Connecting the Dots: Selective Fragment Recovery in ICNLoWPAN

Martine S. Lenders
Freie Universität Berlin
m.lenders@fu-berlin.de

Cenk Gündoğan
HAW Hamburg
cenk.guendogan@haw-hamburg.de

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

Matthias Wählisch
Freie Universität Berlin
m.waehlisch@fu-berlin.de

ABSTRACT

In this paper, we analyze the benefits of integrating 6LoWPAN Selective Fragment Recovery (SFR) in ICNLoWPAN. We present a solution that allows for immediate fragment forwarding—a key feature of SFR—in combination with ICN caching. Our proposal introduces a *Virtual Reassembling Endpoint* (VREP), which acts transparently as an SFR fragment forwarder while simultaneously collecting fragments. Once a datagram is complete, it is exposed to the content cache, effectively making the VREP the new fragmenting endpoint. Our solution complies with current specs defined in the IETF/IRTF. Furthermore, we combine the reverse path forwarding schemes of both SFR and ICNLoWPAN and assess drawbacks and benefits in a testbed. Our evaluation shows that SFR with VREP performs similar to hop-wise reassembly, details depend on the topology, but both outperform SFR without VREP in all scenarios.

CCS CONCEPTS

• **Networks** → **Network protocol design**; **Network experimentation**; **Network performance analysis**.

KEYWORDS

Information-centric networking; Internet of Things; wireless; fragmentation; protocol design; measurement; protocol evaluation

ACM Reference Format:
Martine S. Lenders, Cenk Gündoğan, Thomas C. Schmidt, and Matthias Wählisch. 2020. Connecting the Dots: Selective Fragment Recovery in ICNLoWPAN. In *ACM Conference on Information-Centric Networking (ICN '20), September 29–October 1, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3405656.3418719

1 INTRODUCTION

The Internet of Things (IoT) is an appealing deployment space for Information-centric Networks [3, 8, 28]. The current state-of-art in the IoT envisions a collection of heterogeneous device types offering a wide range of link layer technologies. For low-power and lossy networks (LLN), ICNLoWPAN [11] was introduced to enhance ICN for small data units, which are often encountered in LLNs. This is achieved by utilizing *compression* and *fragmentation* comparable to 6LoWPAN [16] in the IP world.

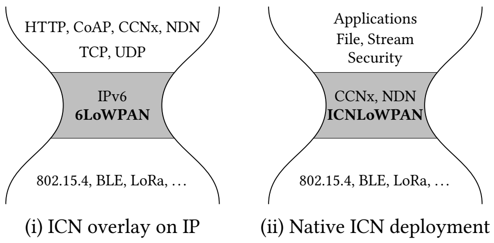


Figure 1: Two types of ICN deployment in the IoT.

Lately, the idea of fragment forwarding—in contrast to the more simple approach of reassembling a fragmented datagram at every hop—gained new attention within the 6LoWPAN community to reduce overall resource usage and potentially decrease latencies [15, 23]. A major challenge in this context of fragmentation *and* forwarding is that only the first fragment includes a forwarding hint, which leads to stateful forwarding techniques. *6LoWPAN Selective Fragment Recovery* (SFR) [24] offers an approach to not only forward fragments, but also to provide higher reliability by utilizing the forwarding state information to acknowledge the reception of the fragmented datagram to the original sender.

Forwarding ICNLoWPAN fragments using the original Selective Fragment Recovery, however, will lead to higher loss, as ICN caching is completely circumvented since nodes do not receive completed content packets anymore. In this paper, we present a solution how both the advantages of SFR and ICN can be used.

In this paper, we make the following contributions. First, we propose a solution that combines current 6LoWPAN fragmentation schemes with ICN to benefit from ICN caching. Second, we compare our proposal to the current state-of-art.

The remainder of this paper is structured as follows. In Section 2, we gave a brief overview of the two protocols we want to combine and present our problem statement. We address these problems in Section 3 by introducing our SFR extension. In Section 4, we evaluate this extension in a testbed. We then follow with an overview of related work in Section 5 and discuss our findings in Section 6. Finally, we conclude our paper and give an outlook in Section 7.

Table 1: Overview of ICNLoWPAN forwarding approaches that support fragmentation.

	Supported Features			
Approach	**Buffer Frugality**	**Fragment Forwarding**	**Fragment Recovery**	**En-route Caching**
Hop-Wise Reassembly **(HWR)** [16]	✘	✘	✘	✓
Fragment Forwarding without Selective Fragment Recovery **(FF w/o SFR)** [25]	✓	✓	✘	✘
FF with SFR without Virtual Reassembling Endpoint **(SFR w/o VREP)** [24]	✓	✓	✓	✘
FF with SFR with VREP **(SFR w/ VREP)** [this paper]	✓	✓	✓	✓

2 BACKGROUND AND PROBLEM STATEMENT

2.1 Fragmentation in 6LoWPAN

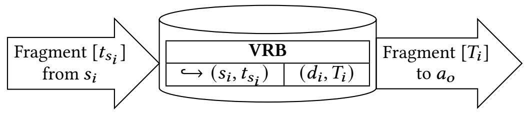
In many low-power and lossy networks, devices can handle only very small PDUs of some 100 bytes. This means packets need to be fragmented. Carrying multi-hop forwarding information in each of the fragments, as IP fragmentation does, would introduce relatively large overhead compared to the overall packet size. To minimize the number of bytes per frame, only the *first fragment* in 6LoWPAN fragmentation includes the forwarding information (*e.g.*, source and destination address in IP or content names in ICN) but subsequent fragments do not.

