Towards Domain-Specific Time-Sensitive Information-Centric Networking Architecture

Marcin Bosk*, Jörg Ott†
TUM School of Computation, Information, and Technology, Technical University of Munich, Germany
Email: *bosk@in.tum.de, †ott@in.tum.de

***Abstract*—Due to increasing bandwidth requirements, Ethernet is replacing traditional technologies such as CAN bus within the industrial automation and automotive networks. To support the deterministic communication required in these systems, it is often combined with the IEEE 802.1Q Time-Sensitive Networking (TSN) group of standards. Information-Centric Networking (ICN) is an appealing architecture for aiding the migration to Ethernet. It can map the content-based message identifiers of the CAN bus and support complex network topologies in those new systems. This work investigates ICN's interaction with TSN and how their combination can support real-time systems. By extending the EnGINE framework with Named-Data Networking (NDN) support, we provide a hardware-based experimental environment tailored to develop ICN-based TSN systems. We further propose an NDN-based architecture tailored for networked systems requiring low latency deterministic communication. We show that NDN on top of TSN is a promising approach on average fulfilling the strict delay requirements of 2 ms and jitter of 125 μs in a seven-hop automotive network, with our approach further reducing the network overhead.**

***Index Terms*—TSN, IVN, IACS, ICN, Network Architecture**

I. Introduction

Time-sensitive and real-time systems require networks supporting low latency and reliable communication. As legacy technologies, e.g., Controller Area Network (CAN) bus, cannot meet the increasing bandwidth demands of modern systems, Ethernet is replacing previously used networks [1]. It is usually combined with the Time Sensitive Networking (TSN) standards [2] that enable deterministic communication, providing support for low latency, low jitter, and lossless packet exchange. In contrast to Ethernet, many legacy systems use signals with message *identifiers* defining the semantics (and priority) of a frame, to which all interested nodes "subscribe", effectively creating a publish-subscribe (pub/sub) system [3]. This introduces challenges for integrating Ethernet in low latency systems and their migration from legacy networks, leading researchers to explore mechanisms above Layer 2 [4].

While building complete connectivity for real-time systems based on Ethernet is feasible with the industry already adopting it, focusing on a single Layer 2 technology may be limiting. The capability to smoothly integrate different Layer 2 technologies across larger systems is more easily provided at Layer 3, as the Internet Protocol (IP) has proven over decades. IP, with its host-based addressing scheme, may not be a perfect fit for the communication based on named message *identifiers* we find in industrial and automotive environments. A similar argument can be made for security in such systems, generally ensured with separate Layer 2 or above mechanisms, e.g., MACsec, IPSec, or TLS [5]. Their integration with TSN is challenging and often requires specialized hardware and careful consideration of traffic shaping configurations. With its inherently content-based addressing and native security support, the Information-Centric Networking (ICN) concept presents a compelling alternative to IP-based solutions [6].

ICN is a Layer 3 technology tailored for request-response style communication in large-scale networks. It has valuable properties for networks built upon TSN, which need interaction with legacy bus systems. ICN supports naming similar to, e.g, Vehicle Signal Specification (VSS)[1], a scheme used in Intra-Vehicular Networks (IVNs), and does not require abstractions to end-host addresses. It further supports data integrity with cryptographic signatures of transmitted data [6]. Finally, ICN includes packet replication methods at network nodes, beneficial specifically for time-shifted multicast [7].

To address the challenges of real-time systems, in this work, we outline the requirements and initial design considerations for an ICN-based architecture tailored for TSN built upon Named-Data Networking (NDN). We propose several protocol extensions, among others, based on the concept akin to Persistent Interests (PI) for real-time communication [8]. To enable experimentation, we build upon our open-source[2] EnGINE [9] framework and extend it with NDN support. Using this setup, we verify the applicability of NDN for real-time systems in a simple network combined with TSN traffic shaping. We further perform an initial verification of the architectural concepts, comparing its low latency and low jitter IVN communication requirements fulfillment against an IP-based deployment.

II. Background and Related Work

In the following, we introduce the relevant technologies and summarize related work investigating the applicability of ICN for low latency and real-time communication, including TSN.

