7th KuVS Fachgespräch on Machine Learning in Networking (MaLeNe), University of Augsburg, Augsburg, Germany, March 19-20, 2026: proceedings

Michael Seufert, Andreas Blenk, Björn Richerzhagen

Angaben zur Veröffentlichung / Publication details:

Seufert, Michael, Andreas Blenk, and Björn Richerzhagen, eds. 2026. “7th KuVS Fachgespräch on Machine Learning in Networking (MaLeNe), University of Augsburg, Augsburg, Germany, March 19-20, 2026: proceedings.” In. Augsburg: Universität Augsburg.

7TH KUVS FACHGESPRÄCH ON MACHINE LEARNING IN NETWORKING (MaLeNe)

PROCEEDINGS

MARCH 19-20
2026

UNIVERSITY OF AUGSBURG, AUGSBURG, GERMANY

7th KuVS Fachgespräch on Machine Learning in Networking (MaLeNe)

The 7th KuVS Fachgespräch on Machine Learning in Networking (MaLeNe, https://www.uni-augsburg.de/de/malene2026/) was a successful event held over two half-days at the University of Augsburg on March 19 and 20, 2026. As the 7th edition of the MaLeNe workshop series, it continued the established tradition of KuVS workshops in this field. The event was organized by the workshop co-chairs Michael Seufert (University of Augsburg, Germany), Andreas Blenk (Siemens AG, Germany) and Björn Richerzhagen (Siemens AG, Germany). In total, 13 full papers were accepted.

On the first day of the workshop, the co-chairs welcomed 36 registered participants. The workshop started with an industry keynote given by Dr. Andreas Mäder (Nokia, Germany) on "AI for 6G and AI for Standards – a transformation journey". He discussed the role of AI as a key enabler for upcoming 6G networks, particularly in the context of evolving standardization processes. Furthermore, he reflected on how development cycles in mobile communications and machine learning are converging, highlighting the need to better align innovation speed with standardization efforts.

Afterwards, the first technical session on security and intrusion detection began. Xenia Wagner (ipoque, Germany) introduced SPARTAN, a persona-driven dataset for improving IoT-based APT detection in smart homes. Marleen Sichermann (University of Wuerzburg, Germany) presented ML-based intrusion detection methods tailored for IPFIX networks. Johannes Schleicher (University of Augsburg, Germany) talked about active learning for open-set intrusion detection to identify previously unseen attacks. Hannes Schwanzer (RheinMain University of Applied Sciences, Germany) presented Bayesian packet header synthesis with expert-based authenticity evaluation.

After a coffee break, the second session started, focusing on learning-based routing and congestion control. Ahmed Nader al-Dulaimy (University of Koblenz, Germany) presented a machine learning-based approach for predicting optimal parent nodes in software-defined wireless sensor networks to improve routing efficiency. Furthermore, Musab Ahmed (Hamburg University of Technology, Germany) introduced a learning-based method for making local routing decisions in sparse aeronautical communication networks to enhance connectivity under challenging conditions. Andreas Boltres (Karlsruhe Institute of Technology, Germany) explored how neural networks can generate latent embeddings to support telemetry-aware greedy routing, enabling more informed forwarding decisions. Michael König (Karlsruhe Institute of Technology, Germany) presented a method for generalizing coordinated congestion control across multiple flows and network topologies.

The second day began with a keynote titled "Industrial AI Today. Autonomous Industry Tomorrow." delivered by Dr. Andreas Blenk (Siemens AG, Germany). The talk highlighted current trends in industrial AI, including automation, digitalization, and future research directions toward autonomous systems.

Following this, the third technical session on AI for network management took place. Zineddine Bettouche (Deggendorf Institute of Technology, Germany) presented a spatial PDE-aware selective state-space model with nested memory for mobile traffic grid forecasting. Ibrahima Ndiaye (Otto von Guericke University Magdeburg, Germany) introduced a proactive load balancing approach for IIoT gateways using federated learning. Jonas Wessner (Ulm University, Germany) explored how large language models can be leveraged to automate network access control.

After a coffee break, the fourth session focused on systems, platforms, and digital twins. Johannes Späth (Technical University of Munich, Germany) presented a high-fidelity virtualized digital twin system designed to emulate ISP core networks with high accuracy. Sebastian Hauschild (University of Luebeck, Germany) introduced a portable IoT platform for experimental cadaverine measurements.

The workshop co-chairs concluded the day with a brief summary and expressed their gratitude to all speakers and participants for the engaging and productive discussions. As the workshop once again demonstrated its ability to foster active collaboration within the research community, a future edition is already being considered.

We also sincerely thank all authors, reviewers, attendees, and local organizers for their valuable contributions to the success of the workshop.

Michael Seufert, Andreas Blenk, Björn Richerzhagen
MaLeNe 2026 & Workshop Co-Chairs

Program

Welcome and Workshop Opening

Keynote 1

AI for 6G and AI for Standards – a transformation journey, Dr. Andreas Maeder (Nokia)

Session 1: Security and Intrusion Detection

1. SPARTAN: A Persona-Driven Smart-Home IoT Security Dataset for APT Detection - Victor Jüttner (Leipzig University), Erik Buchmann (Leipzig University), Miguel Vargas (ipoque, a Rohde & Schwarz company), Xenia Wagner (ipoque, a Rohde & Schwarz company), Julius Jolig (ipoque, a Rohde & Schwarz company) and Christoph Jahn (ipoque, a Rohde & Schwarz company).
2. Rethinking Feature Engineering: ML-Based Network Intrusion Detection in IPFIX Networks - Marleen Sichermann (University of Wuerzburg), Katharina Dietz (University of Wuerzburg), Jochen Kögel (Isarnet Software Solutions GmbH), Rüdiger Gunreben (Isarnet Software Solutions GmbH), Stefan Geißler (University of Wuerzburg) and Tobias Hoßfeld (University of Wuerzburg).
3. Is this a New Attack? Active Learning for Zero Knowledge Open-set Network Intrusion Detection - Johannes Schleicher (University of Augsburg), Leo Stumpf (University of Augsburg), Katharina Dietz (University of Wuerzburg) and Michael Seufert (University of Augsburg).
4. An Expert Evaluation of Efficient Models for Packet Level Traffic Generation - Maurice Falk (RheinMein University of Applied Sciences), Hannes Schwanzer (RheinMein University of Applied Sciences), Adrian Ulges (RheinMein University of Applied Sciences), Martin Gergeleit (RheinMein University of Applied Sciences) and Alexander Prange (consistec GmbH).

Session 2: Learning Based Routing and Congestion Control

5. Machine Learning-Enhanced Parent Node Prediction in Software-Defined Wireless Sensor Networks - Ahmed Nader al-Dulaimy and Prof. Dr. Hannes Frey (University of Koblenz).
6. Learning-Based Local Routing Decisions in Sparse Aeronautical Communication Networks - Musab Ahmed, Kevin Nabiev, Konrad Fuger, Koojana Kuladinithi and Andreas Timm-Giel (Hamburg University of Technology).
7. Can Neural Networks Provide Latent Embeddings for Telemetry-Aware Greedy Routing? - Andreas Boltres (Karlsruhe Institute of Technology and SAP SE), Niklas Freymuth (Karlsruhe Institute of Technology) and Gerhard Neumann (Karlsruhe Institute of Technology).
8. C3S: Towards Generalizing Coordinated Congestion Control Across Flows and Topologies - Michael König and Martina Zitterbart (Karlsruhe Institute of Technology).

Keynote 2

Industrial AI Today. Autonomous Industry Tomorrow, Dr. Andreas Blenk (Siemens AG).

Session 3: AI Network Management

9. Spatial PDE-aware Selective State-space with Nested Memory for Mobile Traffic Grid Forecasting - Zineddine Bettouche, Khalid Ali, Andreas Fischer and Andreas Kassler (Deggendorf Institute of Technology).
10. Proactive IIoT Gateway Load Balancing via Federated Learning - Ibrahima Ndiaye and Mesut Günes (Otto-von-Guericke University Magdeburg).
11. Automated Network Access Control with LLMs - Jonas Wessner (Ulm University), Tobias Meuser (Technical University Darmstadt) and Frank Kargl (Ulm University).

Session 4: Systems, Platforms and Digital Twins

12. A High-Fidelity Virtualized Digital Twin System for ISP Core Networks - Johannes Späth and Georg Carle (Technical University of Munich).
13. Portable IoT Platform for Experimental Cadaverine Measurements - Sebastian Hauschild and Horst Hellbrück (Luebeck University of Applied Sciences).

Wrap-up and Closing Remarks

SPARTAN: A Persona-Driven Smart-Home IoT-Security Dataset for APT Detection

Victor Jüttner, Erik Buchmann
Dept. of Computer Science, Leipzig University
ScaDS.AI Dresden/Leipzig, Germany
Email: {victor.juettner|erik.buchmann}@uni-leipzig.de

Miguel Vargas, Xenia Wagner, Julius Jolig, Christoph Jahn
ipoque, a Rohde & Schwarz company, Leipzig, Germany
Email: {miguel.vargas|xenia.wagner|julius.jolig|christoph.jahn}@rohde-schwarz.com

***Abstract*—The proliferation of smart-home devices has increased security challenges, with advanced persistent threats being difficult to detect due to the complexity of user interactions and device behaviors. This paper introduces SPARTAN (Smart-home Persona-driven Attack Recognition and Trace Analysis Network), a dataset designed to simulate realistic household activities combined with APT attack scenarios. Using persona-driven schedules and an automated system that orchestrates realistic smart-device interactions, we generate diverse benign traffic and inject APT phases, such as reconnaissance and exfiltration, into the network. SPARTAN offers realistic, structured, and parametrizable datasets with synchronized labels for evaluating machine learning-based APT detection methods.**

***Index Terms*—Smart Home, APT detection, Dataset.**

I. Introduction

Smart-home deployments are expanding rapidly [1], integrating sensors, actuators, and data-intensive smart devices into domestic networks. Smart devices offer convenience and automation, but they also broaden the attack surface and complicate security monitoring. As a result, smart devices are increasingly targeted and compromised, contributing to the continued growth of botnets such as Aisuru [2] and Kimwolf [3]. At the same time, smart-home network traffic is closely tied to human activity and user interaction patterns [4, 5], distinguishing it from traditional network environments.

Detecting attacks on smart devices is challenging because diverse household factors—including varying user interactions, number of residents, automation routines, and firmware updates—introduce significant variability and noise into observed behavior. Moreover, smart-home devices are typically closed platforms with limited internal observability. As a result, network-based anomaly detection approaches, such as machine-learning–driven Network Intrusion Detection Systems (NIDS), are an attractive solution [6].

Because smart-home network traffic is shaped by diverse and evolving user interaction patterns, early attack activities such as probing, credential abuse, or lateral movement often blend into legitimate behavior generated by normal device usage or routine maintenance. Effective defenses therefore needs to distinguish benign usage from attacks and ideally identify threats before full device compromise occurs. Improving network-based detection methods requires high-quality training data and datasets that reflect both realistic smart-home usage and systematic attacker behavior. However, existing smart-home and IoT security datasets often fail the complexity of advanced persistent threats (APTs) by labeling individual attack phases at a high level and lack realistic benign traffic generated by multiple distinct users [7, 8, 9].

To address this gap, we propose SPARTAN (Smart-home Persona-driven Attack Recognition and Trace Analysis Network), a novel persona-driven dataset for smart-home security that combines realistic household behavior with comprehensive attack scenarios. We simulate activity schedules from diverse household personas and orchestrate them using the Test and Training Data Automation System (TTDAS). TTDAS is an internal Rohde & Schwarz platform for executing automation scripts over original smartphone apps on Android and iOS devices. This provides the ground truth for benign network behavior. Attack patterns, modeled on all APT stages (reconnaissance, initial access, persistence, lateral movement, command-and-control, collection/exfiltration), are then injected into the network traffic at configurable times and intensities, ensuring both realism and flexibility.

Our dataset is designed to be (1) realistic, incorporating diverse user activity and devices; (2) structured, labels for both benign and malicious traffic; and (3) parametrizable, enabling the introduction of variability in household device mix, usage patterns, and attack scenarios. By combining diverse personas with complete attack lifecycles, we generate synchronized labels for physical activities (e.g., household tasks and daily routines) and corresponding network traffic, enabling fine-grained evaluation of APT detection techniques.

II. Related Work

There is a large variety of IoT-security datasets: Alex et al. review 44 IoT security datasets [7] in their survey, De Keersmaeker et al. list 74 datasets [8], and Kaur et al. review 14 datasets [9]. The three surveys show that across datasets, coverage is dominated by botnets/malware, volumetric attacks (DoS/DDoS), reconnaissance/scanning, brute force, and MITM [8, 9]. At the same time, recurring gaps include rapid device obsolescence, limited realism, poor coverage of IoT-specific protocols, evolving attack trends, and uneven documentation and labeling/ground truth [7, 8, 9]. These problems mirror general NIDS dataset limitations regarding labeling quality, bias, and suitability for specific use cases [9, 10]. Representative datasets include Neto et al. *CICIoT2023* [11]

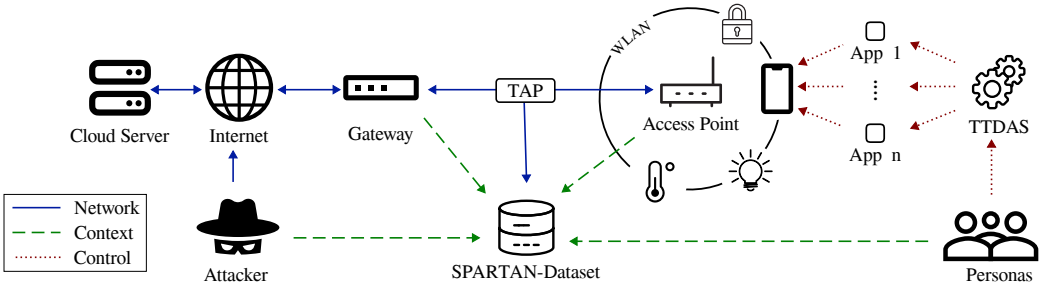

Fig. 1. Overview of the technical workflow used for smart-home traffic generation, attack execution, and dataset collection.

Koroniotis et al. *Bot-IoT* [12] by , Alsaedi et al. *TON_IoT* [13] , Ferrag et al. *Edge-IIoTset* [14], Garcia et al. *IoT-23* [15], Meidan et al. *N-BaIoT* [16], and Mirsky et al. *Kitsune* [17]. These gaps motivate our persona-driven, parametrizable smart-home dataset with activity-aligned labels and APT annotations.

III. Our Approach

A. Smart-Home Testbed and Device Diversity

Figure 1 shows our smart-home testbed. All smart devices are are controlled through a dedicated smartphone running the respective control applications. The access point is connected to the internet through a gateway, providing access to their manufacturers cloud services. Device control and automation are realized using TTDAS, which enables centralized and repeatable execution of predefined interaction sequences. Malicious traffic is injected by an attacker node located outside the test LAN, while all network traffic is tapped between the gateway and the access point using a ProfiShark 1G device.

Our smart-home environment comprises devices with diverse functional roles and communication characteristics, including **sensing devices** (e.g., temperature monitors, glass-break sensors, and water-level detectors), **high-bandwidth multimedia devices** (e.g., security cameras, smart speakers, and smart doorbells), and **actuating control devices** (e.g., heating thermostats and smart air-conditioning units). This diversity yields a wide range of network behaviors from low-rate telemetry and event-triggered messages to continuous data-streaming traffic, generated by both single-purpose and multi-functional smart devices.

B. Persona-Driven Benign Usage Patterns

A major difficulty in smart-home security is that home activities have an impact on network traffic. We address this challenge by modeling household behavior through personas. Each persona corresponds to a distinct type of occupant and is characterized by a daily routine and a set of associated smart-device actions. Those are expressed as activity schedules that map time-of-day to device interactions. To generate realistic long-term behavior, we employ large language models (LLMs) to synthesize multi-week schedules that incorporate variations (e.g., weekend-vs-weekday shifts, occasional deviations) [18]. These schedules are executed automatically using TTDAS. This approach enables repeatable simulation of diverse benign usage patterns across all devices.

C. APT Attack Scenarios

We generate APT-style attacks that can be injected into ongoing benign activity and span the full attack lifecycle, following the attack stages defined by the MITRE ATT&CK framework [19]. In addition to classical external attacker behavior, our setup also supports attacks originating from compromised IoT devices (e.g., Mirai-derivative botnet behavior observed in the wild). Our framework integrates attacks via:

- **Phase-level modeling:** APT progressions from all APT stages to enable flow-level attack labels.
- **Temporal variability:** attacks can be injected at different times/in proximity to benign activities to increase realism.
- **Intensity variability:** parameterized aggressiveness (e.g., scan rate, retry frequency, exfil volume) to model stealthy threshold-evasion behavior.

D. Labeling the Dataset

We record network traces as PCAP and store annotations in a time-aligned activity log, and we export a lightweight, analysis-ready table of per-packet ML statistical derived features with the corresponding labels. For multi-modal learning we also capture device logs from the gateway and access point. Following best practice [20], we also document dataset collection, composition, and use.

Benign annotations. Persona schedules serve as ground truth for benign behavior, enabling labeling of network intervals and events with the underlying user and device activity. We introduce controlled randomness (e.g., timing jitter and optional actions) to increase realism while preserving reproducibility via logged seeds.

Attack annotations. Attack execution produces time-aligned labels indicating attack presence and the corresponding attack phase. This supports evaluations such as phase-aware classification and early-warning performance.

IV. Conclusion

We present SPARTAN, a generator for persona-driven smart-home network traces and cyber-physical attacks. SPARTAN supports (a) training and evaluation of anomaly detection and sequence models for smart-home APT detection, (b) robustness testing under benign variability and vendor-driven background traffic, and (c) comparative feature sets (flows vs. packets) and model classes. It could also be applied to privacy attacks, e.g., tracking or activity recognition.

References

[1] Statista. (2026) Smart home - worldwide. Accessed: 15 January 2026. [Online]. Available: https://www.statista.com/outlook/cmo/smart-home/worldwide?currency=USD

[2] B. Krebs. (2025) Aisuru botnet shifts from ddos to residential proxies. Krebs on Security. Accessed: 15 January 2026. [Online]. Available: https://krebsonsecurity.com/2025/10/aisuru-botnet-shifts-from-ddos-to-residential-proxies/

[3] The Hacker News. (2025) Kimwolf botnet hijacks 1.8 million android tvs, launches large-scale ddos attacks. The Hacker News. Accessed: 15 January 2026. [Online]. Available: https://thehackernews.com/2025/12/kimwolf-botnet-hijacks-18-million.html

[4] M. H. Mazhar and Z. Shafiq, "Characterizing smart home iot traffic in the wild," in *IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2020, pp. 203–215. [Online]. Available: https://doi.org/10.1109/IoTDI49375.2020.00027

[5] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and S. Uluagac, "Peek-a-boo: i see your smart home activities, even encrypted!" in *WiSec '20*. ACM, 2020, p. 207–218. [Online]. Available: https://doi.org/10.1145/3395351.3399421

[6] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019. [Online]. Available: https://doi.org/10.1109/JIOT.2019.2926365

[7] C. Alex, G. Creado, W. Almobaideen, O. Abu Alghanam, and M. Saadeh, "A comprehensive survey for iot security datasets taxonomy, classification and machine learning mechanisms," *Computers & Security*, vol. 132, 2023. [Online]. Available: https://doi.org/10.1016/j.cose.2023.103283

[8] F. De Keersmaeker, Y. Cao, G. Kabasele Ndonda, and R. Sadre, "A survey of public iot datasets for network security research," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1808–1840, 2023. [Online]. Available: https://doi.org/10.1109/COMST.2023.3288942

[9] B. Kaur, S. Dadkhah, F. Shoeleh, E. C. Pinto Neto, P. Xiong, S. Iqbal, P. Lamontagne, S. Ray, and A. A. Ghorbani, "Internet of things (iot) security dataset evolution: Challenges and future directions," *Internet of Things*, vol. 22, p. 100780, 2023. [Online]. Available: https://doi.org/10.1016/j.iot.2023.100780

[10] P. Goldschmidt and D. Chudá, "Network intrusion datasets: A survey, limitations, and recommendations," *Computers & Security*, vol. 156, 2025. [Online]. Available: https://doi.org/10.1016/j.cose.2025.104510

[11] E. C. Pinto Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023. [Online]. Available: https://doi.org/10.3390/s23135941

[12] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2019.05.041

[13] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3022862

[14] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3165809

[15] S. Garcia, A. Parmisano, and M. J. Erquiaga, "Iot-23: A labeled dataset with malicious and benign iot network traffic," 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4743746

[16] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018. [Online]. Available: https://doi.org/10.1109/MPRV.2018.03367731

[17] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2018. [Online]. Available: https://doi.org/10.14722/ndss.2018.23204

[18] V. Jüttner, A. Fleig, and E. Buchmann, "ChatAnalysis revisited: can ChatGPT undermine privacy in smart homes with data analysis?" *i-com*, vol. 24, no. 1, pp. 173–187, 2025. [Online]. Available: https://doi.org/10.1515/icom-2024-0072

[19] B. Strom, A. Applebaum, D. Miller, K. Nickels, A. Pennington, and C. Thomas, "MITRE ATT&CK: Design and philosophy," 2018. [Online]. Available: https://www.mitre.org/sites/default/files/2021-11/prs-19-01075-28-mitre-attack-design-and-philosophy.pdf

[20] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford, "Datasheets for datasets," *Commun. ACM*, vol. 64, no. 12, p. 86–92, 2021. [Online]. Available: https://doi.org/10.1145/3458723

Rethinking Feature Engineering: ML-Based Network Intrusion Detection in IPFIX Networks

Marleen Sichermann*, Katharina Dietz*, Jochen Kögel**, Rüdiger Gunreben**, Stefan Geißler*, Tobias Hoßfeld*
*Chair of Communication Networks, University of Würzburg, Würzburg, Germany,
**Isarnet Software Solutions GmbH, Munich, Germany,
*{marleen.sichermann, katharina.dietz, stefan.geissler, tobias.hossfeld}@uni-wuerzburg.de,
**{jochen.koegel, ruediger.gunreben}@isarnet.de

I. Introduction

Network Intrusion Detection Systems (NIDSs) are essential for defending computer networks against the rising complexity and frequency of cyberattacks. Recently, much research has focused on applying machine learning (ML) to improve NIDSs [1]. Many approaches have been proposed in the literature, yet are rarely deployed in real-world environments, highlighting a gap between research and practical implementation. A major contributing factor is feature engineering: (i) Many works neglect to evaluate the availability of features in real-world deployments, and (ii) authors often omit detailed feature descriptions and instead only mention the used dataset. It is therefore often unclear whether studies rely on packet or flow-level features. Although packet-level features provide detailed information, their availability in high-speed networks is limited by computational and resource demands. Another problem is that studies frequently utilize all available features in a dataset without assessing their realism or practicality for deployment in real-world environments. Therefore, best practices recommend NetFlow or the IP Flow Information Export (IPFIX) protocol, which has become the de facto standard for high-speed networks [2].