How to forward fragments?. The very basic method to handle fragments in low-power lossy networks is simple and implemented by *Hop-wise Reassembly (HWR)*: Each forwarder between source and destination fully reassembles the original datagram before forwarding fragments separately [16]. This simplicity introduces two drawbacks: forwarding delay and higher resource requirements in terms of memory and energy because all fragments that belong to the same datagram need to be buffered before releasing them.

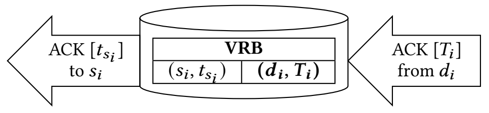
To genuinely consider fragments instead of a datagram while forwarding, *Fragment Forwarding (FF)* [24] is proposed in the IETF. Here, each fragment is independently linked to the first fragment and its forwarding information.

How to virtually link fragments to forwarding information?. The first fragment carries some type of forwarding information, which in combination with the Forwarding Information Base (FIB) of the network layer can be used to determine the next hop. A train of fragments that belong to same datagram (or content chunk) is linked based on a link-unique tag in 6LoWPAN. Consequently, at each node any fragment can be mapped to the next hop by finding that tag of the corresponding first fragment. This idea is implemented in the *Virtual Reassembly Buffer (VRB)* [4], which provides the next hop for a given fragment without storing the fragments (see Figure 2a). To ensure delivery, an implementer may decide to fall back to HWR if the VRB is full.

How to recover fragments?. Losing a fragment is costly if the complete datagram needs to be retransmitted by an upper layer. This is the case for both HWR and plain fragment forwarding. *Selective Fragment Recovery (SFR)* [24], though, enables nodes to retransmit dedicated fragments instead of a full datagram. SFR introduces both a lightweight cumulative acknowledgment (ACK) scheme as well as the option to buffer fragments for retransmission. ACKs can be returned to the datagram source via a reverse lookup in the VRB (see Figure 2b).


Fragment [t_{s_i}] from s_i
VRB
$\hookrightarrow (s_i, t_{s_i})$
(d_i, T_i)
Fragment [T_i] to a_o

(a) VRB creation or lookup for fragments



(b) Reverse lookup for acknowledgments (ACKs)

Figure 2: Virtual Reassembly Buffer (VRB; s_i: link layer src address of incoming fragment i; t_{s_i} link-unique tag in i; d_i next-hop link layer address for i; T_i link-unique tag for i).

2.2 ICNLoWPAN and Challenges of Fragmentation in ICN

ICNLoWPAN [10] contributes a convergence layer that maps NDN packets onto constrained link layer technologies to enable information-centric deployments without running as an overlay on top of IP (see Figure 1). It inherits most of the features from the 6LoWPAN specification, such as link fragmentation, protocol framing on the link layer, and stateless as well as stateful header compression components.

Combining fragmentation and ICN leads to the following key challenge: How to maintain the ICN cache based on separate content fragments? It is worth noting that these fragments differ from content chunks as the lower layer provides fragmentation independently of any ICN semantic.

HWR is transparent to ICN. After reassembling all fragments, the 6LoWPAN stack passes complete content chunks to the ICN stack, which finally caches if necessary.

Fragment forwarding, on the other hand, immediately forwards each fragment using the VRB, effectively bypassing the cache. Even if fragments are stored in the cache they cannot be mapped to incoming interests without extension as they miss any ICN notion.

SFR adds an extra burden to constrained ICN devices, as they add a content store in which they buffer fragments for retransmission in addition to the common ICN cache.

In the next section, we will present the design and implementation of a Virtual Reassembling Endpoint for ICN that allows for all fragmentation features and ICN caching while keeping constrained resources memory resources moderate.

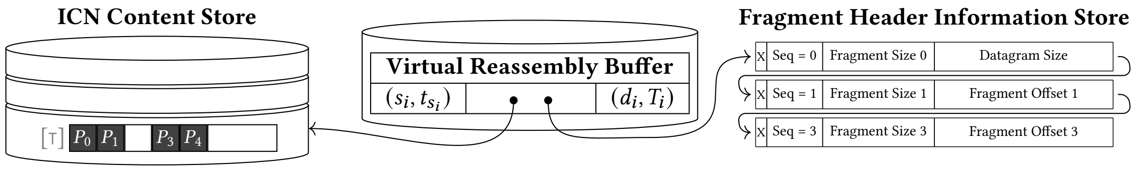

ICN Content Store
[T] P_0 P_1 P_3 P_4
Virtual Reassembly Buffer
(s_i, t_{s_i})
(d_i, T_i)
Fragment Header Information Store
X Seq = 0 Fragment Size 0 Datagram Size
X Seq = 1 Fragment Size 1 Fragment Offset 1
X Seq = 3 Fragment Size 3 Fragment Offset 3

Figure 3: Overview of the architecture at a VREP: An extended virtual reassembly buffer provides a pointer to `TENTATIVE` (`[T]`) content store entries with fragment payloads P_f and holds fragment header information for forwarding.