A. Information-Centric Networking

ICN [10] is an alternative networking concept to the Layer 3 IP protocol. It shifts the focus from host addressing to exchanged information naming. Many ICN realizations are request-response based, where consumers request named

All links are valid as of 27 April 2024.
ISBN 978-3-903176-63-8 © 2024 IFIP

[1]https://covesa.github.io/vehicle_signal_specification/
[2]https://github.com/rezabfil-sec/engine-framework

information from the producers. The network devices contain routing information, data services, and forwarding logic, facilitating the exchange and enabling the network to fulfill consumer requests directly. ICN emphasizes security, with the network further supporting data caching on network nodes.

While several ICN implementations exist, in this work, we focus on NDN [11] and its C++ implementation due to its accessibility and ability to run on hardware-constrained devices. It is also well documented, has numerous supporting tools, and is actively maintained. NDN provides receiver-driven, data-oriented communication architecture that can run as a Layer 2 or 3 protocol. The communication parties exchange two types of packets. The consumer requests information using INTERESTS, with the producer responding using DATA packets. Each type includes a data name, with the INTEREST packet having additional information, e.g., desired age limit. The DATA packet includes the content itself, a digital signature for data integrity, and metadata such as its validity period.

These packets are forwarded using forwarder modules placed on each NDN node. Communication is enabled using faces, which provide an abstraction for the use of underlying network protocols. A content store contains data that is available at a given node. The Pending Interest Table (PIT) includes information on all received INTERESTS, such as name and incoming face. A routing table is contained within the Forwarding Information Base (FIB), including the relation between name prefixes and faces leading to other nodes offering DATA. The forwarder decides how to forward and fulfill INTERESTS based on the aforementioned information, with DATA following the reverse path of request packets.

B. Time-Sensitive Networking

The set of IEEE 802.1Q [2] TSN standards defines methods for low latency and reliable Ethernet communication. This work focuses on the IEEE 802.1Qav standard, supported by the Precision Time Protocol (PTP). For TSN, PTP is used in the form of generic Precision Time Protocol (gPTP) as defined in the IEEE 802.1AS standard [12]. The time on each participating node is synchronized to a grandmaster clock, which is placed on top of a master-slave hierarchy. In this structure, the gPTP instances exchange messages, allowing nodes to accurately synchronize all clocks in the network.

The Multiqueue Priority (MQPRIO) queuing discipline (qdisc) enables mapping of packets belonging to various Traffic Classes (TCLs) and priorities into corresponding Network Interface Card (NIC) hardware queues. It is usually paired with the Credit-Based Shaper (CBS) introduced by the IEEE 802.1Qav standard [2] and can be used for bandwidth allocation to various TCLs. The shaper works according to a credit system where packets are dequeued when TCL's credit is ≥ 0. The amount of TCL credit over time is governed by four parameters: *idleSlope*, *sendSlope*, *hiCredit*, and *loCredit*. These specify the rates at which credit is accumulated and spent during transmission and its maximal and minimal levels. All four parameters depend on, e.g., the traffic's Physical Layer (PHY) packet size, bitrate, and available bandwidth.

C. The EnGINE Framework

In prior work, we built the EnGINE framework [9], an orchestration tool for hardware-based networking experiments based on open-source software and commercial off-the-shelf hardware. It focuses on, but is not limited to, TSN and IVN experiments. The framework is written in Ansible[3] and supports automated node setup, network configuration, traffic generation, and artifact collection and evaluation. Using appropriate hardware, e.g., the *Intel I210 NICs*, EnGINE supports CBS and TAPRIO qdiscs, as well as PTP. As part of the framework, we also provide a methodology [13] covering relevant settings, metrics, and result expectations for TSN experimentation. In this work, we extend EnGINE with support for NDN.

Other TSN experimentation approaches exist such as the TSN-FlexTest [14], focusing on evaluating the performance of a single hardware NIC. Similar evaluation can be done using, e.g., the OMNeT++ [15] discrete-event simulator.

D. Related Work

Prior research investigated the use of ICN for TSN. Papadopoulos, et al. [6] outline the challenges of mapping NDN into IVNs. They focus on the similarity of the VSS and NDN names, considering benefits of caching capabilities. The authors also outline ICN's limitations for IVNs, pointing out the lack of standardization, relevant in the automotive industry.

