If you cite this paper, please use the Computer Communications reference: C. Gündoğan, J. Pfender, P. Kietzmann, T. C. Schmidt, M. Wählisch, On the Impact of QoS Management in an Information-centric Internet of Things, Computer Communications 154 (2020) 160–172. URL https://doi.org/10.1016/j.comcom.2020.02.046

On the Impact of QoS Management in an Information-centric Internet of Things

Cenk Gündoğan[a], Jakob Pfender[b], Peter Kietzmann[a], Thomas C. Schmidt[a], Matthias Wählisch[c]

[a] *Hamburg University of Applied Sciences, Department of Computer Science, Berliner Tor 7, 20099 Hamburg, Germany*
[b] *Victoria University of Wellington, School of Engineering and Computer Science, Wellington, New Zealand*
[c] *Freie Universität Berlin, Inst. of Computer Science, Takustr. 9, 14195 Berlin, Germany*

Abstract

The Internet of Things (IoT) comprises a relevant class of applications that require Quality of Service (QoS) assurances. Information Centric Networking (ICN) has shown promising characteristics in constrained wireless networks, but differentiated QoS has not yet fully emerged. In this paper, we design and analyze a QoS scheme that manages the NDN resources *forwarding and queuing priorities*, as well as the *utilization of caches* and of *forwarding state space*. In constrained wireless networks, these resources are scarce with a potentially high impact due to lossy radio transmission. We explore the two basic service qualities (i) *prompt* and (ii) *reliable* traffic forwarding. We treat QoS resources not only in isolation, but correlate their use on local nodes and between network members. Network-wide coordination is based on simple QoS code points that can be distributed via a routing protocol. Fairness measures that prevent resource starvation are part of this management scheme. Our findings indicate that our coordinated QoS management in ICN does not only effectively prioritize the privileged data chunks, but also improves regular data communication. We can show that appropriate QoS coordination can enhance the overall network performance by more than the sum of its parts and that it exceeds the impact QoS can have in the IP world.

Keywords: Information-Centric Networking, Quality of Service, Network Performance Measurement, Wireless Sensor Network

1. Introduction

The advent of the Internet of Things (IoT) introduced the vision of omnipresent and always connected sensors and actuators that generate business models from new products, innovative processes, and data—a total of 1.6 Zettabytes is soon expected [1] in this rapidly growing business segment. Various use cases are emerging, some of which raise specific requirements for high reliability or low latency. Prominent examples are industrial control systems, autonomous driving, emergency alerts, and disaster scenarios. Common link layers that outreach to the embedded edge such as LowPANs, LoRa [2], or NB-IoT [3] are often of low power lossy nature and do not provide an intrinsic support of the desired service qualities for data transmission. It is thus left to the network layer to optimize the overall performance of the communication system.

ICN has been a promising candidate for networking the IoT edge for a while [4–9]. Recent experimental studies [10, 11] confirmed that ICN can sustain high reliability at moderate latency penalty even on lossy wireless links, and showed how it could be integrated into an emerging 5G edge architecture. Nevertheless, current analyzes and solutions are built purely on equal resource sharing. QoS resource management for ICN is still in its early stage and studies of ICN properties under prioritization and active QoS management are missing.

Quality of Service (QoS) in IP networks has been around for two decades, but so far has experienced remarkably little deployment. Its hesitant adoption is commonly understood to have two reasons: limited scalability (IntServ [12, 13]) and plain resource trading (DiffServ [14, 15])—the latter is often referred to as managed unfairness. While QoS in the IP world is mainly restricted to managing forwarding resources (link capacities and buffer spaces) [16] and the IEEE follows this approach on Layer 2 with Time Sensitive Networking (TSN) [17, 18], Information-Centric Networking (ICN) [19, 20] offers additional resource dimensions such as in-network caches and forwarding states that can shape network performance significantly.

In this paper, we thoroughly explore the impacts of QoS management on NDN [21, 22] in the resource-constrained IoT using RIOT [23]. We extend our early work [24] on the two basic service dimensions *(i) prompt* and *(ii) reliable* traffic forwarding and define a simple yet efficient distributed management scheme. We carefully implement these QoS semantics by employing not only the NDN resources *forwarding capacity, PIT state,* and *cache* alone,

Email addresses: cenk.guendogan@haw-hamburg.de (Cenk Gündoğan), jakob.pfender@ecs.vuw.ac.nz (Jakob Pfender), peter.kietzmann@haw-hamburg.de (Peter Kietzmann), t.schmidt@haw-hamburg.de (Thomas C. Schmidt), m.waehlisch@fu-berlin.de (Matthias Wählisch)

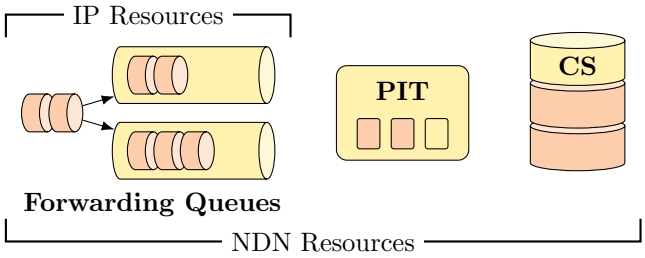

Figure 1: Manageable resources in IP vs. NDN.

but by correlating resources internally on a node and also externally between nodes. These correlations can be performed without additional signaling overhead. We design and analyze fairness measures that prevent network members from starvation and demonstrate the feasibility of our approach in a realistic showcase of a disaster scenario.

Our findings from extensive experiments in a large testbed confirm the efficacy of our approach. We can show that—in contrast to the IP world—these QoS measures do not sacrifice the performance of unprioritized traffic. Even though resources shift to the prime packets, best effort flows still uphold their performance or even improve. A thorough analysis reveals the positive effects of resource correlation, which raises the overall network performance to a higher level than in the state of uncoordinated resource allocation. These results clearly strengthen ICN as a candidate for differentiated IoT network services.

The remainder of this paper is structured as follows. The problem space of managing distributed resources is discussed in Section 2 along with related work. Section 3 introduces the QoS building blocks and how we manage the resources. Our implementations and evaluations of prioritized forwarding are presented in Section 4. Section 5 is dedicated to analyzing the impact of caching on network services. The problems of fairness and countermeasures to resource starvation are derived in Section 6, followed by our use case study in Section 7. We conclude with an outlook in Section 8.

2. The Problem of Distributed Resource Management in ICN

2.1. Problem Statement

Implementing service differentiation and assurance in a network raises the challenge of managing distributed resources without sacrificing them. Common Internet approaches follow a flow-based (e.g., IntServ) or a class-based (e.g., DiffServ) concept. Flow-based resource reservation requires dedicated signaling and state, which quickly reaches scalability limits with resource exhaustion. In the presence of ubiquitous in-network caching and request aggregation, content endpoints are unspecified for ICN and data paths are inherently multi-source, multi-destination, and possibly widely disjoint. This makes ICN flows difficult to identify and to maintain.

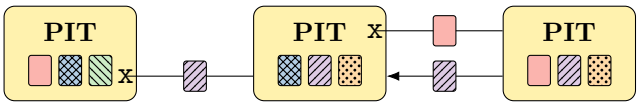

Figure 2: PIT decorrelation terminates data paths.

Resource allocation according to packet classes requires ingress shaping and filtering, since unforeseen traffic bursts quickly exhaust the per-class reservations and counteract service assurances. In several ICN flavors including NDN and CCNx, link occupancy and forwarding demands are steered hop-by-hop in a request-response fashion. Small requests trigger data replies of unknown size, provenance, and timing. This complicates reliable resource predictions for responses in NDN. Shaping and dropping Interests can prevent resource exhaustion, but may leave the network underutilized. Restricting ingress only to data may lead to bursts of unsatisfied Interests, which waste network resources.

In-network caches in ICN enrich the field of manageable resources. Caches reduce latency and forwarding load and often take the role of a (large, delay-tolerant) retransmission buffer. With NDN/CCNx, additional resources come into play in the form of Pending Interest Tables (PITs) that govern stateful forwarding. The overall resource ensemble is visualized in Figure 1 and raises concerns. Capacities in forwarding, caching, and pending Interest state may be largely heterogeneous. A wirespeed forwarder may supply negligible cache memory compared to its transmission capacity, for example. In the IoT the opposite is often true in that flash memory is normally shipped in 'infinite' sizes when compared to the main memory (PIT) or the wireless data rate. A beneficial resource management faces the problem of how to carefully balance these resources and arrive at an overall optimized network performance [25].

Resource complexity, however, extends beyond a single system. The impact of distributed resources is easily flawed if management cannot jointly coordinate contributions. Neighboring caches, for instance, are less effective if filled with identical copies. The effective overall cache capacity of the system can be increased by using cooperative caching strategies, which are needed in particular if caches are too small to hold the requested amounts of data during their validity periods. A more delicate problem arises from PIT state management. If neighboring PITs diverge and no longer represent common forwarding paths (see Figure 2) all data flows terminate and forwarding resources are wasted. This problem of state decorrelation was first reported in [26].

2.2. QoS Qualifiers

QoS extensions for ICN have recently attracted attention and generated various efforts within the IRTF ICN research group [27–31]. A proposed flow classification mechanism [29] differentiates traffic flows based on content name

prefixes using two different methods. The first method, *EC3*, introduces a new message header entry in Data packets that indicates the prefix length of the content name characteristic for the classification process. Once a consumer learns about such an equivalence class, it can also include equivalence class indicators into subsequent Interests. The second method, *ECNCT*, encodes classification indicators directly into names at content creation using a new type of name component. This solution does not inflate messages with additional headers, which is advantageous for constrained IoT deployments. Nevertheless, encoding flow classification indicators in typed name components leads to an inflation of names in the routing system. Identical content published with different classification values will lead to duplicate names that cannot be aggregated in Forwarding Information Bases (FIBs). Analogously, the default matching functions used for the Pending Interest Table (PIT) and Content Store (CS) will consider names that differ only in their flow classification as dissimilar, which conflicts with Interest aggregation and cache utility.