3 A VIRTUAL ICN REASSEMBLING ENDPOINT

3.1 Design

In this section, we briefly explain the design of our proposal to integrate Fragment Forwarding and Selective Fragment Recovery in ICNLoWPAN. The objective of our work is both enable ICN caching in scenarios of fragmented datagrams and allow for fine-grained error handling (*i.e.,* retransmit of fragments) without additional overhead. To prevent storing data twice in the ICN content store as well as SFR retransmission buffer, we propose a *Virtual Reassembling Endpoint (VREP)* that combines both techniques on the system level. Our proposal aims to be compliant with common techniques [10, 11, 24, 25] and satisfies all requirements for LoWPAN networks.

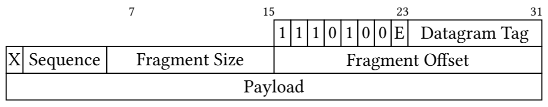
Each VREP node keeps a pointer to the content store entry in the VRB. The fragmented data can then be added to that entry at the offset provided in the fragment header (see Figure 4a).

Fragment ACKs carry information about the received fragments in a bitmap (see Figure 4b) that corresponds to the sequence numbers of the fragments. As such, we keep the fragment header information in addition to the pointer to the content store entry in the VRB. Now, whenever a VREP node forwards an ACK, it can reply with unacknowledged fragments it already received and mark the fragment as received in the outgoing ACK bitmap. This provides us with an en-route caching solution on the SFR fragment layer.

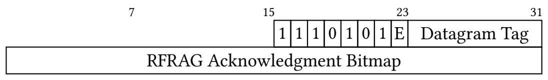
The content store entry associated with the VRB entry, however, must not be used to reply to incoming interests until the fragmented content chunk is complete. Replying to an ICN interest with a single fragment would poison the network with incomplete content chunks. To prevent this, we introduce a `TENTATIVE` flag, which is assigned to content store entries. Entries marked with this flag still follow common ICN cache replacement behavior. The `TENTATIVE` flag remains set as long as the content chunk is incomplete. Completeness of the chunk can be determined in two ways. (*i*) by counting bytes of forwarded fragments and compare the sum to the datagram size provided in the first fragment, or (*ii*) when a VREP forwards an ACK for all fragments.

Figure 3 shows all components of a VREP and their relations.

We want to emphasize that our VREP concept does not contradict our original objective of saving memory when using fragment forwarding: In our approach, the data is stored in the content store, which is typically already available on an ICN node, not in an additional ICNLoWPAN reassembly buffer. If the node has no content store, the VREP concept should not be used. Such a non-VREP node would than only be able to forward fragments in accordance with SFR with no caching benefits.


7
15
23
31
1 1 1 0 1 0 0 E Datagram Tag
X Sequence Fragment Size Fragment Offset
Payload

(a) Recoverable Fragment (RFRAG)



(b) RFRAG-ACK

Figure 4: SFR dispatch types and headers.

Furthermore, it is worth noting that the VREP—while following a cross-layer design—does not break additional interfaces compared to ICNLoWPAN. ICNLoWPAN already requires access to data structures of ICN, *e.g.,* for the name compression feature.

3.2 Implementation

We implement the VREP extension in the RIOT operating system [2] because core building blocks, ICNLoWPAN and SFR, are already available. This allows us to provide a maintained solution to the ICN community in the long term.

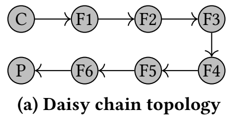
RIOT offers two ICN implementations, CCN-lite and NDN-RIOT [21]. We use the NDN implementation of CCN-lite for our experiments because CCN-lite gives us the option to explore additional ICN flavors in future work.

To implement the VREP, we extend the CCN-lite content store to maintain incomplete, `TENTATIVE` content store entries, as described in Section 3.1. The fragment information is already stored in the reassembly buffer of RIOT, so we only need to extend the VRB to store this information in addition to the pointer to the `TENTATIVE` content store entry. Similar, SFR also needs an extension to collect and reply with previous fragments based on the `TENTATIVE` content store entries.

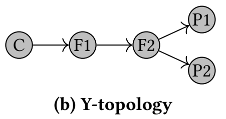
4 EVALUATION AND RESULTS

4.1 Experiment Setup

Testbed. We conduct our experiments in the FIT/IoT-LAB testbed and choose the Grenoble site, as this site allows for large multi-hop deployments. Nodes are constrained IoT devices featuring Cortex-M3 MCUs, 64 kilobytes of RAM, 512 kilobytes of ROM, and IEEE 802.15.4 radios. The radio chip provides basic MAC layer features such as CSMA/CA, link layer retransmissions, and acknowledgements.



(a) Daisy chain topology



(b) Y-topology

Figure 5: Topologies used in experiments, incl. consumer (C), forwarders (F_i), and producer (P). Arrows show FIB routes.

