Computer Communications 164 (2020) 114–123

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Designing a LoWPAN convergence layer for the Information Centric Internet of Things

Cenk Gündoğan [a,*], Peter Kietzmann [a], Thomas C. Schmidt [a], Matthias Wählisch [b]

[a] Hamburg University of Applied Sciences, Department of Computer Science, Berliner Tor 7, 20099 Hamburg, Germany
[b] Freie Universität Berlin, Inst. of Computer Science, Takustr. 9, 14195 Berlin, Germany

ARTICLE INFO

Keywords:
Information Centric Networking
Wireless sensor networks
ICNLoWPAN
Protocol design and evaluation

ABSTRACT

The low-power Internet of Things (IoT) introduces lossy radio links with ultra-constrained frame sizes and high transmission cost for each byte. Information Centric Networking (ICN) is considered a promising communication technology in this regime, as it increases reliability by ubiquitous caching and eases transmission efforts by hop-wise forwarding. Common ICN layers such as NDN, however, were designed for fixed network infrastructure and require an adaptation layer to the constrained wireless—as the common Internet Protocol does.

In this paper, we design and evaluate such an ICN convergence layer for low power lossy links that (1) augments the NDN stateful forwarding plane with a highly efficient name eliding, (2) devises stateless compression schemes for standard NDN use cases with utile data encodings, (3) adapts NDN packets to the small MTU size of IEEE 802.15.4, and (4) generates compatibility with 6LoWPAN so that IPv6 and NDN can coexist on the same LoWPAN links. Our findings indicate that stateful compression can reduce the size of NDN data packets by more than 70 % in realistic examples, while packet fragmentation operates in a predictable way even for high fragment numbers. Our experiments show that for common use cases ICNLoWPAN saves 33 % of transmission resources over NDN, and about 20 % over 6LoWPAN.

1. Introduction

The Internet is evolving toward new applications and deployment regimes. Its wireless edge has been the particular focus of development during the past decade with billions of networked mobile devices in operation today. Tomorrow, the Internet of Things (IoT) will extend this wireless edge even further by low-power embedded sensors and actuators, which often are unattended, highly constrained devices that run on small batteries. Low-power components in the processing and transmission have paved the way to the expected massive deployment of ultra-constrained 'things' on the Internet.

These smart things are foreseen to mainly exchange confined chunks of data in challenged environments—a potential deployment area for Information Centric Networks (ICN) [1]. Infrastructureless access to content, resilient hop-by-hop forwarding, and in-network data replication demonstrated notable advantages over the traditional host-to-host approach on the Internet [2–5]. Named Data Networking (NDN) [6,7] has matured to a prominent, widely implemented flavor of ICN that strictly couples data delivery to consumer requests. Recent studies [8] have shown that NDN can outperform CoAP [9] and MQTT-SN [10], the corresponding IP-based data services for the constrained IoT. In addition, quality of service measures in combination with loosely coordinated caches proved surprisingly beneficial for all NDN users [11]. Unlike IPv6, the NDN protocol lacks mechanisms for packet adaptation and compression that comply with limitations of wireless links prevalent at the low-power IoT edge [12].

Common access networks in the constrained IoT are built from low power and lossy radios (see "LLN" in [13]) such as IEEE 802.15.4 [14], Bluetooth Low Energy (BLE) [15], or LoRA [16]. Characteristics of LLNs include an unreliable environment, low bandwidth transmissions, and increased latencies. IEEE 802.15.4 admits a maximum physical layer packet size of 127 octets. With 6LoWPAN [17,18], the IP-world has created a convergence layer that provides appropriate frame encapsulation formats, packet header compression and link fragmentation for IPv6 packets in IEEE 802.15.4 networks. The ICN world has not yet developed corresponding features for constrained environments.

This paper presents the design of ICNLoWPAN, a full-featured protocol convergence layer for an information centric IoT. It closes the gap for NDN deployment on LoWPANs by extending our previous work [19] with more efficient data encoding and fragmentation. We leverage the NDN potential of stateful forwarding to elide names on paths

* Corresponding author.
E-mail addresses: cenk.guendogan@haw-hamburg.de (C. Gündoğan), peter.kietzmann@haw-hamburg.de (P. Kietzmann), t.schmidt@haw-hamburg.de (T.C. Schmidt), m.waehlisch@fu-berlin.de (M. Wählisch).

https://doi.org/10.1016/j.comcom.2020.10.002
Received 15 February 2020; Received in revised form 4 September 2020; Accepted 7 October 2020
Available online 12 October 2020

and design highly efficient compression primitives that outperform 6LoWPAN. Our evaluations reveal significant gains in packet reduction, energy consumption, and reliability. In addition to stateful and stateless compression, we also contribute a fragmentation scheme as well as a framing compatible to the 6LoWPAN techniques. Real implementations under RIOT OS [20,21] and experimentation on a testbed of current IoT hardware demonstrate the feasibility, robustness and energy efficiency of our approach.

The remainder of this paper is organized as follows. The subsequent Section 2 discusses the problem of NDN LoWPAN adaptation and related work. In Section 3, we introduce our ICNLoWPAN convergence layer and detail out IPv6-ICN coexistence, on-link fragmentation, time encoding, and compression. A thorough theoretical and experimental evaluation of (*i*) compression benefits, (*ii*) encodings, and (*iii*) the impact of fragmentation follows in Section 4. Finally, we conclude with an outlook in Section 5.

2. Problem space and related work

The Internet of Things inherently connects numerous devices of substantial heterogeneity. In this work, we focus on deployment use cases that bundle low-end and battery-operated microprocessors in wireless networks, where packet transmission distinctly dominates power consumption. The challenges we face in such scenarios are manifold and range from limited MTUs, lossy links and mobility to link layers that lack basic protocol features, such as frame encapsulation formats (cf., EtherTypes in Ethernet).

NDN couples name-based routing from TRIAD [22] with stateful forwarding from DONA [23] and seamlessly leverages in-network caching on the forwarding plane. Three deployment scenarios for NDN are currently envisioned: (*i*) the leanest deployment runs NDN directly on top of a link layer, which requires a mapping of names to MAC addresses [24], (*ii*) NDN as an IP overlay allows to operate in the existing Internet infrastructures, and (*iii*) an integration of NDN encodings into IPv6 headers (see hICN [25]) can display synergies from the information-centric and the host-centric world. There is an additional inverse proposal that runs an IP overlay on top of NDN to continue the use of established, IP-bound applications and services, while taking advantage of loosely coupled content replication and in-network caching [26].

The fundamental request–response semantic on the network layer of NDN requires an *Interest* message and a returning *data* message. Both message types utilize flexible Type-Length-Value (TLV) header fields to allow for generic and extensible packet formats to the cost of space efficiency. Name TLVs are essential to NDN and thus always appear in Interest as well as in data messages. Depending on the naming scheme, human-readable Name TLVs make up the largest part of a request, and also of a response for many IoT use cases. We explore related work that copes with strict message length limitations using header compression and fragmentation in IP networks first, and then discuss proposed solutions for NDN.

IPv6 mandates a minimum MTU of 1280 bytes for each link and thus precludes a proper IPv6 operation in low power networks with small-sized MTUs. The IETF designed and extends a set of protocols for constrained IoT deployments where the 6LoWPAN convergence layer is an integral part of. It is situated below the network layer and provides packet encapsulation, stateless and stateful header compression as well as a protocol independent link fragmentation scheme. A generic header compression (GHC) [27] extends 6LoWPAN with an LZ77 flavored approach to deal with headers and header-like payloads that are not covered by the 6LoWPAN compression specification. While 6LoWPAN proves necessary for an interoperable and interconnected host-centric IoT, the same challenges remain open for information-centric IoT deployments.

Shang et al. [28] proposed a lightweight link fragmentation scheme that prepends a 3-byte fragmentation header to each NDN fragment to allow for messages larger than the limiting MTU of IEEE 802.15.4. This custom header further supports a minimal protocol identification by distinguishing normal NDN messages from fragmented messages. However, this protocol encapsulation collides with the 6LoWPAN encapsulation and thus disregards interoperability with 6LoWPAN, especially in multi-interface and multi-protocol deployments.