In contrast, our approach to flow classification is simpler and leaner. We attribute QoS service classes to name prefixes and distribute this information to the ICN node (e.g., via a routing protocol) to be kept as persistent state. This approach omits packet overhead, remains compliant with Interest aggregation and caching, but cannot process content of the same name in different service classes.

In the Internet Research Task Force (IRTF), two approaches to QoS treatment are under discussion. An end-to-end QoS framework [30], in which non-routable QoS markers are appended as suffixes to content names. In analogy to DiffServ, these markers are then used to apply different resource allocation mechanisms to the request and response messages. The FIB is extended to ignore QoS markers, and the PIT is modified to disaggregate pending requests that have differing QoS markers. This disaggregation may lead to PIT inflation in particular for setups with a high diversity of QoS markers. This work has also been presented to the IRTF [31].

Tsilopoulos *et al.* [32] identify three different types of traffic based on two characteristics in ICN traffic. The authors introduce two extensions to CCN in order to handle these traffic types, *Persistent Interests* and *Reliable Notifications*. Persistent Interests are valid for Data packets which are produced during a pre-defined period of time. Reliable Notifications inform receivers that real-time data is available and are propagated reliably on a hop-by-hop basis. Notifications that are not acknowledged in time are retransmitted. If a receiver successfully receives a notification but does not receive data in a given time, new Interests are created to renew the request.

2.3. Distributed Forwarding Resources

MIRCC [33] introduces a rate-based, multipath-aware congestion control scheme for ICN. Each Data message in MIRCC contains a rate value which in turn is used to calculate per-link rates. It is inspired by the Rate Control Protocol (RCP) [34] for IP networks.

Al-Naday *et al.* [35] manages forwarding resources and is experimentally evaluated for the PURSUIT architecture. This work implements a QoS differentiation scheme, where each forwarder manages virtual links that include packet queues with varying traffic rates and a designated traffic shaper. QoS information is encoded into the names and determines the mapping of traffic flows to low or high priority virtual links.

2.4. Distributed Cache Management

Caching policies that employ heuristics to inform a caching decision instead of caching all incoming content can be broadly organized into a number of different families, depending on what information they use to reach their caching decision.

The easiest way to achieve higher cache diversity without increasing the complexity of the caching policy is to cache probabilistically. The static version of this approach, commonly known as $Prob(p)$, uses a static probability p that governs how likely a given node will cache a given content chunk. It has been shown [36, 37] that $Prob(p)$ outperforms the default strategy of caching everything in terms of cache diversity, and that lower values for p correlate with higher diversity [36–41].

Instead of using the same static caching probability for all incoming content, a caching strategy may also dynamically compute a probability for each individual node or even for each content chunk in order to adapt the caching behavior to the state of the network. These strategies may be based purely on node-local information, such as CS saturation or battery levels; on properties of the incoming content, such as its name, freshness, type, or producer; or on information from the wider network, such as the position of the caching node in the network topology or the CS contents of neighboring nodes. Examples include *ProbCache* [38], which computes the caching probability of a given content chunk based on the total number of hops between its producer and the consumer that requested it, and *pCASTING* [37], which considers the freshness of the content as well as the node's battery level and CS saturation when calculating p. Both strategies have been found to increase the cache hit ratio, reduce the average number of hops required to hit requested content, and reduce the number of cache evictions [42]. Various other dynamic probabilistic caching strategies have been proposed, with decisions based on content freshness [37, 43], content popularity [44], or whether the content is already in a neighboring CS [45].

Not all caching strategies use the probabilistic approach. Instead, some exploit knowledge about the network topology [46–51], which has the advantage of taking global knowledge about the network into account but often comes at significant costs such as lengthy setup times, communications and memory overhead, and vulnerability to changes in the topology [51]. Another class of caching policies

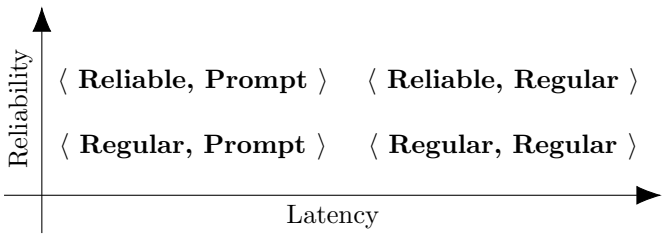

Figure 3: QoS Service Levels.

has nodes cooperate with their neighbors, either explicitly by exchanging information [52] or implicitly by using pre-defined rules [53–55].

3. QoS Building Blocks for NDN

3.1. Distributed Traffic Flow Classification

General purpose networks simultaneously host competing traffic flows that exhibit varying resource requirements and time constraints. This also holds for typical IoT deployments, in which flows originate from sensors and actuators, or from remote cloud services that connect via gateways to the IoT domain. A flow classification is necessary for differentiating packets that belong to separate message flows, whenever the network should allocate distinct resources and treat them in a differentiated manner.

In the IP world, traffic flows are defined by the application endpoints and identified by address/port tuples (IPv4) or addresses plus flow label (IPv6). Since ICN abandons the host-centric paradigm, this definition of traffic flows no longer holds. In the presence of delocalized content and in-network caching, the concept of application endpoints becomes meaningless.

The characterizing property of these packets are content names (or prefixes). In the example of CCNx and NDN, content names appear in related Interest and Data packets. This ubiquity of names and their potential to impose hierarchies on content make them a distinguished component for identifying flows. Accordingly, we propose a traffic flow classification mechanism for NDN that builds on hierarchical names, prefixes, and longest common prefix match. It is explicitly designed to not put a strain on typically resource constrained IoT devices. In this regard, our scheme is computationally simple and does not require an additional overhead in message headers.

In this work, we consider service differentiation with respect to two quality dimensions: *latency* and *reliability*. For simplicity we only use a plain distinction in each quality, which results in a matrix breakdown of service levels as shown in Figure 3. More sophisticated differentiations apply analogously.

Service classes are assigned to flows according to a list of prefixes that are marked with a traffic class and maintained by each node, e.g., as part of the Forwarding Information Base (FIB). Incoming Interest and Data messages are then mapped onto traffic classes by applying a longest prefix match against this list of traffic identifiers. Traffic class names may look as in Tab. 1.

Traffic Class	Priority
/MO/ACM/ICN	<Reliable, Regular>
/MO/ACM/ICN/site/A/alarm	<Reliable, Prompt>
/MO/ACM/ICN/site/B/temp	<Regular, Prompt>

Table 1: Traffic classes and appropriate priority mappings.

Given this example, Interest and Data messages containing the name prefix */MO/ACM/ICN/site/C* would map to the closest valid traffic class. In this case, the longest matching prefix is */MO/ACM/ICN*.

This work assumes such prefix marking to be deployed at all nodes within a network domain. The distribution and maintenance of QoS configurations may be performed by a regular routing protocol or by more specific QoS negotiation protocols. To illustrate the first, we recall that any NDN routing protocol distributes name prefixes across the nodes of its domain to enable a dynamic name-based routing. Prefix (or name) advertisements could be easily augmented by QoS qualifiers so that QoS configurations would allow for actualization within one update cycle of the routing protocol in use. Nevertheless, a general discussion of QoS signaling is beyond the scope of this work, which focuses on a lightweight flow-to-priority mapping for the purpose of quantifying the benefits of QoS resource management.

It may be noted that fine-grained service differentiation within a complex name hierarchy can result in large QoS prefix tables that may inflate the FIB. For the constrained IoT use case, though, we argue that sensor readings and actuator settings are machine type communications (MTC) with (short) names configurable according to processing needs. Hence QoS overheads should easily comply to resource constraints.

3.2. Manageable Resources

3.2.1. Forwarding onto the Link Layer

The link layer manages access to the media and provides space to buffer packets. In low-power wireless networks, media access times are highly susceptible to media saturation and buffer spaces are small. While time slotted technologies such as the IEEE 802.15.4e TSCH mode and Bluetooth Low Energy access media in a deterministically scheduled manner, the unslotted CSMA/CA version of IEEE 802.15.4 or long-range radios such as LoRa are more susceptible to packet collisions between neighboring nodes.

In the IoT, content producers that generate sensor readings may produce egress traffic at a rate that is much higher than the average media access time. In addition, nodes may further need to forward ingress traffic in multihop scenarios. For this purpose, buffering egress traffic is necessary to cope with traffic spikes.

Isolated Decisions	Resource Correlations	Joint Prioritization
Forwarding Queue Expedite *prompt* traffic	**PIT–CS Correlation** If *Prompt* Data has no PI → cached with priority	**PIT Coherence** Aligned node decisions *regular* < *reliable* < *prompt*
Pending Int. Table Evict *regular* for *prompt*	**Fwd–CS Correlation** If *Prompt* Data drops → cached with priority	**CS Efficiency** Aligned node decisions *regular* < *prompt* < *reliable*
Content Store Evict *regular* for *reliable*		

Table 2: Classification of QoS mechanisms and decisions.

Queuing and buffering take the same role in ICN as in the IP world. Class-based forwarding queues will process packets of the *prompt* flow class before packets with a regular priority while buffer space will prevent packet drops. It should be noted, though, that rapidly forwarded packets in NDN will also quickly satisfy PIT entries and thereby free forwarding resources for unprioritized traffic on the same path.

3.2.2. Pending Interest Table

The Pending Interest Table (PIT) enables the stateful forwarding plane of CCNx and NDN and thereby governs the flows in the network. The size of the PIT resource effectively dictates the maximum number of simultaneous open requests and the coherence of PIT entries along a path determines whether flows can propagate without barriers. In normal NDN operation, PIT state is allocated when an Interest message is processed, and it is removed in two scenarios. Either a returning Data message consumes the PIT state, or a timeout after a succession of retransmissions clears the state.

PIT saturation is common even in overprovisioned networks, but is far more likely to occur in IoT deployments. Limited RAM resources, slow processing power, delayed media access, and packet loss in low-power networks with intermittent connectivity can all cause the PIT to reach its maximum capacity.