However, the term NetFlow is often overloaded in literature and used without precise specification: in some cases it refers to Cisco's proprietary flow export protocol, of which multiple versions exist, along with various third-party implementations, while in other cases it denotes the broader combination of packet capture and flow aggregation that leverages such export protocols [3]. However, we argue that deploying a model in a real-world system, particularly in large enterprise networks, requires careful consideration of the underlying flow export technology and its limitations as well as the characteristics of the network infrastructure. In this context, we aim to clarify the concepts behind NetFlow/IPFIX and to investigate how practical flow export constraints affect the performance of ML-based intrusion detection systems.

II. Problem Description

The resulting problem is that many ML approaches from the literature are based on assumptions that do not hold in practice, or when operating under the limitations of operational, large-scale deployments. This has been discussed in multiple related works, such as [2], [4]–[6], and poses the question, how experiments need to be designed to enable the development of ML solutions that alleviate this issue.

First, to design experiments that transfer well to real-world systems we must carefully select the datasets and models used during the development of ML solutions. Public datasets are often synthetic and thus imperfect, yet they remain essential for NIDS research as they provide the only available labeled data [2]. Many datasets are distributed as packet captures (PCAPs), allowing conversion into various flow formats [7]–[9]. To further ensure generalization to real-world conditions, rigorous cross-dataset evaluation (e.g. training on one dataset and testing on another) can be applied [10].

Second, the lack of standardized IPFIX and NetFlow implementations across hardware hinders the development of generalizable ML-based solutions. While industry standards exist, vendor-specific "dialects" and inconsistent feature support produce heterogeneous datasets. For example, critical telemetry like packet inter-arrival times or TCP flags may be exported by high-end switches but omitted by edge routers. This fragmentation forces ML models to either use a "lowest common denominator" feature set and losing the granularity needed to detect sophisticated attacks, or employ complex normalization layers. Consequently, models often face significant feature drift and performance degradation in multi-vendor environments, undermining NIDS scalability and reliability.

Several studies have explored IPFIX-centric feature engineering. The authors of [11] investigated how certain features can be calculated from IPFIX records. In [1] Sarhan et al. used nProbe to convert public datasets into NetFlow format. While this constitutes an initial step, possible constraints of flow export devices were not considered. These works show that, while progress is being made, significant challenges remain.

III. Modern Day Flow Export Technologies

Flow monitoring enables scalable network traffic analysis by summarizing communication behavior in the dataplane rather than exporting individual packets. Thereby, traffic at an observation point (e.g. a switch or router) is aggregated into records based on shared properties (key fields) and subsequently exported. Early approaches lacked a common export standard until Cisco's NetFlow v5 achieved widespread use, followed by the more flexible NetFlow v9 [12]. While other vendors developed similar mechanisms, the Internet Engineering Task

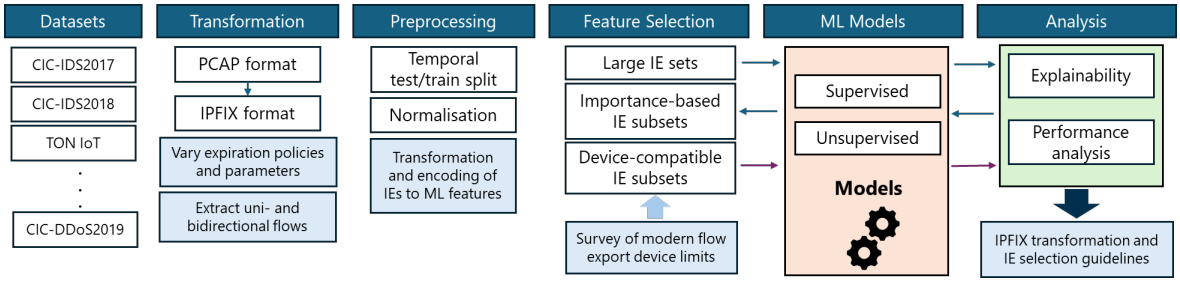

Fig. 1. Proposed workflow for evaluating IPFIX-centric NIDS and real-world constraints.

Force (IETF) eventually standardized IPFIX based on NetFlow v9 [13]. Although IPFIX extends v9's capabilities and shares its principles, they remain mutually incompatible [3]. Consequently, this discussion focuses primarily on IPFIX.

Application of IPFIX generally involves packet observation, flow metering and export, and data collection. Multiple of these stages are often consolidated into routers, switches, or dedicated probes. During metering, packets are aggregated into flows structured by Information Elements (IEs). While the Internet Assigned Numbers Authority (IANA) standardizes common IEs, custom enterprise-specific elements can capture data across Layers 2–7. For instance, Layer 7 IEs can utilize deep packet inspection for application-level insights, and IEs like `ipPayloadPacketSection` can even reference payload segments. The specific IEs collected depend on the exporter's implementation and configuration. These are communicated via templates defining the structure of subsequent data records.

The flow cache manages active flows by categorizing IEs into key and non-key fields. Upon packet arrival, the system identifies a match using key fields. If found, it updates non-key fields (e.g., byte counters). Otherwise, it initializes a new entry. While flows are inherently unidirectional, specialized cache support enables bidirectional tracking [3]. Though, bidirectional flows should not be assumed, as unidirectional reporting may be required to prevent duplicate records in specific environments, such as heterogeneous cloud architectures [14], and symmetric routing is often not guaranteed.

Flow cache sizing depends on available memory, expected flow volume, IE count, and expiration policies. While IPFIX does not mandate exact intervals, it recommends several expiration triggers [3]. Active timers periodically export long-lived flows to ensure regular updates, while inactive timers expire flows after a period of silence. Resource constraints or cache saturation can trigger emergency exports as well. Some implementations use protocol-specific triggers, e.g., TCP flags. Crucially, active and inactive timer configurations directly affect cache occupancy, CPU load, and total export volume.

Flow exporters range from fully software-based implementations, offering flexibility and affordability, to fully hardware-based solutions like Application Specific Integrated Circuits (ASICs), providing higher throughput at the cost of adaptability. Specialized Network Processing Units (NPUs) and configurable ASICs represent a middle ground. Exporter throughput is typically specified by vendors in Gbit/s. However, provided figures often assume high cache hit rates and minimal IE counts. Actual performance decreases as the volume or complexity of IEs increases. Consequently, both quantity and type of IEs must be evaluated, especially for older or hardware-based exporters that support restricted IE sets and may exhibit varying measurement accuracy [3]. This has important implications for feature engineering, since the availability and reliability of IEs determine which features can be computed.

IV. Research Approach

From the previous sections, it becomes clear that feature selection must be aligned with the capabilities of real-world flow export systems. In particular, hardware-based devices support only a limited number of IEs, and high traffic throughput further constrains the ability of collecting additional IEs. To investigate this issue, we present our research approach, with the workflow illustrated in Fig. 1. We plan on converting PCAP datasets into IPFIX format, after which extracted IEs are transformed or encoded into ML features. During preprocessing, the data is temporally split to prevent leakage, or cross-dataset testing can be used instead.

We plan to investigate how IE collectability affects IDS performance by training and comparing ML models using different IE sets. This helps establishing guidelines on the most important IEs, considering both detection utility and practical feasibility. Different IEs may matter for detecting specific attack types, and bidirectional flows could provide additional context but increase collection overhead. Flow expiration policies should also be considered, as they influence which traffic patterns are captured and can impact IDS effectiveness. This supports enterprises in choosing which IEs to enable and picking devices accordingly.

Finally, we will examine the current device landscape to understand which flow export features and IEs are realistically available. This knowledge will inform guidelines on the features and limitations scientists can expect in typical deployment scenarios. Thus, our aim is to translate research into practice and practical constraints back into research.

Acknowledgment

This work is funded by the Bavarian Ministry of Economics, Regional Development and Energy (StMWI) within the project VIPNANO as part of the R&D program for information and communication technology under research grants DIK-2307-0005 and DIK-2307-0006. The authors alone are responsible for the content.

REFERENCES

[1] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, no. 1, pp. 357–370, 2022.

[2] M. Husák, D. Manoj, and P. Kumar, "Machine learning in intrusion detection: An operational perspective," in *IEEE International Conference on Network and Service Management (CNSM)*. IEEE, 2024, pp. 1–7.

[3] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[4] K. Dietz, M. Mühlhauser, J. Kögel, S. Schwinger, M. Sichermann, M. Seufert, D. Herrmann, and T. Hoßfeld, "The missing link in network intrusion detection: Taking AI/ML research efforts to users," *IEEE Access*, vol. 12, pp. 79 815–79 837, 2024.

[5] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *USENIX Security Symposium*, 2022, pp. 3971–3988.

[6] Z. K. Maseer, Q. K. Kadhim, B. Al-Bander, R. Yusof, and A. Saif, "Meta-analysis and systematic review for anomaly network intrusion detection systems: Detection methods, dataset, validation methodology, and challenges," *IET Networks*, vol. 13, no. 5-6, pp. 339–376, 2024.

[7] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8.

[8] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116.

[9] N. Moustafa, "A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets," *Sustainable Cities and Society*, vol. 72, p. 102994, 2021.

[10] L. D'hooge, T. Wauters, B. Volckaert, and F. De Turck, "Inter-dataset generalization strength of supervised machine learning methods for intrusion detection," *Journal of Information Security and Applications*, vol. 54, p. 102564, 2020.

[11] F. Iglesias and T. Zseby, "Analysis of network traffic features for anomaly detection," *Machine Learning*, vol. 101, no. 1, pp. 59–84, 2015.

[12] B. Claise, "Cisco Systems NetFlow services export version 9," RFC 3954, Oct. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3954

[13] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP Flow Information Export (IPFIX) protocol for the exchange of flow information," RFC 7011, Sep. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc7011

[14] M. Sichermann, K. Dietz, J. Kögel, S. Meier, S. Geißler, and T. Hoßfeld, "VIPNANO: Monitoring of virtual private cloud networks for automated anomaly detection," in *4th GI/ITG KuVS Fachgespräch "Network Softwarization"*, 2025.

Is this a New Attack? Active Learning for Zero Knowledge Open-set Network Intrusion Detection

Johannes Schleicher*, Leo Stumpf*, Katharina Dietz†, Michael Seufert*
*University of Augsburg, Chair of Networked Systems and Communication Networks, Augsburg, Germany
{johannes.schleicher | leo.stumpf | michael.seufert}@uni-a.de
†University of Würzburg, Chair of Communication Networks, Würzburg, Germany
katharina.dietz@uni-wuerzburg.de

***Abstract*—Machine Learning (ML) based Network Intrusion Detection Systems (NIDS) often face challenges due to the scarcity of labeled data. Active Learning (AL) can reduce the labeling effort by focusing on uncertain instances. However, traditional AL methods face limitations in open-set scenarios, where the main challenge lies in identifying novel or previously unseen attack types, rather than simply resolving uncertainty among known classes. This paper highlights the possible improvement of intrusion detection systems through the integration of novelty detection techniques into open-set AL to minimize the manual annotation effort. We propose an approach that initially trains a system from scratch using only benign traffic to establish a baseline. It then employs a zero-knowledge learning strategy to incrementally update the NIDS with expert-verified labels for both malicious and benign traffic, enabling adaptation to concept drift as well as newly emerging attack types. Experiments on the CIC-IDS2017 dataset show that novelty detection and in particular AL effectively supports the identification of novel attacks with minimal input from an oracle.**

***Index Terms*—NIDS, Zero-Knowledge, AL, CIC-IDS2017.**

I. Introduction

Modern networks form the backbone of critical infrastructure. As these systems grow in complexity and importance, they become increasingly attractive targets for cyberattacks [1].

With the rapid growth of network traffic and more complex attacks, traditional rule-based Network Intrusion Detection Systems (NIDS) struggle to keep up. Their reliance on predefined signatures often limits their ability to detect complex or novel threats. To address this, Machine Learning (ML)-based NIDS have emerged, enabling the discovery of new patterns and correlations from large volumes of labeled traffic [2].

As traffic evolves, models trained on outdated data lose effectiveness, a phenomenon known as concept drift [3]. However, in dynamic open-set environments, previously unseen traffic can appear at any time. While closed-set NIDS, highlighted in blue in Figure 1, accurately classify benign traffic and known attacks, they often fail to detect unknown threats.

In this paper, we propose using Active Learning (AL) to address the challenge of detecting unknown attacks while adapting to concept drift, as illustrated in the green section of Figure 1. Unlike traditional retraining, AL can quickly identify informative samples through specialized ML models, which can then be labeled by an oracle. Nevertheless, as noted by Dietz et al. [4], the resulting administrative load can still be overwhelming and thus prohibitive.

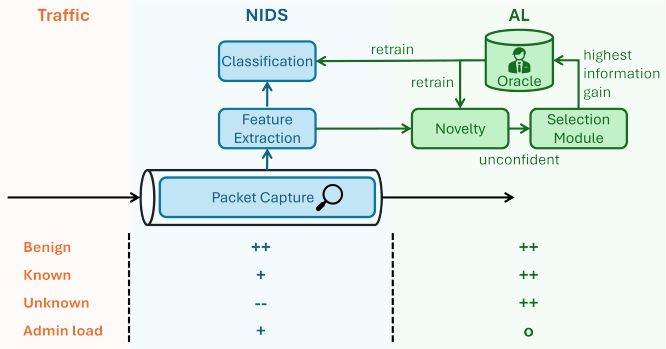

Fig. 1. NIDS functionality and performance with and without AL for benign traffic, known/unknown attacks, and admin load

II. Background & Related Work

As shown in Figure 1, the NIDS (blue) monitors network traffic and classifies it using a dedicated module. Integrating AL (green) allows the classifier to improve by selective expert labeling. Its output can either alert network administrators to suspicious activity or feed a Network Intrusion Prevention Systems (NIPS) for active mitigation. In both cases, achieving high recall and precision is essential to reliably detect attacks while maintaining normal network operation [5].

Labeling all network traffic is infeasible due to its sheer volume. Instead, leveraging knowledge from a small set of labeled samples allows systems to identify uncertain or novel traffic [6], [7]. Previous work shows that labeling only a small subset can already yield substantial improvements in detection performance [8]. However, effective AL requires more than selecting the most anomalous samples. As emphasized by Dietz et al. [4], an effective query strategy is crucial to capture informative samples without overloading human operators.

Traditional closed-set AL assumes all data belong to a fixed set of known classes [9], [10], limiting its ability to detect novel attacks. In practice, networks frequently face previously unseen traffic (open-set), which often contains the most informative samples and should thus be prioritized by novelty detection techniques [11], [12], [13].

Beyond novelty, querying diverse samples, including benign traffic, is essential to counter concept drift as normal traffic also evolves, while maintaining a balance to avoid missing genuinely suspicious activities [7], [14].

III. Benefits of AL-based NIDS

A. System Description

Creating high-quality annotated datasets for every network is impractical. We therefore propose a zero-knowledge AL-based NIDS framework that first learns a baseline from a short period of benign traffic. It then iteratively improves its detection capabilities through AL by identifying and labeling uncertain samples. This process enables the detection of both known and novel threats, including zero-day attacks, without the need of an extensive dataset.

As shown in Figure 1, network traffic is parsed into one-second time slots and processed through two parallel pipelines. One performs real-time closed-set classification for immediate detection (blue), while the other stores samples in an unlabeled pool for offline open-set AL (green). A novelty detection module selects the most informative or anomalous samples for expert labeling. The newly labeled samples can then be incorporated with the already labeled ones to retrain both the classification and novelty detection models. This zero-knowledge approach allows the system to continuously adapt without impacting the real-time performance.

As Random Forest classifier are widely used [8] and have demonstrated strong performance due to their robustness and ability to handle high-dimensional data [9], we decided to employ them for the classification throughout our investigations.

B. Real-Time Feature Set.

To support near real-time analysis, we implement a slotting mechanism inspired by [15] that computes flow features for all active flows every second. We extend the flow-level metrics of Gray et al. [15], accepting higher computational costs [16], [17] to improve robustness in open-set scenarios, where diverse indicators are needed to detect novel attacks.

We simulate real-time network traffic using the widely adopted CIC-IDS2017 [18] dataset, which includes diverse attack scenarios. We replay raw PCAPs to generate one-second long snapshots of active flows [15], providing a realistic and repeatable real-time classification. The corrected labels from [19] provide an accurate ground truth for model evaluation.

C. Evaluation

To highlight the limitations of closed-set NIDS, we train an initial model using Monday's benign traffic and 10% of the traffic from Tuesday and Wednesday. This includes attacks such as Patator, Denial of Service (DoS), and Heartbleed, while others like Distributed Denial of Service (DDoS), Infiltration, Portscan, and Web remain unseen, representing unknown threats to the NIDS. For evaluation, the remaining 90% of the traffic from Tuesday to Friday is further split, with 40% used to evaluate the closed-set system and support novelty detection. To simulate the AL loop, the flagged samples are incorporated into retraining. The updated AL-based NIDS performance is then evaluated on the remaining 60%. Table I summarizes the results, with known classes in regular font and initially unknown classes in italics. The closed-set Random Forest classification detects known attacks with over 99% precision and recall, but achieves only about 86% precision on benign traffic. Its 0% recall for unknown attacks underscores the inherent limitation of closed-set classification.

TABLE I
Performance evaluation of closed-set, novelty, and AL NIDS

Class	NIDS		NIDS+Novelty		NIDS+AL	
	P [%]	R [%]	P [%]	R [%]	P [%]	R [%]
Benign	86.14	99.99	91.49	98.72	99.34	99.99
SSH-Patator	100	99.89	100	99.89	100	99.89
FTP-Patator	100	99.83	100	99.83	100	99.88
Heartbleed	100	99.78	100	100	100	100
DoS	99.90	99.87	99.91	99.88	99.95	99.94
DDoS	0.00	0.00	100	85.33	100	99.93
Infiltration	0.00	0.00	0.01	0.01	99.96	85.70
Botnet	0.00	0.00	100	3.87	98.69	60.04
Portscan	0.00	0.00	0.01	0.01	99.95	99.42
Web	0.00	0.00	100	1.80	100	3.09

To address this, we implement an One-Class Support Vector Machine (OC-SVM) for novelty detection. As a proof of concept, we flag the top 20% most uncertain samples as malicious. This threshold ensures detection of at least one instance of each unknown attack type, while only about 12% of traffic is actually malicious.

As the results show, flagging 20% inevitably misclassifies some benign flows, reducing recall to just under 99% and potentially blocking thousands of legitimate connections. Nevertheless, the novelty enhanced system can detect previously unseen attacks. DDoS, Botnet, and Web are identified with nearly 100% precision, though recall for Botnet and Web remains below 5%. Detection of Portscan and Infiltration remains marginal, with near-zero precision and recall.

To adapt to concept drift, uncertain samples can also be incorporated into an AL loop, where they are sent to an oracle for labeling instead of being directly blocked. The resulting labels are then added to the training set to retrain both the classifier and the OC-SVM novelty module. As shown in Table I, this approach achieves the best overall performance. All attack types reach roughly 99% precision, and recall rises to at least 60%, except for Web. For benign traffic, recall approaches 100%, while precision peaks at 99%.

Labeling the top 20% of uncertain samples captures all attack types but involves thousands of flows, making it impractical for a single administrator. This underscores the need for a more efficient novelty detection strategy.