Threet, et al. [16] verify the applicability of NDN for enhanced security in IVNs. They investigate vulnerabilities of CAN systems and show how NDN signing schemes alleviate various attack types. The authors show that NDN can mitigate most attacks while highlighting the need for more research.

Nagaraj, et al. [7], [17] investigate the applicability of ICN for Industrial Automation Control Systems (IACS). The authors outline the benefits of in-network caching and investigate cache placement in NDN-enabled IACS. They further focus on Traffic Control Subsystem (TCS) and its impact on latency and throughput, while highlighting the applicability of NDN naming schemes for IACS. The authors conclude that the NDN implementation used in their experiments did not work correctly with the TCS, outlining the need for further research.

Moll, et al. [8], [18] investigate push-based conversational services in ICN. They formalize the concept of PIs in NDN. With such INTERESTS, each PIT entry may result in one or more DATA packets. The authors show improved performance in the push-based variant compared to the pull-based operation, specifically lowering forwarder CPU load. PIs inspired parts of the architecture design presented in this work.

IVN architectures are also subject to extensive research. Häckel, et al. in [1] investigate a Software Defined Networking (SDN) approach combined with TSN for a zonal IVN architecture. They utilize SDN's matching pipeline to manage traffic flows and integrate those with TSN traffic shaping and policing mechanisms. The authors show that such flow isolation enhances the security and removes the attacker's ability to influence the system.

[3]https://www.ansible.com

III. Analysis and Design

Real-time systems used, e.g., in IVNs or IACS, must fulfill strict latency and jitter requirements for high-priority flows. Such flows generally belong to Stream Reservation (SR) classes A and B [13]. Over a network with seven hops, these classes require a delay lower than 2 ms and 10 ms, as well as jitter of less than 125 µs and 1000 µs, respectively. Such strict requirements are needed to ensure proper functionality of safety-critical subsystems within IACS and IVNs.

A. State of the Art

With increasing bandwidth demands as seen, e.g., for autonomous driving, we observe an increasing adaptation of Ethernet, replacing legacy technologies such as CAN, LIN, or MOST [1]. Generally, to achieve low delays with Ethernet networks, traffic policing and shaping methods from the IEEE 802.1Q family of standards are applied. The settings of these shapers consider traffic characteristics, such as the packet generation frequency or packet size. This switch to Ethernet further necessitates redesigning the underlying network architecture from a centralized into a zonal, decentralized one.

Ethernet addresses communication end-points. This proves challenging in TSN systems where devices push data. Generated information is often aimed at multiple destinations, meaning that a suitable architecture should also support a form of multicast or pub/sub communication. This requires a protocol on Layer 3 or above, usually combined with some Ethernet extensions [19]. Some approaches utilize Internet Group Management Protocol (IGMP) [20] as, e.g., OPC UA when used for IVNs [21]. Alternatively, Ethernet multicast may be used. Its applicability is limited as available hardware handles a limited number of multicast groups, e.g., 16 with the Intel I210 NIC[4]. The groups are further generally defined by IGMP snooping, still relying on end-point IP addressing.

With bus technologies such as CAN bus, IVNs and IACS generally used signals to exchange information throughout the network [3]. The data placed on the bus are broadcast to all receivers. These signals use message *identifiers* that can be related to the type of specific information contained within the frames. Such *identifiers* can extend to names associated with transmitted information. Prominent examples include the VSS, defining semantic names for sensor and actuator data in IVNs. Such an approach decouples the addressing from end-points as used, e.g., in IP, and addresses the information.

Security is not directly integrated into protocols used in IVNs and IACS. It is generally applied through related standards, e.g., MACsec for Layer 2 or IPSec for Layer 3. Their integration with TSN is challenging and no readily available solutions exist [1]. This is also the case for OPC UA when used for pub/sub, as no security profiles are available [21].