The typical way of handling incoming Interest messages at a saturated PIT is to drop them in order to avoid cancelling active but incomplete request operations. The penalty for dropping such Interests is an increase in latency due to retransmissions, which usually happen on the scale of seconds. To avoid high latencies for time-sensitive traffic flows, an elaborate PIT eviction strategy is necessary, which accounts for (*i*) unhindered forwarding of prioritized Data, and harmonizes with (*ii*) the retransmission mechanisms of the regular NDN.

3.2.3. Content Store

Due to memory constraints, the Content Store (CS) is typically small, necessitating heuristics for deciding what content to cache at which node instead of indiscriminately caching all incoming content. There is a wealth of existing research on how to make the most efficient use of limited CS space, with a number of different strategies employing various heuristics to decide whether or not to cache incoming content (see Section 2.4). This aspect of caching is called the *caching decision strategy*.

The introduction of traffic flow priorities adds an additional dimension to the caching decision. Regardless of which specific caching decision strategy is employed, content marked as *reliable* should always be cached as it is imperative that this content is available throughout the network. Thus, reception of *reliable* content should not trigger the caching decision strategy; instead, control should be handed directly to the cache replacement strategy (see below). The question whether *prompt* content should be cached with a higher priority than content with *regular* latency requirements is not as clear-cut. Caching *prompt* content with higher priority would have a positive effect on future transmissions of that content object (either by retransmission of the original request or by new requests) and thus have a positive effect on latency, although the potential gain in this aspect is dependent on path length. Any content that is marked as *regular* in both QoS dimensions should be treated as normal; in other words, the caching decision strategy is consulted.

After a node has decided to cache a new content object, an additional step may have to be taken in case the CS is at capacity. This aspect of caching is the *cache replacement strategy*. In most cases, CS contents will be replaced using a simple heuristic such as Least Recently Used (LRU). However, once again the introduction of traffic flow priorities adds an additional dimension to this decision.

In general, incoming content should not replace content of a higher priority. Therefore, content with *regular* latency requirements should not replace *prompt* content and content with *regular* reliability should not replace *reliable* content. When it comes to the correlation between latency and reliability, the primary goal of the CS should be to ensure content availability, which places a stronger emphasis on the reliability aspect. Thus, *reliable* content with *regular* latency should be able to replace *prompt* content if no other content is eligible to be replaced. If all content is of the same priority class, regular replacement

rules (e.g., LRU) should apply.

In probabilistic caching, as introduced in Section 2.4, each node caches incoming content according to a certain probability p. Regardless of how exactly p is determined (whether statically or by one of the dynamic methods discussed in Section 2.4), the probabilistic approach may be refined by differentiating between two separate probabilities p_{reg} for regular content and p_{rel} for reliable content, with $p_{rel} > p_{reg}$. This has the effect that a CS at each node will have different contents, thus contributing to CS diversity across the network by making a larger range of content available as cached copies, while giving consideration to service classes ensures that higher-priority content is still treated preferentially.

3.3. Distributed QoS Management

We are now ready to present our approaches to distributed resource management for supporting QoS in ICN. The corresponding mechanisms fall into three classes, depending on the level of interdependence between resources on the same device or between devices. A summarizing table is further given in Tab. 2.

3.3.1. Locally Isolated Decisions

The straightforward allocation of independent resources to packet forwarding follows three simple rules:

Prioritized forwarding applies to flows marked as *prompt*.

Cache (re-)placement decisions obey the priority order *reliable* (highest) to *regular* (lowest). In the presence of probabilistic caching strategies, the weights are set accordingly.

PIT management operates in favor of rapid packet forwarding, so PIs enter the PIT in the order *prompt* (highest) to *regular* (lowest). Newly arriving Interests that meet a PIT saturated with entries of equal or higher priority will be dropped. Otherwise, a premature PIT eviction would lead to incoherent PIT states along common forwarding paths (see Sec. 2.1).

3.3.2. Local Resource Correlations

These are decisions that entail interaction between mechanisms on the same device (intra-device correlations). This includes the correlation between the caching decision and cache replacement strategies, where e.g. the caching decision may pre-empt the cache replacement decision in the case of *reliable* flows, while the cache replacement decision may drop content even in the case of a positive caching decision if no content of the same or lower service class can be replaced. In detail we take the following steps:

w/ PIT entry If arriving Data meets a valid PIT entry, Data is forwarded according to priorities *and cached* with priority, if marked as *reliable*. In the case of exhausted prioritized forwarding queue, *prompt* traffic will be cached with the highest priority.

w/o PIT entry If arriving Data meets no valid PIT entry, cache placement will still be initiated for *prompt* and *reliable* data in subsequent order. For probabilistic caching, weights are adjusted accordingly.

In balanced, unconstrained NDN networks, returning regular Data meets open PIT states. For saturated PITs, however, PIT entries may time out quickly, or resource management may enforce eviction of PIT entries in favor of other requests. Allowing Data without corresponding PIT entries to be cached may introduce the threat of cache poisoning attacks. However, a simple rate limiting on incoming Data packets and a reduced cache time for these CS entries may reduce the attack surface in our constrained environment. Further analysis of related effects is left to future work.

3.3.3. Joint Resource Prioritization

Such mechanisms affect resources across multiple or all devices in the network (inter-device correlations). These include maintaining PIT coherence by ensuring that all nodes apply uniform QoS mechanisms when replacing content of different service classes, as well as achieving CS diversity by introducing probabilistic caching based on priority classes. In our system, a joint resource prioritization is achieved as follows.

PIT coherence is increased by applying the same PIT eviction strategy at all nodes, i.e., evict *regular* before *reliable* before *prompt*.

Cache efficiency increases with probabilistic caching using aligned configurations of equal cache weights at all nodes. It is noteworthy that probabilistic caching reduces the risk of starvation for low priority content due to higher cache diversity, even if high priority flows dominate the network.

A summary of the different QoS decisions while processing Interest and Data messages are visualized in a flow description in Figure 4.

4. Evaluating Competing Traffic

4.1. Implementation and Experiment Setup

Testbed. We conduct our experiments on the FIT IoT-Lab testbed using typical class 2 [56] IoT devices that feature an ARM Cortex-M3 MCU with 64 kB RAM and 512 kB ROM. Each device further contains an Atmel AT86RF231 2.4 GHz transceiver [57] to operate on the IEEE 802.15.4 radio.

QoS aware nodes. All devices run on RIOT OS [58] version `2019.04` with the integrated NDN network stack CCN-lite [59], which we extended with our QoS management scheme. In addition to the PIT and CS management strategies, a very lightweight prioritized forwarding was

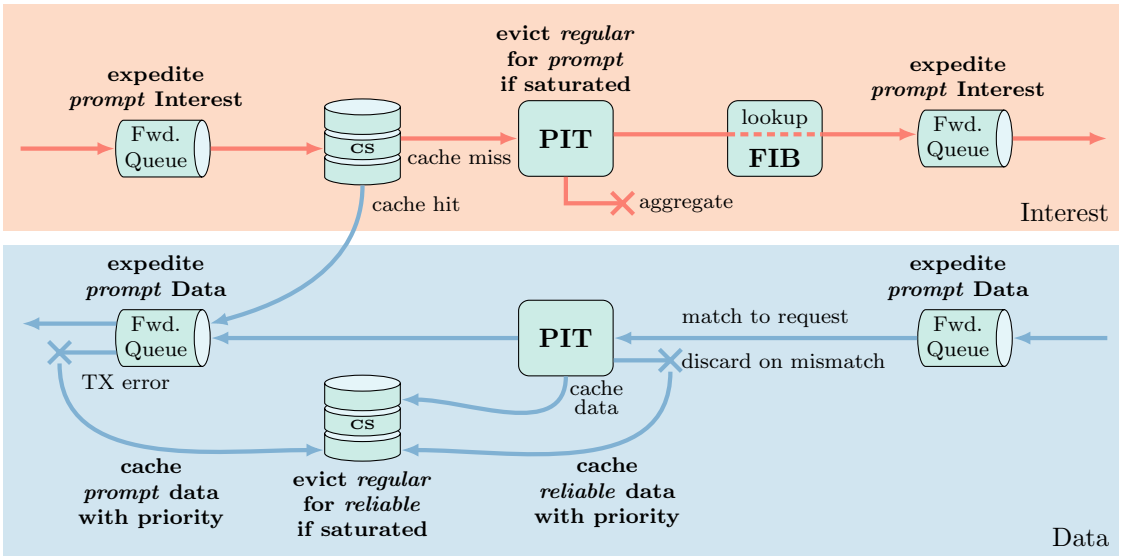

Figure 4: Flow description for Interest and Data messages in a QoS enabled NDN forwarder.

implemented using a single packet double-buffer that allows for pairwise packet re-ordering. We also note that network stack performance usually exceeds link speed. While IEEE 802.15.4 provides a theoretical maximum of 250 kbit/s, I/O and data processing in the network stack is at least one order of magnitude faster [60].

System parameters. Following the large-scale deployment in [10], we configure a maximum number of four retransmissions and a retransmission interval of two seconds for CCN-lite. To analyze our approach under different levels of network saturation, we configure the maximum capacities of PIT and CS to range between 5 and 30 elements. Notably, the estimated RAM usage for 30 PIT entries and 30 CS elements with name lengths of ≈ 32 bytes is already approximately 11 KiB, which is around 17% of the total available RAM for our hardware platform.

Caching parameters. We consider the caching procedure to consist of two fundamental steps: (*i*) caching decision, and (*ii*) cache replacement. In our experiments, we use two different caching decision strategies and one cache replacement strategy.

The first decision strategy is to *cache always* incoming Data packets, with the restriction for regular traffic that Data packets without a corresponding PIT entry are dropped and not cached. For *reliable* traffic, we adjust this strategy, such that Data packets without PIT entries are still considered for caching. The second decision strategy is to *cache probabilistically*. Every node caches incoming Data packets with a probability p_{reg} of 30% for regular traffic, and with a probability p_{rel} of 70% for *reliable* traffic. For a saturated CS, our cache replacement strategy evicts content store elements using the *least recently used (LRU)* policy.