To quantify the required memory, we compile our applications with GCC v9.2 for ARM on Ubuntu 18.04, optimized for size.

Communication scenarios. In our experiments, we compare three fragmentation approaches, (*i*) classic 6LoWPAN fragmentation with hop-wise reassembly, short *HWR*, (*ii*) SFR without a VREP implementation, short *SFR w/o VREP*, and (*iii*) SFR implementing our VREP proposal, short *SFR w/ VREP*; each deployed in two different topologies (see Figure 5).

As we are not interested in topology changes, we use static routes in all FIBs. The daisy chain topology consists of 7 hops between consumer and producer. This topology will uncover caching behavior [9, 19] in particular. The Y-topology consists of 1 consumer and 2 producers, consumer and producers are separated by 2 additional hops. This topology is chosen to show that the reassembly buffer in HWR may be a bottleneck but can be bypassed by the VRB in SFR [23, 25].

The producers are configured to reply with 500 bytes of data to any ICN interest with a pre-configured name prefix, resulting in 6 fragments. The consumer sends 300 distinct, unfragmented interests for that pre-configured prefix with a uniformly distributed delay of 1.0 ± 0.25 s between each interest. Each scenario runs 10 times for each fragmentation approach and topology setting.

Software configuration. RIOT offers a variety of compile-time configuration parameters to adapt to use cases. In most scenarios, we can use default configurations. We need, however, to adapt some parameters to be able (*i*) to show the effects of packet loss and (*ii*) reduce the effects of setup adaptations such as the rudimentary congestion control in SFR.

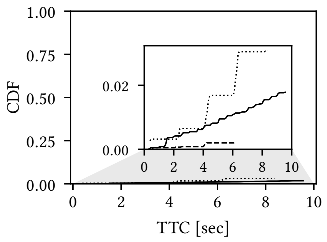
CCN-lite is configured to expire cache entries after 30 s and PIT entries after 10 s. Interests are repeated at most 3 times using a retransmission timer of 2 s.

In SFR, we set the initial window size to 1, as we learned from prior experiments (not shown) that the simple congestion control mechanism proposed in [24] quickly leads to that. Furthermore, as different latencies occur in each topology because of different path lengths, we adjust the ACK request (ARQ) timeout in SFR (Y: 150 ms, daisy chain: 500 ms) and the number of retransmissions (Y: 4 retransmissions, daisy chain: 2 retransmissions).

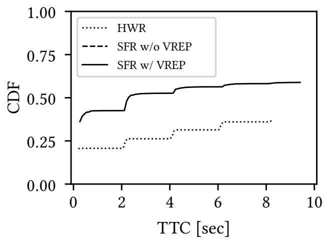
Each node has a reassembly buffer size of 1 entry and a VRB size of 32 entries. For each of the overall 33 entries we assign a timeout of 1 s. The Fragment Header Information Store is capped at 198 entries (6 fragments per 33 entries).

4.2 Time to Completion

The time to completion quantifies the delay between sending an interest and receiving all fragments of the corresponding data chunk. Figure 6 displays our measurement results, including completed



(a) Daisy chain topology



(b) Y-topology

Figure 6: Time to completion (TTC). In the Y-topology, SFR w/o VREP does not complete any requested content.

and failed requests. To highlight details, Figure 6a presents a zoom that rescales the *y*-axis. It is clearly visible that the performance depends on both the fragmentation scheme and the topology in use.

In the daisy chain topology, all three fragmentation approaches lead to very few successful content deliveries, with HWR exhibiting marginal better performance (see Figure 6a). In the Y-topology, SFR w/ VREP outperforms HWR (see Figure 6b), and SFR w/o VREP does not lead to any successful content request. To verify our observations, we also configured a moderate request rate of 10 ± 2.5 s (not shown). Here, both HWR and SFR w/ VREP completed 95% of the requested content items in the range of the round-trip times of the topologies. The remaining 5% did not exceed the configured ARQ timeout for SFR and were delivered based on fragment retransmissions of the SFR layer.

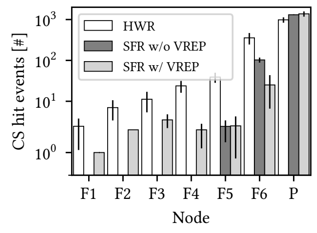
In the Y-topology, the performance benefit of SFR w/ VREP over HWR is due to the single reassembly buffer entry in HWR, which becomes a bottleneck at the forwarders. We will discuss more implications of that bottleneck in Section 4.3 and Section 4.4.

The key reason for our observations is that SFR introduces additional messages (*i.e.,* ACKs), which increase the load in the network and amplify potential message loss. Without our VREP extension, NDN cannot cache the content, *i.e.,* any SFR node needs to retransmit interests to the original content source. Those interest retransmits are clearly visible in the 2 s steps of the CDF. Retransmissions of the fragments caused by SFR are visible at the step with length of the respective ARQ timeout lengths within the CDF.

