.

[6] C. Bormann, M. Ersue, and A. Keranen. 2014. *Terminology for Constrained-Node Networks*. RFC 7228. IETF.

[7] Sem Borst, Varun Gupta, and Anwar Walid. 2010. Distributed Caching Algorithms for Content Distribution Networks. In *Proc. of IEEE Infocom*. IEEE, 1–9.

[8] CCN Lite 2014. CCN Lite: Lightweight implementation of the Content Centric Networking protocol. (2014). http://ccn-lite.net

[9] Jia Dai, Zhan Hu, Bo Li, Jiangchuan Liu, and Baochun Li. 2012. Collaborative Hierarchical Caching with Dynamic Request Routing for Massive Content Distribution. In *Proc. of IEEE Infocom*. IEEE, 2444–2452.

[10] LL Dai and P. Basu. 2006. Energy and Delivery Capacity of Wireless Sensor Networks with Random Duty-Cycle. In *ICC*. IEEE, 3503–3510.

[11] Adam Dunkels. 2011. *The contikimac radio duty cycling protocol*. Technical Report. Swedish Institute of Computer Science.

[12] Adam Dunkels, Juan Alonso, and Thiemo Voigt. 2003. *Making TCP/IP viable for wireless sensor networks*. Technical Report. Swedish Institute of Computer Science.

[13] Adam Dunkels, Luca Mottola, Nicolas Tsiftes, Fredrik Österlind, Joakim Eriksson, and Niclas Finne. 2011. The announcement layer: Beacon coordination for the sensornet stack. In *European Conference on Wireless Sensor Networks*. Springer, 211–226.

[14] Oliver Hahm and others. 2016. Operating systems for low-end devices in the internet of things: A survey. *IEEE Internet of Things Journal* 3, 5 (2016), 720–734.

[15] Oliver Hahm, Cédric Adjih, Emmanuel Baccelli, Thomas C. Schmidt, and Matthias Wählisch. 2016. ICN over TSCH: Potentials for Link-Layer Adaptation in the IoT. In *Proc. of 3rd ACM Conf. on Information-Centric Networking (ICN 2016), Poster Session*. ACM, 195—196.

[16] Oliver Hahm, Emmanuel Baccelli, Matthias Wählisch, Thomas C. Schmidt, and Cedric Adjih. 2016. A Named Data Network Approach to Energy Efficiency in IoT. In *IEEE GLOBECOM Workshops: Information Centric Networking Solutions for Real World Applications (ICNSRA)*. IEEE, Washington, USA, 1–6.

[17] 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 *Wired/Wireless Internet Communications*. Springer, 313–326.

[18] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. 2000. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 56–67.

[19] 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*. ACM, 1–12.

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

[21] S. Y. Oh, D. Lau, and M. Gerla. 2010. Content Centric Networking in tactical and emergency MANETs. In *2010 IFIP Wireless Days*. IEEE, 1–5.

[22] H. Petersen, M. Lenders, M. Wählisch, O. Hahm, and E. Baccelli. 2015. Old Wine in New Skins? Revisiting the Software Architecture for IP Network Stacks on Constrained IoT Devices. In *ACM MobiSys Workshop on IoT Challenges in Mobile and Industrial Systems (IoT-Sys)*. ACM, ACM, 31–35.

[23] Ioannis Psaras, Wei Koong Chai, and George Pavlou. 2012. Probabilistic In-Network Caching for Information-Centric Networks. In *Proc. of ACM ICN Workshop*. ACM, 55–60.

[24] Jose Quevedo, Daniel Corujo, and Rui Aguiar. 2014. Consumer-driven information freshness approach for content centric networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 482–487.

[25] J. Redi and others. 2008. JAVeLEN : An ultra-low energy ad hoc wireless network . In *Ad Hoc Networks*. Elsevier, 108–126.

[26] Elisha J. Rosensweig and Jim Kurose. 2009. Breadcrumbs: efficient, best-effort content location in cache networks. In *Proc. of IEEE Infocom*. IEEE, 2631–2635.

[27] Sumanta Saha, Andrey Lukyanenko, and Antti Ylä-Jääski. 2013. Cooperative Caching through Routing Control in Information-Centric Networks. In *Proc. of IEEE Infocom*. IEEE, 100–104.

[28] D. Schmidt, M. Krämer, T. Kuhn, and N. Wehn. 2007. Energy modelling in sensor networks. *Advances in Radio Science* 3, 5 (June 2007), 347–351.

[29] Wenato Shang, Alex Afanasyev, and Lixia Zhang. 2016. The Design and Implementation of the NDN Protocol Stack for RIOT-OS. In *Proc. of IEEE GLOBECOM 2016*. IEEE, Washington, DC, USA, 1–6.

[30] Zach Shelby, Klaus Hartke, and Carsten Bormann. 2014. *The constrained application protocol (CoAP)*. RFC 7252. IETF.

[31] Zhengguo Sheng, Shusen Yang, Yifan Yu, Athanasios V Vasilakos, Julie A McCann, and Kin K Leung. 2013. A Survey on the IETF Protocol Suite for the Internet of Things: Standards, Challenges, and Opportunities. *Wireless Communications, IEEE* 20, 6 (2013), 91–98.

[32] Fred Stann and John Heidemann. 2003. RMST: Reliable data transport in sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*. IEEE, 102–112.

[33] Liang Wang, Gareth Tyson, Jussi Kangasharju, and Jon Crowcroft. 2016. FairCache: Introducing Fairness to ICN Caching. In *Proc. of IEEE ICNP*. IEEE, 1–10.

[34] Thomas Watteyne and others. 2016. Industrial Wireless IP-Based Cyber-Physical Systems. *Proc. IEEE* 104, 5 (2016), 1025–1038.