4.2. Topology Setup

The testbed provides access to multiple sites with varying numbers of M3 IoT devices and network characteristics. We deploy our applications on the *Grenoble* site, as this supports significantly complex multi-hop paths. We choose 31 M3 devices to host a rich network topology with varying fan-outs, chains, and branch sizes. As such, this single topology represents well a non-trivial IoT edge network and therefore facilitates the reproducibility of our results. In this topology, one device acts as a gateway node and the other 30 devices act as sensors and actuators. Since convergecast is the most predominant traffic pattern in common IoT scenarios, we arrange our devices to form a *Destination Oriented Directed Acyclic Graph (DODAG)* that is rooted at the gateway node. Approximately 60% of the nodes are reachable from the root within 4–5 hops, while the remaining devices have path lengths up to 12 hops. The topology is visualized in Figure 5. Extended left and right wings in the routing topology result from long hallways in the *Grenoble* site.

4.3. Experiment Scenario 1: Mixed Sensors and Actuators

We want to quantify the efficiency of QoS enhanced forwarding in challenged multi-hop deployments that display typical traffic patterns. With this in view, we analyze our approach first in a scenario with mixed traffic of unprioritized sensor readings and prioritized actor commands.

The gateway node requests temperature readings from the 30 sensor nodes every 10 s with $\pm$ 2 s jitter interval. Thus, on average, the request rate at the gateway approximates to 3 *packets/s* and including the reception rate of responses, the gateway handles 6 *packets/s*. The naming scheme for each request consists of a prefix, a device-specific *node id*, and an increasing sequence number. We refer to this traffic equally as *sensor readings* or *sensor traffic*.

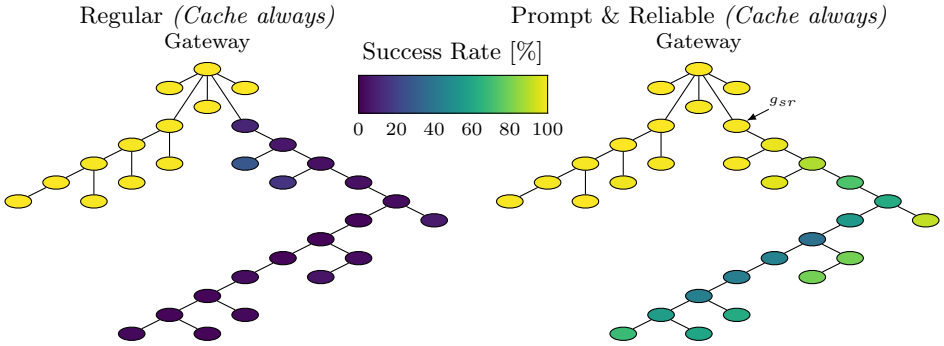

Figure 5: Nodal success rates for *Scenario 1* using regular traffic (left) and *reliable* actuator traffic (right).

In addition, all 30 devices further act as actuators that periodically request a device-specific state from the gateway node every 5 s with ± 1 s jitter. This yields a request reception rate of 6 *packets/s* on the root node. The naming scheme for these requests similarly consists of a prefix, a device-specific *node id*, and an increasing sequence number. We name this traffic *actuator traffic*.

In this scenario, sensor and actuator data are device-specific and thus only scattered destinations benefit from on-path caching during the narrow window of Interest retransmissions.

4.4. Results

We measure and record the *success rates*, *goodputs* and *time to completion* for each node in the topology, while we analyze the network utilization for gateway and actuator traffic separately with and without the proposed QoS features enabled.

Success rates. In our first experiment we focus on the nodal success rates using the first scenario with a PIT and CS limitation of 5 entries. The gateway is configured with a PIT limitation of 50. Figure 5 shows the resulting success rates for (*i*) the regular operation of NDN on the left hand side, and (*ii*) a setup with prioritized actuator traffic using the *prompt* and *reliable* QoS service levels on the right hand side. The success rates per node are color coded and range from 0% (purple) to 100% (yellow).

Figure 5 clearly depicts huge differences in success rates for both configurations. In the normal NDN operation, nodes close to the gateway, as well as the left wing of the topology, achieve 100% success rates. Strikingly, the right wing exhibits major network stress, with nearly all actuators having a success rate below 10% due to PIT overflows. Conversely, with QoS service levels enabled, the right wing shows much enhanced network performance, which results in overall higher success. With this configuration, 70–100% of the packets arrive at actuators close to the gateway, and 40–70% at more distant nodes. Leaf nodes farther away show a greater improvement in success rates than forwarding ancestor nodes.

This striking example nicely illustrates the positive effect of PIT coordination obtain from QoS differentiation. While in the regular case Interests at nodes with exhausted PIT are discarded as they randomly arrive, the QoS marking preselects those requests that are prioritized throughout the network, leading to fewer retransmissions and a more efficient use of the overall PIT space available in the network. A more detailed analysis that compares with enhanced caching effects will be discussed in Section 5 and is visualized in Figure 10.

Network utilization. In typical convergecast settings, a majority of the traffic traverses the gateway to reach remote endpoints. Thus, a robust operation of the gateway node is crucial to ensure adequate performance of the entire IoT network. Due to the increasing sequence numbers used in the naming scheme for *Scenario 1*, virtually no response contributes to future in-network cache hits. To gauge the network load on the gateway, we analyze the number of outgoing requests and incoming responses during a setup in which actuator traffic is added to the traffic from the gateway after approximately eight minutes. We first perform this experiment without any QoS features enabled, and then repeat it with the adjustment that actuator traffic is prioritized using the *reliable* and *prompt* service levels. The PIT of the gateway is configured to a maximum of 50 entries, while the remaining nodes have a PIT maximum of 5 entries. The CS is limited to 5 entries for all nodes.

We observe in Figure 6 that the sensor traffic exhibits a steady request-response flow of about 180 *packets/min* for both requests and responses. As soon as the actuators initiate their periodic requests (at minute eight), the network load at the gateway increases. The number of requests spikes threefold due to network layer retransmissions, while the number of returning responses drops to half. In contrast, the setup with prioritized actuator traffic clearly admits a reduced number of requests at the gateway while achieving a higher response rate.

These results reveal that prioritizing the actuator traffic has a positive effect on the overall network load due to reduced retransmissions.

Goodputs. Figure 7 shows that while PIT sizes have a significant impact on the goodput at the gateway and at each

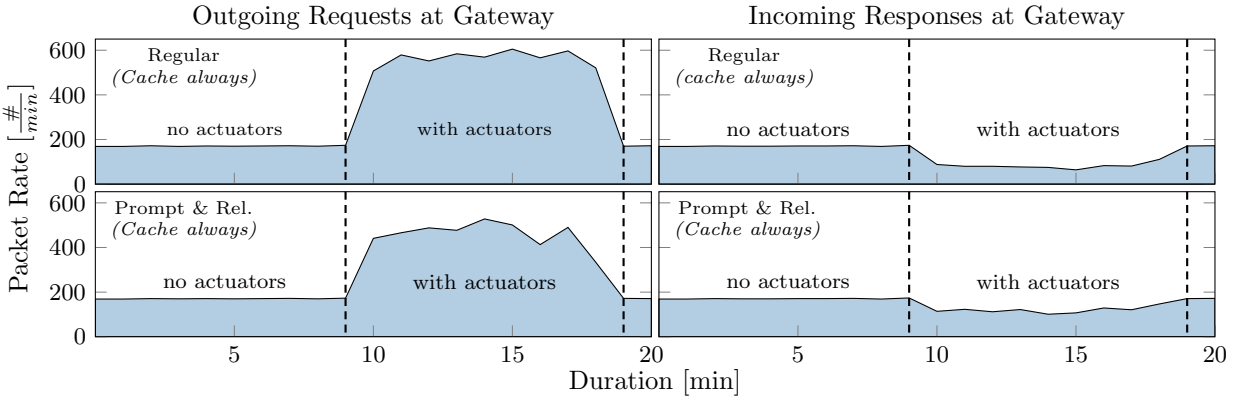

Figure 6: Packet transmission rate per minute for requests and responses with and without (prioritized) cross traffic measured at the gateway.

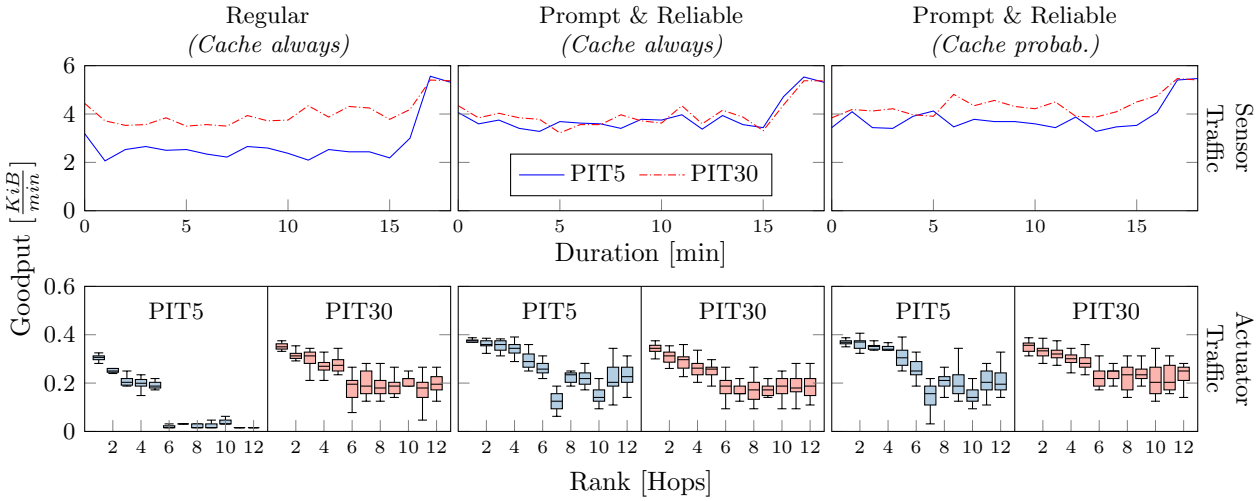

Figure 7: Goodput evolution for *Scenario 1* with actuator and gateway traffic using a CS size of 5.

actuator during normal operation, this effect is reduced when QoS service classes are introduced. When using service classes and the *always* caching decision strategy, network members with small PIT sizes can reach a level of goodput that is comparable to those of the largest PIT size during normal operation. With *probabilistic* caching in operation, the network performance increases even further for larger PITs, which support a higher number of concurrent flows that can leverage cache diversity better. This overall enhanced transport performance is caused by flows that complete with delay based on segment-wise available network capacities and retransmissions. The corresponding temporal effects can be observed from Figure 8.