B. Initial Architecture Design

To address these challenges, we propose an ICN-based architecture design for IACS and IVNs. Following best practices,

[4]https://www.intel.com/content/www/us/en/content-details/333016/intel-ethernet-controller-i210-datasheet.html

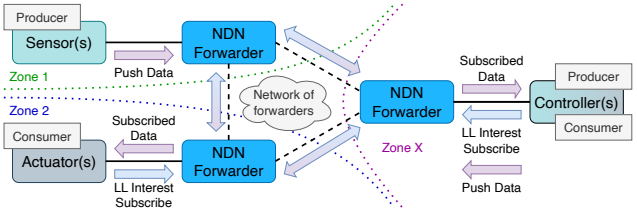

Fig. 1. Simplified overview of the proposed network architecture.

we employ Ethernet and include support for various IEEE 802.1Q standards. The timing accuracy is ensured using gPTP. At Layer 3, we employ a modified version of NDN to connect various consumers with producers. Data is named according to VSS or an equivalent scheme. As indicated in Figure 1, we follow the zonal architecture concept with all components interconnected using a decentralized network. We assume these components are directly connected to the forwarders.

While NDN enables information addressing, it brings further challenges. The packet size is influenced by NDN's request-response behavior and additional data encoded in the header, e.g., the name with variable length. This variance impacts traffic shapers, e.g., CBS, which are sensitive to packet sizes, requiring individual settings per direction to guarantee delay and jitter. This is especially relevant for Interests, directly influencing the latency of the Data packets.

To support pub/sub and multicast communication, we propose Long-Lived (LL) Interests, a concept similar to a minimal version of PIs [8], [18]. Both approaches extend the NDN Interest with an additional type field indicating its nature. If present, the forwarder retains the PIT entry even after a Data packet arrived and was forwarded. PIT entries containing the LL flag are soft-state. Stale ones are avoided using timeouts dictated by the *interest lifetime* type.

Due to the different system requirements, we approach the push-based communication with LL Interests differently when compared to PIs. The LL Interests and Data packets do not encode sequence numbers in the name to avoid its bloating and improve compatibility with traffic-shaping algorithms expecting consistent packet sizes. These numbers may be included in their payload or as a separate type field in the NDN header. The exact encoding is out of this work's scope.

In the data-push configuration, a producer creates Data packets according to its traffic pattern and continuously transmits those to the nearest NDN forwarder. The forwarders assist in its distribution, forwarding the newest Data according to LL PIT entries. The packet's *freshness period* parameter ensures no obsolete information is distributed through the network. This results in a targeted multicast capability, where multiple consumers can receive a data from a single producer, using only one Interest per client. Support for request-response Interest to Data exchange is retained.

Further considerations must be made for Quality of Service (QoS) in NDN. Some solutions consider its encoding in the name [22]. Such an approach might result in data duplication across the network if it is available under multiple priorities. Caching could also reduce the load introduced by non-critical

traffic streams if request-response communication is used. However, it may significantly impact the system because the TSN configuration is influenced by the packet generation frequency. Investigation of caching and priority encoding is subject to ongoing work and is out of scope for this paper.

IV. Initial Verification

To validate the initial architecture design, we prepare and verify an NDN-enabled experimental environment based on the EnGINE framework. We then extend the NDN implementation with support for LL INTERESTS. The resulting system enables verification of ICN concept in the context of TSN and is provided as an open-source repository[5].

A. Experimental Environment

To enable experimentation on ICN-enabled TSN systems, we extend the EnGINE framework with support for NDN, following the four-phase experiment execution flow of EnGINE. In the first phase, *install*, the experimental nodes are booted. The second phase, *setup*, is responsible for installing all required dependencies. We extend it to install software relevant for ICN. As we are using C++ NDN implementation, these dependencies include the *ndn-cxx*[6], *NDN Forwarding Daemon (NFD)*[7], and *ndn-traffic-generator*[8]. These are compiled from source, enabling experimentation with modified NDN code.

The experiments are set up and executed in the third phase, *scenario*. Firstly, we extend EnGINE's experiment configuration with the ability to place NFD on select nodes. We then utilize EnGINE's Open vSwitch-based flow configuration capabilities to enable point-to-point, NDN-enabled connections. As outlined in Figure 2, each of these flows ① follows a pre-defined path over a static network and terminates at selected nodes with virtual flow interfaces ②. The orchestration framework automatically associates these with their respective NFD faces ③. Those are used to associate traffic flowing through a NFD face with a Virtual LAN Priority Code Point, as well as Linux Socket Buffer priority ④. Priorities are applied using the *cgroups*[9] utility, specifying traffic priority on the flow interfaces. Such arrangement enables per-face QoS enforcement using TSN traffic shapers ⑤, following EnGINE's qdisc configuration capabilities.