IV. Conclusion

Labeling data is costly, and existing NIDS datasets are scarce and can quickly become outdated. Therefore, this paper investigates whether AL can enhance NIDS performance in an open-set scenario by implementing a zero-knowledge learning strategy. Our initial findings demonstrate that AL-based NIDS outperform static, pretrained models by continuously refining their detection capabilities. However, applying AL in open-set environments also requires effective novelty detection methods to keep the manual labeling efforts of network administrators as low as possible.

REFERENCES

[1] Allianz, "Allianz risk barometer 2024," 2024, accessed: 2024-09-05. [Online]. Available: https://www.agcs.allianz.com

[2] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 50–60.

[3] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and H. Bouchachia, "A Survey on Concept Drift Adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, 04 2014.

[4] K. Dietz, M. Hajizadeh, N. Wehner, S. Geißler, P. Casas, M. Seufert, and T. Hoßfeld, "Certainly uncertain: Demystifying ml uncertainty for active learning in network monitoring tasks," in *2024 20th International Conference on Network and Service Management (CNSM)*, 2024, pp. 1–7.

[5] K. Dietz, M. Mühlhauser, J. Kögel, S. Schwinger, M. Sichermann, M. Seufert, D. Herrmann, and T. Hoßfeld, "The Missing Link in Network Intrusion Detection: Taking AI/ML Research Efforts to Users," *IEEE Access*, 2024.

[6] K. Dietz, N. Wehner, P. Casas, T. Hošfeld, and M. Seufert, "Demystifying User-based Active Learning for Network Monitoring Tasks," in *2nd Workshop on Machine Learning & Netwoking (MaLeNe), co-located with the 5th International Conference on Networked Systems (NetSys 2023), Potsdam, Berlin, September 4, 2023: proceedings*, M. Seufert, A. Blenk, and O. Landsiedel, Eds., 2023.

[7] J. Talpini, F. Sartori, and M. Savi, "Enhancing Trustworthiness in ML-Based Network Intrusion Detection with Uncertainty Quantification," 2024.

[8] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, "A Survey of Deep Active Learning," 2021.

[9] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "Active Learning for Network Traffic Classification: A Technical Study," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 1, pp. 422–439, 2022.

[10] CrowdStrike, "CrowdStrike Global Threat Report 2024," 2024, accessed: 2024-09-05. [Online]. Available: https://www.crowdstrike.com

[11] K.-P. Ning, X. Zhao, Y. Li, and S.-J. Huang, "Active Learning for Open-set Annotation," 2022.

[12] C.-C. Zong, Y.-W. Wang, K.-P. Ning, H.-B. Ye, and S.-J. Huang, "Bidirectional Uncertainty-Based Active Learning for Open Set Annotation," 2024.

[13] B. Safaei, V. VS, C. M. de Melo, and V. M. Patel, "Entropic Open-set Active Learning," 2023.

[14] I. Elezi, Z. Yu, A. Anandkumar, L. Leal-Taixe, and J. M. Alvarez, "Not All Labels Are Equal: Rationalizing The Labeling Costs for Training Object Detection," 2021.

[15] N. Gray, K. Dietz, M. Seufert, and T. Hossfeld, "High Performance Network Metadata Extraction Using P4 for ML-based Intrusion Detection Systems," in *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1–7.

[16] X. Li, P. Yi, W. Wei, Y. Jiang, and L. Tian, "LNNLS-KH: A Feature Selection Method for Network Intrusion Detection," *Security and Communication Networks*, vol. 2021, pp. 1–22, 01 2021.

[17] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection," *IEEE Access*, vol. 8, pp. 132 911–132 921, 2020.

[18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *International Conference on Information Systems Security and Privacy*, 2018.

[19] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study," in *2021 IEEE Security and Privacy Workshops (SPW)*, 2021, pp. 7–12.

An Expert Evaluation of Efficient Models for Packet Level Traffic Generation

Maurice Falk[1], Hannes Schwanzer[2], Adrian Ulges[1], Martin Gergeleit[1]
RheinMain University of Applied Sciences
Wiesbaden, Germany
[1]{firstname.lastname}@hs-rm.de, [2]{firstname.lastname}@student.hs-rm.de

Alexander Prange
consistec GmbH
Saarbrücken, Germany
alexander.prange@consistec.de

***Abstract*—While state-of-the-art synthetic network traffic generation relies on resource-intensive, byte-level autoregressive models, their computational overhead poses a barrier to scalability. This paper explores first steps towards a more efficient model by utilizing Bayesian networks (BN) for packet-level header synthesis, leveraging structural dependencies and domain heuristics rather than raw byte sequences. Besides protocol compliance checks and training data suitability (TSTR), we evaluate generation approaches through a novel expert discrimination test. Preliminary results show that, while BN-generated flows currently exhibit lower fidelity than byte-level models, they achieve high TCP protocol compliance and experts' error is 53% when discriminating real traffic from generated one.**

***Index Terms*—flow generation, packet header, bayesian network, data synthesis, discrimination test**

I. Introduction

While ML-based network traffic analysis and security is gaining interest in an increasingly digitized world, it is hampered by the lack of annotated real-world network traffic. Research datasets are either constructed in laboratory environments to examine specific aspects such as intrusion detection [12], [13] or encrypted traffic classification [3], [6], are limited to campus networks [14], [17], or are kept private altogether [5], [7], [8]. Training directly on real-world target networks – though known to be vital for ML models [15] – is often infeasible due to privacy concerns. Therefore, recent research has turned towards generating network traffic that replicates the statistical properties of a target network's traffic.

This work addresses traffic generation at the packet level, which – in contrast to aggregated flow-level statistics – offers significantly higher fidelity and greater flexibility, and enables the generation of standard Packet Capturing (PCAP) files, which can be analyzed using tools like *Wireshark* [16]. Recent work frames such packet-level traffic generation as a language modeling task, generating header bytes as words/tokens with large-scale models such as transformers [8], [9], [18] and State Space Models (SSM) [2], as in Chu et al. recent *NetSSM* model, which achieves state-of-the-art results [1].

In this work, we address two open issues:
(1) Model Efficiency: Models like NetSSM feature >100 mio. parameters, and generate raw packet headers byte by byte (up to 94 tokens per packet), which is resource-intensive. Therefore, we argue that certain fields of a packet header can be heuristically derived (e.g., increasing Sequence Numbers) or do not change across a single flow (e.g., ethernet addresses / IPs / ports). Other generator architectures that only utilize derived fields from the packet headers, for example, NetShare [19] (GAN) or NetDiff [20] (Diffusion), but struggle to model stateful protocol characteristics [1], [18]. Inspired by Schoen et al. [11], who showed that Bayesian networks (BNs) outperform GAN-based models for generating network flows on an aggregated level, we ask the question how a much simpler and efficient BN-based approach fares on *packet-level* flow generation compared to much larger language models.
(2) Model Evaluation: While prior work has evaluated the authenticity of generated traffic only indirectly (by benchmarking classifiers trained on it, a strategy known as "Train on Synthetic, Test on Real" (TSTR) [4]) or sparsely (by checking some heuristic constraints such as TCP handshakes), generated traffic's plausibility to human experts has not been evaluated directly to our knowledge. To this end, we present an expert discrimination test, in which experts are asked to label flows as either real or generated. Particularly, we investigate how well expert-perceived authenticity aligns with the above prior evaluation strategies (i.e., domain checks and as TSTR). We share our code and expert annotations at *https://github.com/lavis-nlp/2026-malene-packet-generation*.

II. Flow Generation using Bayesian Networks

We generate packet data using a BN, where a packet's header fields depend only on the preceding packet. Each network node represents a packet header feature (namely, *payload length*, *inter-packet time*, *direction* (client/server), and *TCP flags*), while edges encode conditional dependencies between them. Other nodes include header fields of the previous packet, the packet index within the flow, a protocol label (such as *WEB_HTTP*), and a binary indicator denoting the flow's last packet. All continuous features are discretized using quantile binning ($n = 10$). The network structure is learned using Hill-Climbing search with the K2 score, and maximum likelihood estimation. To preserve causality, we restrict the structure such that input features, namely the preceding packet and protocol labels, cannot depend on features to be generated, namely the next packet and the end-of-flow indicator. The generation process can be either initialized by a separate BN trained only on first packets or by extracting the required features from a seed packet.

III. Experiments

We compare the flows generated by our BN generation model to the byte-level generation model NetSSM in three tasks: a novel expert discrimination test, domain-specific checks, and ML-based classifier training (TSTR).

Setup: Our experiments were performed using the *CIC-IDS-2017* [12] and Non-VPN traffic of the *ISCX-VPN-2016* [3] dataset. For each dataset, bi-directional flows were identified using the standard 5-tuple [10] with a 120-sec timeout. Since NetSSM targets TCP traffic, all non-TCP flows were discarded. For each dataset, we held out 20% of the flows for testing. As both our BN and NetSSM utilize a class label of the flow, we annotated a majority of the flows using a combination of deep packet inspection, expert domain knowledge, and heuristic rules, assigning their specific protocol or application (e.g., *FTP*). Since there is currently no official implementation of NetSSM shared by the authors, we re-implemented the base model according to the original specifications. However, we added a special end-of-sequence (EOS) token to each flow during training to explicitly determine flow completion during inference. Training of the models on 671k flows took $\approx$ 31 min (BN) and $\approx$ 153 h (NetSSM), respectively.

Expert Discrimination Test: In this experiment the realism of the generated flows is evaluated by experts through a manual discriminative test. Three network experts (with 2, 8, and 25 years of professional experience in the field) are individually given 200 flows for each model (100 real/synthetic) as PCAPs and asked to annotate each flow as either *real*, *likely real*, *likely generated*, or *generated*. The annotators did not have knowledge whether a flow is real or generated.

For evaluation consistency, real and synthetic flows were reconstructed into PCAPs using a shared feature subset. Static fields like IPs and ports were inherited from seed packets, while inter-packet times were fixed to a constant value to accommodate NetSSM's lack of temporal data.

TABLE I
MEAN AND STANDARD DEVIATION OF THE DISCRIMINATION TEST. ON AVERAGE, GENERATED FLOWS WHERE CLASSIFIED AS *(likely) real* IN 53% (BN) AND 70% (NETSSM) OF CASES.

Ground truth	Real	Likely Real	Likely Generated	Generated
NetSSM				
Real	13.00±12.53	51.00±17.09	17.67±15.7	18.33±20.79
Generated	17.67±17.04	52.33±18.04	15.33±11.85	14.67±11.02
Bayesian Network				
Real	20.33±17.79	51.00±25.87	15.00±7.00	13.67±13.61
Generated	4.67±4.16	48.67±21.55	26.00±12.29	20.67±7.37

The results in Table I indicate that for both models, the experts would generally tend to classify a flow as *(likely) real*, confirming that the generated flows look realistic (at high standard deviation between experts). For NetSSM, the experts rated truly generated flows as *(likely) real* more often (70%), while the generated flows of the BN-based generator were classified as *(likely) real* around 53%. For the BN, experts occasionally observed implausible TCP behavior, such as data being transmitted after FIN flags, while NetSSM showed more isolated errors, e.g., a TCP packet without flags. Overall, the NetSSM-generated flows were commented as 'more realistic'.

Domain Checks Following previous works [1], we evaluate whether the generated flows comply with TCP session rules. For this, we generated samples for up to 3000 flows per dataset. The generation took $\approx$ 13 mins for BN (single core) and $\approx$ 28 h (129$\times$ more) for NetSSM (batch size of 1). We report the accuracy for different heuristics in Table II.

TABLE II
EVALUATION SCORES FOR DOMAIN CHECKS OF THE REAL/GENERATED FLOWS (UPPER) AND WHEN USED AS TRAINING DATA FOR FLOW CLASSIFICATION (LOWER). c = NUMBER OF CLASSES

	Bayesian Network	NetSSM	Real
Domain Checks			
TCP handshake observed	0.94	**0.96**	**0.96**
No data before handshake	**1.00**	**1.00**	**1.00**
No duplicate handshake	0.89	**0.93**	**0.93**
TCP termination observed	0.96	**0.99**	**0.99**
No data after termination	0.57	0.82	**0.83**
F1 Macro			
CIC-IDS-2017 ($c = 14$)	0.29 ± 0.03	0.72 ± 0.05	$\mathbf{0.80 \pm 0.02}$
ISCX-VPN-2016 ($c = 7$)	0.25 ± 0.04	0.76 ± 0.08	$\mathbf{0.78 \pm 0.07}$

The results show that the generated flows from both models mostly comply with the patterns of typical TCP flows. Similar to the previous test, the generated flows of the BN sometimes continue even after a TCP termination is observed. The scores of NetSSM are close to the real data.

Traffic Classification Utility: A primary use case of the synthetic data is for training of ML models on downstream task. To evaluate whether the generated flows suffice as training data, we compare the performance of Random Forest flow classifiers after supervised trained on the generated/real data. For each flow, we extracted a feature vector of 22 common statistical values [10] (e.g., number of packets send) and use their seed packet's protocol/appl. labels (see **Setup**) as targets.

The results (Table II) show a decline in classifier performance for training on the generated flows, especially for BN-generated flows, but only slightly for NetSSM's flows.

IV. Conclusion

In this paper we presented a Bayesian network approach for network packet header synthesis, and an expert discrimination test to evaluate the authenticity of the generated flows by experts. While the BN-generated flows are more reliably identified than those from the byte-level generator, they are also found as real in 53% of the times, indicating that structural dependencies capture significant protocol semantics. Further, domain checks show a high compliance with TCP session behavior–something similar models struggle with. Despite a current decline in downstream classifier performance, the BN requires just 0.3% and 0.8% of the training and inference time of NetSSM, respectively. These results highlight a promising research direction; our future work will focus on narrowing the fidelity gap and evaluating the utility of BN-generated traffic for training ML-based network security tools.

ACKNOWLEDGMENT

This project was funded by German Federal Ministry of Research, Technology, and Space (BMFTR), Program "KI4KMU", Project "FlowLens" (ID: 01IS24017B).

REFERENCES

[1] Andrew Chu, Xi Jiang, Shinan Liu, Arjun Bhagoji, Francesco Bronzino, Paul Schmitt, and Nick Feamster. Netssm: Multi-flow and state-aware network trace generation using state-space models. *arXiv preprint arXiv:2503.22663*, 2025.

[2] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024.

[3] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of encrypted and vpn traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, pages 407–414, 2016.

[4] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[5] Jorge Luis Guerra, Carlos Catania, and Eduardo Veas. Datasets are not enough: Challenges in labeling network traffic. *Computers & Security*, 120:102810, 2022.

[6] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. Characterization of tor traffic using time based features. In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - ICISSP*, pages 253–262. INSTICC, SciTePress, 2017.

[7] Jonas Höchst, Lars Baumgärtner, Matthias Hollick, and Bernd Freisleben. Unsupervised traffic flow classification using a neural autoencoder. In *2017 IEEE 42Nd Conference on local computer networks (LCN)*, pages 523–526. IEEE, 2017.

[8] Xuying Meng, Chungang Lin, Yequan Wang, and Yujun Zhang. Netgpt: Generative pretrained transformer for network traffic. *arXiv preprint arXiv:2304.09513*, 2023.

[9] Jian Qu, Xiaobo Ma, and Jianfeng Li. Trafficgpt: Breaking the token barrier for efficient long traffic analysis and generation. *arXiv preprint arXiv:2403.05822*, 2024.

[10] Ola Salman, Imad H Elhajj, Ayman Kayssi, and Ali Chehab. A review on machine learning–based approaches for internet traffic classification. *Annals of Telecommunications*, 75(11):673–710, 2020.

[11] Adrien Schoen, Gregory Blanc, Pierre-François Gimenez, Yufei Han, Frédéric Majorczyk, and Ludovic Me. A tale of two methods: unveiling the limitations of gan and the rise of bayesian networks for synthetic network traffic generation. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 273–286. IEEE, 2024.

[12] Iman Sharafaldin, Arash Habibi Lashkari, Ali A Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1(2018):108–116, 2018.

[13] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012.

[14] Stanislav Špaček, Petr Velan, Pavel Čeleda, and Daniel Tovarňák. Encrypted web traffic dataset: Event logs and packet traces. *Data in Brief*, 42:108188, 2022.

[15] Zahra Taghiyarrenani and Hamed Farsi. Domain adaptation with maximum margin criterion with application to network traffic classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 159–169. Springer, 2022.

[16] The Wireshark Foundation. Wireshark, 2026. Network protocol analyzer.

[17] UNIBS. Available traces with associated ground truth. http://netweb.ing.unibs.it/~ntw/tools/traces/. Accessed: 2025-10-23.

[18] Qineng Wang, Chen Qian, Xiaochang Li, Ziyu Yao, Gang Zhou, and Huajie Shao. Lens: A foundation model for network traffic. *arXiv preprint arXiv:2402.03646*, 2024.

[19] Yucheng Yin, Zinan Lin, Minhao Jin, Giulia Fanti, and Vyas Sekar. Practical gan-based synthetic ip header trace generation using netshare. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 458–472, 2022.

[20] Shiyuan Zhang, Tong Li, Depeng Jin, and Yong Li. Netdiff: a service-guided hierarchical diffusion model for network flow trace generation. *Proceedings of the ACM on Networking*, 2(CoNEXT3):1–21, 2024.

Machine Learning-Enhanced Parent Node Prediction in Software-Defined Wireless Sensor Networks

Ahmed Nader al-Dulaimy
Institute for Computer Science
University of Koblenz
Koblenz, Germany
Email: aldulaimy@uni-koblenz.de

Prof. Dr. Hannes Frey
Institute for Computer Science
University of Koblenz
Koblenz, Germany
Email: frey@uni-koblenz.de

***Abstract*—Node mobility management in wireless sensor networks (WSNs) presents significant challenges due to dynamic environments and strict resource constraints. This study introduces a machine learning-enhanced framework for Software-Defined WSNs (SDWSNs) that predicts optimal parent node transitions in dynamic tree topologies. The integration of XGBoost classification with hierarchical IPv6 subnetting enables proactive and efficient topology management. OMNeT++ simulations demonstrate substantial improvements over existing approaches, including 94.20% prediction accuracy, an 89.78% F1-score, a 1.76 percentage point increase in packet delivery ratio, a 73.6% reduction in latency, a 7.4% extension in network lifetime, and an 32.2% decrease in overhead compared to reactive baselines.**

***Index Terms*—Machine Learning, SDN, WSN, XGBoost, Parent Node Prediction**

I. Introduction

Software-Defined Networking (SDN) decouples the control plane from the data plane, thereby centralizing network intelligence and streamlining management [1]. In the context of wireless sensor networks (WSNs), SDN supports efficient data aggregation at the controller, which in turn enables practical integration of machine learning for diverse applications, including the Internet of Things (IoT), smart cities, and industrial monitoring [2] [3].

However, deploying SDN in wireless environments necessitates careful evaluation of the trade-offs between its benefits and the operational overhead of topology maintenance. In dynamic WSNs, frequent parent node changes due to mobility highlight the limitations of reactive strategies, including delayed detection and energy-intensive route discovery. To address these challenges, this study integrates machine learning-based predictions with hierarchical IPv6 subnetting to enable proactive parent node transitions before link failures occur.

The primary contributions of this paper are as follows: (1) development of an XGBoost-based prediction model for parent node transitions achieving 94.20% accuracy; (2) implementation of IPv6-based hierarchical subnetting to support scalable and efficient topology management; and (3) comprehensive evaluation demonstrating significant improvements in packet delivery ratio (PDR), latency, energy efficiency, and overhead reduction.

II. System Architecture

A. SDN-WiSE Topology Discovery

This work adopts the SDN-WiSE topology discovery (TD) protocol [4], which facilitates the exchange of TD packets containing information on battery level and hop distance. Nodes assess potential parent nodes using three criteria: (1) hop count, (2) received signal strength indicator (RSSI), and (3) residual battery energy. By maintaining updated neighbor lists, the protocol establishes a robust foundation for the controller's network-wide perspective and informed predictions.

B. IPv6-Based Hierarchical Subnetting

Building on Sub-WSN [1], our approach implements subnetting directly at the IPv6 network layer. The hierarchical segmentation of the network positions the sink as the primary subnet parent, with cluster heads overseeing subordinate subnets. Each node stores a flow table populated with IPv6-based forwarding rules issued by the controller. Leveraging Dijkstra's algorithm applied to topology data, the controller allocates addresses and subnet masks, while parent nodes supplement flow tables with entries for their respective children.

C. Mobility Management

Periodic TD packets support ongoing neighbor tracking and facilitate timely detection of disconnections. The controller oversees an IPv6 address inventory, deliberately reserving additional space to accommodate mobile nodes. Upon handover, the prospective parent assigns a new address from its available inventory. Notably, only the hierarchical segment of the address changes, allowing the node ID to remain unchanged and significantly reducing reconfiguration overhead [5].