At actuator nodes far from the gateway, our QoS mechanisms mitigate the effects of PIT exhaustion, which in normal operation leads to an abrupt collapse of the throughput at around a rank of 6. Our QoS mechanisms cause a smooth, gradual decline in performance instead.

Time to completion. We can see from Figure 8 that content arrival times are significantly reduced for smaller PIT sizes when QoS service classes are introduced—for traffic at gateways and even more at the prioritized actuators. Simultaneously, we see again a signature of enhanced traffic delivery for QoS-coordinated flows that shows doubled success rates from 40% to 80%. It is worthwhile to (re-)observe that the sensor traffic also experiences improvements due to a more efficient balancing of resources—small PIT sizes in particular.

We now quantify the content arrival times specifically for each quality dimension. Figure 9 illustrates the nodal content arrival times with PIT and CS sizes both set to 5. In all three displayed configurations, completion time increases with the distance to the gateway. Content arrival times range from 5 ms to 350 ms for the majority of *regular* requests, and the *reliable* configuration does not change this. In contrast, the *prompt* configuration yields noticeably faster delivery times for all nodes, in particular nodes far away from the gateway experience a reduction of about 100 ms ($\approx 30\%$). This shows that the very lightweight priority queuing based on a simple double-buffer already shows an important impact throughout the constrained network.

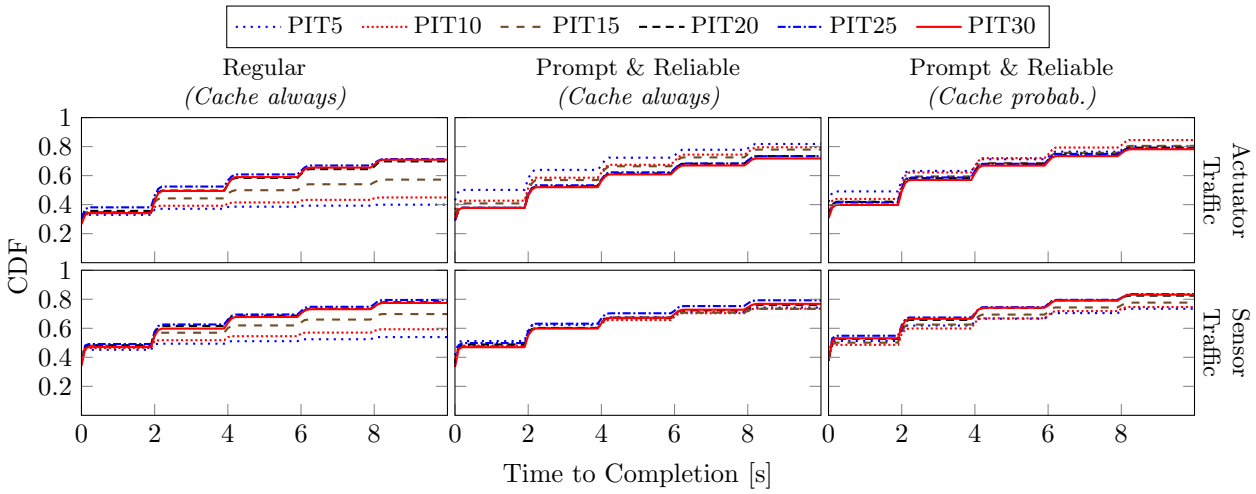

Figure 8: Time to completion for *Scenario 1* with actuator and gateway traffic using a CS size of 5.

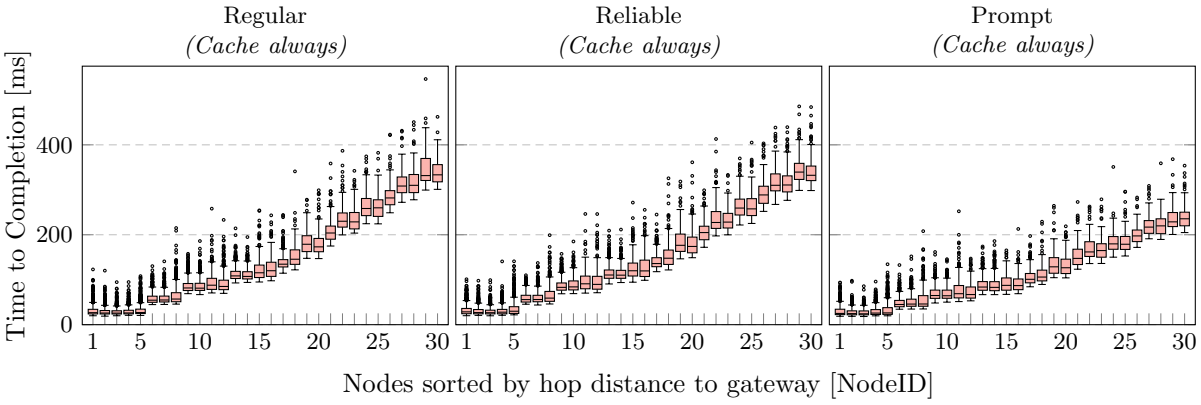

Figure 9: Time to completion per actuator and quality dimension in *Scenario 1* using PIT and CS sizes of 5.

5. The Impact of Caching

5.1. Experiment Scenario 2: Sensing and Lighting Control

In the second scenario, we change the role of actuators but leave the sensor readings unchanged. Instead of independent actuators that receive disjoint instructions, we envision a scenario of lighting in which groups of fixtures switch lights in a coordinated way. Hence, these groups receive identical commands and caching becomes applicable.

To explore the event space by mixing group memberships, our experiments proceed as follows. For each request, an actuator randomly joins one out of five possible 'lighting' groups. The naming scheme for such requests is changed to include the selected *group id*, instead of a device-specific *node id*. Besides naming, we use the same request parameters for the actuator traffic as in the first scenario. We repeat this process 240 times for each configuration in order to explore the state space of unevenly distributed groups and converge statistics.

In this scenario, prioritized actuator traffic flows from the gateway to multiple destinations and thus benefits from on-path caching. Accordingly, we expect the network performance to improve over that for Scenario 1.

5.2. Results

Success rates. The overall success results presented in Figure 10 confirm these expectations. With this figure we dig deeper into network reliability and examine the success rates for a range of PIT sizes (scenario 1) and Content Store sizes (Scenario 2) plotted against the node ranks.

Success in content delivery for the second scenario nicely approaches 100% in most QoS settings, while in contrast Scenario 1 experiences significant loss rates above 50% at the edges for all sizes of pending Interest tables. With caching even for the most constrained number of 5 PIT entries, an increase from 5 to 30 of the CS sizes suffices to turn traffic from QoS class *reliable* into a fully reliable service. While we do see some failures at higher ranks for the small CS size of 5 similar to *Scenario 1*, increasing the CS capacity to 10 is already sufficient to attain success rates above 80%. The most striking contrast is found in comparison to the results of regular NDN operation, where even with a maximum CS size of 30 the failure rate stays

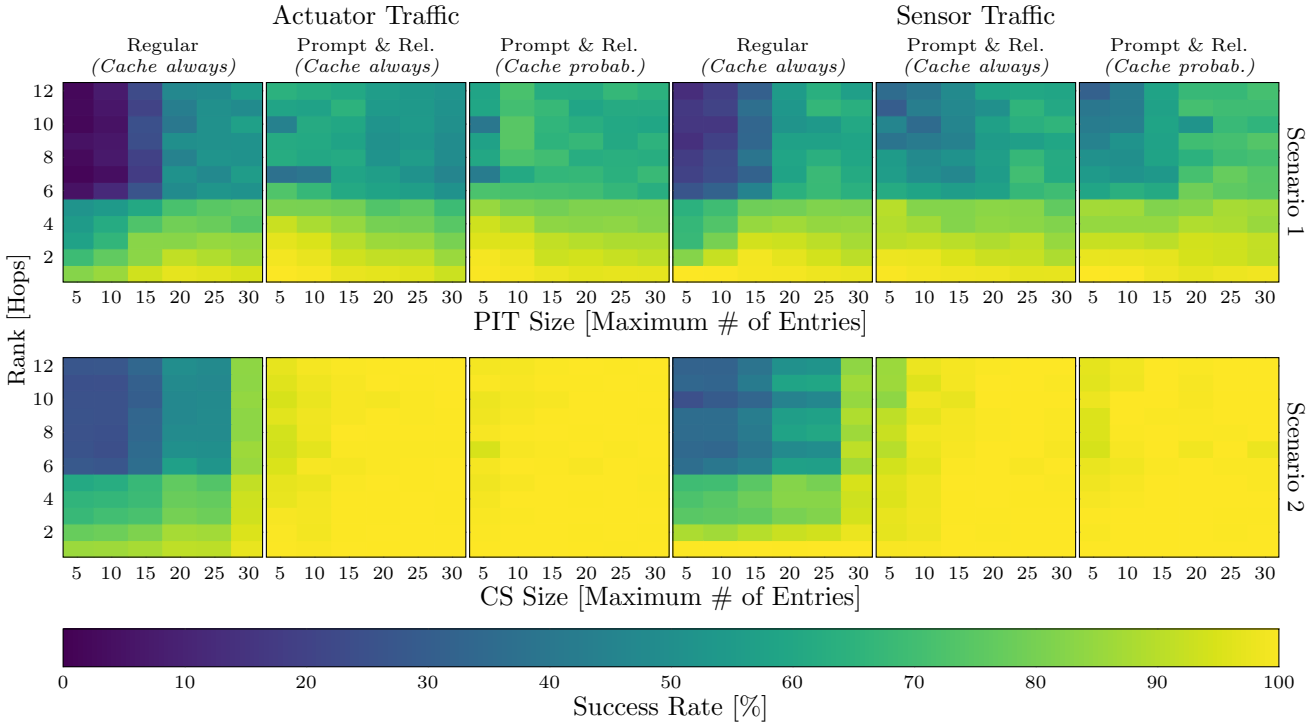

Figure 10: Success rates per rank for *Scenario 1* and *Scenario 2* using varying PIT and CS sizes.

at 30–40% at higher ranks. Regular NDN traffic apparently profits less from cacheable content. This is mainly due to PIT decorrelation, which breaks content flows.