In the next step, the NFD content store is disabled, and the NDN routing is configured. As indicated in Figure 2, all reachable nodes on a given priority are associated with an appropriate face. A similar static configuration would typically be used in systems with deterministic guarantees. While we defer a final design for naming, in our prototype, the scheme always includes `/node-X/prioY/data-name`, with `node-X` specifying the node, `prioY` specifying the priority, and a `data-name` indicating specific DATA. This naming scheme follows suggestions introduced in [22] and can easily be adjusted in the future. To generate traffic on the NDN-enabled nodes, we utilize a customized implementation of the *ndn-traffic-generator* outlined in Section IV-B. We abstract its settings, utilizing the EnGINE configuration format.

The result collection follows the EnGINE pipeline, with packets being recorded by the *tcpdump* utility. To process the recorded information, in the fourth phase, *process*, we modify the pipeline that automatically parses collected packet traces to additionally extract NDN-specific details. Collected fields include the packet's timestamp, type (INTEREST or DATA), and sequence number embedded in the name or the payload by the *ndn-traffic-generator*, as indicated in Section IV-B. If the sequence number is absent, we use the nonce associated with the packets. However, this method is not always reliable or applicable. This information is collected and saved into a CSV file for each defined data name on every interface in the network. With PTP actively synchronizing clocks across network nodes in all experiments, these files can then be correlated to directly calculate the end-to-end delay or jitter.

B. Support for LL INTERESTS

We realize LL INTERESTS as an extension to *ndn-cxx* and *NFD*. The LL Type-Length-Value field is implemented as an additional description in *ndn-cxx* definition of INTERESTS, alongside all required methods to set and read it. To support these, the implementation of NDN face was also slightly adjusted, allowing the consumer to receive multiple data packets responding to a single INTEREST. Note that the initial implementation does not consider or support caching.

The *NFD*'s INTEREST and DATA handling is enhanced with support for the LL type. The PIT entries include a flag indicating whether the incoming INTEREST contained the LL type. The data forwarding pipeline is modified to not remove LL PIT entries when they matched an incoming DATA. These entries are only removed when an appropriate NACK packet is received or the timer based on the *interest lifetime* expires.

Using *ndn-traffic-generator*'s *ndn-traffic-server*, we implement *ndn-traffic-push* that can generate push data traffic. The server's functionality is modified to only register the name prefix with the forwarder without including the methods usually used to react to incoming INTERESTS. The DATA packets are then generated in a loop based on a configured interval and subsequently sent to the NFD. Both the server and push application are further extended to include sequence numbers encoded in the payload. This is later used to accurately correlate the DATA packets between the producer and the consumer. We do not adjust the metrics calculation of the *ndn-traffic-generator*, as we calculate relevant ones using EnGINE's evaluation capabilities. We further extend the *ndn-traffic-client* to support generating INTERESTS with LL type field and accept multiple returning DATA packets for each.

V. Experiments and Evaluation

To validate the experimental environment and the LL INTERESTS concept, we build a simple network of eight nodes interconnected using *Intel I210 NICs*. Each node runs Linux

[5]https://github.com/rezabfil-sec/engine-framework
[6]https://github.com/named-data/ndn-cxx
[7]https://github.com/named-data/NFD
[8]https://github.com/named-data/ndn-traffic-generator
[9]https://docs.kernel.org/admin-guide/cgroup-v2.html