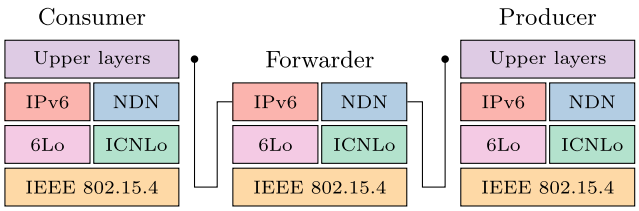


Fig. 1. Stack traversal in a 6LoWPAN and ICNLoWPAN.

Another approach was presented by Mosko et al. [29], which extends NDN with a new message type that encapsulates each message fragment. It also adds complexity to the state machine for each peer to initialize fragment sequence numbers and perform corrective actions in case of drifting sequences after packet loss. A similar approach is the NDN Link Protocol (NDNLP) [30], which features fragmentation and reassembly as well as ARQ mechanisms. It provides packet encapsulation to distinguish between normal NDN messages and link acknowledgments. The last two approaches add overhead in terms of memory consumption and error-control messages which is a disadvantage for low power use cases.

A secure fragmentation for content-centric networks that does not rely on hop-by-hop reassembly and therefore decreases latency was derived by Ghali et al. [31]. Each fragment is securely signed according to NDN semantics and can be cached on intermediate routers. The authors propose a new ContentFragment message type that includes a Name TLV for forwarding purposes. Since NDN names are theoretically of unlimited length, duplicating names for each ContentFragment message adds a significant overhead, which naturally is controversial in constrained IoT networks. Individual fragment forwarding in the IP world has also shown several unwanted effects, which degrade network performance [32]. Furthermore, sporadic disruptions and mobility in LLNs do not guarantee a successful handover of each individual fragment to complete the reassembly.

Yang et al. [33] focused on bandwidth reduction and an improved storage utilization by translating long names into short names for local communication. In this regard, a sensor node registers a prefix at a sink node and receives a shorter prefix, e.g., a hash as a name replacement. Ingress messages that traverse the sink node are updated to include the short name and egress messages respectively are updated to include the long name. The proposed local name translation is transparent to nodes outside the local IoT network. We argue nevertheless that a centralized approach to handle name translations with registration and cancellation procedures adds complexity, which limits the scalability in large deployments.

In the following section, we concentrate on a fully distributed convergence layer that preserves compatibility with IP LoWPAN, but takes advantage of the information-centric characteristics to obtain a generically applicable, efficient though uncomplex compressive encoding named ICNLoWPAN.

3. ICNLoWPAN

ICNLoWPAN provides a convergence layer that maps ICN packets onto constrained link layer technologies to enable pure NDN deployments without running as an overlay on top of IP. Our convergence layer includes features such as link fragmentation, protocol separation on the link layer level as well as stateless and stateful header compression mechanisms. Fig. 1 shows the overall network stack of a

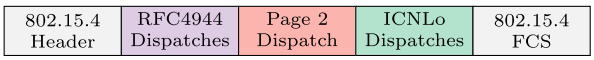


Fig. 2. IEEE 802.15.4 encapsulated ICNLoWPAN message.

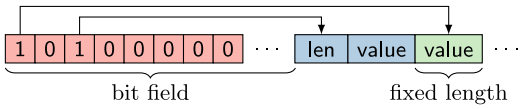


Fig. 3. Eliding *type* and *length* fields using compact bit fields.

6LoWPAN and an ICNLoWPAN deployment in parallel for a consumer, a forwarder, and a producer. Both convergence layers are situated between the link layer and the actual network layer, such that each message traverses them. This allows for a transparent operation without the need for modifications on the network layer.

3.1. 6LoWPAN dispatching framework

6LoWPAN defines a dispatching framework [17], where each frame is prepended with a 1-byte dispatch type and a possible dispatch header. Several dispatch types exist already, e.g., for stateless IPv6 compression, a mesh header for mesh-under routing purposes, link fragmentation, and extensions to expand the limited universe of possible dispatch types. One of those extensions is the 1-byte page switch dispatch [34], which arranges dispatch types into 16 pages and signals a context switch to the packet parser to choose the proper page before interpreting subsequent dispatch types.

ICNLoWPAN integrates into the 6LoWPAN dispatching framework by defining four new dispatch types for Interest and data messages that are either compressed or uncompressed. Since page 0 and page 1 are already reserved for 6LoWPAN usage, we allocate these dispatches from page 2 to allow for coexistence with 6LoWPAN deployments. A prepended page switch dispatch before the very first ICNLoWPAN dispatch is thus necessary as an indicator to the dispatch parser.

A typical ICNLoWPAN message encapsulated in an IEEE 802.15.4 frame is shown in Fig. 2. RFC4944 dispatches are optional and may include all dispatch types defined in [17]. Note the 1-byte page 2 dispatch before the first ICNLoWPAN dispatch. To switch back to 6LoWPAN dispatches after an ICNLoWPAN dispatch, another page switch dispatch to page 0 or 1 is necessary.

A major benefit of reutilizing the 6LoWPAN dispatching framework is to share a common code base for dispatch handling. Notably for multi-interface and multi-protocol deployments that use IPv6 and NDN simultaneously, having the same code components in resource-constrained devices is exceptionally valuable for minimizing RAM and ROM requirements.

3.2. Fragmentation

Reusing the 6LoWPAN dispatching framework enables ICNLoWPAN to seamlessly benefit from the protocol independent link fragmentation scheme defined in [17]. It is thus possible to fragment large NDN messages to fit the limited maximum physical packet sizes of low power link layers, such as 127 bytes for IEEE 802.15.4.

Practically, a fragmented NDN message includes a 4-byte fragmentation dispatch header that lists the original datagram size and a datagram tag to identify fragments of differing packets. Subsequent fragments further include an additional 1-byte datagram offset of the payload in the dispatch header. Fragments are reassembled on the next hop and passed to the NDN network stack as typical NDN Interest or data messages. The 6LoWPAN fragmentation scheme does not define ARQ mechanisms to recover lost fragments, but rather relies on corrective actions of the link layer. This allows for implementations with minimal memory footprints.

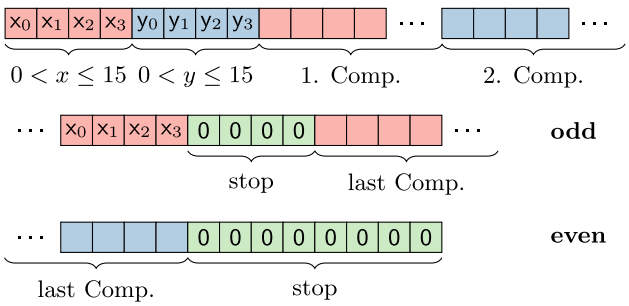


Fig. 4. Stateless name compression and stop marker for odd and even number of name components.

6LoWPAN Fragment Forwarding (6FF) [35] and Selective Fragment Recovery (SFR) [36] are upcoming mechanisms that replace the original fragmentation scheme [17] with refined approaches. Instead of a hop-wise reassembly, 6FF forwards single fragments between endpoints. SFR further adds congestion control capabilities and enables a selective recovery of lost fragments, which has been extensively evaluated in an IoT testbed [32]. These mechanisms open up new possibilities for further optimizations of the ICNLoWPAN integration [37].

3.3. Stateless compression

ICNLoWPAN defines a stateless header compression scheme with the main purpose of reducing header overhead of NDN packets. This is of particular importance for link layers with small MTUs and for increasing energy conservation of battery-operated and wirelessly connected devices. Corresponding dispatch headers in the ICNLoWPAN packet provide the rule set for decompressing NDN messages before handing the packet over to the NDN network stack.

3.3.1. TLV compression

The NDN header format is solely composed of TLV fields to encode header data. The advantage of TLVs is a native support of variable-sized data. The main disadvantage of TLVs is its verbosity that results in two extra bytes for each header field from storing the *type* and *length* of the encoded data.

The stateless header compression scheme of ICNLoWPAN makes use of compact bit fields to indicate the presence of optional TLVs in the uncompressed packet as illustrated in Fig. 3. Each *type* that is present in the bit field is thus elided from the actual TLV representation, which translates to a reduction from 1 byte to 1 bit. Further compression is achieved with eliding the *length* of TLVs that either represent fixed-length header data, or where the length can be assumed from surrounding TLVs. We also achieve smaller encodings by specifying sane default configurations for IoT use cases.