As previously observed, the success rate for *regular* actuator traffic collapses at higher ranks. The sensor traffic performs similarly poorly. Increasing the maximum PIT size gradually improves the nodal success rates. In addition to the *regular* traffic, we further show actuator traffic with *prompt & reliable* QoS service levels. Overall, the setups with prioritized traffic show great improvements for the smaller PIT sizes, while the *probabilistic* caching decision strategy performs slightly better than the *always* strategy. A surprising effect is observed with a PIT size of 30: the success rates for all configurations decline slightly for lower ranks. This is caused by an increased retransmission overhead per node, resulting in link saturation.

Time to completion. QoS service classes have a significant impact on the content arrival times for both the actuators and the sensors, as was already observed for *Scenario 1*. Figure 11 reflects the same qualitative picture for *Scenario 2*, but at significantly reduced probabilities of retransmission. The latter is due to actuator traffic that is coordinated in QoS classes and coherently serviced from caches—a large reduction in overall network load. Results slightly improve for enhanced cache diversity in probabilistic caching, with 90% of the packets arriving promptly (< 100 ms) at cache sizes of at least 10 packets. Similar to *Scenario 1*, CS sizes become less relevant in the presence of QoS marking, since prioritized traffic arrives quickly at its destination and remains unaffected by regular cache replacement.

While we observe that increased CS sizes contribute to reduced completion times and improved success rates, we also notice that even a CS size of 30 does not suffice for the *regular* configuration. On the other hand, the configurations that use QoS service levels display close to 100% success rate, even with severely limited CS sizes, and about 70% of all requests for each traffic type complete in less than 100 ms.

Cache hits. Analysing the cache efficiencies supports these observations. Figure 12 displays the relative in-network cache hits for actuator traffic in setups with varying CS sizes. While the regular NDN operation yields a marginally improving cache hit ratio for increasing CS sizes, both QoS enabled setups exhibit a noticeable enhancement. This improved cache efficacy is caused by the privileged cache resource utilization for data of the *reliable* actuator traffic, whereas data of the sensor traffic is more likely to be evicted. Another expected observation is that *probabilistic* caching further improves the cache hit ratio thanks to its increased CS diversity.

6. Preventing Starvation of the Commons

6.1. The Problem of Resource Starvation

Given that content in the CS is only replaced if new content arrives and that unprioritized content may not replace prioritized content according to the rules laid out in

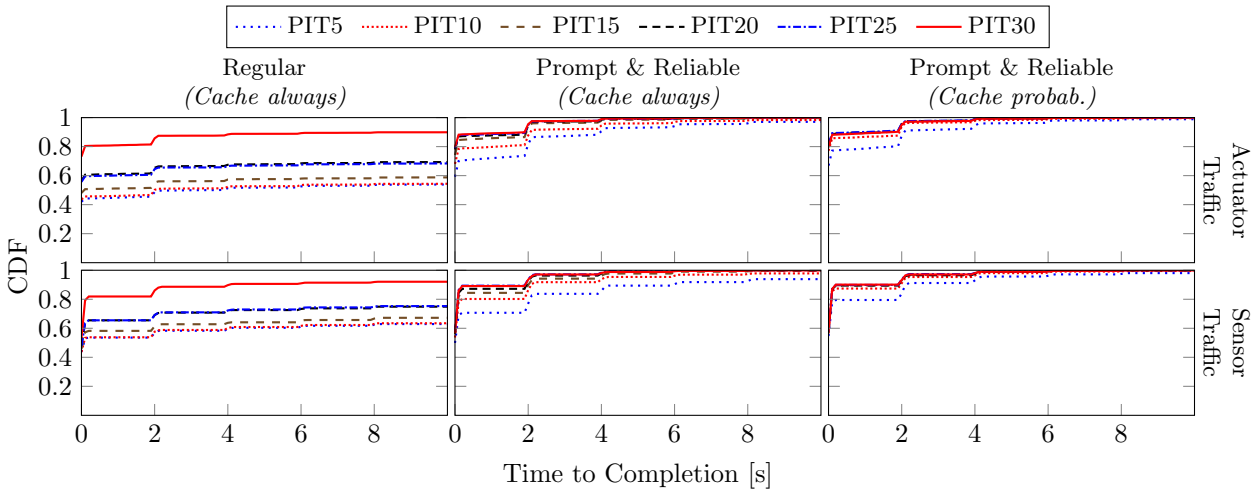

Figure 11: Time to completion for *Scenario 2* with actuator and gateway traffic using a PIT size of 5.

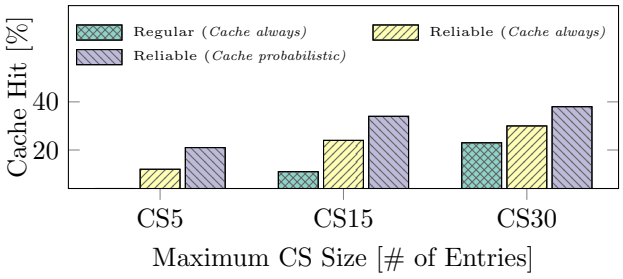

Figure 12: Cache hits for actuator traffic in *Scenario 2* with a PIT size of 5.

Section 3.2, it is easy to construct a scenario in which all CS space is occupied by prioritized content, thus starving the unprioritized. An initially distributed set of prioritized data chunks, for example, could block caches for an unlimited time if followed only by content of lower priority. Moreover, if content request rates reach a frequency at which retransmissions become necessary in order to fulfil the requests, the fact that unprioritized content cannot rely on the CS as a retransmit buffer means that unprioritized flows may no longer be delivered at all.

One might expect the PIT to be affected by starvation in a similar manner, with *prompt* entries crowding out *regular* entries; in practice, however, this does not pose a problem, since as we have shown in Sections 4.4 and 5.2, the improved PIT coordination gained through QoS coordination leads to a more efficient utilization of the PIT space on the whole, while at the same time PIT entries are erased quickly or time out.

6.2. Countermeasures and Experimental Evaluation

We can prevent starvation of CS utility for unprioritized content by introducing fairness measures that prevent prioritized content from blocking the CS and can guarantee that some non-zero amount of unprioritized content can always be cached.

Countermeasure for Cache Blocking. The first countermeasure is the introduction of a priority decay time τ. Any prioritized content that has been in the CS for a time equal to or larger than τ is reclassified as unprioritized and may thus be replaced by unprioritized content. In practice, τ may be chosen as an average time of cache utility. The firing of this timer will prevent prioritized content from blocking the cache for longer than useful in a mixed environment.

Countermeasure for Cache Squeeze Out. The previous measure alone, however, may not be sufficient if the rate of incoming prioritized content is high, as the replacement rules defined in Section 3.2 state that unprioritized content is always replaced before prioritized content, meaning that the actual time the unprioritized content is allowed to stay in the CS may be too short to be useful.

There are many ways to counter this threat of squeezing unprioritized content completely out of the caches, such as preallocated cache resources and modified timers. Most of these countermeasures depend on parameters specific to the traffic patterns and are therefore difficult to deploy in a general way. We argue for a simple, generic solution as given by probabilistic caching, which as we recall from Section 5 also enhances cache diversity. We will now quantify the effects of probabilistic caching in a starvation scenario.

Experiment Configuration. We want to measure resource utilizations of the PIT and CS data structures in setups with increased levels of network stress that are prone to starvation. As before, we limit the PIT and CS resources on every node in the topology to hold a maximum of 5 elements. We consider Scenario 2 as described above, but reverse the priority roles: Sensor traffic is now prioritized, while the cache-dependent actuator traffic remains regular. In this setting, we thus emphasize the effects of cache starvation for unprioritized data traffic.

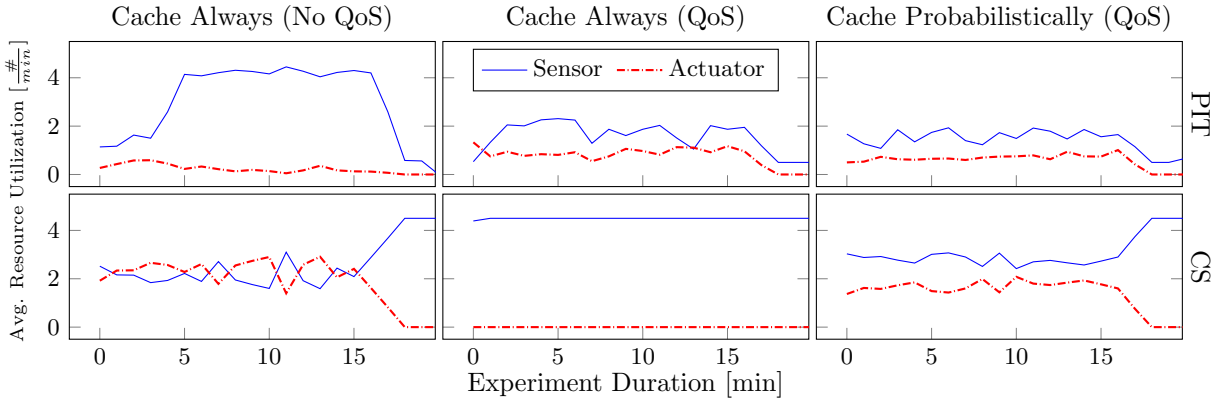

Figure 13: Evolution of resource utilizations for a successor node of the gateway and an actuator request interval of 5±1 s.