III. ML-Based Parent Change Prediction

A. XGBoost Model

XGBoost is employed to perform binary classification (parent change: true or false), leveraging its effectiveness with tabular WSN data and its ability to model complex feature interactions. The model optimizes the following objective function [6]:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

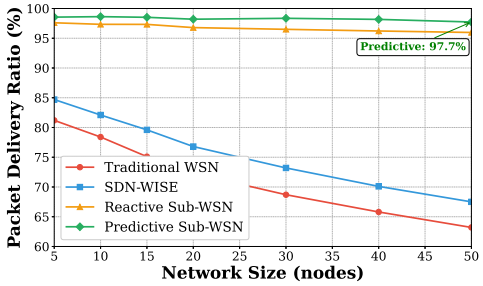

Fig. 1. Packet Delivery Ratio (PDR) for Predictive Sub-WSN, Reactive Sub-WSN, SDN-WISE, and Traditional WSN across network sizes of 5–50 nodes.

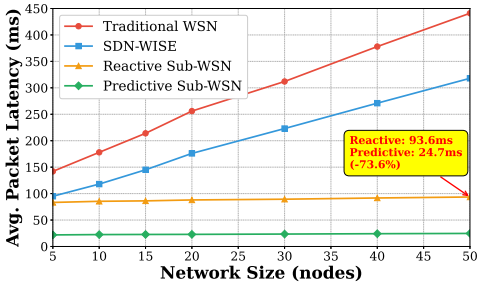

Fig. 2. End-to-end latency for Predictive Sub-WSN, Reactive Sub-WSN, SDN-WISE, and Traditional WSN (5–50 nodes).

where l is the logistic loss, $\hat{y}_i$ and y_i are predicted and true labels, and $\Omega(f_k)$ is the regularization term defined as: $\Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$ with complexity penalty γ, leaf count T, L2 parameter λ, and leaf weights w.

B. Feature Engineering

The feature set includes signal and temporal characteristics: **RSSI Statistics:** Mean RSSI $\overline{r}_i = \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} r_{ij}$, standard deviation σ_{r_i}, range (max-min RSSI), and current parent RSSI, where $\mathcal{N}_i$ denotes neighboring nodes and r_{ij} the RSSI between nodes i and j. **Temporal Features:** Parent change indicator c_i^t, recent change count $\sum_{k=1}^{3} c_i^{t-k}$, time since last change τ_i^t, and stability ratio $S_i^t = \frac{\tau_i^t}{\sum_{k=1}^{3} c_i^{t-k}+1}$.

IV. Evaluation

A. Simulation Setup

OMNeT++ simulations employed an IEEE 802.15.4 topology, scaling from 5 to 50 nodes at a target density of 8 nodes per 100×100 m area. Nodes exhibit mass mobility, with velocities sampled from a truncated normal distribution $s_i \sim \text{TruncNormal}(1.0, 0.5)$ m/s. The energy model initializes each node with 21,600 J and specifies consumption rates for transmission (60 mW), reception (10 mW), idle (2 mW), and sleep (1 mW) states. Propagation effects are simulated using log-normal shadowing with a path loss exponent of 2.5 [7].

B. ML Model Performance

The training process used 989,406 sequences collected from 99 nodes, partitioned chronologically into 60% for training, 20% for validation, and 20% for testing. The resulting model achieved 94.20% accuracy, an 89.78% F1-score, 91.38% precision, 88.23% recall, 95.96% ROC-AUC, and 93.85% PR-AUC on the test set.

C. Network Performance

The evaluation benchmarks the Predictive Sub-WSN against Reactive Sub-WSN [5], SDN-WiSE, and a traditional AODV-based WSN. For each network size, 100 simulation runs were conducted, each lasting 10 hours and using 30-second traffic intervals.

vs. SDN-WiSE (50 nodes): 30.2 pp PDR improvement (97.73% vs 67.5%, as depicted in Figure 1), 92.2% latency reduction (24.67ms vs 318ms, as shown in Figure 2), 35.6% network lifetime extension (72.12h vs 53.2h, see Table I), 66% overhead reduction (2.87 vs 8.45).

TABLE I
Network lifetime at 50 nodes comparing Predictive Sub-WSN, Reactive Sub-WSN, and SDN-WISE to Traditional WSN.

Approach	50 nodes	vs Traditional
Predictive Sub-WSN	72.12h	+46.9%
Reactive Sub-WSN	67.14h	+36.7%
SDN-WiSE	53.2h	+8.4%
Traditional WSN	49.1h	Baseline

vs. Reactive Sub-WSN (50 nodes): 1.76 pp PDR gain (97.73% vs 95.97%), 73.6% lower latency (24.67ms vs 93.57ms), 7.4% better lifetime (72.12h vs 67.14h), 32.2% overhead decrease (2.87 vs 4.23).

D. Scalability Analysis

Predictive Sub-WSN exhibits robust scalability. As the network grows from 5 to 50 nodes, the packet delivery ratio (PDR) remains high at 97.73%, declining by just 0.82 percentage points—half the reduction seen with the reactive approach. Latency rises slightly by 9.7% (from 21.88 ms to 24.67 ms). Notably, per-node energy consumption stays nearly constant, and protocol overhead remains low with a control-to-data ratio of 2.87.

This performance arises from several key factors: centralized machine learning enables global optimization; distributed execution with make-before-break handovers ensures seamless transitions; hierarchical addressing controls protocol overhead; and proactive adaptation closes 88.5% of connectivity gaps, as indicated by the 88.23% recall rate.

V. Conclusion

By combining machine learning with hierarchical SDN architecture, Predictive Sub-WSN addresses core wireless sensor network challenges through proactive topology management. The XGBoost model attains 94.20% accuracy and an 89.78% F1-score, delivering significant gains: 97.73% PDR, 24.67 ms latency at 50 nodes, and 72.12 hours network lifetime—all while sustaining scalability.

Future work should extend prediction horizons, model cascading topology shifts, incorporate temporal and environmental dynamics, and explore alternative machine learning architectures for online and adaptive learning.

REFERENCES

[1] A. N. al Dulaimy and H. Frey, "Subnet addressing in software defined wireless sensor networks," in *2019 12th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 2019, pp. 32–38.

[2] B. Gates, "Gates annual letter: Our big bet for the future," *Retrieved August*, vol. 15, p. 2016, 2015. [Online]. Available: https://www.gatesnotes.com/2015-annual-letter

[3] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the internet of things: A survey," in *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*. IEEE, 2011, pp. 1–6.

[4] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," *Proceedings - IEEE INFOCOM*, vol. 26, pp. 513–521, 2015.

[5] A. N. Al-Dulaimy and H. Frey, "Subneting software defined wireless sensor networks to support mobility," in *2021 13th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 2021, pp. 1–8.

[6] T. Chen, "Xgboost: A scalable tree boosting system," *Cornell University*, 2016.

[7] T. S. Rappaport, *Wireless communications: principles and practice*. Cambridge University Press, 2024.

Learning-Based Local Routing Decisions in Sparse Aeronautical Communication Networks

Musab Ahmed, Kevin Nabiev, Konrad Fuger, Koojana Kuladinithi, Andreas Timm-Giel
Hamburg University of Technology, Institute of Communication Networks, Hamburg, Germany
{musab.ahmed, kevin.nabiev, k.fuger, koojana.kuladinithi, timm-giel}@tuhh.de

***Abstract*—Geographic greedy routing provides the scalability required for future L-band Digital Aeronautical Communications System (LDACS) Air-to-Air (A2A) networks. Yet, its topology blindness becomes a critical limitation in sparse airspaces. This blindness induces a path-dependent process, recently characterized as the "memory effect," where suboptimal local choices steer packets toward dead ends multiple hops away. These failures are not arbitrary: they follow learnable patterns embedded in the local network structure. We propose a decentralized, density-adaptive routing scheme that replaces standard geographic progress toward the destination with learned neighbor ranking. The model is trained using the *Topological Advance (TA)* metric as a ground-truth label to indicate whether a chosen next-hop reduces the shortest-path hop distance to the destination. By predicting TA from three-hop local features, nodes improve routing reliability. Evaluations in realistic French airspace scenarios show a success ratio exceeding 0.93 over topologically connected nodes in sparse networks, a 33% improvement over traditional greedy routing, while maintaining a near-optimal hop stretch of 1.03.**

***Index Terms*—LDACS, Air-to-Air communication, geographic greedy routing, betweenness centrality, machine learning.**

I. Introduction

Modernizing air traffic management to handle projected density growth requires L-band Digital Aeronautical Communications System (LDACS), which is currently standardized to replace legacy analog systems with Air-to-Ground (A2G) and Air-to-Air (A2A) links [1–3]. For A2A multi-hop communication, geographic greedy routing (Greedy-1), which forwards to the direct neighbor closest to the destination, is attractive for its scalability [4, 5]. Yet, it fails in sparse airspaces (fewer than ten neighbors) even when topological paths exist [5]. While traditionally attributed to a local minimum, where no reachable neighbor is closer to the destination, this failure is now understood as a path dependency termed the "memory effect" [6]. As shown in Figure 1, a sequence of locally greedy choices can constrain future options, making the eventual local minimum a late symptom of a suboptimal trajectory. Increasing the neighborhood radius (Greedy-k) does not resolve this, as the protocol remains blind to underlying topological constraints [6]. However, the structural nature of this path dependency implies that local routing decisions follow predictable patterns that can be learned.

Key challenges remain when applying machine learning to the routing problem in sparse scenarios. Prior methods often target dense scenarios to optimize delay [7–9] or assume impractical global topology state [10, 11]. Others attempt trap escape reactively after a packet is already stuck [12], a stage where recovery is difficult in sparse networks. A proactive, topology-aware policy for sparse scenarios is currently missing. We bridge this gap with a decentralized, density-adaptive neighbor ranker. Our approach utilizes three-hop local features and is supervised by Topological Advance (TA) metric, a measure of hop-distance progress toward the destination. This enables forwarding decisions that proactively avoid path-dependent traps. Our main contributions are:

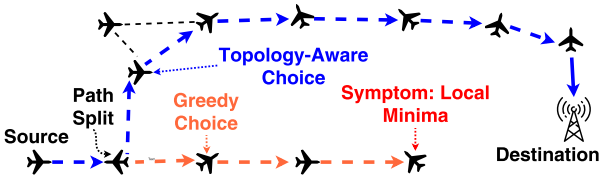

Fig. 1: Memory effect.

- Application of the TA metric as a ground-truth label for neighbor ranking based on shortest-path progress.
- A density-adaptive neighbor-ranking policy for proactive next-hop selection without global topology state.
- A performance evaluation in realistic French airspace scenarios representing different LDACS deployment stages.

II. Ranking-based Framework

The proposed scheme transforms local three-hop topology into a ranking task to select neighbors offering the maximum predicted TA toward the destination. If a topological path exists, at least one immediate neighbor lies on a shortest path toward the destination and thus yields TA. To avoid path-dependent traps, each node constructs a local representation using its three-hop neighborhood. Input features include geometric metrics (e.g., distance and angle to the destination) calculated by each node for its neighbors and structural metrics (e.g., degree and Betweenness Centrality (BC)) calculated by nodes and shared with their neighbors. These structural indicators identify "bridge nodes" that, while geographically distant, are topologically critical for routing. We assume full three-hop neighborhood visibility for BC calculations, which could be obtained and maintained via periodic beacons. Prior research shows this is feasible within LDACS beacon size constraints [13].

Given that routing dynamics differ significantly across network densities, the routing scheme employs a two-phase architecture. First, a lightweight classifier evaluates local structural features to determine the current density condition, using a

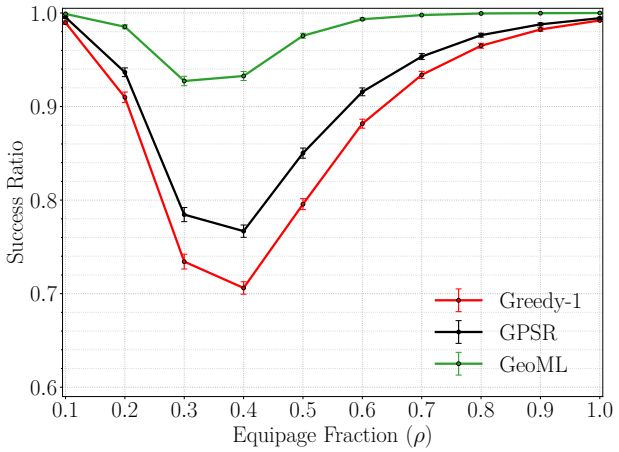

Fig. 2: Average success ratio with varying equipage fraction.

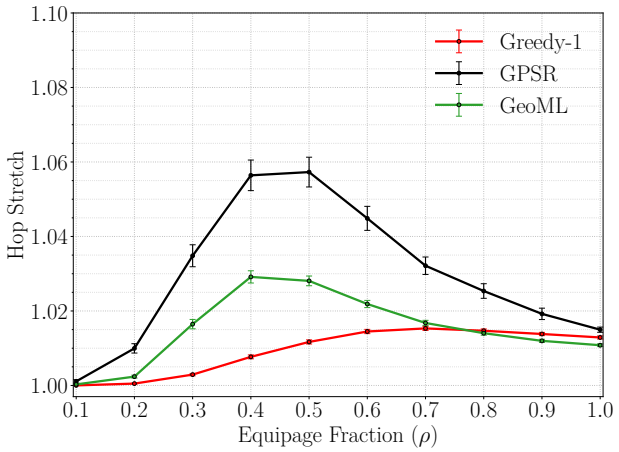

Fig. 3: Average hop stretch with varying equipage fraction.

threshold of ten neighbors to distinguish between sparse and dense environments. Based on this classification, the system selects a specialized ranking model optimized for that density condition. Next-hop selection is formulated as a ranking problem rather than simple classification, treating the decision as a preference task. This allows the model to generalize across varying topologies by ordering neighbors based on their predicted topological advance score. During training, the two rankers are supervised using binary topological advance labels derived from global shortest-path information, where a label of 1 is assigned to neighbors that decrease the shortest-path hop distance to the destination and 0 otherwise. At runtime, the node feeds its current three-hop features into the active ranker to score all candidates. The neighbor with the highest score is selected as the next-hop, promoting hop-by-hop topological progress and bridging the gap between decentralized local heuristics and global shortest-path routing.

III. Results and Discussion

The proposed routing scheme, denoted as GeoML, employs Light Gradient Boosting Machine (LightGBM) [14] for density classification and neighbor ranking. Evaluation is performed on 2000 French airspace test snapshots not used in training (trained on 150 snapshots). Monte Carlo simulations feature 500 aircraft communicating with a ground station. GeoML is benchmarked against Greedy-1 and Greedy Perimeter Stateless Routing (GPSR) (Greedy-1 with perimeter recovery). The equipage fraction (ρ) represents the percentage of aircraft with LDACS capability, where $\rho = 0.4$ corresponds to initial sparse deployment and $\rho = 0.8$ to network maturity. The density classifier labels scenarios with $\rho \geq 0.5$ (average degree $\geq$ 10) as dense. Performance metrics, reported at 95% confidence intervals, are: the *success ratio* (for a given protocol, the fraction of topologically connected sources that reach the destination) and the *hop stretch factor* (the ratio of the protocol's path length to the global shortest-path length).

At $\rho = 0.4$, Greedy-1 and GPSR success ratios drop to approximately 70% and 77%, respectively (Figure 2). This degradation highlights the fundamental limitation of distance-based heuristics in sparse A2A topologies, consistent with the path dependency discussed in Section I. The $\rho = 0.4$ scenarios are particularly challenging: the network remains connected, yet topological traps are frequent. In contrast, GeoML maintains a success ratio above 93%. Unlike reactive methods like GPSR, GeoML utilizes local three-hop neighborhood structure to identify and avoid routing traps before reaching local minima. The hop stretch factor results show that while GPSR provides a 0.07 absolute increase in success ratio over Greedy-1, it inflates hop stretch factor to around 1.06 due to perimeter detours (Figure 3). GeoML maintains high reliability with a stretch of only 1.03. The lower stretch observed for Greedy-1 in sparse scenarios reflects survivorship bias, as it often fails on complex paths. As the network matures ($\rho > 0.8$), success ratios converge. Furthermore, GeoML slightly outperforms legacy baselines in the hop stretch factor, demonstrating effective adaptation across density scenarios.

IV. Conclusion

In sparse A2A topologies, geographic greedy routing fails due to path-dependent memory effects. The frequently encountered local minima (voids) are often late symptoms of earlier suboptimal decisions rather than isolated local accidents. We introduced a density-adaptive neighbor ranking policy that replaces greedy geographic progress heuristics with topology-aware selection using only local three-hop information. By training with the TA metric as ground truth, nodes learn to proactively avoid routing traps from local observations without global topology knowledge. Evaluated on realistic French airspace snapshots, our GeoML achieves a success ratio exceeding 93% over connected sources with a hop stretch factor of only 1.03. These results outperform both Greedy-1 and GPSR without incurring reactive detours. Grounded in the path-dependency analysis in [6], these findings suggest a practical route toward reliable LDACS A2A networking, particularly in critical early deployment phases where sparse connectivity challenges reliability. Future work will extend the evaluation to additional airspaces to further assess model generalization under different mobility patterns.

REFERENCES

[1] Miguel A. Bellido-Manganell and Michael Schnell. "Towards Modern Air-to-Air Communications: The LDACS A2A Mode". In: *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. Sept. 2019, pp. 1–10. DOI: 10.1109/DASC43569.2019.9081678.

[2] Eurocontrol. *Aviation Outlook 2050*. Main report. Apr. 13, 2022.

[3] Nils Mäurer, Thomas Gräupl, and Corinna Schmitt. *L-Band Digital Aeronautical Communications System (LDACS)*. Request for Comments RFC 9372. Internet Engineering Task Force, Mar. 2023. 35 pp. DOI: 10.17487/RFC9372.

[4] Daniel Medina et al. "A Geographic Routing Strategy for North Atlantic In-Flight Internet Access Via Airborne Mesh Networking". In: *IEEE/ACM Transactions on Networking* 20.4 (Aug. 2012), pp. 1231–1244. DOI: 10.1109/TNET.2011.2175487.

[5] Musab Ahmed et al. "Enhancing Geographic Greedy Routing in Sparse LDACS Air-to-Air Networks through k-Hop Neighborhood Exploitation". In: *2024 IEEE 49th Conference on Local Computer Networks (LCN)*. Oct. 2024, pp. 1–7. DOI: 10.1109/LCN60385.2024.10639650.

[6] Musab Ahmed et al. "Modeling of Geographic Greedy Routing in Sparse LDACS Air-to-Air Networks Using Absorbing Markov Chains". In: *2025 IEEE 50th Conference on Local Computer Networks (LCN)*. Oct. 2025, pp. 1–9. DOI: 10.1109/LCN65610.2025.11146308.

[7] Zhigang Jin, Qinyi Zhao, and Yishan Su. "RCAR: A Reinforcement-Learning-Based Routing Protocol for Congestion-Avoided Underwater Acoustic Sensor Networks". In: *IEEE Sensors Journal* 19.22 (Nov. 2019), pp. 10881–10891. DOI: 10.1109/JSEN.2019.2932126.

[8] Dong Liu et al. "Deep-Learning-Aided Packet Routing in Aeronautical Ad Hoc Networks Relying on Real Flight Data: From Single-Objective to Near-Pareto Multiobjective Optimization". In: *IEEE Internet of Things Journal* 9.6 (Mar. 2022), pp. 4598–4614. DOI: 10.1109/JIOT.2021.3105357.

[9] Jianmin Liu, Dan Li, and Yongjun Xu. "Deep Distributional Reinforcement Learning-Based Adaptive Routing With Guaranteed Delay Bounds". In: *IEEE/ACM Transactions on Networking* 32.6 (Dec. 2024), pp. 4692–4706. DOI: 10.1109/TNET.2024.3425652.

[10] Shanshan Jiang, Zhitong Huang, and Yuefeng Ji. "Adaptive UAV-Assisted Geographic Routing With Q-Learning in VANET". In: *IEEE Communications Letters* 25.4 (Apr. 2021), pp. 1358–1362. DOI: 10.1109/LCOMM.2020.3048250.

[11] Yaqi Ke et al. "Distributed Routing Optimization Algorithm for FANET Based on Multiagent Reinforcement Learning". In: *IEEE Sensors Journal* 24.15 (Aug. 2024), pp. 24851–24864. DOI: 10.1109/JSEN.2024.3415127.

[12] Chi Wei et al. "QFAGR: A Q-learning-based Fast Adaptive Geographic Routing Protocol for Flying Ad Hoc Networks". In: *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*. Dec. 2023, pp. 4613–4618. DOI: 10.1109/GLOBECOM54140.2023.10437104.

[13] Musab Ahmed et al. "Poster: Can Betweenness Centrality Mitigate Greedy Routing's Sparse Network Problem?" In: *2025 IEEE Vehicular Networking Conference (VNC)*. June 2025, pp. 1–2. DOI: 10.1109/VNC64509.2025.11054179.

[14] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

Can Neural Networks Provide Latent Embeddings for Telemetry-Aware Greedy Routing?