3.3.2. Name compression

A Name TLV is substantial to NDN messages and usually consists of several name components, each of variable size. An Interest message essentially carries the Name TLV and a data message returns either the same TLV, or a more specific variant with an equal prefix. The NDN TLV encoding requires at least two bytes for each name component (*type* + *length*) and an extra two bytes for the outer-most Name TLV. The TLV overhead for a name is displayed in Eq. (1), where $|c|$ is the number of components.

$$TLV\ overhead = 2 + 2 \cdot |c| \tag{1}$$

ICNLoWPAN provides a compression scheme for Name TLVs that drastically reduces the TLV overhead of each nested component. This compression encodes length fields of two consecutive component TLVs into one byte, using 4 bits each as displayed in Fig. 4. This process

limits the length of a component TLV to 15 bytes. To further elide the outer-most *length* field of the name, this scheme utilizes a stop marker. For an odd number of components, the stop marker is encoded into the least significant 4 bits of the current *length* byte. On an even number of components, the full *length* byte is already occupied with two name components. In this case, a stop byte is appended to the last component as shown in Fig. 4.

Compressed names yield a significantly lower TLV overhead as displayed in Eq. (2), where $|c|$ again is the number of components. The ceil operator handles both cases of odd and even for $|c|$.

$$TLV\ overhead = \left\lceil \frac{|c|+1}{2} \right\rceil \tag{2}$$

The total TLV overhead reduction for names thus follows from Eqs. (1) and (2) and is a function of $|c|$. The reduction is therefore given as $\lceil 1.5 \cdot |c| \rceil + 1$.

3.3.3. Efficient encoding of timestamps

NDN utilizes timestamps for various protocol operations and thereby differentiates between absolute and relative values. Relative timestamps denote positive time offsets that add to the absolute message reception time and provide a general sense of recency. Absolute time values indicate the elapsed time since the Unix epoch and grant a higher precision for global time and date values throughout the network.

The current NDN packet format describes three TLVs that encode temporal values. Interest messages optionally carry `InterestLifetime` in relative time. Data messages optionally include `ContentFreshness` in relative time and the `SignatureTime` as an absolute timestamp. All timestamps in respective packet headers are expressed as milliseconds.

At the time of writing, the current absolute time in milliseconds is encoded in 40 bits. Absolute timestamps in TLV form will thus use 8 bytes to denote current or future moments in time. Relative time offsets on the other hand encode in fewer bytes and still provide an effective protocol operation: A 2-byte value represents a maximum offset of around 65 s and a 4-byte value holds a maximum of around 49 days. In typical IoT use cases with long sleep cycles and disrupted network connectivity, the 4-byte form is much more likely to be utilized for the lifetime of Interests and content freshness.

ICNLoWPAN uses a compressed representation for time offsets, *i.e.,* `Interestlifetime` and `ContentFreshness`, similarly to previous work based on CCNx [38]. Instead of a linear representation, ICNLoWPAN encodes time values in dynamic range to limit the size of offsets to a single byte. This form allows for high precision between lower time values and at the same time enables wide time ranges. Following the standard for floating-point arithmetic [39], an 8-bit time code describes the values for exponent and mantissa as shown in the following equations. A configurable number of high-order bits denotes the exponent value and the remaining low-order bits indicate the mantissa value. Eq. (3) shows a subnormal form inspired from IEEE 754 [39], which applies when the exponent value is 0. The subnormal form allows to represent 0 s and fills the underflow gap with time values to enable a gradual underflow. For all other cases, the normalized form in Eq. (4) is used to convert a time code to a time value in seconds. In both equations, $m_{\max}$ denotes the maximum mantissa value that a time code can represent and b is a bias that is statically configured.

8-bit time code

x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7
⟵ exponent (e) ⟶				⟵ mantissa (m) ⟶			

$$t(\mathrm{e}, \mathrm{m}, b) = \begin{cases} \left(0 + \dfrac{\mathrm{m}}{m_{\max}}\right) \cdot 2^{(1+b)}\ s & e = 0 \quad (3) \\ \left(1 + \dfrac{\mathrm{m}}{m_{\max}}\right) \cdot 2^{(\mathrm{e}+b)}\ s & e > 0 \quad (4) \end{cases}$$

time value in seconds

Suitable sizes for exponent, mantissa, and bias determine the effectiveness and applicability of this efficient encoding of timestamps. Section 4.2 explores different configurations and outlines recommended values.

3.4. Stateful compression

ICNLoWPAN further employs two stateful compression schemes to enhance size reductions. These mechanisms rely on shared contexts that are either distributed and maintained in the whole LoWPAN, or are generated and maintained on-demand for a particular Interest-data path. Our stateless and stateful compressions are applied in succession to produce huge compression savings, which we show in Section 4.

3.4.1. LoWPAN-local state

A context identifier (CID) is a 1-byte number that refers to a particular conceptual context between networked devices and may be used to replace frequently appearing information, like name prefixes, suffixes, or meta information, such as an Interest lifetime. This allows for a reduction of potentially long data to a single byte. Shared context has to be initially distributed on compile-time or dynamically maintained on run-time in order for a device to properly encode and decode NDN messages.

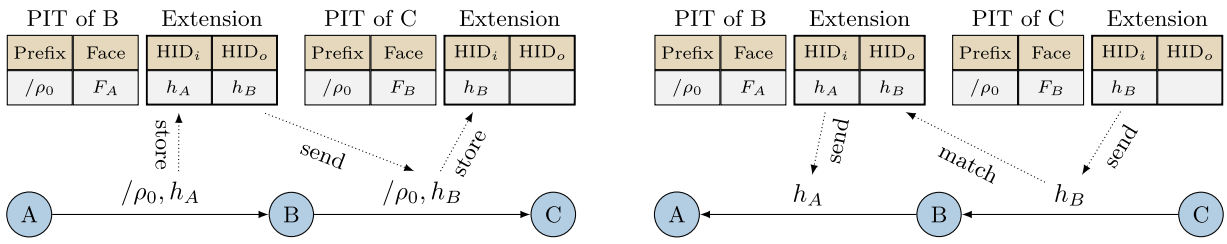
The convergence layer replaces header fields of outgoing Interest and data packets with CIDs maintained in a CID state table. Context identifiers follow the last ICNLoWPAN dispatch, while the most significant bit of a CID signals the presence of a subsequent CID. On reception, the original packet is restored and passed to the network layer.

3.4.2. En-route state

An NDN Interest requests data by a name or a prefix which is then returned in the corresponding data packet. This duplication generates large overhead in particular for long names. To deduplicate we make use of ephemeral 1-byte HopIDs that replace the name in data responses and link them to entries of the *Pending Interest Table (PIT)*. The PIT is a fundamental component of NDN that hop-wise matches the returning responses to open requests by name. HopIDs must be unique within the local PIT and only exist during the lifetime of a PIT entry. We extend the PIT by two new columns to manage these HopIDs. HID_i for inbound HopIDs and HID_o for outbound HopIDs as visualized in Fig. 5. We emphasize that even though PIT entries are extended by two additional bytes, the overall RAM consumption on link commitment reduces due to smaller packets and corresponding message buffers.

Before sending an Interest, a node generates a HopID and stores it in the local HID_o column. This Interest then includes the generated HopID along with the name. On the next hop, the HopID is extracted from the Interest and stored in the HID_i column of the respective PIT entry. The forwarder then generates a new HopID, stores it in the HID_o column of the particular PIT entry, and puts this HopID into the Interest message before it is forwarded to the next hop. This process is repeated for each hop until the request can be satisfied with the corresponding response as displayed in Fig. 5a.

The producer of a returning data message reverses this process by obtaining a HopID from the HID_i column of a PIT entry and encodes it into the response message. If the returning name equals the original name, then it is fully elided. Otherwise, the distinct suffix is included along with the HopID. When a response is forwarded, the contained HopID is extracted and used to match against the correct PIT entry by performing a lookup on the HID_o column. The HopID is then replaced with the corresponding HopID from the HID_i column before forwarding the response, as visualized in Fig. 5b.



(a) Collecting compression state en-route (Interest).

(b) Eliding Name TLVs using en-route state (data).

Fig. 5. Stateful header compression using en-route forwarding state.