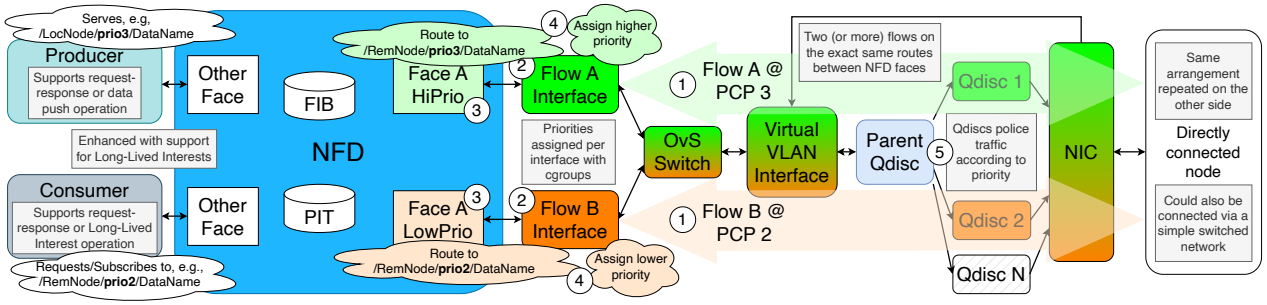

Fig. 2. Overview of components required to integrate per-face Layer 2 priority assignment and enforcement in EnGINE.

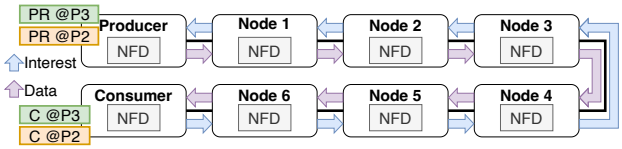

Fig. 3. Overview of the network used for experiments. C and PR indicate consumer or producer with @PX defining their priority X.

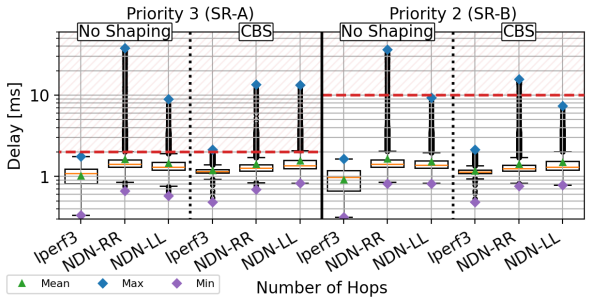

Fig. 4. Measured end-to-end delay in experimental environment validation

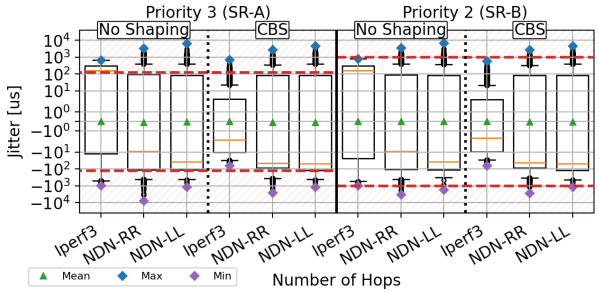

Fig. 5. Measured jitter of end-to-end delay for environment validation

with `5.15.0-89-lowlatency` kernel on *Intel Xeon E3-1265L V2* CPU, forming a line topology with seven hops, as shown in Figure 3, with NFDs placed on each hop. We follow EnGINE CPU management policies and isolate the NFD and other relevant components. We designed two experiment types to validate the system, following the guidelines in [13]. The first experiment uses only MQPRIO and does not apply any traffic shaping. We then include adequate shaping on each involved interface using the CBS. Each experiment lasts 40 s.

On the **Producer**, we configure two generators offering DATA on **priority 3** (higher) and **priority 2** (lower). On the **Consumer**, we place two clients, each requesting data on the respective priorities. We distinguish between two scenarios. The first one, NDN-RR, utilizes the normal NDN request-response communication with sequence numbers encoded in the name. The consumer sends INTERESTS every 250 µs with PHY size of 88 B. The producer fulfills the INTERESTS with DATA of 1250 B, resulting in PHY frames of up to 1373 B. In the second scenario, NDN-LL, we use our implementation utilizing LL INTERESTS. The clients send INTERESTS every 500 ms with the LL type set and size of 88 B. The producer pushes DATA packets every 250 µs with a payload of 1250 B. The sequence number is encoded in the payload, enabling post-processing and resulting in PHY frames of 1364 B.

For the CBS experiments, the shaper is configured on two hardware queues, individually for each priority. The shaping is set up adequately for the traffic patterns and packet sizes expected in each direction. We verified that the respective qdiscs shape the traffic accordingly for all experiments.