Andreas Boltres*†, Niklas Freymuth* and Gerhard Neumann*
* Autonomous Learning Robots, Karlsruhe Institute of Technology, Germany
† SAP SE, Walldorf, Germany

***Abstract*—Telemetry-Aware routing promises to increase efficacy and responsiveness to traffic surges in computer networks. Recent research leverages Machine Learning to deal with the complex dependency between network state and routing, but sacrifices explainability of routing decisions due to the black-box nature of the proposed neural routing modules. We propose *Placer*, a novel algorithm using Message Passing Networks to transform network states into latent node embeddings. These embeddings facilitate quick greedy next-hop routing without directly solving the all-pairs shortest paths problem, and let us visualize how certain network events shape routing decisions.**

***Index Terms*—routing, machine learning, ns-3, latent embedding, graph neural network**

I. Introduction

In computer networks, routing denotes the task of selecting paths along which data packets are forwarded. A good routing mechanism is essential for an efficient use of the given infrastructure. Routing optimization has been a long-standing subject of study because traffic often fluctuates with unpredictable patterns [2, 27], influencing the optimal routing solution. Recent work has suggested that routing decisions be based on live telemetry data [26, 29] and to re-optimize routes within milliseconds [11, 13] to improve routing responsiveness. Yet, for existing methods, the complex, non-linear dependency between high-dimensional network state and suitable routing becomes challenging to navigate, especially as networks increase in scale [30]. Leveraging Machine Learning (ML) for network routing has been a subject of study for decades [7] that has seen differing problem formulations, network types, use cases and goals. Current research derives splitting ratios for precomputed sets of shortest paths [28, 31] from traffic aggregated over minutes to hours, and infers routes directly for individual packets [22] and packet flows [33]. Related work constructs node-level network embeddings that facilitate greedy routing by minimizing the remaining distance to the destination in the embedding space [19, 17, 4]. With a suitable network embedding, these approaches promise competitive routing without the computational cost of computing shortest paths. While existing work uses ML to obtain node-level representations from IP network information [20], none have yet used ML to turn network state graphs containing telemetry data into node embeddings for greedy routing.

This research work was carried out as part of the project "Apeiro Reference Architecture", in short ApeiroRA, under the funding ID 13IPC007, financed by the European Union – NextGenerationEU and funded by the German Federal Ministry for Economic Affairs and Energy.

Correspondence to: Andreas Boltres <andreas.boltres@partner.kit.edu>

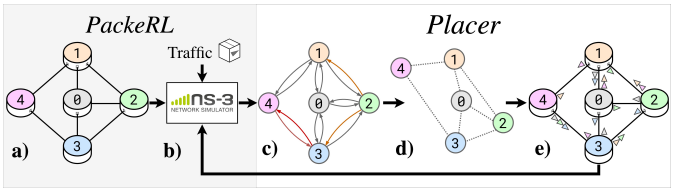

Fig. 1: **a)**: Five-node network topology *mini-5* used in our experiments. **b, c)**: *PackeRL*'s simulation backend turns network topology, incoming traffic and routing actions into an attributed state graph. **d, e)**: *Placer* obtains latent node embeddings using its Message Passing Network, then greedily converts these into next-hop selections.

We propose to connect both aspects, providing telemetry-aware embeddings for IP routing in milliseconds. Our approach, *Placer*, uses Message Passing Networks (MPNs) [12] to obtain latent Euclidean node embeddings from telemetry-infused network state graphs. To obtain informative network states within realistic network environments, we use *PackeRL* [6], a training and evaluation framework for sub-second telemetry-aware routing built on top of the packet-level network simulator *ns-3* [14] and its capabilities for in-band network telemetry [16].

II. Problem Setting

We follow Boltres et al. [6] and view routing in IP networks as a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r \rangle$. States $S_t = (V_t, E_t, \mathbb{X}_{V_t,t}, \mathbb{X}_{E_t,t}, \mathbf{x}_{u,t}) \in \mathcal{S}$ are attributed directed graphs that hold topology information like link data rate or packet buffer size, as well as telemetry data obtained via *ns-3*'s *FlowMonitor* [8]. The action $\mathbf{a}_t = \{(u, z) \mapsto v \mid u, v, z \in V_t, v \in \mathcal{N}_u\} \in \mathcal{A}$ selects next-hop neighbors per routing node $u \in V_t$ for each possible destination node $z \in V_t$. The transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ evolves the current network state using the latest routing actions. In *PackeRL*, this means installing the actions and invoking *ns-3*'s simulator for a predetermined duration. The global reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assesses a routing action in the given network state. We use the global goodput, measured as the sum of MB received per step across all nodes of the topology. We train a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ to maximize the expected discounted cumulative future reward $J_t := \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})} \left[\sum_{k=0}^{\infty} \gamma^k r(\mathbf{s}_{t+k}, \mathbf{a}_{t+k}) \right]$ [25], and thus the long-term global goodput.

III. From Node Embeddings to Greedy Routing

Placer consists of two phases: First, it uses a four-layer MPN with a hidden dimension of 32 to obtain latent node embeddings $\mathbf{x}_i \in \mathbb{R}^d$ of configurable output dimensionality d from the current network state S_t. Then, it decomposes each embedding $\mathbf{x}_i$ into its radius $r_i = ||\mathbf{x}_i||$ and unit direction $u_i = \frac{\mathbf{x}_i}{r_i}$, and computes pairwise squared distances $\Delta^2 \in [0,4)^{|V|\times|V|}$ as $\Delta^2_{ij} = \rho_i^2 + \rho_j^2 - 2\rho_i\rho_j(u_i^\top u_j)$, using the law of cosines in polar form with softly constrained radii $\rho_i = \tanh(r_i) \in [0,1)$. For any given pair of routing node u and destination node z, greedy routing in the embedding space chooses the next-hop node $v = \text{argmin}_{v' \in \mathcal{N}_u} \Delta(v', z)$ that minimizes the remaining distance to z.

To train *Placer*, we use the Proximal Policy Optimization (PPO) [24] implementation and hyperparameters of Boltres et al. [6]. We implement *Boltzmann exploration* [15] by treating the values of Δ^2 as negative logits for $|V|^2$ next-hop sampling distributions, one per pair of (u, z).

IV. Experiments and Results

We use the episodic experiment setup of Boltres et al. [6] and a five-node network topology shown in Figure 1**a)** which we call *mini-5*. We simulate $H = 400$ steps of 5 ms each per episode, injecting synthetic TCP and UDP flows which, in total, exceed the network's capacity. Flow sizes and arrival times are generated akin to Boltres et al. [6], following distributions found in real-world data centers [3]. For *Placer* (ablated with $d = i \in \{1, 2, 32\}$ and written as $\textit{Placer}_{d=i}$) and the baseline algorithm *M-Slim* [6], we use 40 PPO iterations of 16 episodes, each providing 4 unique traffic sequences repeated 4 times. We report the interquartile mean over 8 random seeds [1], with performance averaged over 30 unseen traffic sequences. Lastly, we use the routing scheme of Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP) [23] as a telemetry-oblivious baseline which, by default, computes shortest paths from link data rate and delay values.

	EIGRP	***M-Slim***	***Placer***		
			$d=1$	$d=2$	$d=32$
Goodput (MB, ↑)	209.2	220.45	236.9	237.0	**237.3**
Avg. Delay (ms, ↓)	9.99	**9.59**	9.93	9.92	9.93
Dropped (%, ↓)	0.47	2.92	0.36	0.35	**0.30**
Fluctuation (%)	0.0	22.80	0.71	0.58	0.06

TABLE I: Mean routing performance over 30 evaluation episodes on *mini-5*. Arrows denote whether lower (↓) or higher (↑) values are better for the respective metric.

Table I shows the results of the aforementioned experiment for the total goodput per episode, the average packet delay, and the percentage of dropped data. Additionally, the "Fluctuation" row denotes the percentage of changes in next-hop decisions between consecutive time steps. Both *M-Slim* and *Placer* notably increase the episodic goodput over the telemetry-oblivious EIGRP baseline. *M-Slim* also notably reduces packet latency, however at the expense of more packet drops. In contrast, *Placer* shows the highest goodput rates and the lowest

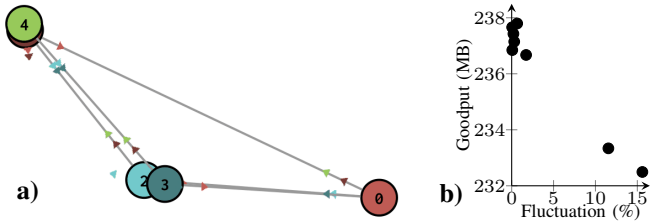

Fig. 2: **a)** An embedding of *mini-5* produced by $\textit{Placer}_{d=2}$. **b)** Goodput vs. fluctuation of $\textit{Placer}_{d=2}$ for 8 random seeds.

drop percentages overall, with packet latencies comparable to EIGRP. While its performance correlates weakly with increasing d, interestingly, $\textit{Placer}_{d=1}$ appears to represent *mini-5* well enough for competitive routing performance.

Despite the promising goodput values for *Placer*, the "Fluctuation" row of Table I and the contents of Figure 2 hint at a caveat: *Placer* converges to a practically telemetry-oblivious embedding of *mini-5*, as indicated by the low fluctuation values. Illustrating the embeddings for $\textit{Placer}_{d=2}$, as shown in Figure 2**a)**, is of limited use to human experts, as telemetry data only leads to minimal and inconsequential embedding changes. Moreover, Figure 2**b)** reveals that, for $\textit{Placer}_{d=2}$, those random seeds that resulted in more dynamic node embeddings also resulted in much lower goodput performance. We conjecture that a possible reason for the quasi-static embeddings is *Placer*'s centralized single-agent inference. It imposes a symmetry on the routing actions, whereas telemetry data may depict asymmetric utilization patterns of links and packet buffers that call for asymmetric routing choices, i.e., different links traversed from u to z than from z to u.

V. Conclusion and Future Work

We propose *Placer*, a routing algorithm that uses Message Passing Networks (MPNs) to turn telemetry-infused network states into latent node embeddings for greedy routing. A fully developed algorithm of this kind could provide sub-second routing responsiveness without relying on shortest-path computations. Additionally, low-dimensional network embeddings in particular could be visualized to help explain the routing decisions made by neural networks. First experiments with *Placer* show that some geometric structure can be extracted, resulting in competitive routing for a five-node network topology example. Yet, more work is needed to fully understand the problem structure, as changes in telemetry data do not yet result in consequential shifts of the node embeddings. This may be a consequence of the current centralized inference design, which may be incapable of providing the asymmetric routing choices required to handle asymmetric network load. Future work may obtain more meaningful node embeddings by deploying *Placer* locally at each node and, perhaps, training it using multi-agent variants of Proximal Policy Optimization (PPO) [10, 32]. Finally, as many larger networks are more efficiently embedded in hyperbolic spaces [18, 5], we suggest that the Euclidean MPN used by *Placer* may be replaced with a hyperbolic variant [21, 9].

References

[1] Rishabh Agarwal et al. "Deep Reinforcement Learning at the Edge of the Statistical Precipice". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021, pp. 29304–29320.

[2] Mohammad Alizadeh et al. "CONGA: Distributed Congestion-Aware Load Balancing for Datacenters". In: *ACM SIGCOMM Conference 2014*. New York, NY, USA: Association for Computing Machinery, Aug. 2014, pp. 503–514.

[3] Theophilus Benson, Aditya Akella, and David A Maltz. "Network Traffic Characteristics of Data Centers in the Wild". In: *ACM SIGCOMM Internet Measurement Conference (IMC'10)*. 2010, pp. 267–280.

[4] Thomas Bläsius et al. "Hyperbolic Embeddings for Near-Optimal Greedy Routing". In: *ACM J. Exp. Algorithmics* 25 (Mar. 2020), 1.3:1–1.3:18.

[5] Marián Boguñá, Fragkiskos Papadopoulos, and Dmitri Krioukov. "Sustaining the Internet with Hyperbolic Mapping". In: *Nature Communications* 1.1 (Sept. 2010), p. 62.

[6] Andreas Boltres et al. "Learning Sub-Second Routing Optimization in Computer Networks requires Packet-Level Dynamics". In: *Transactions on Machine Learning Research* (2024).

[7] Justin Boyan and Michael Littman. "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach". In: *Advances in Neural Information Processing Systems 1993 (NeurIPS '93)*. Vol. 6. Morgan-Kaufmann, 1993.

[8] Gustavo Carneiro, Pedro Fortuna, and Manuel Ricardo. "FlowMonitor: A Network Monitoring Framework for the Network Simulator 3 (NS-3)". In: *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*. VALUETOOLS '09. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Oct. 2009, pp. 1–10.

[9] Ines Chami et al. "Hyperbolic Graph Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[10] Christian Schroeder de Witt et al. *Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?* Nov. 2020.

[11] Steven Gay, Renaud Hartert, and Stefano Vissicchio. "Expect the Unexpected: Sub-second Optimization for Segment Routing". In: *IEEE Conference on Computer Communications (INFOCOM 2017)*. May 2017, pp. 1–9.

[12] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *34th International Conference on Machine Learning (ICML 2017)*. PMLR. 2017, pp. 1263–1272.

[13] Fei Gui et al. "RedTE: Mitigating Subsecond Traffic Bursts with Real-time and Distributed Traffic Engineering". In: *Proceedings of the ACM SIGCOMM 2024 Conference*. ACM SIGCOMM '24. New York, NY, USA: Association for Computing Machinery, Aug. 2024, pp. 71–85.

[14] Thomas R. Henderson, Mathieu Lacage, and George F. Riley. "Network Simulations with the ns-3 Simulator". In: *SIGCOMM Demonstration* 14.14 (2008), p. 527.

[15] L. P. Kaelbling, M. L. Littman, and A. W. Moore. "Reinforcement Learning: A Survey". In: *Journal of Artificial Intelligence Research* 4 (May 1996), pp. 237–285.

[16] Changhoon Kim et al. "In-band Network Telemetry via Programmable Dataplanes". In: *ACM SIGCOMM Conference 2015*. Vol. 15. 2015, pp. 1–2.

[17] R. Kleinberg. "Geographic Routing Using Hyperbolic Space". In: *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. May 2007, pp. 1902–1909.

[18] Dmitri Krioukov et al. "Greedy Forwarding in Dynamic Scale-Free Networks Embedded in Hyperbolic Metric Spaces". In: *2010 Proceedings IEEE INFOCOM*. Mar. 2010, pp. 1–9.

[19] Fabian Kuhn et al. "Geometric Ad-Hoc Routing: Of Theory and Practice". In: *Proceedings of the Twenty-Second Annual Symposium on Principles of Distributed Computing*. Boston Massachusetts: ACM, July 2003, pp. 63–72.

[20] Mingda Li et al. "Deep Learning IP Network Representations". In: *Proceedings of the 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*. Big-DAMA '18. New York, NY, USA: Association for Computing Machinery, Aug. 2018, pp. 33–39.

[21] Qi Liu, Maximilian Nickel, and Douwe Kiela. "Hyperbolic Graph Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[22] Xuan Mai, Quanzhi Fu, and Yi Chen. "Packet Routing with Graph Attention Multi-Agent Reinforcement Learning". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. Dec. 2021, pp. 1–6.

[23] Donnie Savage et al. *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*. Request for Comments RFC 7868. Internet Engineering Task Force, May 2016.

[24] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv:1707.06347* (2017).

[25] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. The MIT Press, 2018.

[26] Belma Turkovic et al. "Fast Network Congestion Detection and Avoidance Using P4". In: *Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies*. NEAT '18. New York, NY, USA: Association for Computing Machinery, Aug. 2018, pp. 45–51.

[27] Patrick Wendell and Michael J. Freedman. "Going Viral: Flash Crowds in an Open CDN". In: *ACM SIGCOMM Internet Measurement Conference (IMC'11)*. 2011, pp. 549–558.

[28] Yang Xiao et al. "Leveraging Deep Reinforcement Learning for Traffic Engineering: A Survey". In: *IEEE Communications Surveys & Tutorials* 23.4 (2021), pp. 2064–2097.

[29] Shanghan Xie et al. "A Delay-guaranteed Routing Mechanism Based on Active Network Telemetry in Deterministic Network". In: *Proceedings of the 2021 ACM International Conference on Intelligent Computing and Its Emerging Applications*. ACM ICEA '21. New York, NY, USA: Association for Computing Machinery, Jan. 2022, pp. 217–222.

[30] Dahai Xu, Mung Chiang, and Jennifer Rexford. "Link-State Routing With Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering". In: *IEEE/ACM Transactions on Networking* 19.6 (Dec. 2011), pp. 1717–1730.

[31] Zhiying Xu et al. "Teal: Learning-Accelerated Optimization of WAN Traffic Engineering". In: *ACM SIGCOMM Conference 2023*. New York, NY, USA: Association for Computing Machinery, Sept. 2023, pp. 378–393.

[32] Chao Yu et al. "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games". In: *Advances in Neural Information Processing Systems* 35 (Dec. 2022), pp. 24611–24624.

[33] Lianming Zhang et al. "Flowlet-Level Routing Optimization with GNN-Based Multi-Agent Deep Reinforcement Learning". In: *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*. Dec. 2023, pp. 5214–5219.

C3S: Towards Generalizing Coordinated Congestion Control Across Flows and Topologies

Michael König, Martina Zitterbart
Institute of Telematics, Karlsruhe Institute of Technology, Karlsruhe, Germany, {m.koenig, martina.zitterbart}@kit.edu

***Abstract*—We revisit C3, a centralized reinforcement learning–based approach to coordinated congestion control, and introduce C3S, a modular, permutation-invariant neural architecture that enables scalable guidance for an arbitrary number of flows. Our results demonstrate effective scaling and fair bandwidth sharing, while highlighting the importance of topology-aware aggregation for generalization to complex network settings.**

***Index Terms*—Congestion Control, Single Agent Reinforcement Learning (SARL), TCP**

I. Introduction

Traditional congestion control relies on distributed, end-to-end algorithms with limited visibility into the global network state, often leading to slow and suboptimal convergence.

In prior work, we proposed *Coordinated Congestion Control (*C3*)* [1], a hybrid approach that augments end-to-end control with a centralized reinforcement learning agent operating on a domain-wide view. Unlike approaches that deploy one agent per sender or per flow (e.g., [2, 3, 4]), C3 uses a single agent per domain to guide flows toward globally efficient operating points.

Our evaluation showed that centralized guidance improves convergence time, latency, and fairness compared to unmodified TCP Cubic. However, scaling C3 to larger numbers of flows and more complex topologies requires rethinking its original fully connected neural architecture. To this end, we introduce C3-Scalable *(*C3S*)*, a scalable, modular, and permutation-invariant extension of C3 designed for efficient learning and control in large and heterogeneous network settings.

II. Background: C3 in a Nutshell

C3's existing setup employs a fully connected neural network and PPO [5] as the learning algorithm for its single, centralized agent.

The network's input layer consists of two feature groups: per-switch interface features (e.g., queue occupancy, link capacity) and per-flow features (e.g., RTT, loss rate). The sizes of these groups scale linearly with the number of switch interfaces and flows in the domain, respectively. The output layer contains N neurons, each producing a $cWnd_{target}$ value for one flow. This value is then used to adjust the $cWnd$ of the corresponding flow within a sender's end-to-end congestion control algorithm. The input and output are connected through two hidden layers, each consisting of 128 neurons with ReLU activations [6].

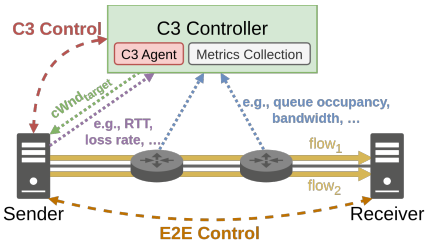

Fig. 1: C3 & C3S High-Level Architecture

Our evaluations [1] show that centralized guidance of end-to-end congestion control is feasible and effective, improving convergence speed, latency, and fairness, while remaining robust under conditions that are typically challenging for congestion control mechanisms (e.g., reverse-path congestion and heterogeneous buffer sizes).

III. Limitations of the C3's Neural Architecture

While these results demonstrate the feasibility of centralized guidance, the neural network architecture of the original proof-of-concept has two fundamental limitations.

A. Limited Scalability

The dimensionality of the input and output layers grows with the number of flows, significantly increasing training and inference costs. Moreover, each flow is processed by a separate set of parameters, which prevents weight sharing across semantically identical entities. This design makes the model sensitive to arbitrary input orderings, preventing weight sharing across identical entities and leading to inefficient learning.

However, flows and switch interfaces are interchangeable in this context: their identities carry no semantic meaning, and only their features are relevant. Consequently, their processing should be permutation-invariant and based on shared modules. This preserves symmetry among identical entities, reduces the parameter count, and enables scalable learning.

B. Topology Dependence

The fully connected design also limits generalization to more complex topologies. Because all flow and switch interface features are jointly processed, a flow's $cWnd_{target}$ can be influenced by irrelevant entities that do not share any bottleneck or path segment. For example, two flows within one pod of a fat-tree topology should not react to congestion in another pod. However, the current architecture lacks a mechanism to enforce such locality.

Therefore, the model must aggregate only features that can directly influence a given flow and must do so in a permutation-invariant manner.