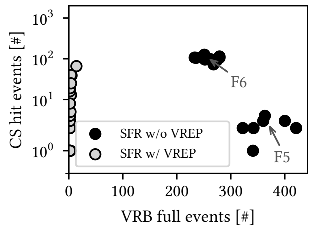
We do not see much of the promised benefits of SFR compared to HWR in any of the scenarios but only moderate performance improvements. This has two reasons: (*i*) The window size is 1, so every fragment needs to be acknowledged before the next fragment is sent. In total, this gives only slightly more time to complete a datagram compared to HWR. (*ii*) In SFR w/o VREP, a higher delay is the result of the single-antenna radios used in our experiments. In combination with CSMA/CA, such deployment may impact performance as shown in [15]. The latter can be subverted by using a coordinating MAC protocol such as IEEE 802.15.4e.

4.3 Cache Hits and Reassembly Buffer Usage

Figure 7a displays the average number of cache hits per node when fetching data from the content store in the daisy chain topology. Nodes are sorted based on the topology in Figure 5a.



(a) Cache hits per node.



(b) Cache hits coinciding with fallback to HWR due to VRB being full at the forwarders.

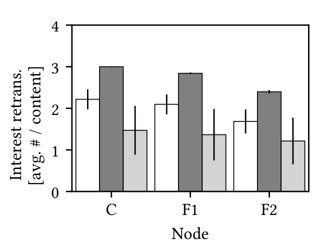
Figure 7: Cache hits in daisy chain topology (see Figure 5a). In Figure 7b, arrows refer to the mean of each cluster, which belongs to forwarders F5 and F6 in SFR w/o VREP.

When using HWR or SFR w/ VREP cached content is available on all forwarders. In general, both SFR approaches experience less cache hits close to the consumer, which leads to increased content store access at the producer, compared to HWR.

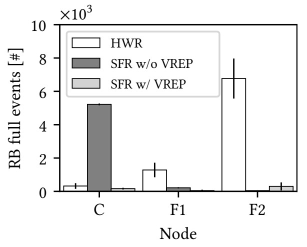
To better understand the impact of the fallback strategy when the VRB is full, Figure 7b shows the number of cache hits (y-axis) depending on the number of events when the VRB is overloaded (x-axis). This scatter plot reveals two clusters for SFR w/o VREP, each cluster belongs to a single intermediate forwarder (arrows refer to the mean). Whenever the VRB is full, SFR w/o VREP falls back to HWR, in turn filling the cache with completed datagrams, resulting into higher and stable cache hit ratio per node. In contrast to this, when SFR w/ VREP is used, the VRB is rarely full because most of the cache hits come via the VREP.

The impact of the cache on the network performance can be observed in Figure 8. We focus on the Y-topology as this includes multiple producers. Figure 8a shows the number of interest retransmissions per content per node. For SFR w/o VREP, the number of retransmissions is significantly higher at every node compared to other approaches, while SFR w/ VREP reduces the number of retransmissions again compared to HWR. Figure 8b shows the number of events in which the reassembly buffer is full per node relatively over all runs. The results vary significantly and depend on the node and scheme in use. SFR w/o VREP leads to a full reassembly buffer at the consumer (C), whereas HWR fills the buffer of the branching node (F2).

The reasons for our observations are as follow: SFR w/o VREP requires a high amount of Interest retransmissions. Due to the missing cache, each retransmit causes the emission of a new fragmented content chunk at the producer. This triggers the creation of a new reassembly buffer entry once the first fragment of each content reaches the consumer. If the reassembly buffer is full, however, we lose this content, as there is no way to reconstruct it. This accumulation of incomplete content items prevents releasing reassembly buffer entries. HWR, however, loses data due to the reassembly buffer, which introduces a forwarding bottleneck at the forwarder closest to the producers (F2): There is only 1 reassembly buffer entry in HWR but 32 VRB entries in SFR. This leads to a full reassembly buffer more often in HWR compared to both SFR approaches.



(a) Interest retransmissions per content



(b) <u>R</u>eassembly <u>B</u>uffer (RB) full events

Figure 8: Reassembly Buffer at consumer being filled due to cascading Interest retransmissions in Y-topoloy (see Figure 5b).

Our analysis shows that a content chunk can be delivered closer to the consumer in case of loss as long as the VREP is in use. The use of the reassembly buffer at every hop can be a hindrance for HWR if a lot of forwarding bottlenecks are on path in the network. Here SFR w/ VREP can perform better. For two distinct reasons, we do not measure more cache hits for SFR w/ VREP compared to HWR. First, when there is more loss with SFR w/ VREP, cache entries remain `TENTATIVE` more often due to that loss. Second, when there is less loss, there are less Interest retransmissions in case of SFR w/ VREP, leading to less cache lookups in the first place.