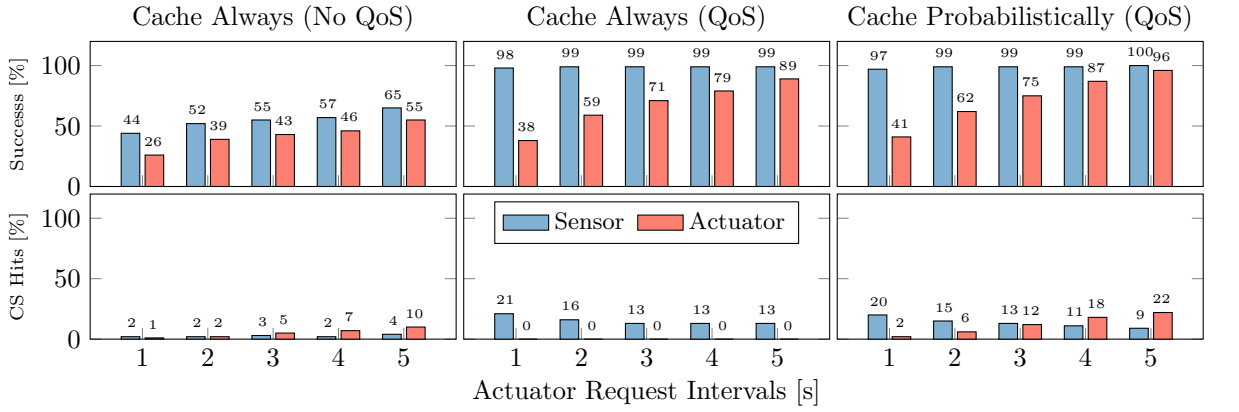

Figure 14: Average success rates and cache hits for different levels of actuator request intervals.

Resource Utilization. In this experiment, we measure the resource utilization over time in order to identify starved traffic flows in environments with limited PIT and CS resources and increasing network loads. Figure 13 depicts the evolution of resources from the point of view of node g_{sr}, which is a successor to the gateway and resides on the right wing of Figure 5. g_{sr} is root of a subtree with 18 descendants and thus will experience extensive traffic in both directions. Actuators emit unprioritized request-response flows in groups and the gateway transmits prioritized sensor readings.

We first observe the distribution of sensor- and actuator-bound PIT entries. Without prioritization, the PIT is mostly saturated with gateway traffic and repeatedly denies actuator requests. This results from the close proximity of g_{sr} to the gateway node and the concomitant fast placement of gateway requests into the PIT. Consistent with our observations in the previous sections, PIT exhaustion relaxes with QoS enabled and resource occupancies almost equal out between the traffic types due to better PIT coordination and faster consumption of pending Interests. This improves slightly further with probabilistic caching in place, since effectiveness increases and more content can be served from caches. In both cases with QoS prioritization enabled, no signs of starvation are visible at the PIT level.

In contrast, the occupation of the Content Store clearly shows how unprioritized content gets blocked from caching for the priority-guided cache always policy. QoS features hence lead to a straight starvation of the CS resource for unprioritized actor traffic. Probabilistic caching—used with the previously established probabilities 0.7 for prioritized and 0.3 for regular traffic—seamlessly resolves this starvation and enables about 30% cache placement for regular packets on average.

Success of Network Operations. Figure 14 displays the corresponding success rates for packet deliveries and cache hits for a series of actuator request intervals. Network degradation and cache underutilization are clearly visible in the absence of QoS as a result of uncoordinated network resources. Activating QoS with the cache-always policy significantly improves the packet delivery success for both traffic classes (as observed before), but prevents cache hits due to the cache starvation observed above.

In contrast, probabilistic caching maintains the utility of the caches for the unprioritized actuator traffic. Moreover, it improves the success rates of actuator traffic without sacrificing performance of privileged communication.

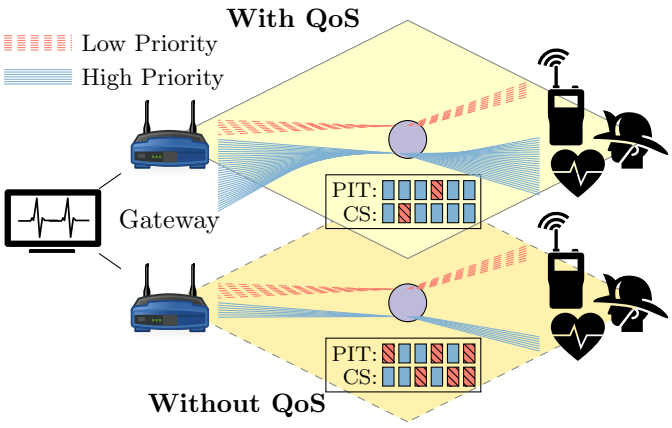

Figure 15: Resource allocation with and without QoS mechanisms in a disaster scenario.

Figure 16: Heartbeat signals with and without QoS mechanisms.

7. Showcase: QoS Impact in Disaster Scenarios

We now demonstrate how differentiated ICN resource management can serve the needs of challenged deployments such as constrained IoT edge networks in disaster scenarios. Using realistic implementations on RIOT, we demonstrate how very constrained devices in harsh environments can reliably communicate, provided QoS measures are in place. These devices gradually invoke traffic flows of different priority levels. In this setup, we contrast regular bulk traffic admitting degradation in flow latency and reliability with QoS-enhanced traffic differentiation and visualize the improved flow resource consumption of high priority traffic on all nodes.

Challenges in disaster scenarios. In unforeseen disasters, a quick response is imperative to prevent serious harm to people or the environment and to minimize collateral damage. With this objective, first responders are sent into the field to (i) assess the situation on-site, (ii) immediately administer first aid, and (iii) call for appropriate further emergency support.

For an efficient operation, search and rescue forces require operational guidance by command and control centers. Since regular communication infrastructures are often not available, first aiders are forced to deploy spontaneous radio networks to enable data flows of different importance under intermittent connectivity.

Typically, hand-held communication devices attach to field units and report back readings from sensors that are deployed in the equipment or the immediate vicinity. Sensor devices are battery-operated with a small form factor in order not to interfere with the mobility and maneuverability of rescuers. NDN deployments have shown great potential to successfully operate in disaster scenarios [61] by inherently providing in-network caches and intrinsic multicast capability as well as support of consumer mobility.

Networked sensor devices form a spontaneous low power lossy network (LLN), where device limitations can significantly impact the overall network performance. To mitigate performance degradations for continuous and life-critical traffic flows such as ECG heartbeat monitoring, the network must be able to differentiate between traffic flows, become aware of flow-specific priorities, and balance available resources according to these priorities.

The rescue case. Figure 15 illustrates our general setup for the two configurations, with and without QoS features enabled. Our gateway continuously requests heartbeat sensor readings and displays them on a dashboard. In addition, our communication device polls new audio messages every 10 seconds to receive up to date fireground assistance. Once an audio message exists, the message is bulk requested, thereby leading to resource saturation on the forwarder and a disruption of continuous heartbeat readings on the gateway. We repeat this setup with and without QoS features enabled and assign a high prioritization to the heartbeat flow. On the dashboard, we illustrate an uninterrupted heartbeat signal to indicate the low latency and high reliability characteristics of our prioritized traffic flow, while slightly distorted audio messages are still being dispatched to our communication device on the fire ground. Figure 16 pictures the results: While heartbeats without QoS support barely arrive at the gateway, we see strong signals after the QoS features are enabled.

8. Conclusions and Future Research

We presented and analyzed QoS extensions to NDN that are suitable for constrained devices. Starting from a name-oriented flow classification scheme, we introduced the two service dimensions *prompt* and *reliable* network forwarding. Strategies were defined that not only foster local, isolated resource allocations, but take into account coordinative actions between different internal resources of a node, as well as correlations between nodes. Here, we exploited the rich set of forwarding and caching options that NDN includes, while protecting resources from starvation by applying fairness measures.

We were able to validate our approach in real-world experiments on a large testbed using a realistic multi-hop wireless setup and in a realistic use case implementation for the rescue domain. Moreover, we learned that QoS

management in NDN is not confined to simple resource trading, but can lead to a global enhancement of network performance by optimizing the interplay between various resource consumptions. In particular, we found evidence that (*i*) coordination of PIT and CS has a prevailing effect on the overall performance of the networked system, and (*ii*) incorporation of Interests in QoS treatment is vital to cater for resource coordination.

Future research shall explore the effects of these coordinative resource actions in an Internet backbone and study the impact against resource decorrelation when large multiplicities of competing flows are crossing in a densely meshed core network.

Acknowledgements

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] A. Markkanen, D. Shey, Edge Analytics in IoT, Tech. rep., ABI Research (April 8 2015).

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

[3] 3GPP, Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2, Tech. Rep. TS 36.300 13.4.0, 3GPP (2016).

[4] 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`

[5] G. C. Polyzos, N. Fotiou, Building a reliable Internet of Things using Information-Centric Networking, Journal of Reliable Intelligent Environments 1 (1) (2015) 47–58.

[6] M. Amadeo, C. Campolo, A. Iera, A. Molinaro, Information Centric Networking in IoT scenarios: The case of a smart home, in: Proc. of IEEE International Conference on Communications (ICC), IEEE, Piscataway, NJ, USA, 2015, pp. 648–653.

[7] 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.

[8] O. Ascigil, S. Reñé, G. Xylomenos, I. Psaras, G. Pavlou, A Keyword-based ICN-IoT Platform, in: Proc. of 4th ACM Conference on Information-Centric Networking (ICN), ACM, New York, NY, USA, 2017, pp. 22–28.

[9] 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.

[10] 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.
URL `https://doi.org/10.1145/3267955.3267967`

[11] A. Chakraborti, S. O. Amin, A. Azgin, S. Misra, R. Ravindran, Using ICN Slicing Framework to Build an IoT Edge Network, in: Proceedings of the 5th ACM Conference on Information-Centric Networking, ICN '18, ACM, New York, NY, USA, 2018, pp. 214–215.

[12] R. Braden, D. Clark, S. Shenker, Integrated Services in the Internet Architecture: an Overview, RFC 1633, IETF (June 1994).

[13] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, RFC 2205, IETF (September 1997).

[14] K. Nichols, S. Blake, F. Baker, D. Black, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC 2474, IETF (December 1998).