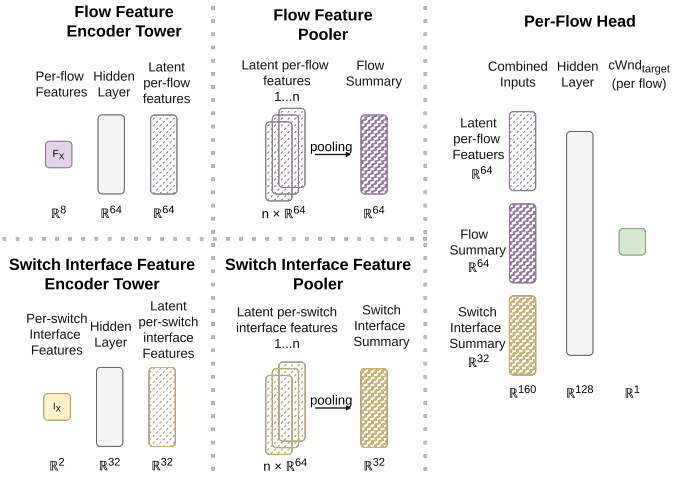

Fig. 2: C3S's new Neural Network Architecture

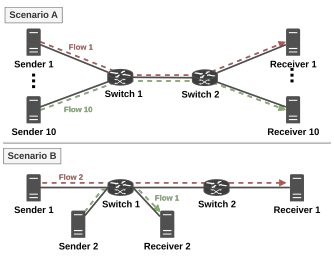

Fig. 3: Evaluation Scenario A & B

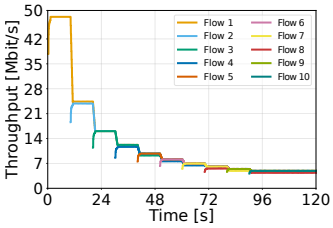

Fig. 4: Fair throughput sharing among an increased number of C3S-guided flows (Scenario A)

IV. C3-Scalable (C3S)

We introduce C3S, a new neural architecture that extends C3 with improved scalability while retaining proven components such as PPO and the existing reward formulation.

A. C3S *New Neural Network Architecture*

The C3S's architecture (cf. Fig. 2) consists of three components: two per-entity encoder towers, two permutation-invariant poolers, and one per-flow decision head.

1) Per-Flow and Per-Switch Interface Encoder Towers: C3S employs two encoder towers: one for per-flow features and one for per-switch interface features. Each tower is implemented as a multilayer perceptron (MLP) that maps raw inputs into a latent embedding space, capturing relevant relationships and interactions among the input features.

The per-flow tower maps 8-dimensional inputs to a 64-dimensional embedding via two hidden layers (8–64–64). The per-switch interface tower follows the same structure with reduced width, mapping 2-dimensional inputs to a 32-dimensional embedding (2–32–32).

2) Per-Flow and Per-Switch-Interface Poolers: To aggregate features across flows and switch interfaces, we employ two pooling components inspired by the Deep Sets paradigm [7]. These apply permutation-invariant mean pooling to the corresponding embeddings, yielding fixed-size summaries of dimension 64 (flows) and 32 (switch interfaces). This enables the model to handle variable numbers of entities while preserving symmetry across identical instances.

3) Per-Flow Head: For each flow to be guided, three inputs are fed into the per-flow head. It is a lightweight MLP with a single hidden layer of width 128. Specifically, the per-flow head receives the following inputs: (i) Latent per-flow features of the flow to be guided, i.e., the embedding produced by the per-flow encoder tower; (ii) The flow summary, i.e., the output of the per-flow pooler, representing the global flow context; and (iii) The switch interface summary, i.e., the output of the per-switch-interface pooler, representing the global switch interface context. The MLP outputs the corresponding $cWnd_{target}$ value for that specific flow.

B. *The Importance of Topology-Aware Aggregation*

Initial results indicate that the new architecture scales to an arbitrary number of flows. Fig. 4 depicts the throughput of 10 flows that start sequentially and transmit data from a sender on the left side of the dumbbell topology (cf. Scenario A in Fig. 3) to a receiver on the right, thereby competing for the same bottleneck link with a capacity of 50 Mbit/s. The C3S-guided flows rapidly converge to their fair share and efficiently utilize the available bandwidth.

In Scenario B, two flows are simultaneously active. Flow 1 transmits between two hosts on the left side of the dumbbell, while an unrelated flow 2 traverses the full dumbbell topology from left to right. Since the two flows do not share a bottleneck, they should not compete for bandwidth. Nevertheless, we observe unintended interactions between them due to global pooling, which aggregates too broadly and mixes unrelated information.

This observation underscores the importance of incorporating topological structure into the aggregation process—a direction we are actively pursuing. In particular, feature aggregation should be restricted to topologically and semantically related entities, such as flows that share one or more bottlenecks or switch interfaces that lie along a flow's path. This can be enabled by explicitly encoding routing and topology information. In parallel, we are exploring architectures that learn this contextualization directly, including attention-based mechanisms and graph neural networks (GNNs). GNNs are especially well-suited for this setting, as they naturally constrain feature aggregation to topologically relevant neighbors, thereby preserving locality and preventing interference between unrelated flows.

V. Conclusion & Outlook

We present a revised neural architecture for C3, termed C3S, which addresses key scalability limitations of the original fully connected design through modular, permutation-invariant components. This enables scalable guidance for an arbitrary number of flows. While our results confirm the effectiveness of this approach, they also highlight the importance of incorporating topological context into the aggregation process. In particular, naive global pooling can induce interactions between unrelated flows, motivating ongoing and future work on topology-aware aggregation via path-based filtering as well as graph- and attention-based models.

REFERENCES

[1] Michael König and Martina Zitterbart. "Augmenting End-to-End Congestion Control with Centralized Reinforcement Learning". In: *IEEE/IFIP Network Operations and Management Symposium 2026* (May 2026).

[2] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. "Classic meets modern: A pragmatic learning-based congestion control for the internet". In: *Proceedings of the ACM SIGCOMM 2020*. 2020.

[3] Salma Emara, Baochun Li, and Yanjiao Chen. "Eagle: Refining congestion control by learning from the experts". In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE. 2020, pp. 676–685.

[4] Bo He et al. "DeepCC: Multi-agent deep reinforcement learning congestion control for multi-path TCP based on self-attention". In: *IEEE Transactions on Network and Service Management* 18.4 (2021), pp. 4770–4788.

[5] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv:1707.06347* (2017).

[6] Abien Fred Agarap. "Deep learning using rectified linear units (relu)". In: *arXiv:1803.08375* (2018).

[7] Manzil Zaheer et al. "Deep sets". In: *Advances in neural information processing systems* 30 (2017).

Spatial PDE-aware Selective State-space with Nested Memory for Mobile Traffic Grid Forecasting

Zineddine Bettouche, Khalid Ali, Andreas Fischer, Andreas Kassler
Deggendorf Institute of Technology
Dieter-Görlitz-Platz 1, 94469 Deggendorf
{zineddine.bettouche, khalid.ali, andreas.fischer, andreas.kassler}@th-deg.de

***Abstract*—Traffic forecasting in cellular networks is a challenging spatiotemporal prediction problem due to strong temporal dependencies, spatial heterogeneity across cells, and the need for scalability to large network deployments. Traditional cell-specific models incur prohibitive training and maintenance costs, while global models often fail to capture heterogeneous spatial dynamics. Recent spatiotemporal architectures based on attention or graph neural networks improve accuracy but introduce high computational overhead, limiting their applicability in large-scale or real-time settings. We study spatiotemporal grid forecasting, where each time step is a 2D lattice of traffic values, and predict the next grid patch using previous patches. We propose NeST-S6, a convolutional selective state-space model (SSM) with a spatial PDE-aware core, implemented in a nested learning paradigm: convolutional local spatial mixing feeds a spatial PDE-aware SSM core, while a *nested-learning* long-term memory is updated by a learned optimizer when one-step prediction errors indicate unmodeled dynamics. On the mobile-traffic grid (Milan dataset) at three resolutions ($20^2, 50^2, 100^2$), NeST-S6 attains lower errors than a strong Mamba-family baseline in both single-step and 6-step autoregressive rollouts. Under drift stress tests, our model's nested memory lowers MAE by 48-65% over a no-memory ablation. NeST-S6 also speeds full-grid reconstruction by 32× and reduces MACs by 4.3× compared to competitive per-pixel scanning models, while achieving 61% lower per-pixel RMSE.**

***Index Terms*—mobile traffic forecasting; state-space models; spatiotemporal prediction; partial differential equations (PDEs); drift robustness; efficient inference.**

I. Introduction

Accurate low-latency mobile traffic forecasting is key in dynamic resource management and capacity planning in closed-loop 5G/6G automation. In many deployments, traffic is observed as a spatiotemporal *2D grid* (e.g., a tessellated city map) sampled over time; the task is to predict the next grid frame or a local patch from recent history. The core challenge is to jointly model (i) per-location temporal dynamics and (ii) predominantly *local* spatial coupling, while remaining efficient at realistic resolutions. Prior work spans several approaches [1]–[7]; however, global attention and per-location processing become costly, in addition to accuracy degradation due to traffic non-stationarity and its distribution shift. SSMs provide linear-time sequence modeling with hardware-friendly recurrence [8], but applying them to *grid-structured* forecasting raises two practical requirements: (a) efficient *local* spatial mixing and (b) robustness to drift with stable free-running rollouts.

We address these requirements with **NeST-S6**, a convolutional PDE-aware SSM (building on the S6 family) for grid patches. NeST-S6 combines local spatial mixing (depthwise convolution plus windowed attention) with a spatial PDE-aware SSM core, and augments it with a *nested-learning* slow learner that maintains persistent spatial memory. This memory is updated by a learned optimizer when one-step prediction errors indicate unmodeled dynamics [9]. To reflect networking practice, we report not only one-step accuracy but also multi-step autoregressive rollouts and controlled drift stress tests (Tables I-II).

We present NeST-S6, a spatially local PDE-aware SSM for grid-based traffic forecasting that enables efficient patch-based prediction and delivers improved robustness in multi-step autoregressive rollouts under drift.

II. Problem, Data, and NeST-S6 Method

A. Problem setup and data

Let $M_t \in \mathbb{R}^{H\times W}$ denote the traffic grid at time t. We use a patch-wise formulation: each frame is tiled into non-overlapping patches of size $H_p \times W_p$ with stride $(s_h, s_w) = (H_p, W_p)$. For a fixed patch index, the input is a tensor of the last T=6 patches (last hour), $X_t \in \mathbb{R}^{T\times H_p\times W_p}$, and the target is the next patch $Y_t \in \mathbb{R}^{H_p\times W_p}$ at the same location. Predicting full patches preserves local structure and avoids reconstructing coherent neighborhoods from pointwise supervision [10]. At inference, we predict all patches in parallel and stitch them to reconstruct the grid. We evaluate on the Milan mobile-traffic dataset [11], provided as 100^2 grids sampled every 10 minutes. We report results at three granularities by choosing $H_p\times W_p \in \{20^2, 50^2, 100^2\}$ (Sec. III).

B. NeST-S6: Fast Prediction with Slow Adaptation

We adapt the SSM to spatiotemporal data by implementing the linear recurrence as a convolutional recurrent network with a spatial *PDE-aware* SSM core (S6). Per-pixel states evolve with input-conditioned, spatially varying parameters $\{\Delta_t, \mathbf{B}_{\text{eff},t}, \mathbf{C}_{\text{eff},t}\}$ and optional low-rank $\mathbf{A}_{\text{eff},t}$. NeST-S6 couples a *Fast Learner* for one-step patch prediction with a *Slow Learner* for persistent spatial memory updates based on a *surprise* signal [9]. The code and the model architecture figures are made available in our public repository [12]. The internal operations of the S-PDE SSM block are defined by the following discretized recurrence.

Fast Learner: The input $\mathbf{u}_t = \text{concat}(\mathbf{x}_t, \mathbf{x}_t - \mathbf{x}_{t-1})$ is projected to a latent context $\mathbf{z}_t$ via a convolutional stem. The *Current Context* is a stack of proposed Convolutional SSM blocks that performs local spatial mixing followed by temporal modeling. The per-location recurrence is:

$$\mathbf{h}_t = \exp(\mathbf{A}_{\text{eff},t} \odot \Delta_t) \odot \mathbf{h}_{t-1} + \big(\Delta_t \odot \mathbf{x}_t\big) \odot \mathbf{B}_{\text{eff},t}, \quad (1)$$

$$\mathbf{y}_t = \sum_{s=1}^{D_s} \mathbf{h}_t^{(s)} \odot \mathbf{C}_{\text{eff},t}^{(s)} + \mathbf{D}_{\text{skip}} \odot \mathbf{x}_t, \quad (2)$$

where Δ_t, $\mathbf{B}_{\text{eff},t}$, and $\mathbf{C}_{\text{eff},t}$ are predicted by 1×1 convolutions. $\mathbf{A}_{\text{eff},t}$ is stabilized via $-\exp(\cdot)$ with optional low-rank modulation. We call the core *PDE-aware* because it mirrors a stable exponential discretization of a spatial dynamical system with input-conditioned, spatially varying coefficients $(\mathbf{A}_{\text{eff},t}, \mathbf{B}_{\text{eff},t}, \mathbf{C}_{\text{eff},t}, \Delta_t)$, and we additionally regularize predictions with a Laplacian penalty to encourage physically plausible spatial smoothness.

Slow Learner: The *Deep Optimizer 2D* maintains a spatial memory $\mathbf{M}_t \in \mathbb{R}^{B\times D\times H_p\times W_p}$ updated via the context $\mathbf{z}_t$ and a surprise signal $\mathbf{S}_t$:

$$\mathbf{M}_t = \lambda \mathbf{M}_{t-1} + (1-\lambda)\, \phi_{\text{opt}}(\mathbf{z}_t, \mathbf{S}_t), \quad (3)$$

$$\tilde{\mathbf{z}}_t = \mathbf{z}_t + \sigma(\mathbf{g}_t) \odot \mathbf{M}_t, \quad (4)$$

where λ is a learned decay and $\sigma(\mathbf{g}_t)$ is a gate injecting memory into the Fast Learner. During free-running rollouts, $\mathbf{S}_t$ is unavailable and $\mathbf{M}_t$ evolves only via decay.

Training objective: The model is trained using a SmoothL1 loss and a 3×3 Laplacian penalty to enforce spatial consistency.

III. Evaluation

A. Protocol and metrics

Models are trained with one-step-ahead objective and evaluated on the chronological test split. We report MAE/RMSE after reversing the global z-score. To assess deployment-relevant stability, we assess the *autoregressive rollout*, which feeds predictions back as inputs over a 6-step horizon.

B. Accuracy, rollout stability, and drift

Table I summarizes one-step accuracy and 6-step rollout accumulation, across the granularities. Against VMRNN-D, NeST-S6 improves one-step MAE/RMSE at all resolutions. In rollouts, NeST-S6 exhibits less accumulation than VMRNN-D at all resolutions except MAE at 50^2.

Drift stress tests (robustness): To probe non-stationarity without fine-tuning, we apply inference-time input shifts: (i) scale/offset $x' = \alpha x + \beta$ with α=1.25, β=0.25; (ii) spatial shift by k=5 cells (zero padding); (iii) dynamics volatility via additive noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with σ=0.25. We compare NeST-S6 to a *no-memory* ablation that disables memory injection and memory writes (equivalently, $\mathbf{M}_t \equiv \mathbf{0}$). As evinced in Table II, the nested memory consistently reduces MAE by 4.88-6.98 across shifts.

TABLE I
SINGLE-STEP TEST PERFORMANCE AND 6-STEP ROLLOUT ACCUMULATION ON MILAN. ACCUMULATION IS $\Delta\text{MAE/RMSE} = \text{MAE/RMSE}_{h=6} - \text{MAE/RMSE}_{h=1}$.

Model	Metric	20^2	50^2	100^2
VMRNN-D	MAE (h=1)	4.442	4.461	4.476
	RMSE (h=1)	11.353	11.485	11.919
	ΔMAE (6−1)	4.264	3.804	3.613
	ΔRMSE (6−1)	10.093	8.946	8.858
NeST-S6	MAE (h=1)	3.854	3.822	3.906
	RMSE (h=1)	8.401	8.354	8.422
	ΔMAE (6−1)	3.404	5.359	3.530
	ΔRMSE (6−1)	8.259	8.827	8.736

TABLE II
DRIFT STRESS TEST (MAE) ON MILAN (ONE-STEP) SHOWING THE NESTED MEMORY EFFECT

Drift	Nested Memory	No Nested Memory	ΔMAE
None	3.800	10.712	+6.912
Scale/offset	5.303	10.182	+4.879
Spatial shift	3.810	10.788	+6.978
Dyn. volatility	5.992	12.418	+6.426

Efficiency: For full-grid (100^2) reconstruction, NeST-S6 predicts patches and tiles them, whereas scalar predictors must be executed per location. When compared to the competitive scalar predictor HiSTM, NeST-S6 reduces MACs by 4.3× and latency by 32× per-location scanning on an A100 GPU, supporting scalable forecasting for multi-step decision making. NeST-S6 also achieves lower per-pixel RMSE than the HiSTM in 61% of the grid.

IV. Conclusion

We formulated mobile traffic grid forecasting as patch-wise spatiotemporal prediction and introduced **NeST-S6**, a convolutional state-space model with a PDE-inspired spatial SSM core and a nested-learning memory mechanism. By decoupling fast prediction from slow, error-driven memory updates, NeST-S6 enables low-latency inference while adapting to non-stationary dynamics without frequent weight updates.

Across three spatial resolutions on the Milan dataset, NeST-S6 consistently outperforms strong mamba-based baseline in one-step accuracy, exhibits smoother error growth under autoregressive rollout, and shows strong robustness to controlled distribution shifts. Patch-based tiling further enables efficient full-grid forecasting, reducing inference cost by over 30× while achieving lower per-pixel error across most of the grid.

Future work will extend NeST-S6 to longer-horizon and continual forecasting, richer drift scenarios and real network events, and multi-modal inputs and control-oriented objectives. The future work also takes into account the explainability analysis to unpack the behavior of the core, along with CUDA kernels for a hardware-aware implementation that fits the standard in SSM developments.

Overall, NeST-S6 provides a practical, drift-aware foundation for scalable traffic forecasting in operational 5G/6G systems.

ACKNOWLEDGMENT

This work was partly funded by the Bavarian Government through the HighTech Agenda (HTA).

REFERENCES

[1] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional lstm network: a machine learning approach for precipitation nowcasting," in *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, MA, USA, 2015, p. 802–810.

[2] Z. He, C.-Y. Chow, and J.-D. Zhang, "Stcnn: A spatio-temporal convolutional neural network for long-term traffic prediction," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, 2019, pp. 226–233.

[3] A. Deihim, E. Alonso, and D. Apostolopoulou, "Sttre: A spatio-temporal transformer with relative embeddings for multivariate time series forecasting," *Neural Networks*, vol. 168, pp. 549–559, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608023005361

[4] Y. Tang, P. Dong, Z. Tang, X. Chu, and J. Liang, " VMRNN: Integrating Vision Mamba and LSTM for Efficient and Accurate Spatiotemporal Forecasting ," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2024, pp. 5663–5673. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPRW63382.2024.00575

[5] Y. Wang, Z. Gao, M. Long, J. Wang, and P. S. Yu, "Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning," in *International conference on machine learning*. PMLR, 2018, pp. 5123–5132.

[6] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 231–240. [Online]. Available: https://doi.org/10.1145/3209582.3209606

[7] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *International Conference on Learning Representations*, 2023.

[8] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *First conference on language modeling*, 2024.

[9] A. Behrouz, M. Razaviyayn, P. Zhong, and V. Mirrokni, "Nested learning: The illusion of deep learning architectures," in *NeurIPS 2025*, 2025, also available at: https://abehrouz.github.io/files/NL.pdf. [Online]. Available: https://neurips.cc/virtual/2025/poster/116123

[10] Z. Bettouche, K. Ali, A. Fischer, and A. Kassler, "Histm: Hierarchical spatiotemporal mamba for cellular traffic forecasting," in *Proceedings of the International Conference on Network Systems (NetSys), Workshop on Machine Learning in Networking (MaLeNe)*, Ilmenau, Germany, September 2025.

[11] G. Barlacchi, M. D. Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of milan and the province of trentino," *Scientific Data*, vol. 2, p. 150055, 2015. [Online]. Available: https://www.nature.com/articles/sdata201555

[12] Z. Bettouche, K. Ali, A. Fischer, and A. Kassler, "Nest-s6 (reference implementation)," 2026. [Online]. Available: https://github.com/ZineddineBtc/NeST-S6

Proactive IIoT Gateway Load Balancing via Federated Learning

Ibrahima Ndiaye Mesut Güneş
Faculty of Computer Science, Otto-von-Guericke University Magdeburg, Germany

***Abstract*—We study proactive gateway load balancing in industrial sensor networks. Conventional schemes are reactive and centralized. We instead predict per-gateway load via federated learning (FL) and trigger local reassignment before overload. Gateways train Long Short-Term Memory (LSTM) models on local traffic and share only model updates; no raw data leaves devices. The aggregated model enables decentralized, privacy-preserving control with a lightweight coordinator using Multi-Gateway Centralized Utility-Based Load Balancing (MCUBE). In simulation, our approach improves utilization (+14.5%), reduces load imbalance (–30.0%) and packet loss (–63%) versus a reactive baseline, with a modest rise in reassignments.**

***Index Terms*—Load balancing; Industrial IoT; Federated learning; LSTM**

I. Introduction

The Industrial Internet of Things (IIoT) connects sensors through multiple gateways that forward traffic to backend systems [1], [2]. When gateway loads become imbalanced, congestion and loss increase [3], [4]. CUBE improves utilization by reassigning LM–gateway links, but remains reactive and centralized [4], [5], [6].