4.4 Memory Consumption

The binary of the ICNLoWPAN fragmentation module requires 1.7 kilobytes (KiB) of ROM and 1.2 KiB of RAM for HWR; it requires 5.5 KiB of ROM and 5.2 KiB of RAM for SFR w/o VREP; and 6.4 KiB of ROM and 7.5 KiB of RAM for SFR w/ VREP. The cache entry extension that implements the `TENTATIVE` flag in CCN-lite adds around 900 bytes of ROM. It is worth noting, however, that in all cases only 1 reassembly buffer entry is used, so HWR is only able to handle one datagram at a time. On the other hand, both SFR approaches allow for 32 VRB entries—in addition to that 1 reassembly buffer entry—which allows to handle up to 33 datagrams concurrently. The increase in size, hence, comes from new data structures adding complexity. If we would increase the reassembly buffer to 32 entries, the RAM usage of HWR would increase to 5.4 KiB to store meta-data alone.

Likewise, the packet buffer usage of ICNLoWPAN to store the actual fragments and datagrams would massively increase at the forwarders. Figure 9 shows the average usage of the 8 KiB dynamic packet buffer used by ICNLoWPAN [14] at a request rate of 10 ± 2.5 s in the daisy chain topology averaged over the course of all runs (see Section 4.2). In this scenario, side-effects on the packet buffer usage, *e.g.*, from retransmissions, are minimal, which provides us a good baseline for the packet buffer usage of each approach. Nevertheless, the packet buffer usage of HWR is much higher as for the SFR approaches at the forwarders because every datagram needs to be stored completely before a node can forward it. At the producer and consumer the packet buffer usage exhibits similar results for fragmentation and reassembly. Increasing the reassembly buffer size would also multiply the packet buffer usage at the forwarders for HWR, requiring even more RAM.

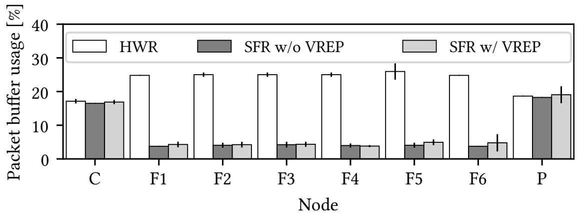


Figure 9: Packet buffer usage (of 8 KiB) for 300 interests emitted every 10.0 ± 2.5 s in the daisy chain topology (see Figure 5a).

In conclusion, while additional complexity and data structures of both SFR approaches add to RAM and ROM usage, the increase of the required, more expensive RAM is small compared to the additional RAM overhead that is required to allow HWR to yield comparable performance in complex topologies.

5 RELATED WORK

Fragmentation and the support of selective acknowledgments is common in IP-based IoT. Kent *et al.* [12, 13] discussed potential harms already at the beginning of the Internet and thus paved the way for proper protocol design. Instead of using bitmaps, protocols such as NETBLT [6] use the last consecutive received sequence number to selectively acknowledge fragmented messages.

Lenders *et al.* [15] recently evaluated the drawbacks of fragment forwarding in low-latency lossy networks compared to hop-wise reassembly in 6LoWPAN. Alternative approaches beyond the solutions proposed by the IETF mainly focus on datagram prioritization [26, 27]. There is also a hop-wise selective, negative acknowledgment approach proposed by Chowdhury *et al.* [5], which is not compatible to the SFR proposal by the IETF. As we aim for IETF/IRTF-compliant solutions, we did not consider such proposals in this paper.

The NDN Link Protocol (NDNLP) [22] features fragmentation and link acknowledgments, but it does not allow for selective acknowledgments. Potentially, this increases the number of exchanged messages. Furthermore, NDNLP is not compatible with ICNLoWPAN and thus does not benefit from ICNLoWPAN features such as parallel deployment in ICN and IP networks. A similar approach was proposed by Mosco *et al.* [17], which has the same drawbacks and requires a synchronization of sequence numbers. Synchronization is an additional source of error and can increase the fragmentation overhead even more. This is not necessary using fragment forwarding with selective fragment recovery.

An NDN-specific fragment forwarding approach with en-route content caching was proposed by Ghali *et al.* [7] and followed up on by Mosko and Wood [18]. In contrast to the fragment forwarding techniques presented in this paper, NDN semantics are used from scratch. A new fragment message type is introduced that identifies the complete content chunk by its name. Unfortunately, there is no limits regarding the length of an NDN name. Consequently, this protocol extension seems unsuitable for low-power lossy networks with small link layer PDU.

Shang *et al.* [21] used a dedicated 3-byte fragmentation header in their NDN-RIOT implementation to communicate over IEEE 802.15.4. The header re-uses the 6LoWPAN Mesh header dispatch and thus might collide with standard 6LoWPAN communication. Moreover, it only employs hop-wise reassembly. The 5-bit sequence number value space offers potential for an SFR extension similar to [24]. This was however not visited by them.

6 DISCUSSION

In Section 4, we showed advantages of VREP compared to SFR without any ICN support. We were not able to show the timing benefits of SFR over HWR, though. As single-antenna radios can easily be overwhelmed with rapidly incoming and outgoing messages when just using CSMA/CA, the VREP was not able to enfold its full potential and thus falls back into HWR. We want to emphasize two side observations. First, most fragment forwarding schemes for 6LoWPAN are conceptualized for much slower data rates than we used in our experiments, which would lead to accumulated time-to-completion in HWR. Second, such forwarding schemes are typically based on the IEEE 802.15.4e Time Slotted Channel Hopping (TSCH) MAC layer [15, 23], which provides a more robust media access—a moderate amount of fragment loss prevents overloading the constrained reassembly buffer and VRB. Full 802.15.4e TSCH support is on-going activity in the RIOT community and our implementation can benefit from this as soon as available.