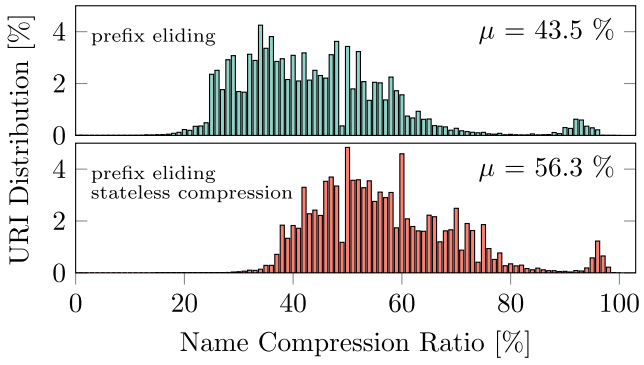


Fig. 6. Distribution of percental name compression ratios.

3.5. Name compression performance

Our proposed stateless and stateful name compression mechanisms are of simple nature and do not exhaust scarcely available CPU resources. The compression ratio is highly dependent on (*i*) the number of name components as they dictate the overall TLV overhead and (*ii*) the length of the prefix omitted from the name. To quantify the effects, we analyze the average name compression performances in Fig. 6 for $356 \cdot 10^6$ real-world URI paths obtained from the WWW, where each component does not exceed 15 bytes. Before applying our compression scheme, we encode all names as NDN Name TLVs. First, we elide the hostname from each URI as part of our stateful compression in Section 3.4. This yields an average compression ratio of 43.5%. Second, we apply the stateful name compression described in Section 3.3.2 to reduce the TLV overhead and observe a compression ratio of 56.3% on average. We believe that typical names in a low power IoT edge network will yield similar, if not better, compression performances.

4. Evaluation

4.1. Experiment setup

4.1.1. Protocol comparison

We consider two different NDN deployments for low power IoT networks. In the first experiment, we run NDN directly on top of IEEE 802.15.4. Due to the small MTU of this particular link layer, we limit packet sizes to a maximum of 100 bytes. In the second experiment, NDN runs on top of ICNLoWPAN with activated stateless and stateful compression. We further compare our NDN setups with a typical 6LoWPAN operation that uses UDP as a transport protocol and CoAP as an application protocol. We specifically compare against the GET method of CoAP as it provides a request–response pattern analog to NDN. We deploy our devices in two different network configurations.

Single-hop. A consumer device has managed FIB entries to a producer device and vice versa. In our NDN deployment, the producer initially creates all content objects. The 6LoWPAN producer configures a callback function as a CoAP endpoint for a particular URI to trigger a response.

Multi-hop. An extra forwarding device is added to the network topology, such that requests and responses between the consumer and producer traverse through the forwarder.

4.1.2. Hardware & software platform

We conduct all our experiments on typical class 2 [13] devices that feature an ARM Cortex-M0+ MCU with 32 kB RAM, 256 kB ROM and up to 48 MHz CPU frequency. Each device further provides an Atmel AT86RF233 [40] 2,4 GHz IEEE 802.15.4 radio transceiver. We set the radio transmission power to 0 dBm, the receiver sensitivity to −94 dBm and enable the *Smart Receiving* feature, an energy saving mode for idle listening. For our power consumption measurements we make use of on-board current measurement headers on each of our devices. We measure currents using a Keithley DMM7510 $7\frac{1}{2}$ digit graphical sampling multimeter with 1 MHz sampling rate and control it with external I/O lines to trigger start and stop from events generated by our network stack. Devices under test are powered by a regulated external DC power supply and connect via UART to a Linux control node to obtain experiment results.

In each experiment, our devices operate RIOT OS version 2018.10. Our NDN deployments use the CCN-lite [41] package and our 6LoWPAN experiments are based on the default GNRC network stack of RIOT OS. We integrate ICNLoWPAN into the 6LoWPAN module of RIOT OS to reutilize the code base of the dispatching framework and link fragmentation.

4.1.3. Name configuration

Name lengths proportionally affect processing times, packet lengths and consequently energy expenditure during transmissions. This especially impacts NDN as names are included in requests as well as in responses. We use two different names in our experiments to measure the effects of our stateless and stateful compression mechanisms.

Name$_{short}$. We use a short name with 4 components to denote temperature readings produced by a sensor. The name is of the form `/org/example/temp/` id_x, where id_x is an increasing number for each request. A CID is configured for `/org`, such that this prefix is elided from all messages.

Name$_{long}$. We use a long name with 10 components of the form `/org/example/building/1/floor/4/room/481/temp/` id_x. A CID is configured for the prefix `/org/example/building/1/floor/4/room/481` to elide a considerable portion of the name.

4.2. Theoretical evaluation

4.2.1. Packet dissection

In this first evaluation, we analyze NDN and CoAP packet sizes for a typical IoT scenario using Name$_{long}$ as a CoAP endpoint and as a naming scheme for NDN. We further configure responses of both protocols to return 4-byte signed integer values that represent temperature sensor readings.

Fig. 7 depicts the actual packet sizes for each protocol. Our CoAP request has a packet size of 97 bytes, where 3 bytes are used for 6LoWPAN dispatches, 32 bytes for the compressed IPv6 header and 6 bytes

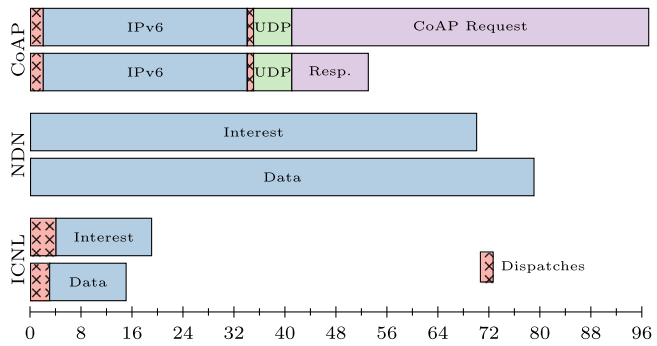


Fig. 7. Packet length and structure for different protocols.

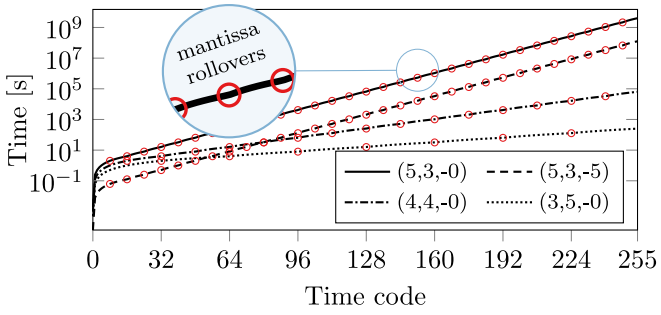


Fig. 8. Precision and range of various timestamp encoding configurations.

for the UDP header. The remaining 56 bytes are used by the actual CoAP message. The respective CoAP response requires considerably less, which follows from CoAP omitting URIs in responses and using tokens to match against open requests on the requesting node. In our setup, CoAP uses 2-byte tokens for each request and returns the exact token in the response. The NDN Interest message nets to 70 bytes, whereas the returning data message requires 79 bytes. Contrary to CoAP, returning responses in NDN include the name of the request, or even a more specific and longer name. The displayed data message further contains empty Signature TLVs. The equivalent ICNLoWPAN compressed NDN messages are significantly shorter, where the Interest message reduces down to 19 bytes (72% savings) and the data message down to 15 bytes (81% savings). For Interests, this gain is mainly due to leveraging the configured stateful name compression and data messages naturally benefit from eliding the full returning name. In addition to the compressed messages, we require a 1-byte page dispatch, 1-byte ICNLoWPAN dispatch, and a 1-byte HopID for Interest and data. The compressed Interest packet further includes a 1-byte CID indicating the elided prefix for the stateful name compression.