As a baseline, we run experiments with UDP traffic in the DATA direction generated with *Iperf3*. The same payload of 1250 B (PHY size of 1320 B) every 250 µs is sent. In CBS experiments, we configure the traffic shaper accordingly.

To validate NDN's use with TSN, we look at end-to-end statistics of NDN DATA and *Iperf3* packets. Figure 4 shows delay box plots, with the dashed horizontal lines representing 2 ms and 10 ms target for SR classes A and B. The whiskers are configured to show values within 1.5 times inter-quartile range. The left half of the figure shows **priority 3** and the right one results for **priority 2** traffic. We observe generally lower latencies when *Iperf3* is used, with both NDN-RR and NDN-LL performing comparably. Most of the measured delay values fall within the requirements of SR classes A and B.

When no shaping is used, *Iperf3* on **priority 3** observed an average latency of 1.02 ms, lower by 0.61 ms when compared to NDN-RR and by 0.43 ms when compared to NDN-LL. Recorded values and differences were comparable for **priority 2** flows. With CBS traffic shaping, the average delay was also lower for *Iperf3* on both priorities and measured at 1.18 ms. The mean latency in NDN-RR scenario was higher by 0.24 ms and in NDN-LL by 0.37 ms. Recorded values and differences were again comparable for flows placed on **priority 2**.

Figure 5 shows jitter, being the variation in latency between two consecutive packets. The figure and boxplot structure follows this of Figure 4, with the red lines indicating the jitter target of 125 µs and 1000 µs for SR classes A and B. We again observe that most recorded values fall within the respective

requirements of each class. While we notice generally higher jitter in NDN experiments, when the traffic was policed with CBS, the jitter for *Iperf3* flows was significantly lower.

Comparing the experiments further shows the overhead reduction. In NDN-RR, each consumer on each priority generated INTERESTS at a rate of $2816.05\,\mathrm{kbit/s}$. For NDN-LL, this was reduced to $1.43\,\mathrm{kbit/s}$. The bitrate ratio of DATA per INTEREST is improved from 15.6 in NDN-RR to $30\,610$ in NDN-LL. We also observe a significant change in ratios of packet numbers from 1 to 2000 DATA packets per INTEREST. Those values are specific to this scenario and will differ when different payload sizes and traffic patterns are used.

We believe that NDN combined with TSN traffic shaping is a feasible approach to the realization of an ICN-enabled real-time system. We observe a somewhat worse performance of NDN when compared with IP-based results using *Iperf3*. The classic NDN implementation performs similarly to that with LL INTERESTS, indicating that push-based traffic generation does not negatively impact the system. However, the NDN implementations present a worst-case scenario as its code is executed in user space. With a kernel-space or hardware-based implementation of the forwarder, we would likely observe improved fulfillment of SR class A and B requirements.

VI. CONCLUSION AND FUTURE WORK

In this work, we introduced a hardware-based environment for the assessment of ICN-based solutions in TSN systems. Our solution is based on the open-source EnGINE framework, into which we integrate support for NDN. Using this environment, we show that NDN is a promising alternative to IP, fulfilling the requirements of $2\,\mathrm{ms}$ delay and $125\,\mu\mathrm{s}$ jitter for most recorded values. We further proposed an ICN-based architecture for time-sensitive networks. The architecture aims to ease the interaction with or migration from legacy bus systems to TSN-based Ethernet networks. Utilizing long-lived interests, we enable a pub/sub-like behavior in NDN, reducing the overhead of otherwise used request-response exchanges. Being ICN-based, the architecture supports content addressing with named message *identifier* schemes such as VSS.

In the future, the correlation of named traffic with TCL priorities should be investigated further. Adequate priority encoding could also improve caching support. Best-effort and lower-priority consumers could benefit from such cached data on the forwarders, minimizing the overhead of their request-response communication. Routing is another aspect requiring further investigation. The currently considered static case should be extended with dynamic routing. Route reconfiguration is especially relevant for safety in systems such as IVNs. These networks must quickly react to failures while maintaining support for safety-critical functions.