We propose Federated Learning (FL)-Multi-Gateway Centralized Utility-Based Load Balancing (MCUBE): gateways train local LSTM predictors via FL and proactively trigger reassignment locally, achieving decentralized control without sharing raw data.

Reconfiguration is triggered when predicted utilization crosses adaptive thresholds, acting before overload while preserving data locality and privacy.

Contributions: Local LSTM predictors for load forecasting, federated learning without sharing raw data, and proactive threshold-based reassignment. Simulation shows +14.5% utilization, 63% loss reduction, and better load balance.

We next summarize related work, describe the model/protocol, and present results.

II. Related Work

Prior work falls into three groups: reactive utility-based reassignment, centralized ML prediction/control, and federated learning in IoT.

1) Utility-Based Reassignment (Reactive Centralization)

CUBE/MCUBE [4] reassigns managers after overload is observed. This is simple but reactive and centralized, which increases control overhead and delays response under bursty traffic.

2) Centralized Machine Learning for Prediction and Control

Centralized ML methods [5], [7], [8] improve proactivity via global predictors, but they require raw data collection and centralized control, creating privacy, scalability, and autonomy limitations in IIoT.

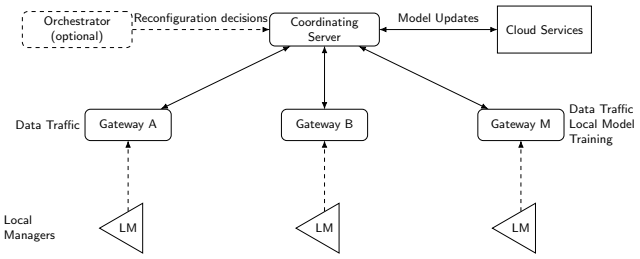

Fig. 1: Decentralized gateways with local LSTMs, federated aggregation, and proactive MCUBE reassignment with migration costs and hysteresis.

3) Federated Learning in IoT

FL [9], [6], [10] preserves privacy by sharing model updates instead of raw data. Existing IoT FL studies mainly target sensing/analytics tasks, with limited focus on proactive gateway-level control.

4) Gap and Our Contribution

FL-MCUBE closes the gap by combining proactive prediction with decentralized reassignment: each gateway predicts overload locally, collaborates through FL without exposing raw data, and triggers migration-aware MCUBE decisions autonomously.

III. System Model and FL-MCUBE Protocol

In brief, we consider an IIoT network with gateways $\mathcal{G}$ and local managers (LMs) $\mathcal{L}$. Each LM attaches to one gateway and contributes to its channel utilization. Gateways record their utilization history $u_k(t)$ and operate under bandwidth/compute constraints [3], [2]. Gateways train local LSTM predictors on $u_k(t)$ and collaboratively refine them via FL (FedAvg [11]); only model updates are exchanged.

Given the past τ utilization values

$$X_k = \{u_k(t-\tau), \ldots, u_k(t-1)\},$$

the LSTM outputs a next-step prediction

$$\hat{u}_k(t) = f_{\text{LSTM}}(X_k; w_k),$$

where w_k are local weights refined via FL. Reassignment is triggered when $\hat{u}_k(t+1)$ exceeds an adaptive threshold for h consecutive slots; MCUBE executes locally with a migration cost to avoid churn.

FL-MCUBE is intentionally lightweight: local LSTMs avoid global traffic collection, FedAvg limits coordination

Algorithm 1 FL-MCUBE: Adaptive proactive load balancing protocol

Require: Gateways $\mathcal{G}$; Local Managers (LMs) $\mathcal{L}$; thresholds $\theta_k(t{+}1)$; lookback τ; hysteresis h; FL rounds R
Ensure: Balanced gateway load

for each gateway $g_k \in \mathcal{G}$ **do**
  Predict $\hat{u}_k(t{+}1)$ with $f_{\mathrm{LSTM}}(X_k; w_k)$
end for
for $r = 1$ to R **do**
  Securely aggregate local w_k; compute global w via FedAvg; redistribute
end for
for each gateway $g_k \in \mathcal{G}$ **do**
  if $\hat{u}_k(t{+}1) > \theta_k(t{+}1)$ for h consecutive slots **then**
    Identify candidate LMs; select subset S minimizing $\sum_{\ell \in S} c_{\mathrm{mig}}(\ell)$ under utilization constraints
    Trigger proactive LM reassignment via MCUBE (executed locally)
  end if
end for

Table I: FL-MCUBE evaluation results (30-run average $\pm$ std. dev.). Util.: Utilization.

Setup	Util. [%]	Std. Dev.	Loss [%]	Switches
Baseline	55	0.300	7.5	2.0
FL-only	58	0.260	5.5	1.9
FL-MCUBE (ours)	**63**	**0.210**	**2.8**	**3.8**

overhead, and threshold-based triggering prevents continuous optimization. Compared to fully centralized predictors, this design trades global optimality for scalability and privacy, appropriate for IIoT deployments with constrained gateways. Hysteresis and migration costs explicitly control churn, making proactive reassignment stable in practice.

IV. Results

We compare a reactive centralized MCUBE baseline, an FL-only setup (prediction but no reassignment), and the proposed FL-MCUBE.

The baseline follows MCUBE [4]: centralized, threshold-triggered, and purely reactive (no prediction).

Simulations use a custom Python 3.10/TensorFlow 2.15.0 environment; all hyperparameters are fixed across baselines.

Setup: six gateways (bursty, stable, rising, smooth, sporadic, periodic); local LSTMs with short lookback; adaptive threshold with hysteresis; three FL rounds with FedAvg and secure aggregation [12], [11]. Each experiment runs 300 steps and is replicated 30 times with different seeds. We also evaluate scalability with 20 gateways.

Key outcomes (30-run means) are in Table I; lower std. dev. means better balance.

Prediction Accuracy: From aggregated stored runs (6 gateways, 30 runs), LSTM yields Mean Absolute Error (MAE) 0.091 and Root Mean Squared Error (RMSE) 0.133; the moving-average baseline (window 10) yields MAE 0.081 and RMSE 0.122.

Scalability: In the 20-gateway case (Table II), FL-MCUBE keeps the same improvement trend over baselines.

Table II: FL-MCUBE scalability: 20-gateway experiment (10-run average).

Setup	Util. [%]	Loss [%]	Switches
Baseline	52	9.2	2.1
FL-only	56	7.1	2.0
FL-MCUBE (ours)	60	3.5	4.2

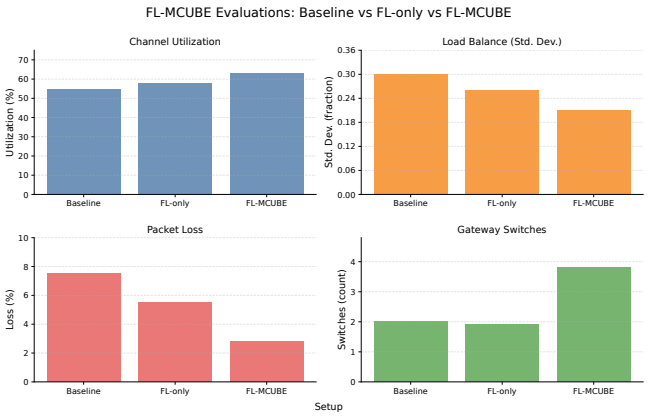

Fig. 2: Comparative evaluation across three schemes over 30-run means: reactive MCUBE baseline (red), FL-only (green), and proactive FL-MCUBE (blue). Left: utilization (%) increases with proactivity. Center: standard deviation of load decreases, indicating improved balance. Right: packet loss (%) drops significantly with prediction. Modest increase in switches reflects proactive reassignment overhead, but load balance gains outweigh churn cost.

Communication Overhead: Each FL round transmits $\sim$12 KB per gateway; for 3 rounds and 20 gateways this is $\sim$720 KB, trading bandwidth for privacy.

FL-MCUBE achieves the best balance/loss with a modestly higher switch count due to proactive moves.

1) Analysis and Insights

All reported means are over 30 runs; FL-MCUBE vs. baseline is significant ($p < 0.05$, t-test).

Three findings emerge: (1) FL-only already improves balance/loss over reactive baseline, showing prediction value; (2) FL-MCUBE reaches highest utilization (63% vs. 55% baseline) by acting before overload; (3) the higher switch count (3.8 vs. 2.0) is offset by large loss reduction (2.8% vs. 7.5%).

V. Conclusion

We presented FL-MCUBE, a decentralized predictive load-balancing framework for IIoT gateways. By combining local LSTM forecasting with federated learning, gateways trigger proactive reassignment without sharing raw data.

Across experiments, FL-MCUBE improves utilization (63% vs. 55%), reduces packet loss (2.8% vs. 7.5%), and improves balance (std. dev. 0.210 vs. 0.300), with a moderate increase in switches. The 20-gateway setting shows similar trends.

The main contribution is integrating FL directly into the control loop so gateways learn collaboratively and act autonomously. Future work includes testbed validation, stronger robustness under client dropout, and mobility-aware extensions.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[3] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1389–1406, 2013.

[4] M. d. C. Lucas-Estañ and J. Gozalvez, "Load balancing for reliable self-organizing industrial IoT networks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 5052–5063, 2019.

[5] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 12:1–12:19, 2019.

[7] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[8] H. Li, Y. Wang, and X. Li, "Deep learning-based traffic prediction for industrial IoT," in *2018 IEEE 44th Annual Conference of the Industrial Electronics Society (IECON)*. IEEE, 2018, pp. 384–389.

[9] X. Wang, S. Garg, H. Lin, J. Hu, and M. S. Hossain, "Toward accurate anomaly detection in industrial internet of things using hierarchical federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7110–7119, 2022.

[10] C. Zhang, S. Dang, B. Shihada, and M. Alouini, "Dual attention-based federated learning for wireless traffic prediction," in *IEEE INFOCOM 2021 – IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[11] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017)*, ser. Proceedings of Machine Learning Research, A. Singh and X. Zhu, Eds., vol. 54. Fort Lauderdale, FL, USA: PMLR, 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[12] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Dallas, TX, USA: ACM, 2017, pp. 1175–1191.

Automated Network Access Control with LLMs

Jonas Wessner
Ulm University
Email: jonas.wessner@uni-ulm.de

Tobias Meuser
Technical University Darmstadt
Email: tobias.meuser@kom.tu-darmstadt.de

Frank Kargl
Ulm University
Email: frank.kargl@uni-ulm.de

***Abstract*—Access control policies provide an essential layer of security for organizations' systems and data by defining rules for allowed and disallowed accesses, thereby preventing unauthorized access to resources. Access policies are inherently dynamic, as they must adapt to evolving organizational requirements such as infrastructural, operational, or personnel changes. As networks grow larger, the complexity of access policies and the frequency of required updates increase, traditional manual approaches to access control configuration no longer scale. In this paper, we explore the potential of large language models (LLMs) for automating network access control by automatically translating help desk tickets into policy changes. We summarize the challenges of using LLMs for network access control automation and present insights from preliminary experiments. Our early experiments show that LLMs are capable of understanding user intent and applying configuration changes in isolated examples when relevant background information is provided within the model's context. At the same time, we identify integration into production systems and reliability as major challenges to be addressed in future work.**

***Index Terms*—access control, intent-based networking, autonomous networking, LLM, natural language**

I. Introduction

Network access policies are a central building block of security architectures, defining rules for allowed and disallowed accesses to resources such as data, applications, and physical infrastructure. Requirements for accessibility of resources are usually highly dynamic. For instance, applications may be moved from testing to production environments, or user permissions may change as they switch departments or teams. In traditional workflows, users can request access control changes through help desk tickets. However, as systems grow in the number of users and resources, and as the landscape of access control mechanisms becomes increasingly complex and heterogeneous, manual implementation of continuous access control changes no longer scales. With IT help desks becoming a bottleneck in access control configuration, prolonged queuing of user tickets can hinder company productivity and potentially lead to slow reactions to system vulnerabilities. These challenges highlight the need for more agile and automated approaches to access control configuration.

The rise of Large Language Models (LLMs) provides a new angle to this topic, as their strengths in natural language understanding and reasoning can be helpful for automating help desk ticket processing. LLMs can be used to automatically understand user tickets, interpret them, and generate policy updates, as illustrated in Figure 1. However, LLMs are prone to hallucinations, which can be fatal if they result in incorrect policies [1]. Furthermore, user requests may be vague or lack essential technical details, or even propose nonsensical updates, making it difficult to synthesize precise network access control policies directly from user requests. In this paper, we identify key challenges in automating access control configuration and share insights from preliminary experiments. Our results highlight the potential of state-of-the-art LLMs for understanding user requests and constructing structured policies, while also identifying several challenges for their use in real-world systems. Based on these insights, we suggest future work to further explore LLM-based systems for access control management to serve the dynamic requirements of current and future secure IT infrastructure.

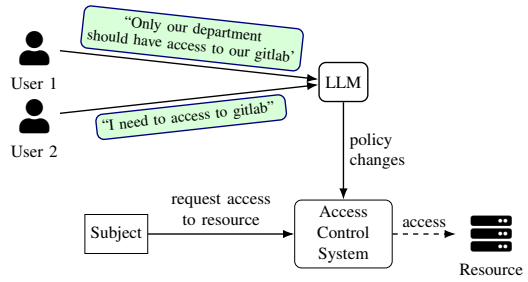

Fig. 1. Overview of an LLM-based access control system. On the control plane, an LLM is used to translate help desk tickets into access policies. On the data plane, these policies are enforced by the access control system.

II. Key Challenges in LLM-Based Access Control

In this section, we discuss central challenges for automated network access control policy synthesis from natural language.

A. Vague and Incomplete User Requests

Typically, users who report access control changes via help desks do not come from a technical background. Thus, such help desk tickets are inherently vague and lack technical details. For instance, the message "I need access to gitlab" cannot be directly translated into a low-level policy for two reasons: First, entities mentioned in the text (e.g., "gitlab") are ambiguous without further context (e.g., could refer to one of multiple existing gitlab servers). Second, users refer to high-level entities without specifying or knowing technical details, such as IP addresses and ports, that are necessary for implementing the policies. Additionally, users might even completely omit important information, which suggests that an LLM-based access control system should be able to query the user for additional information in such cases.

B. Heterogeneous Landscape of Access Control Mechanisms

In recent years, access control has become increasingly complex and heterogeneous, with trends such as zero trust [2] and micro-segmentation [3] pushing towards more layered and fine-grained access control. Specifically, access control policies should not only be enforced at a single point (such as a perimeter firewall), but at as many security layers as possible, including, for example, SDN switches, host firewalls, and application-level logins. This complex landscape of access control mechanisms poses challenges not only for manual implementation of access control policies but also for LLM-based approaches. For instance, it is known from previous works that LLMs struggle with large contexts and nuanced distinctions between software versions, both of which are relevant for correctly managing large-scale and heterogeneous infrastructure [4], [5].

C. User Intent Conflicts

An access control system that serves change requests from end users, similar to how a help desk works, is a multi-user system. Thus, requests from one user can interfere with or be contradictory to requests from other users. This calls for a conflict resolution approach, possibly through defining precedence among requests or an authorization scheme.

III. Lessons from Early Exploration

We obtained a small set of real-world help-desk tickets from a university computing center, containing user messages related to access control policy changes. Using insights from this dataset, we conducted preliminary experiments to explore the potential as well as revealing challenges of using LLMs for automating access control management. In the following, we share lessons learned from our experiments, which may support future research in this field.

A. Potential of LLMs for Understanding User Intent

As a case study, we presented state-of-the-art LLMs with help desk tickets and manually selected necessary background information that help desk staff needed to correctly process such tickets, providing this information as part of the prompt. This contextual information included (1) the identity of the user and related information such as previous requests or their relationship to other relevant parties, (2) relevant excerpts of the access control configuration files, and (3) any additional background information about the organization relevant to this ticket. We then asked an LLM to construct an access policy in a given JSON format and compared the result with the expected policy outcome. We found that current LLMs can often accurately understand user intent and can adjust access control configuration accordingly. Furthermore, we find that current models benefit from in-context examples [6]: When we illustrate the task using the example of another ticket as part of the prompt, the model behavior is closer to our expectations. In production systems, tickets previously processed by humans could be utilized to provide such examples to the LLM.

B. Challenges of Full Automation in Real-World Settings

There remains a gap between isolated use cases and complex production environments. Real-world use cases with large, unstructured code bases of configuration files and large organizations pose challenges to the problem understanding of LLMs [5]. We identify the automated selection of relevant context information and navigation of configuration code bases to identify relevant sections as core challenges. To address these challenges, we are exploring viable approaches. To select contextual information, embedding-based similarity search [7], similar to that used in Retrieval Augmented Generation (RAG) [8], might be useful. For navigating the code base, approaches based on LLM agents [9] seem promising, allowing for iterative exploration of code bases [10], [11]. Agentic approaches might also be suitable for asking follow-up questions to the user if their request is missing essential information.

IV. Reliability of LLM Decisions

While we find that LLMs can often accurately understand user intent, we also identify cases in which LLMs respond to user tickets with incorrect solutions. Especially when the user message is not written clearly or is missing important information, we find that the LLM tends to output wrong answers instead of suggesting to ask the user for clarification. Such inaccuracies highlight that, while LLMs are a powerful tool for network access control automation, they must be integrated into production systems with caution. We suggest that future work create datasets that allow measuring the accuracy of specific LLM-based designs, which could help estimate the utility and risk of such systems. Additionally, certain security checks, such as rule-based checks and human reviews, should be performed before deploying LLM-generated policy changes to avoid misconfiguration. At the same time, human intervention should be kept minimal to not degrade the efficiency of the system, highlighting the trade-off between automation and human supervision.

V. Conclusion

In this paper, we presented the concept of LLM-based network access control management. We highlighted the need for automated access control solutions to handle dynamic and increasingly complex IT infrastructure and how LLMs emerge as a new tool for automating this task. We further shared insights from preliminary experiments. On the one hand, we found that state-of-the-art LLMs can often correctly understand user intent and edit access control configurations accordingly, while on the other hand, we identified challenges in scaling to complex real-world use cases and ensuring correctness of LLM outputs in production settings. Our work provides evidence that LLM-based access control is a promising field of research and worth further investigation. Open questions to be addressed are how to integrate LLMs into real-world systems and how to prove or estimate the reliability of LLM-based access control systems.

REFERENCES

[1] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin *et al.*, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *Transactions on Information Systems*, 2025.

[2] N. F. Syed, S. W. Shah, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "Zero Trust Architecture (ZTA): A Comprehensive Survey," *IEEE Access*, 2022.

[3] N. Sheikh, M. Pawar, and V. Lawrence, "Zero Trust Using Network Micro Segmentation," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021.

[4] T. Wu, W. Wu, X. Wang, K. Xu, S. Ma, B. Jiang, P. Yang, Z. Xing, Y.-F. Li, and G. Haffari, "Versicode: Towards Version-Controllable Code Generation," *arXiv preprint arXiv:2406.07411*, 2024.

[5] T. Li, G. Zhang, Q. D. Do, X. Yue, and W. Chen, "Long-Context LLMs Struggle with Long In-Context Learning," *arXiv preprint arXiv:2404.02060*, 2024.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Curran Associates, Inc., 2020.

[7] N. Reimers and I. Gurevych, "Sentence-bert: Sentence Embeddings Using Siamese BERT-Networks," *arXiv preprint arXiv:1908.10084*, 2019.

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2020.

[9] X. Dong, X. Zhang, W. Bu, D. Zhang, and F. Cao, "A Survey of LLM-based Agents: Theories, Technologies, Applications and Suggestions," in *International Conference on Artificial Intelligence, Internet of Things and Cloud Computing Technology (AIoTC)*, 2024.

[10] T. Gupta, L. Weihs, and A. Kembhavi, "CodeNav: Beyond Tool-Use to Using Real-World Codebases with LLM Agents," *arXiv preprint arXiv:2406.12276*, 2024.

[11] K. Zhang, J. Li, G. Li, X. Shi, and Z. Jin, "Codeagent: Enhancing Code Generation with Tool-Integrated Agent Systems for Real-World Repo-Level Coding Challenges," *arXiv preprint arXiv:2401.07339*, 2024.