4.2.2. Configurations for compressed timestamps

In the next evaluation, we explore different configurations of the enhanced timestamp encoding described in Section 3.3.3. Fig. 8 illustrates the precision and maximum ranges in logarithmic scale for selected configurations. The number of bits for exponent, mantissa, and the value for bias are noted in the form `(exponent,mantissa,bias)` for each plot. Red dots symbolize mantissa rollovers and indicate an exponent increase by one. The configuration (3,5,−0) allocates five mantissa bits and thus shows a precision loss that declines minimally across mantissa rollovers as indicated by the distances between the red circles. Conversely, the remaining three bits for the exponent allows only for a small maximum range of 252 s, while the minimum non-zero number is 62.5 ms. This configuration is unsuited for scenario deployments that expect a prolonged protocol operation (*e.g.,* request lifetime, content freshness) due to intermittent connectivity and network partitioning. Configuration (4,4,−0) increases the range to about 17 h to the cost of precision. The maximum time value boosts with configuration (5,3,−0): Time codes can represent time values as high as 127 years, but experience a considerable precision loss, even in the lower seconds range. 250 ms is the minimum non-zero time that can be represented. After 32 s, this configuration yields a 4-second resolution, which decreases to an 8-second resolution after 64 s. At around 10 minutes, the resolution is at 64 s. Configuration (5,3,−5) applies a bias of −5 to the previous setting and makes a good compromise by shifting the co-domain further into the sub-seconds range. It thereby sacrifices the maximum range and inherits the same precision decline from the latter configuration. The maximum representable time value degrades from 127 years to approximately four years. However, the negative bias adds a decent precision to the sub-seconds range, which is important in real-time scenario deployments that operate in milliseconds. This setting has a fine-grained resolution of 7.8 ms between 0–125 ms and 15.6 ms between 125–250 ms. The decent precision in lower sub-second ranges and the maximum range of four years allows this configuration to provide an effective operation for most of the IoT deployment scenarios.

4.3. Experimental evaluation

The theoretical evaluation indicates that ICNLoWPAN substantially reduces packet sizes in Named-Data IoT networks with pull-driven traffic patterns. In this experimental evaluation, we want to explore the effects of our convergence layer on resource-constrained nodes to gauge

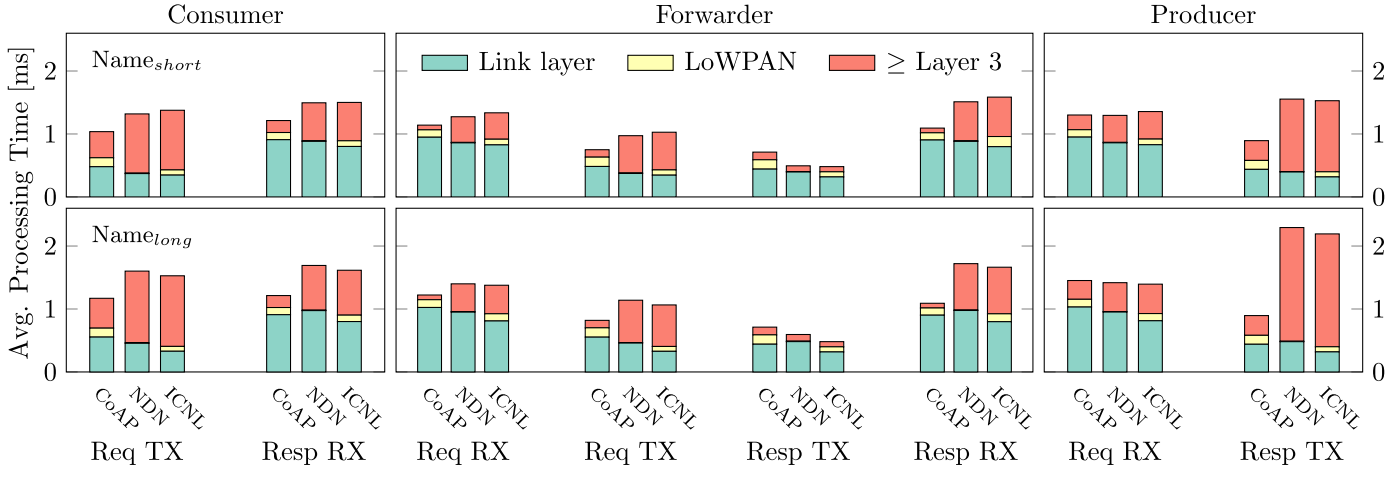


Fig. 9. Intra-stack average processing times for CoAP, NDN, and ICNLoWPAN.

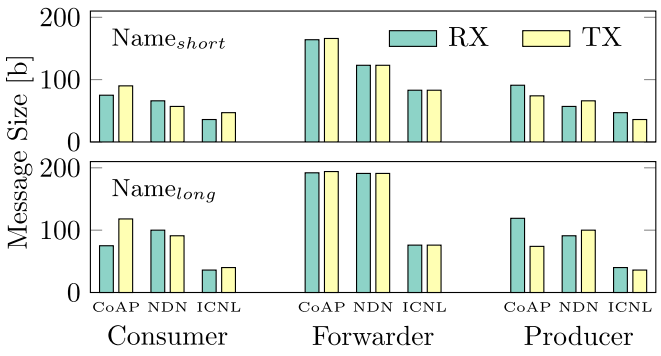


Fig. 10. Average bytes per request—response.

the feasibility for real-world low power networks. In particular, we measure (*i*) intra-stack processing times, (*ii*) average message overhead for a request–response handshake, (*iii*) energy expenditure during transmissions for single-hop and multi-hop deployments, (*iv*) and effects on reliability when periodically requesting sensor values while we enable randomized and bursty cross-traffic.

4.3.1. Processing times

We first evaluate intra-stack processing times in a multi-hop deployment, where a consumer requests every 500 ms a specific temperature reading from a producer using Name$_{short}$ and Name$_{long}$. Timestamps for the link layer, convergence layer (LoWPAN), and upper layers are recorded for each packet using a hardware timer on each device with µs precision. The link layer time depicts operations of the RIOT OS radio driver including SPI transfer of the packet to the radio frame buffer with 5 MHz. This measurement does not contain the actual transmission or reception of a frame over the wireless medium, as this procedure does not load the CPU. Time spent in the LoWPAN module includes the handling of dispatch headers and packet (de-)compression. Processing times for the network layer and beyond either include IPv6, UDP, and CoAP or NDN operations in addition to the actual application on top that issues or satisfies requests.

Fig. 9 displays experiment results for the different roles of a consumer, forwarder, and producer. We first observe that for the Name$_{short}$ configuration, the additional processing overhead of ICNLoWPAN does not pay off on the link layer. Thus, some measurements with ICNLoWPAN take slightly more CPU resources than plain NDN. Conversely, savings on the link layer for our Name$_{long}$ configuration visibly outperform the ICNLoWPAN processing overhead, especially for response packets by a decrease of ≈ 100 µs per packet.

4.3.2. Message sizes

We now analyze the amount of bytes that are transmitted between consumer, forwarder, and producer when performing a request–response handshake in Fig. 10. The captured packets include the message lengths of Fig. 7 in addition to 21 bytes for an IEEE 802.15.4 header and 2 bytes for FCS. While comparing the Name$_{short}$ and Name$_{long}$ configurations, we observe an increase in sent bytes for the CoAP and NDN consumer, whereas ICNLoWPAN reduces the amount of sent bytes due to a better utilization of our name compression scheme. We interestingly see that the amount of sent bytes for the CoAP producer stagnates, while the amount for our NDN producer increases. This follows from the fact that CoAP responses do not include the URI component, but rather a fixed-length token to match against open requests. In contrast, the NDN data message does include the full name that is obtained from the request, which leads to significantly increased message sizes from Name$_{short}$ to Name$_{long}$.

4.3.3. Energy consumption

ICNLoWPAN decreases the total amount of bytes over the air with both name configurations compared to NDN. We observe a strong reduction of bytes for responses at the producer for both name configurations, since the name is fully elided. The reduction is most prominent for the Name$_{long}$ configuration at the forwarder with a drop by 60% from 191 bytes down to 76 bytes. This gain is to be expected, considering that the forwarder is involved in four transmissions.

The packet length during transmission has an immediate effect on the energy consumption. We measure the current while sending and receiving messages for each role separately in a single-hop setup and display the results in Fig. 11 for Name$_{long}$. The graphs involve transmission over the wireless, radio turnaround time as well as link layer frame acknowledgment. In our setup, sending draws slightly higher current than receiving and the duration of each transmission depends on the packet length. In fact, the duration of each depicted measurement correlates with the respective message size displayed in Fig. 7 and the results showed in Fig. 10, so that larger messages yield longer periods of operation for sending and receiving.