ACKNOWLEDGMENT

This work was supported by the German Federal Ministry of Education and Research joint project 6G-life (16KISK002). We thank Atacan Iyidogan for his support implementing NDN into the EnGINE framework.

REFERENCES

[1] T. Häckel, P. Meyer, F. Korf, and T. C. Schmidt, "Secure time-sensitive software-defined networking in vehicles," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 35–51, 2023.
[2] "Ieee standard for local and metropolitan area networks–bridges and bridged networks," *IEEE Std 802.1Q-2022*, pp. 1–2163, 2022.
[3] R. B. GmbH, "Can specification version 2.0," Stuttgart, Germany, 1991.
[4] L. L. Bello and W. Steiner, "A perspective on ieee time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.
[5] R. A. Peña, M. Pascual, A. Astarloa, D. Uribe, and J. Inchausti, "Impact of macsec security on tsn traffic," in *2022 37th Conference on Design of Circuits and Integrated Circuits (DCIS)*, 2022, pp. 01–06.
[6] C. Papadopoulos, S. Shannigrahi, and A. Afanaseyv, "In-vehicle networking with ndn," in *Proceedings of the 8th ACM Conference on Information-Centric Networking*, New York, USA, 2021, p. 127–129.
[7] A. H. Nagaraj, M. P. Tahiliani, D. Tandur, and H. Satheesh, "Leveraging named data networking for industrial automation: Opportunities and challenges," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1–6.
[8] P. Moll, D. Posch, and H. Hellwagner, "Investigation of push-based traffic for conversational services in named data networking," in *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2017, pp. 315–320.
[9] F. Rezabek, M. Bosk, T. Paul, K. Holzinger, S. Gallenmüller, A. Gonzalez, A. Kane, F. Fons, Z. Haigang, G. Carle, and J. Ott, "Engine: Flexible research infrastructure for reliable and scalable time sensitive networks," *Journal of Network and Systems Management*, vol. 30, no. 4, p. 74, 2022.
[10] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. C. Schmidt, and M. Wählisch, "Information-Centric Networking (ICN) Research Challenges," RFC 7927, Jul. 2016.
[11] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 66–73, jul 2014.
[12] "Ieee standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications," (2020).
[13] M. Bosk, F. Rezabek, K. Holzinger, A. G. Marino, A. A. Kane, F. Fons, J. Ott, and G. Carle, "Methodology and infrastructure for tsn-based reproducible network experiments," *IEEE Access*, vol. 10, pp. 109 203–109 239, 2022.
[14] M. Ulbricht, S. Senk, H. K. Nazari, H.-H. Liu, M. Reisslein, G. T. Nguyen, and F. H. P. Fitzek, "Tsn-flextest: Flexible tsn measurement testbed," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023.
[15] A. Varga, "Omnet++," *Modeling and tools for network simulation*, pp. 35–59, 2010.
[16] Z. Threet, C. Papadopoulos, W. Lambert, P. Podder, S. Thanasoulas, A. Afanasyev, S. Ghafoor, and S. Shannigrahi, "Securing automotive architectures with named data networking," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 2663–2668.
[17] A. H. Nagaraj, B. Kataria, A. Sohoni, M. P. Tahiliani, D. Tandur, and H. Satheesh, "On the importance of traffic control subsystem in icn-based industrial networks," in *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2020.
[18] P. Moll, S. Theuermann, and H. Hellwagner, "Persistent interests in named data networking," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. IEEE, 2018, pp. 1–5.
[19] S. E. Deering, "Host extensions for IP multicasting," RFC 1112, Aug. 1989. [Online]. Available: https://www.rfc-editor.org/info/rfc1112
[20] B. Fenner, H. He, B. Haberman, and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")," RFC 4605, Aug. 2006. [Online]. Available: https://www.rfc-editor.org/info/rfc4605
[21] B. Leander, B. Johansson, T. Lindström, O. Holmgren, T. Nolte, and A. V. Papadopoulos, "Dependability and security aspects of network-centric control," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2023, pp. 1–8.
[22] I. Moiseenko and D. R. Oran, "Flow Classification in Information Centric Networking," Jan. 2021. [Online]. Available: https://datatracker.ietf.org/doc/draft-moiseenko-icnrg-flowclass/07/