In addition to the deployment of TSCH, we identify a second option for improvement: proper congestion control in SFR by leveraging acknowledgments. The SFR specification [24] does not define a specific congestion control mechanism to adapt window sizes. Exploring this will be part of our future work as our preliminary results are promising. ICN caching in combination with selective fragment recovery enabled by the VREP extension can help to increase the content delivery rate when the VRB is balanced.

7 CONCLUSIONS AND OUTLOOK

In this paper, we presented the potential of combining ICNLoWPAN and Selective Fragment Recovery (SFR). Our proposed solution is based on a Virtual Reassembling Endpoint (VREP) extension for SFR that enables en-route reassembly of fragments in the content store of an ICN node. In turn, SFR forwarders are enhanced to restore fragments en-route when missing fragments are requested by the reassembling endpoint, instead of forwarding the requesting acknowledgment to the fragmenting endpoint. This provides caching benefits on the SFR layer.

In a topology exhibiting long paths, our findings indicate that the performance of SFR with the VREP extension (SFR w/ VREP) is comparable to hop-wise reassembly (HWR) with a slight advantage for HWR, while SFR without the VREP extension (SFR w/o VREP) is struggling to complete requested contents. In topologies with many converging paths, SFR w/ VREP leads to a better content delivery rate compared to HWR, SFR w/o VREP again performs poorly. As such, it is highly advisable to assess the use case for the LLN stub of the ICN first, before deciding if SFR or HWR should be picked.

For future work, we identify three topics. (*i*) Consideration of more complex MAC layers such as IEEE 802.15.4e, (*ii*) analysis of advanced congestion control mechanisms employed in SFR, and (*iii*) an analysis of the impact of different ICN caching strategies on the VREP extension.

A Note on Reproducibility

We explicitly support reproducible research [1, 20]. Our experiments have been conducted in an open testbed. The source code of our implementations (including scripts to setup the experiments, RIOT measurement apps *etc.*) will be available on Github at

https://github.com/5G-I3/ACM-ICN-2020-SFR.

Acknowledgments

We would like to thank our shepherd Karen Sollins and the anonymous reviewers for their feedback on this paper. This work was supported in parts by the German Federal Ministry of Education and Research within the 5G project *I3* and the VIP+ project *RAPstore*.

REFERENCES

[1] ACM. Jan., 2017. Result and Artifact Review and Badging. http://acm.org/publications/policies/artifact-review-badging.

[2] Emmanuel Baccelli, Cenk Gündogan, Oliver Hahm, Peter Kietzmann, Martine Lenders, Hauke Petersen, Kaspar Schleiser, Thomas C. Schmidt, and Matthias Wählisch. 2018. RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT. *IEEE Internet of Things Journal* 5, 6 (December 2018), 4428–4440. http://dx.doi.org/10.1109/JIOT.2018.2815038

[3] 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 Information-Centric Networking (ICN-2014)*. ACM, New York, 77–86. http://dx.doi.org/10.1145/2660129.2660144

[4] Carsten Bormann and Thomas Watteyne. 2020. *Virtual reassembly buffers in 6LoWPAN.* Internet-Draft – work in progress 02. IETF.

[5] Aminul Haque Chowdhury, Muhammad Ikram, Hyon-Soo Cha, Hassen Redwan, SM Shams, Ki-Hyung Kim, and Seung-Wha Yoo. 2009. Route-over vs Mesh-under Routing in 6LoWPAN. In *Proc. of the 2009 International Conference on Wireless Communications and Mobile Computing (IWCMC): Connecting the World Wirelessly.* ACM, 1208–1212. https://doi.org/10.1145/1582379.1582643

[6] D.D. Clark, M.L. Lambert, and L. Zhang. 1987. *NETBLT: A bulk data transfer protocol.* RFC 998. IETF.

[7] Cesar Ghali, Ashok Narayanan, David Oran, Gene Tsudik, and Christopher A. Wood. 2015. Secure Fragmentation for Content-Centric Networks. In *Proc. of the 14th International Symposium on Network Computing and Applications.* 47–56.

[8] Cenk Gündogan, Peter Kietzmann, Martine Lenders, Hauke Petersen, Thomas C. Schmidt, and Matthias Wählisch. 2018. 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, 159–171. https://doi.org/10.1145/3267955.3267967

[9] Cenk Gündogan, Peter Kietzmann, Thomas C. Schmidt, and Matthias Wählisch. 2018. HoPP: Robust and Resilient Publish-Subscribe for an Information-Centric Internet of Things. In *Proc. of the 43rd IEEE Conference on Local Computer Networks (LCN).* IEEE Press, Piscataway, NJ, USA, 331–334. http://doi.org/10.1109/LCN.2018.8638030

[10] Cenk Gündogan, Peter Kietzmann, Thomas C. Schmidt, and Matthias Wählisch. 2019. ICNLoWPAN – Named-Data Networking in Low Power IoT Networks. In *Proc. of 18th IFIP Networking Conference.* IEEE Press, Piscataway, NJ, USA, 1–9. http://dx.doi.org/10.23919/IFIPNetworking.2019.8816850

[11] Cenk Gundogan, Thomas Schmidt, Matthias Waehlisch, Christopher Scherb, Claudio Marxer, and Christian Tschudin. 2020. *ICN Adaptation to LoWPAN Networks (ICN LoWPAN).* Internet-Draft – work in progress 08. IETF.