On the consumer, we observe that our CoAP request requires 5 ms to complete, while the respective NDN request is transmitted in 4 ms, including the reception of acknowledgments for both. Conversely, the CoAP response is received by the consumer in 3.8 ms, while the NDN response completes in 4.2 ms, including the sending of acknowledgments. With ICNLoWPAN in operation, we notice a decrease of transmission times by around ≈ 50% on the consumer due to compressed messages and the resulting shortened media utilization. As expected, the reduction for responses is more prominent due to fully eliding Name

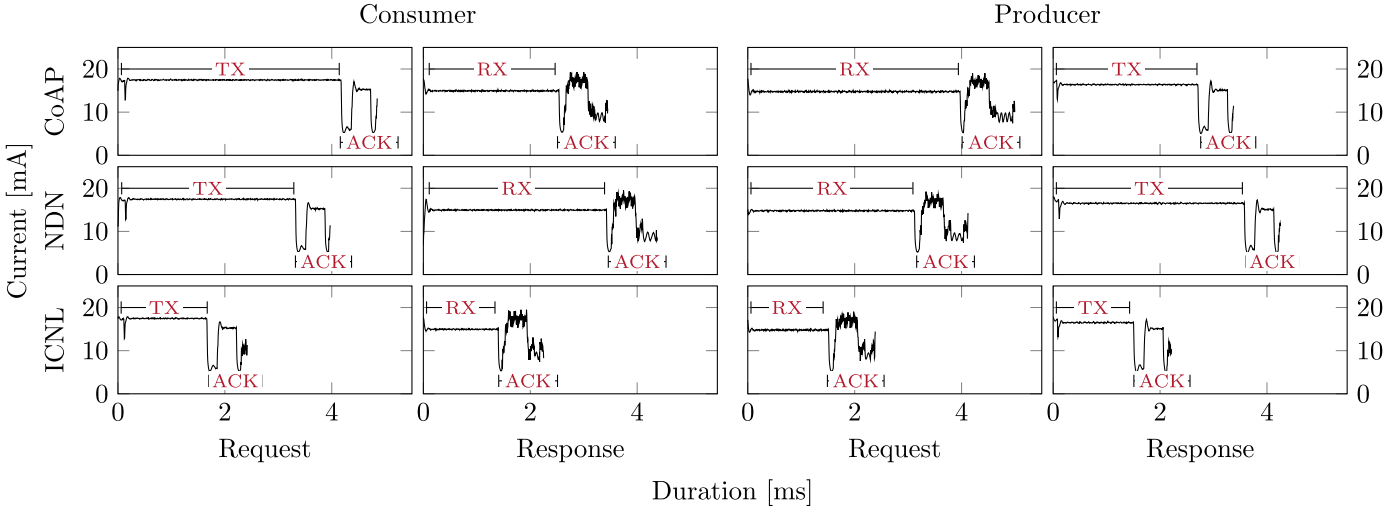


Fig. 11. Current consumption for send and receive operations.

Table 1
Energy consumption in µJ.

	Consumer		Forwarder		Producer	
	$Name_{short}$	$Name_{long}$	$Name_{short}$	$Name_{long}$	$Name_{short}$	$Name_{long}$
CoAP	548.58 µJ	612.24 µJ	967.41 µJ	1072.07 µJ	464.73 µJ	517.96 µJ
NDN	526.23 µJ	687.26 µJ	880.68 µJ	1152.02 µJ	422.55 µJ	584.82 µJ
ICNL	466.09 µJ	487.32 µJ	769.17 µJ	773.97 µJ	369.84 µJ	395.19 µJ

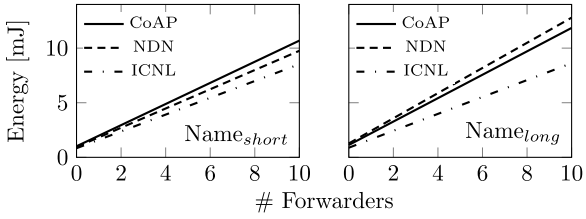


Fig. 12. Total energy consumption for multi-hop networks.

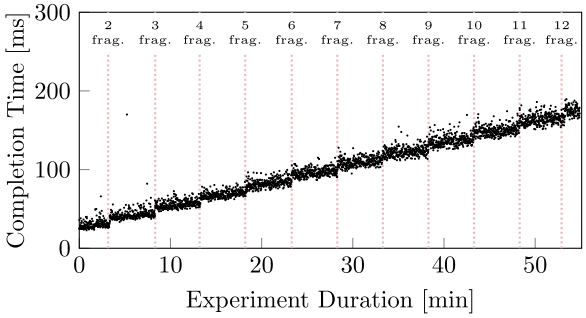


Fig. 13. Content arrival times with gradually increasing payload sizes.

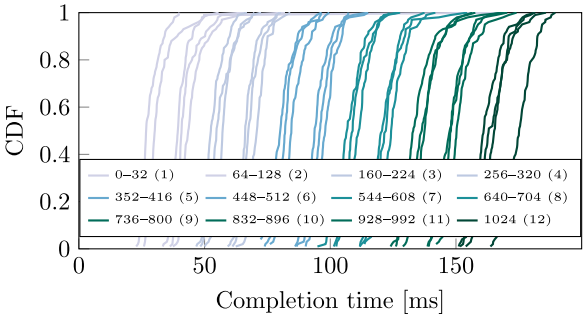


Fig. 14. Temporal distributions of content arrival times for different payload sizes in bytes. The amount of fragments is given in parentheses.

TLVs. On the producer, we naturally observe mirrored results for each operation.

Given the fact that current draws for transmissions in Fig. 11 are mainly similar, the actual energy consumption is predominated by the transmission durations. We thus analyze the overall power consumption of a request–response handshake in a multi-hop setup for $Name_{short}$ and $Name_{long}$ including processing times and transmission durations in Table 1. Our results indicate an increase in energy usage for each role with the $Name_{long}$ configuration compared to the $Name_{short}$ configuration. We further notice that our producer spends the least amount of energy, followed by the consumer, and our forwarder expends nearly double the amount of energy than the producer. The increased power consumption is inherently consistent with the fact that the forwarder is involved in two request and subsequently in two response transmissions.

The NDN consumer device uses 4% less energy for $Name_{short}$ and 12% more energy for $Name_{long}$ compared to the CoAP consumer. This turnaround in energy expenditure for $Name_{long}$ is twofold. (*i*) NDN has a more verbose name encoding than CoAP and (*ii*) CoAP does not include the URI in the response. ICNLoWPAN reduces energy usage of our NDN consumer by 11% for $Name_{short}$ and 29% for $Name_{long}$. Our NDN producer consumes 9% less energy for $Name_{short}$ and 13% more energy for $Name_{long}$ compared to our CoAP producer. The energy consumption reduces by 12% for $Name_{short}$ and 32% for $Name_{long}$ with an enabled ICNLoWPAN operation compared to NDN. Since our forwarder interacts with four transmissions, we observe a natural increase in overall expenditures. The NDN forwarder consumes 9% less power for $Name_{short}$ and 7% more energy for $Name_{long}$ compared to the CoAP forwarder. In contrast, ICNLoWPAN reduces the expenditure by 13% for $Name_{short}$ and 33% for $Name_{long}$. The trend that ICNLoWPAN yields higher energy savings for $Name_{long}$ becomes apparent.

Finally, we calculate the energy consumption of a full request–response handshake for a multi-hop setup with a varying number of forwarders between a consumer and producer in Fig. 12. For all setups, we see an increase in expenditure for CoAP, NDN, and ICNLoWPAN with an increasing number of forwarders. We again notice that the power consumption for NDN surpasses the consumption for CoAP using the $Name_{long}$ configuration due to returning Name TLVs in the response. ICNLoWPAN clearly reduces the overall energy consumption of an NDN handshake for both configurations $Name_{short}$ and $Name_{long}$. Interestingly, despite the increase in power consumption for NDN versus CoAP, ICNLoWPAN manages to cut expenditures by ≈ 25% for a setup with ten forwarders compared to NDN.

4.3.4. The impact of fragmentation

ICNLoWPAN employs a hop-wise fragmentation scheme for messages that do not fit the maximum physical packet size of 127 bytes in IEEE 802.15.4 environments. In this experiment, we gradually increase the payload size in data messages by 32 bytes after every 100th packet to examine the impact of fragmentation in our multi-hop scenario. We start with a payload size of 0 bytes and end at 1024 bytes. A consumer periodically requests content from every 1 ± 0.1 s using the $Name_{short}$ configuration. The requests traverse a forwarding node to reach the producer device.