[15] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An Architecture for Differentiated Services, RFC 2475, IETF (December 1998).

[16] M. Welzl, Network Congestion Control - Managing Internet Traffic, Wiley, Hoboken, NJ,USA, 2005.

[17] IEEE 802.1 Working Group, IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks, Tech. Rep. Std 802.1Q-2018, IEEE, (Revision of IEEE Std 802.1Q-2014) (July 2018).

[18] P. Meyer, T. Steinbach, F. Korf, T. C. Schmidt, Extending IEEE 802.1 AVB with Time-triggered Scheduling: A Simulation Study of the Coexistence of Synchronous and Asynchronous Traffic, in: 2013 IEEE Vehicular Networking Conference (VNC), IEEE Press, Piscataway, New Jersey, 2013, pp. 47–54.

[19] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, A Survey of Information-Centric Networking, IEEE Communications Magazine 50 (7) (2012) 26–36.

[20] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos, A Survey of Information-Centric Networking Research, IEEE Communications Surveys and Tutorials 16 (2) (2014) 1024–1049.

[21] 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.

[22] 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.

[23] 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 of Things Journal 5 (6) (2018) 4428–4440.
URL `http://dx.doi.org/10.1109/JIOT.2018.2815038`

[24] C. Gündogan, J. Pfender, M. Frey, T. C. Schmidt, F. Shzu-Juraschek, M. Wählisch, Gain More for Less: The Surprising Benefits of QoS Management in Constrained NDN Networks, in: Proc. of 6th ACM Conference on Information-Centric Networking (ICN), ACM, New York, 2019, pp. 141–152.
URL `https://doi.org/10.1145/3357150.3357404`

[25] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, M. Waehlisch, Information-Centric Networking (ICN) Research Challenges, RFC 7927, IETF (July 2016).

[26] M. Wählisch, T. C. Schmidt, M. Vahlenkamp, Backscatter from the Data Plane – Threats to Stability and Security in Information-Centric Network Infrastructure, Computer Networks 57 (16) (2013) 3192–3206.

URL http://dx.doi.org/10.1016/j.comnet.2013.07.009
[27] D. Oran, Considerations in the development of a QoS Architecture for CCNx-like ICN protocols, Internet-Draft – work in progress 03, IETF (December 2019).
[28] D. Oran, Maintaining CCNx or NDN flow balance with highly variable data object sizes, Internet-Draft – work in progress 02, IETF (February 2020).
[29] I. Moiseenko, D. Oran, Flow Classification in Information Centric Networking, Internet-Draft – work in progress 05, IETF (January 2020).
[30] A. Jangam, P. suthar, M. Stolic, QoS Treatments in ICN using Disaggregated Name Components, Internet-Draft – work in progress 01, IETF (September 2019).
[31] C. Gündogan, T. C. Schmidt, M. Wählisch, M. Frey, F. Shzu-Juraschek, J. Pfender, Quality of Service for ICN in the IoT, IRTF Internet Draft – work in progress 01, IRTF (July 2019).
URL http://tools.ietf.org/html/draft-gundogan-icnrg-iotqos
[32] C. Tsilopoulos, G. Xylomenos, Supporting Diverse Traffic Types in Information Centric Networks, in: Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking, ICN '11, ACM, New York, NY, USA, 2011, pp. 13–18.
[33] M. Mahdian, S. Arianfar, J. Gibson, D. Oran, MIRCC: Multipath-aware ICN Rate-based Congestion Control, in: Proceedings of the 3rd ACM Conference on Information-Centric Networking, ACM-ICN '16, ACM, New York, NY, USA, 2016, pp. 1–10.
[34] N. Dukkipati, Rate control protocol (rcp): Congestion control to make flows complete quickly, Ph.D. thesis, Stanford, CA, USA (2008).
[35] M. F. Al-Naday, A. Bontozoglou, V. G. Vassilakis, M. J. Reed, Quality of Service in an Information-Centric Network, in: Proceedings of Globecom 2014, 2014, pp. 1861–1866.
[36] M. Zhang, H. Luo, H. Zhang, A Survey of Caching Mechanisms in Information-Centric Networking, IEEE Communications Surveys & Tutorials 17 (3) (2015) 1473–1499.
[37] M. A. Hail, M. Amadeo, A. Molinaro, S. Fischer, Caching in Named Data Networking for the Wireless Internet of Things, in: International Conference on Recent Advances in Internet of Things (RIoT), IEEE, 2015, pp. 1–6.
[38] I. Psaras, W. K. Chai, G. Pavlou, Probabilistic In-Network Caching for Information-Centric Networks, in: Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking, Helsinki, Finland, 2012, pp. 55–60.
[39] S. Tarnoi, K. Suksomboon, W. Kumwilaisak, Y. Ji, Performance of Probabilistic Caching and Cache Replacement Policies for Content-Centric Networks, in: Proceedings of the IEEE 39th Conference on Local Computer Networks (LCN), Edmonton, AB, Canada, 2014, pp. 99–106.
[40] M. A. M. Hail, M. Amadeo, A. Molinaro, S. Fischer, On the Performance of Caching and Forwarding in Information-Centric Networking for the IoT, in: Proceedings of the International Conference on Wired/Wireless Internet Communication (WWIC), Springer, Malaga, Spain, 2015, pp. 313–326.
[41] S. Arshad, M. A. Azam, M. H. Rehmani, J. Loo, Information-Centric Networking Based Caching and Naming Schemes for Internet of Things: A Survey and Future Research Directions, arXiv preprint arXiv:1710.03473.
[42] J. Pfender, A. Valera, W. K. G. Seah, Performance Comparison of Caching Strategies for Information-Centric IoT, in: Proceedings of the 5th ACM Conference on Information-Centric Networking, ICN '18, ACM, New York, NY, USA, 2018, pp. 43–53.
[43] D. Doan Van, Q. Ai, An Efficient in-Network Caching Decision Algorithm for Internet of Things, International Journal of Communication Systems 31 (8) (2018) e3521.
[44] B. Chen, L. Liu, Z. Zhang, W. Yang, H. Ma, BRR-CVR: A Collaborative Caching Strategy for Information-Centric Wireless Sensor Networks, in: 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), IEEE, 2016, pp. 31–38.
[45] L. Zhang, J. Zhao, Z. Shi, LF: A Caching Strategy for Named Data Mobile Ad Hoc Networks, in: Proceedings of the 4th International Conference on Computer Engineering and Networks, Springer, 2015, pp. 279–290.
[46] W. K. Chai, D. He, I. Psaras, G. Pavlou, Cache "Less for More" in Information-Centric Networks (Extended Version), Computer Communications 36 (7) (2013) 758–770.
[47] S. Naz, R. N. B. Rais, P. A. Shah, S. Yasmin, A. Qayyum, S. Rho, Y. Nam, A Dynamic Caching Strategy for CCN-Based MANETs, Computer Networks.
[48] L. Zhou, T. Zhang, X. Xu, Z. Zeng, Y. Liu, Broadcasting Based Neighborhood Cooperative Caching for Content Centric Ad Hoc Networks, in: IEEE/CIC International Conference on Communications in China (ICCC), IEEE, 2015, pp. 1–5.
[49] Y. Sun, T. Zhang, R. Wang, W. Feng, P. Chen, Topology Potential Based Probability Caching Strategy for Content Centric Ad Hoc Networks, Journal of Residuals Science & Technology 13 (6).
[50] S. Ioannidis, E. Yeh, Jointly Optimal Routing and Caching for Arbitrary Network Topologies, in: 4th ACM Conference on Information-Centric Networking, ACM-ICN '17, ACM, New York, NY, USA, 2017, pp. 77–87.
[51] J. Pfender, A. Valera, W. K. G. Seah, Easy as ABC: A Lightweight Centrality-Based Caching Strategy for Information-Centric IoT, in: Proceedings of the 6th ACM Conference on Information-Centric Networking, ICN '19, ACM, New York, NY, USA, 2019, pp. 100–111.
[52] Y. Liu, D. Zhu, W. Ma, A Novel Cooperative Caching Scheme for Content Centric Mobile Ad Hoc Networks, in: IEEE Symposium on Computers and Communication (ISCC), IEEE, 2016, pp. 824–829.
[53] Z. Li, G. Simon, Time-Shifted TV in Content Centric Networks: The Case for Cooperative In-Network Caching, in: Proceedings of the IEEE International Conference on Communications (ICC), Kyoto, Japan, 2011, pp. 1–6.
[54] Y. Zeng, X. Hong, A Caching Strategy in Mobile Ad Hoc Named Data Network, in: 6th International ICST Conference on Communications and Networking in China (CHINACOM), IEEE, 2011, pp. 805–809.
[55] O. Hahm, E. Baccelli, T. C. Schmidt, M. Wählisch, C. Adjih, L. Massoulié, Low-power Internet of Things with NDN and Cooperative Caching, in: Proc. of 4th ACM Conference on Information-Centric Networking (ICN), ACM, New York, NY, USA, 2017, pp. 98–108.
[56] C. Bormann, M. Ersue, A. Keranen, Terminology for Constrained-Node Networks, RFC 7228, IETF (May 2014).
[57] Atmel, Low Power 2.4 GHz Transceiver for ZigBee, IEEE 802.15.4, 6LoWPAN, RF4CE, SP100, WirelessHART, and ISM Applications, Atmel Corporation (September 2009).
URL http://www.atmel.com/images/doc8111.pdf
[58] 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.
[59] C. Tschudin, C. Scherb, et al., CCN Lite: Lightweight implementation of the Content Centric Networking protocol (2018).
URL http://ccn-lite.net
[60] M. Lenders, P. Kietzmann, O. Hahm, H. Petersen, C. Gündogan, E. Baccelli, K. Schleiser, T. C. Schmidt, M. Wählisch, 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 (January 2018).
URL https://arxiv.org/abs/1801.02833
[61] J. Seedorf, M. Arumaithurai, A. Tagami, K. Ramakrishnan, N. Blefari-Melazzi, Research Directions for Using ICN in Disaster Scenarios, Internet-Draft – work in progress 07, IETF (June 2019).