[12] Christopher A. Kent and Jeffrey C. Mogul. 1987. *Fragmentation Considered Harmful.* Vol. 17. Western Research Laboratory.

[13] Christopher A. Kent and Jeffrey C. Mogul. 1995. Fragmentation Considered Harmful. *SIGCOMM Comput. Commun. Rev.* 25, 1 (Jan. 1995), 75–87. https://doi.org/10.1145/205447.205456

[14] Martine Lenders, Peter Kietzmann, Oliver Hahm, Hauke Petersen, Cenk Gündogan, Emmanuel Baccelli, Kaspar Schleiser, Thomas C. Schmidt, and Matthias Wählisch. 2018. *Connecting the World of Embedded Mobiles: The RIOT Approach to Ubiquitous Networking for the Internet of Things.* Technical Report arXiv:1801.02833. Open Archive: arXiv.org. https://arxiv.org/abs/1801.02833

[15] Martine S. Lenders, Thomas C. Schmidt, and Matthias Wählisch. 2019. 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.

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

[17] Marc Mosko and Christian Tschudin. 2016. *ICN 'Begin-End' Hop by Hop Fragmentation.* Internet-Draft – work in progress 02. IETF.

[18] Marc Mosko and Christopher A Wood. 2015. Secure fragmentation for content centric networking. In *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems.* IEEE, 506–512. https://doi.org/10.1109/MASS.2015.51

[19] Jakob Pfender, Alvin Valera, and Winston KG Seah. 2018. Performance comparison of caching strategies for information-centric IoT. In *Proceedings of the 5th ACM Conference on Information-Centric Networking.* 43–53.

[20] Quirin Scheitle, Matthias Wählisch, Oliver Gasser, Thomas C. Schmidt, and Georg Carle. 2017. Towards an Ecosystem for Reproducible Research in Computer Networking. In *Proc. of ACM SIGCOMM Reproducibility Workshop.* ACM, New York, NY, USA, 5–8.

[21] 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.

[22] Junxiao Shi and Beichuan Zhang. 2012. *NDNLP: A Link Protocol for NDN.* NDN, Technical Report NDN-0006. NDN Team.

[23] Yasuyuki Tanaka, Pascale Minet, and Thomas Watteyne. 2019. 6LoWPAN Fragment Forwarding. *IEEE Communications Standards Magazine* 3, 1 (July 2019), 35–39. https://doi.org/10.1109/MCOMSTD.2019.1800029

[24] Pascal Thubert. 2020. *6LoWPAN Selective Fragment Recovery.* Internet-Draft – work in progress 21. IETF.

[25] Thomas Watteyne, Pascal Thubert, and Carsten Bormann. 2020. *On Forwarding 6LoWPAN Fragments over a Multihop IPv6 Network.* Internet-Draft – work in progress 15. IETF.

[26] Andreas Weigel. 2017. *Forwarding strategies for 6LoWPAN-fragmented IPv6 datagrams.* Ph.D. Dissertation. Technische Universität Hamburg-Harburg.

[27] Andreas Weigel, Martin Ringwelski, Volker Turau, and Andreas Timm-Giel. 2013. Route-over forwarding techniques in a 6LoWPAN. In *International Conference on Mobile Networks and Management.* Springer, 122–135.

[28] Zhiyi Zhang, Edward Lu, Yanbiao Li, Lixia Zhang, Tianyuan Yu, Davide Pesavento, Junxiao Shi, and Lotfi Benmohamed. 2018. NDNoT: A Framework for Named Data Network of Things. In *Proceedings of the 5th ACM Conference on Information-Centric Networking.* ACM, New York, NY, USA, 200–201.

APPENDIX: A NOTE ON STATEFUL FORWARDING

In this paper, we focused on solving the caching challenges introduced by SFR. In our original design, we also considered to combine the stateful forwarding mechanisms of NDN and SFR. For an Interest, an NDN node stores a pair of name and incoming face in the pending interest table (PIT) and sends Data packet back to the consumer using that PIT entry. SFR, on the other hand, links fragment tag and forwarding information in the VRB to incoming fragments and sends back acknowledgments from the virtual reassembling end-point to the fragmenting end-point.

Aligning both states would require cross-layer intervention while bringing little gain. Primarily, the problem arises from the allocation of packet sizes for request and response being reversed in NDN and SFR, respectively: In SFR, the responding acknowledgment does not carry any data and is thus shorter than the payload carrying fragment. In NDN, the response is a content chunk, so in most—if not all—cases they are larger than the requesting interest. Consequently, combining the forwarding states of both protocols would only be helpful in a very small number of use cases, *i.e.,* when the interest would be fragmented but the data chunk is not.