Fig. 13 portrays content arrival times at the consumer over the duration of the experiment. The gradual increase of returning payloads leads to elevated completion times as indicated by the previous experiment in Section 4.3.3. We observe a moderate incline of around 3 ms for payload sizes that do not provoke further fragmentation. As soon as sizes force an added fragment, the average completion time for subsequent content requests jumps by roughly 10 ms. We also note the increasing dispersion with each additional fragment due to accumulating link-layer activities, such as carrier sensing related back-offs and packet retransmissions. The scattering naturally shows its extreme spread for data messages with 11 and 12 fragments. Temporal distributions for content arrival times reflect in Fig. 14 and confirm our previous results: Configurations with payload sizes that fit into the same amount of fragments are roughly 3 ms apart, while new fragments add another 10 ms to completion times. While 50% of packets with a payload size of 0–32 bytes finish below 25–30 ms, we observe an increase that is almost six-fold for payload sizes of 1024 bytes. This

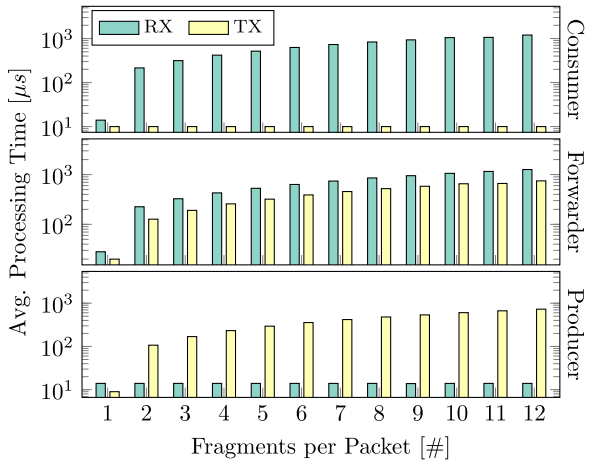


Fig. 15. Average processing overhead for fragmentation and reassembly in a multi-hop scenario.

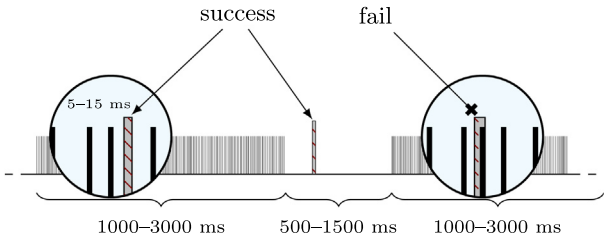


Fig. 16. Radio interference due to bursty cross-traffic.

configuration generates 12 fragments per packet and 50% of requests complete within 175 ms.

We further want to quantify the processing overhead that results from fragmented messages. Fig. 15 displays the average processing time spent within the fragmentation module for the previous experiment. Interest and Data packets that fit into a single fragment exhibit only a minimal processing overhead of 10–14 µs for transmission and reception on the consumer and producer side, respectively. The forwarder demonstrates a doubling due to processing Interest and Data messages in both directions. We observe a sudden increase in processing time for Data messages as soon as the fragmentation strategy activates: The producer shows an accumulated average processing time of 100 µs for fragmenting the data, while the consumer spends around 200 µs during the reassembly of two fragments. These numbers linearly increase with the number of fragments by approximately 70 µs for an added fragment and about 100 µs for the reassembly. For 12 fragments, the absolute numbers multiply up to 1.2 ms for the producer and forwarder on fragmentation and 0.7 ms on reassembly for the forwarder and consumer.

4.3.5. Reliability

In this experiment we investigate the reliability of a typical data retrieval setup in our single-hop deployment, where a consumer periodically requests temperature values from a producer every 300 ms. We additionally generate bursty cross-traffic in randomized intervals to a third party in order to mimic a dense network with multiple devices that periodically request sensor readings. Fig. 16 illustrates the configured traffic pattern of our cross-traffic with each burst consisting of 200 UDP packets in succession with a 5–15 ms delay in between each transmission and a radio silence interval of 500–1500 ms. It is worth noting that our cross-traffic only adds wireless interference as the IEEE 802.15.4 frames are not delivered to the devices due to automatic MAC address filtering in hardware of the radio module. We further disable CSMA/CA and frame retransmissions of the radio transceiver to explore reliability gains without distortions of automatic corrective actions in hardware. Indeed, disabling such hardware features in real deployments is not a far-fetched scenario as sophisticated link layer protocols that employ time-slotted algorithms must satisfy strict time constraints, while CSMA/CA and retransmissions lead to indeterministic and drifting media accesses. In fact, several radio transceivers that dual operate IEEE 802.15.4 and Bluetooth Low Energy do not even support such mechanisms in hardware, but rather rely on software implementations.

Table 2
Packet Reception Ratio (PRR).

	CoAP	NDN	ICNL
Producer	93.53%	94.05%	94.40%
Consumer	73.25%	75.98%	93.06%

Table 2 lists the packet reception ratio (PRR) for consumer and producer roles using the Name_{long} configuration. For each deployment the number of received requests on the producer lies within ≈ 93–94%. Conversely, the percentage of successfully received responses on each consumer clearly varies between ≈ 73–76% for CoAP and NDN and ≈ 93% for our ICNLoWPAN operation.

The performance gain of ICNLoWPAN results from strongly compressed packets which lead to a significantly reduced on-air time for the low power wireless transmission (see Fig. 11). This reduces collision probability with interferer traffic, especially when responses are sent during a burst as shown in Fig. 16. Furthermore, responses of a producer always follow the successful reception of a request. Reduced transaction times with ICNLoWPAN leave more time to the next interferer transmission within a burst, which further reduces the probability of overlapping cross-traffic compared to the NDN and CoAP operation.

5. Conclusions

IoT networking has proven to benefit from information-centric architectures in several directions. In this paper, we worked out the components for adapting NDN to a LoWPAN edge: encoding, compression, framing, and fragmentation. By leveraging the NDN stateful forwarding for compression on path, we could again take particular advantage of the information-centric approach in low power lossy IoT networks.

Theoretical estimates and extensive measurements of our protocol implementation on IoT hardware revealed these benefits in comparison with plain NDN and the IP world (CoAP over 6LoWPAN). Our experimental results clearly showed that ICNLoWPAN outperforms NDN and CoAP in terms of media utilization as well as energy consumption. ICNLoWPAN further reduces end-to-end latencies in multi-hop scenarios, and contributes to an improved reliability in lossy environments while preserving battery resources. Depending on the use case, savings typically range from 20% to 33%.

With these results, we hope to contribute insights to the community, to substantiate corresponding standard development [42], and to encourage deployment of NDN in the constrained IoT. Our future work will concentrate on extending ICNLoWPAN to different low power link technologies, such as BLE and align with adaptation layers for wide-area and cellular technologies to enable LoRA and NB-IoT for a versatile, efficient, and robust information-centric Internet of Things.

CRediT authorship contribution statement

Cenk Gündoğan: Conceptualization, Methodology, Software, Investigation, Visualization, Data curation, Writing - original draft. **Peter Kietzmann:** Conceptualization, Methodology, Investigation, Validation. **Thomas C. Schmidt:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Supervision, Project administration, Funding acquisition. **Matthias Wählisch:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was inspired by many fruitful discussions in the IRTF ICN research group, as well as by industrial deployment demands. It was supported in part by the German Federal Ministry for Education and Research (BMBF) within the projects *I3—Information Centric Networking for the Industrial Internet*, *RAPstore—RIOT App Store*, and the Hamburg *ahoi.digital* initiative with *SANE*.

References

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A survey of Information-Centric Networking, IEEE Commun. Mag. 50 (7) (2012) 26–36.

[2] E. Baccelli, C. Mehlis, O. Hahm, T.C. Schmidt, M. Wählisch, Information centric networking in the IoT: Experiments with NDN in the Wild, in: Proc. of 1st ACM Conf. on Information-Centric Networking (ICN-2014), ACM, New York, 2014, pp. 77–86, URL http://dx.doi.org/10.1145/2660129.2660144.

[3] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, L. Zhang, Named data Networking of Things (Invited Paper), in: Proc. of IEEE International Conf. on Internet-of-Things Design and Implementation (IoTDI), IEEE Computer Society, Los Alamitos, CA, USA, 2016, pp. 117–128.

[4] E.M. Schooler, D. Zage, J. Sedayao, H. Moustafa, A. Brown, M. Ambrosin, An architectural vision for a Data-centric IoT: Rethinking Things, trust and clouds, in: IEEE 37th Intern. Conference on Distributed Computing Systems (ICDCS), IEEE, Piscataway, NJ, USA, 2017, pp. 1717–1728.

[5] K. Pentikousis, B. Ohlman, D. Corujo, G. Boggia, G. Tyson, E. Davies, A. Molinaro, S. Eum, Information-Centric Networking: Baseline Scenarios, RFC 7476, IETF, 2015.

[6] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, Networking named content, in: 5th Int. Conf. on Emerging Networking Experiments and Technologies (ACM CoNEXT'09), ACM, New York, NY, USA, 2009, pp. 1–12.

[7] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, Named data Networking, SIGCOMM Comput. Commun. Rev. 44 (3) (2014) 66–73.

[8] C. Gündogan, P. Kietzmann, M. Lenders, H. Petersen, T.C. Schmidt, M. Wählisch, NDN, CoAP, and MQTT: A comparative measurement study in the IoT, in: Proc. of 5th ACM Conference on Information-Centric Networking (ICN), ACM, New York, NY, USA, 2018, pp. 159–171, http://dx.doi.org/10.1145/3267955.3267967.

[9] Z. Shelby, K. Hartke, C. Bormann, The Constrained Application Protocol (CoAP), RFC 7252, IETF, 2014.

[10] A. Stanford-Clark, H.L. Truong, MQTT for Sensor Networks (MQTT-SN) Version 1.2, Protocol Specification, IBM, 2013, URL http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf.

[11] C. Gündogan, J. Pfender, P. Kietzmann, T.C. Schmidt, M. Wählisch, On the impact of QoS management in an information-centric Internet of Things, Comput. Commun. 154 (2020) 160–172, http://dx.doi.org/10.1016/j.comcom.2020.02.046.

[12] D. Kutscher, S. Eum, et al., Information-Centric Networking (ICN) Research Challenges, RFC 7927, IETF, 2016.

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

[14] IEEE 802.15 Working Group, IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs), Tech. Rep. IEEE Std 802.15.4™–2011, IEEE, New York, NY, USA, 2011.

[15] B. S. I. Group, Bluetooth Core Specification Version 5.1, Tech. rep., 2019, URL https://www.bluetooth.com/specifications/bluetooth-core-specification.

[16] LoRa Alliance, A Technical Overview of LoRa and LoRaWan, Tech. rep., 2015, URL https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf.

[17] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, Transmission of IPv6 Packets over IEEE 802.15.4 Networks, RFC 4944, IETF, 2007.

[18] J. Hui, P. Thubert, Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, RFC 6282, IETF, 2011.

[19] C. Gündogan, P. Kietzmann, T.C. Schmidt, M. Wählisch, ICNLoWPAN – named-data networking in low power IoT Networks, in: Proc. of 18th IFIP Networking Conference, IEEE Press, Piscataway, NJ, USA, 2019, pp. 1–9, http://dx.doi.org/10.23919/IFIPNetworking.2019.8816850.

[20] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, T.C. Schmidt, RIOT OS: Towards an OS for the Internet of Things, in: Proc. of the 32nd IEEE INFOCOM. Poster, IEEE Press, Piscataway, NJ, USA, 2013, pp. 79–80.

[21] E. Baccelli, C. Gündogan, O. Hahm, P. Kietzmann, M. Lenders, H. Petersen, K. Schleiser, T.C. Schmidt, M. Wählisch, RIOT: an open source operating system for low-end embedded devices in the IoT, IEEE Internet Things J. 5 (6) (2018) 4428–4440, http://dx.doi.org/10.1109/JIOT.2018.2815038.

[22] M. Gritter, D.R. Cheriton, An architecture for content routing support in the Internet, in: Proc. USITS'01, USENIX Association, Berkeley, CA, USA, 2001, p. 4.

[23] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K.H. Kim, S. Shenker, I. Stoica, A data-oriented (and beyond) Network Architecture, SIGCOMM Comput. Commun. Rev. 37 (4) (2007) 181–192.

[24] P. Kietzmann, C. Gündogan, T.C. Schmidt, O. Hahm, M. Wählisch, The need for a name to MAC address mapping in NDN: Towards quantifying the resource gain, in: Proc. of 4th ACM Conference on Information-Centric Networking (ICN), ACM, New York, NY, USA, 2017, pp. 36–42.

[25] G. Carofiglio, L. Muscariello, J. Augé, M. Papalini, M. Sardara, A. Compagno, Enabling ICN in the Internet Protocol: Analysis and evaluation of the hybrid-ICN architecture, in: Proceedings of the 6th ACM Conference on Information-Centric Networking, in: ICN '19, ACM, New York, NY, USA, 2019, pp. 55–66.

[26] D. Trossen, M.J. Reed, J. Riihijärvi, M. Georgiades, N. Fotiou, G. Xylomenos, IP over ICN - the better IP?, in: 2015 European Conference on Networks and Communications (EuCNC), IEEE Computer Society, Los Alamitos, CA, USA, 2015, pp. 413–417.

[27] C. Bormann, 6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs), RFC 7400, IETF, 2014.

[28] W. Shang, A. Afanasyev, L. Zhang, The design and implementation of the NDN protocol stack for RIOT-OS, in: Proc. of IEEE GLOBECOM 2016, IEEE, Washington, DC, USA, 2016, pp. 1–6.

[29] M. Mosko, C. Tschudin, ICN 'Begin-End' Hop By Hop Fragmentation, Internet-Draft – Work in Progress 02, IETF, 2016.

[30] J. Shi, B. Zhang, NDNLP: A Link Protocol for NDN, NDN, Technical Report NDN-0006, NDN Team, 2012.

[31] C. Ghali, A. Narayanan, D. Oran, G. Tsudik, C.A. Wood, Secure fragmentation for content-centric networks, in: Proc. of the 14th International Symposium on Network Computing and Applications, IEEE Press, Piscataway, NJ, USA, 2015, pp. 47–56.

[32] M.S. Lenders, T.C. Schmidt, M. Wählisch, A lesson in scaling 6LoWPAN – minimal fragment forwarding in lossy networks, in: Proc. of the 44rd IEEE Conference on Local Computer Networks (LCN), IEEE Press, Piscataway, NJ, USA, 2019.

[33] Y. Yang, T. Song, Local name translation for succinct communication towards named data Networking of Things, IEEE Commun. Lett. 22 (2018) 2551–2554.

[34] P. Thubert, R. Cragie, IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Paging Dispatch, RFC 8025, IETF, 2016.

[35] T. Watteyne, P. Thubert, C. Bormann, On Forwarding 6LoWPAN Fragments over a Multihop IPv6 Network, Internet-Draft – Work in Progress 15, IETF, 2020.

[36] P. Thubert, 6LoWPAN Selective Fragment Recovery, Internet-Draft – Work in Progress 21, IETF, 2020.

[37] M.S. Lenders, C. Gündogan, T.C. Schmidt, M. Wählisch, Connecting the dots: Selective fragment recovery in ICNLoWPAN, in: Proc. of 7th ACM Conference on Information-Centric Networking (ICN), ACM, New York, 2020.

[38] C. Gündogan, T.C. Schmidt, D. Oran, M. Wählisch, An alternative delta time encoding for CCNx using interval time from RFC5497, in: IRTF Internet Draft – Work in Progress 00, IRTF, 2019, URL https://tools.ietf.org/html/draft-gundogan-icnrg-ccnx-timetlv.

[39] IEEE 754 Working Group, IEEE Standard for Floating-Point Arithmetic, Tech. Rep. IEEE Std 754-2019 (Revision of IEEE Std 754-2008), IEEE, New York, NY, USA, 2019, http://dx.doi.org/10.1109/IEEESTD.2019.876622.

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

[41] C. Tschudin, C. Scherb, et al., CCN Lite: Lightweight Implementation of the Content Centric Networking Protocol, 2018, URL http://ccn-lite.net.

[42] C. Gündogan, T.C. Schmidt, M. Wählisch, C. Scherb, C. Marxer, C. Tschudin, ICN Adaptation to LowPAN Networks (ICN LoWPAN), IRTF Internet Draft – Work in Progress 08, IRTF, 2020, URL https://tools.ietf.org/html/draft-irtf-icnrg-icnlowpan.