A High-Fidelity Virtualized Digital Twin System for ISP Core Networks

Johannes Späth, Georg Carle
Technical University of Munich, Germany
{spaethj, carle}@net.in.tum.de

***Abstract*—ISP core networks are critical for today's Internet, and fault management is a major concern. Digital Twins (DTs) offer a safe environment for what-if analyses and can help generating incident datasets, thereby supporting Machine Learning (ML) applications. Existing approaches often use high abstraction, limiting accuracy and performance, and frequently focus on specific metrics. Furthermore, they often rely on custom interfaces, impeding real-world application. We present a virtualization-based DT framework capable of low abstraction, high fidelity, and near-hardware performance. It spans multiple network planes via standardized NETCONF/YANG interfaces and exports telemetry through YANG models, IPFIX, and BMP.**

***Index Terms*—Digital Twin, YANG, Testbed, Core Network**

I. Introduction

Fault management is a central concern for operators of computer networks, especially for critical infrastructure like ISP core networks. Digital Twins (DTs) offer significant potential to support multiple use cases in this domain. As networks become increasingly complex, understanding the effects of configuration changes is often infeasible. DTs enable what-if analyses to test changes and validate mitigations. Further, Machine Learning (ML)-based network management approaches require large amounts of training data. While live systems generate substantial amounts of data, insights from real incidents are most valuable. Since introducing faults in a production system is not feasible, DTs provide a practical means for generating incident datasets.

Existing DT approaches typically operate at a high abstraction level, like with simulations or emulations, suffering from inaccuracies and low performance. Many approaches focus solely on specific aspects, *e.g.*, latency modeling. In addition, they frequently rely on custom interfaces and do not support ISP-grade monitoring standards.

Our approach uses virtualization in a testbed and thus provides low abstraction, high fidelity, and near-hardware performance. It covers multiple planes via standardized interfaces based on NETCONF/YANG and supports telemetry export using standardized YANG models, IPFIX, and BMP.

II. Background and Related Work

This section covers relevant background and related work.

A. Network Digital Twins

Various techniques have been proposed for replicating computer networks, including simulation [1], [2], emulation [3], [4], virtualization [5], as well as bare-metal deployments.

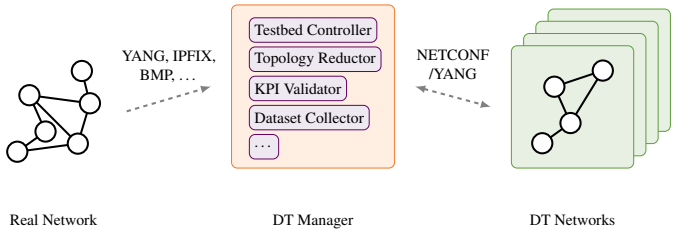

Fig. 1: Overview of the DT system.

Wu et al. [6] present a comprehensive survey on network DTs, in which DTs are defined as replicas of physical network elements. The authors identify key challenges for network DTs and discuss a wide range of application domains. Almasan et al. [7] introduce the concept of a network DT providing accurate and performant network models. Their approach leverages ML techniques such as Graph Neural Networks (GNNs) combined with suitable abstractions to achieve scalability and real-time performance. Our system focuses on ISP core networks and uses standardized YANG models and real-world data from monitoring protocols like IPFIX and BMP.

B. The YANG Modeling Language

YANG [8], [9] is a data modeling language standardized by the IETF for use with NETCONF [10], a network management protocol for configuring network devices. The IETF has standardized a wide range of YANG models covering multiple layers and components of the ISO/OSI stack, including interfaces, routing, and layer 2+3 topology [11], [12], [13], [14], [15], [16], [17].

C. ISP Network Telemetry

We distinguish between configuration data and telemetry in ISP networks. YANG models support configuration and operational state data, whereas dedicated telemetry protocols are limited to the latter. Several standardized telemetry protocols are used in ISP networks, *e.g.*, IPFIX [18] for flow-level data-plane telemetry, and BMP [19] for routing-related control-plane telemetry.

III. Approach

In this section, we describe our high-fidelity virtualized DT system, which is depicted in Figure 1. It consists of a real network, a central management component, and multiple DT networks deployed in a testbed. Network state is exported from the real network using YANG-modeled data, IPFIX telemetry,

and BMP messages, and the architecture can be extended to support additional sources. The DT networks are configured via NETCONF and also export telemetry using YANG.

A. Real Network

This component is a production-grade network consisting of multiple routers. There are no special constraints regarding the routing software used, but it needs to support configuration and telemetry export using the IETF-standardized YANG models.

B. DT Manager Components

At the core of the system is the *DT Manager*, a central component that provides multiple functionalities and supports extensibility, *i.e.*, further features can be added on demand.

a) Testbed Controller: This component orchestrates the virtualized router replicas, configures links, and manages load generation. Depending on the use case, it can also be used to keep the DT synchronized with the real network and to execute actions at specific timestamps.

b) Topology Reductor: This part performs sampling and/or aggregation of the original topology based on predefined goals or the resources available for the DT. Typically, this step also includes flow- and link-rate reduction.

c) KPI Validator: This component evaluates the current state of the real network and the DTs against given Key Performance Indicators (KPIs) using the telemetry.

d) Dataset Collector: This module allows creating datasets for ML using the DT networks. Since DTs are replicas of the production environment, multiple problem instances can be generated safely without affecting live operations. To produce diverse datasets for ML training, the parameter space of problems is defined and samples are balanced across it. Faults can be injected using the Testbed Controller's timed execution feature. After a fault is injected, the corresponding raw data exported from the DT network can be correlated to the fault, allowing multiple datapoints to be extracted from a single long-running measurement.

C. DT Networks

The network DTs are deployed in a testbed using virtualization. Routers are replicated by Linux machines based on FRR [20], which implements a northbound API for YANG export that supports NETCONF via a plugin [21]. Each router runs on an individual Virtual Machine (VM).

IV. Parameter Evaluation

In this section, we evaluate the number of parameters that we expect to be collected for networks of a certain size. For that, we restrict ourselves to three IETF-standardized models (L2 topology [13], L3 topology [12], and interface management [15]) and group their fields by data type (*Identifier*, *Bool*, *Numeric*, and *Other*). We generate random graphs using the Barabási-Albert preferential attachment model [22] with $m = 2$, *i.e.*, for each new node, two edges are added. For each graph, we estimate the number of YANG fields, assuming that every optional field is present and every field with zero or more elements has exactly one instance.

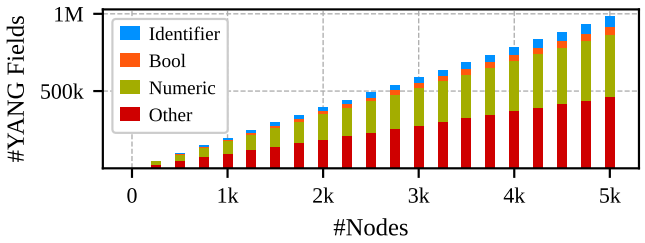

Fig. 2: Number of YANG fields resulting from three IETF models for an increasing number of nodes.

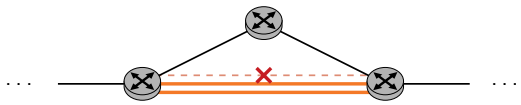

Fig. 3: Overload in a LAG caused by a link failure. The remaining two links are not able to handle the full load.

Figure 2 shows a high number of parameters for three models and small networks (49k fields/250 nodes), which significantly grows for larger networks (1M fields/5k nodes). This underlines the high fidelity of network modeling using our YANG-based approach, however, it introduces complexity for the ML models operating on this data.

V. Example Use Case: LAG Overload Scenario

In this section, we introduce an example scenario involving a Link Aggregation Group (LAG), the relevant YANG models for this scenario, potential symptom fixes, and use cases for our proposed DT system. The scenario considers a LAG comprising three 1 Gbit/s links, carrying a total of 2.5 Gbit/s of traffic (see Figure 3). If one link fails, the remaining two experience an overload situation, since their combined capacity of 2 Gbit/s is exceeded.

In YANG, the LAG is represented using the fields `lag` and `member-link-tp` from the L2 topology model [13]. The `oper-status` field from the interface management model [15] indicates the failed link, while `out-discards` signals the overload on the remaining links.

Potential symptom mitigation strategies include adapting rate limiting on the remaining links if sufficient headroom exists, or re-routing traffic over alternative paths. Using our system, the failure can be automatically detected via the KPI validator, two DT instances can be instantiated to test the mitigations, and the best-performing solution can then be applied to the real network. Additionally, our system can generate datasets of similar scenarios to train ML models for predictive management.

VI. Conclusions and Outlook

In this work, we presented an extensible system for creating high-fidelity virtualized DTs of ISP core networks. Our approach is based on standardized data models and protocols used in production networks. Thus, it provides a basis for fault management in ISP core networks. By automating incident dataset generation, it enables ML-based approaches for fault recovery and thus contributes to efforts toward closed-loop systems and self-driving networks.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), projects HyperNIC (503359370) and SLICES-SUSTAINABILITY (566292327), by the EU Horizon Europe programme, project GreenDIGIT (101131207), by the German Federal Ministry of Research, Technology and Space (BMFTR), project 6G-life (16KIS2414), and by the Bavarian Ministry of Regional Development and Energy, project 6G Future Lab Bavaria. Results presented in this publication were obtained using the SLICES-DE research infrastructure.

References

[1] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," ICST, May 2010. DOI: 10.4108/ICST.SIMUTOOLS2008.3027.

[2] G. F. Riley and T. R. Henderson, "The ns-3 Network Simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34. DOI: 10.1007/978-3-642-12331-3_2.

[3] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, ser. Hotnets-IX, Monterey, California: Association for Computing Machinery, 2010. DOI: 10.1145/1868447.1868466.

[4] M. Peuster, H. Karl, and S. van Rossem, "MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov. 2016, pp. 148–153. DOI: 10.1109/NFV-SDN.2016.7919490.

[5] F. Wiedner, M. Helm, S. Gallenmüller, and G. Carle, "HVNet: Hardware-Assisted Virtual Networking on a Single Physical Host," in *IEEE INFOCOM WKSHPS: Computer and Networking Experimental Research using Testbeds (CNERT 2022) (INFOCOM WKSHPS CNERT 2022)*, Virtual Event, May 2022. DOI: 10.1109/INFOCOMWKSHPS54753.2022.9798351.

[6] Y. Wu, K. Zhang, and Y. Zhang, "Digital Twin Networks: A Survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021. DOI: 10.1109/JIOT.2021.3079510.

[7] P. Almasan et al., "Network Digital Twin: Context, Enabling Technologies, and Opportunities," *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, 2022. DOI: 10.1109/MCOM.001.2200012.

[8] M. Björklund, *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*, RFC 6020, Oct. 2010. DOI: 10.17487/RFC6020. [Online]. Available: https://www.rfc-editor.org/info/rfc6020.

[9] M. Björklund, *The YANG 1.1 Data Modeling Language*, RFC 7950, Aug. 2016. DOI: 10.17487/RFC7950. [Online]. Available: https://www.rfc-editor.org/info/rfc7950.

[10] R. Enns, M. Björklund, A. Bierman, and J. Schönwälder, *Network Configuration Protocol (NETCONF)*, RFC 6241, Jun. 2011. DOI: 10.17487/RFC6241. [Online]. Available: https://www.rfc-editor.org/info/rfc6241.

[11] A. Clemm, J. Medved, R. Varga, N. Bahadur, H. Ananthakrishnan, and X. Liu, *A YANG Data Model for Network Topologies*, RFC 8345, Mar. 2018. DOI: 10.17487/RFC8345. [Online]. Available: https://www.rfc-editor.org/info/rfc8345.

[12] A. Clemm, J. Medved, R. Varga, X. Liu, H. Ananthakrishnan, and N. Bahadur, *A YANG Data Model for Layer 3 Topologies*, RFC 8346, Mar. 2018. DOI: 10.17487/RFC8346. [Online]. Available: https://www.rfc-editor.org/info/rfc8346.

[13] J. Dong, X. Wei, Q. Wu, M. Boucadair, and A. Liu, *A YANG Data Model for Layer 2 Network Topologies*, RFC 8944, Nov. 2020. DOI: 10.17487/RFC8944. [Online]. Available: https://www.rfc-editor.org/info/rfc8944.

[14] M. Björklund, *A YANG Data Model for IP Management*, RFC 7277, Jun. 2014. DOI: 10.17487/RFC7277. [Online]. Available: https://www.rfc-editor.org/info/rfc7277.

[15] M. Björklund, *A YANG Data Model for Interface Management*, RFC 8343, Mar. 2018. DOI: 10.17487/RFC8343. [Online]. Available: https://www.rfc-editor.org/info/rfc8343.

[16] S. Litkowski, D. M. Yeung, A. Lindem, Z. Zhang, and L. Lhotka, *YANG Data Model for the IS-IS Protocol*, RFC 9130, Oct. 2022. DOI: 10.17487/RFC9130. [Online]. Available: https://www.rfc-editor.org/info/rfc9130.

[17] L. Lhotka and A. Lindem, *A YANG Data Model for Routing Management*, RFC 8022, Nov. 2016. DOI: 10.17487/RFC8022. [Online]. Available: https://www.rfc-editor.org/info/rfc8022.

[18] P. Aitken, B. Claise, and B. Trammell, *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*, RFC 7011, Sep. 2013. DOI: 10.17487/RFC7011. [Online]. Available: https://www.rfc-editor.org/info/rfc7011.

[19] J. Scudder, R. Fernando, and S. Stuart, *BGP Monitoring Protocol (BMP)*, RFC 7854, Jun. 2016. DOI: 10.17487/RFC7854. [Online]. Available: https://www.rfc-editor.org/info/rfc7854.

[20] FRRouting, *FRRouting*, Online, visited on: 2026-01-15. [Online]. Available: https://frrouting.org/.

[21] FRRouting, *FRRouting Developer's Guide – Northbound API*, Online, visited on: 2026-01-15, 2017. [Online]. Available: https://docs.frrouting.org/projects/dev-guide/en/latest/northbound/northbound.html.

[22] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999. DOI: 10.1126/science.286.5439.509.

Portable IoT Platform for Experimental Cadaverine Measurements

1st Sebastian Hauschild
Luebeck University of Applied Sciences
Center of Excellence CoSA
Luebeck, Germany
sebastian.hauschild@th-luebeck.de

2nd Horst Hellbrück
Luebeck University of Applied Sciences
Center of Excellence CoSA
Luebeck, Germany
horst.hellbrueck@th-luebeck.de

***Abstract*—The use of cadaverine-selective cantilever sensors on IoT platforms to predict the shelf life of meat represents a novel approach in food processing. These sensors generate large volumes of measurement data directly at the source, requiring software-based analysis and machine learning. Assessing meat freshness in industrial environments is challenging because network stability and availability cannot always be guaranteed indoors. This may interrupt data evaluation and hinder real-time analysis as well as data uploads. To address these issues, this work proposes a portable IoT measurement platform designed to receive experimental sensor data. The platform integrates multiple wireless technologies and middleware components to enable local data acquisition, processing, storage and visualization of sensor data.**

***Index Terms*—IoT-Platform, Machine Learning, Database, Distributed Systems, Cantilever**

I. Introduction

The use of sensors on IoT platforms has become increasingly important in food processing. The sensors generate large amounts of measurement data at the source [1]. Analyzing and interpreting this data requires software based analysis and machine learning methods [2]. To execute the algorithms IoT platforms are often connected to powerful cloud systems via the internet [3]. One special application on IoT platforms in food processing is the use of wireless cadaverine-selective cantilever sensors based on cyclam derivatives to detect the freshness of meat in industrial environments [4]. A complex issue in this environment is the interruption of the evaluation of measurement data, because the stability and availability of network connections are not always guaranteed when the sensors are used indoors [5], [6]. In particular the problem is critical for processes that rely on real time data analysis and for scenarios involving large scale data uploads from multiple cadaverine sensor systems [7]. Our approach is to extend the wireless cadaverine sensors with a federated AI evaluation. We will design and present a draft of a semi-stationary system that enables the wireless integration of sensors for detecting meat freshness, thereby reducing their energy consumption and processing time of computationally intensive AI algorithms. In addition, a portable signal generator and oscilloscope will be implemented to support testing and validation procedures. The contributions of this work are:

- Development of a portable IoT measurement platform for meat freshness detection based on cadaverine-selective cantilever sensors.
- Design and implementation of a distributed application network for recording, distributing, storing and processing sensor data for meat freshness detection.

This work is funded by the European Regional Development Fund Interreg Germany-Denmark within the project PRECISE.

II. Materials and System Draft

In previous work, we have investigated the problem of processing AI algorithms on systems with low computing power compared to cloud processing. Local systems with low processing power were found to be able to execute AI algorithms reliably and quickly. Especially in the case of a poor wireless connection between cloud and local system, local evaluation can be useful [8]. Regarding to our previous investigations we develop a semi-stationary IoT platform for cadaverine measurements. The raw sensor data of the cantilever sensor were stored in a SQL database in the earlier project phase. For experimental tests, the IoT platform implements the existing database and additionally sets up a data-processing pipeline to present, evaluate, and control the cantilever-sensor measurements for AI-training purposes. The training of the data is performed on the system-control block. The draft of the data-processing pipeline is shown in Figure 1.

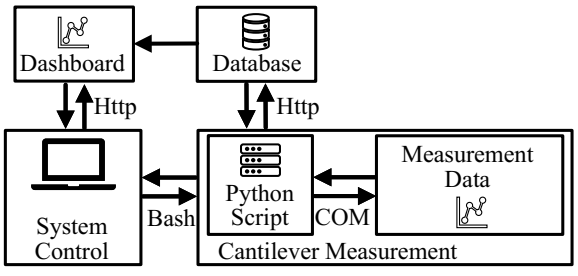

Fig. 1. Block diagram of the data processing pipeline

The IoT systems hardware is supplied with a total of 120 W via a POE switch GS305EPP. The power supplies (PS) for the

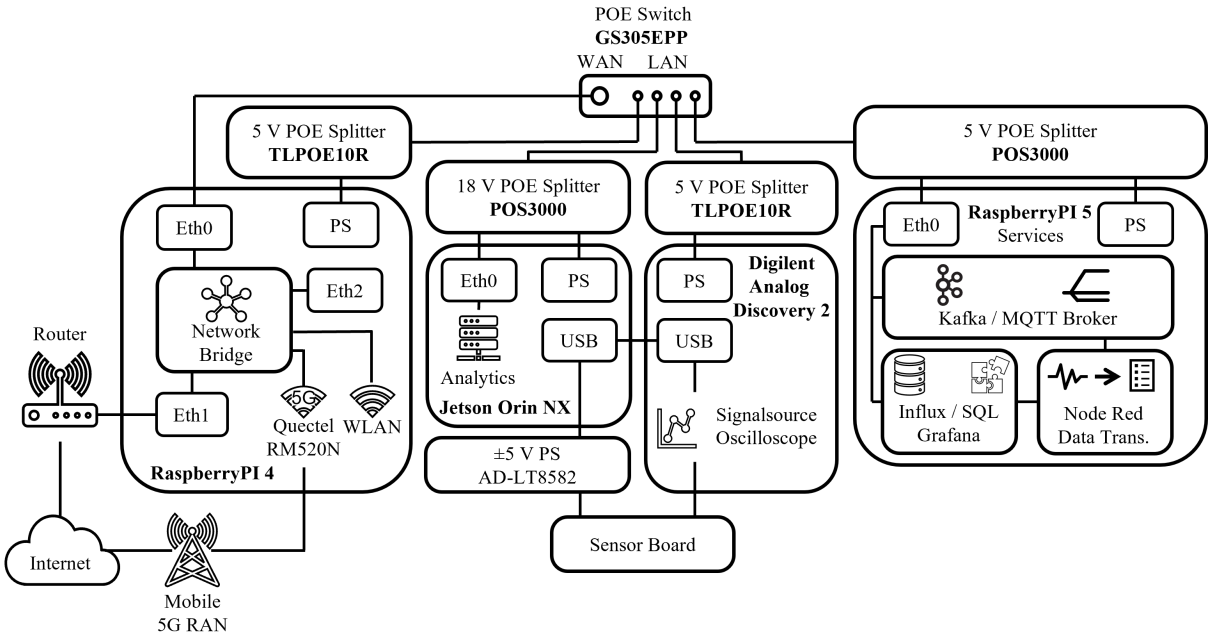

Fig. 2. Block diagram of the developed IoT platform showing network infrastructure, services, and measurement devices.

Raspberry Pi 4, Raspberry Pi 5, Digilent Analog Discovery 2, and Jetson Orin NX are provided by PoE-to-jack splitters (TLPOE10R, POS3000) connected to the GS305EPP PoE switch. An additional power supply board (AD-LT8582) for external sensors with an output power of ± 5 V is powered via USB by the Jetson Orin NX board as shown in Figure 2.

To provide external access to services and devices, Ethernet, WLAN and an experimental 5G interface (Quectel-RM520N) are connected to and managed by the Raspberry Pi 4.

Middleware services and databases such as Kafka, MQTT, NodeRed, InfluxDB and SQL are deployed on the Raspberry Pi 5 with a 1 TB NVMe using Docker to stream and store the sensor data. The sensor values during measurements are visualized via the InfluxDB platform together with a Grafana dashboard.

The Jetson Orin NX is used to achive the sensor data and to electronically control the sensors with help of the Digilent Analog Discovery 2. For further AI and machine-learning processing, parameters are saved in a structured H5 file on the Jetson Orin NX to ensure reproducible and compact storage. The file structure is related to the existing database schema from previous project measurements. An OLED display with keypad is connected to the Jetson Orin NX as the interface and operating unit. The result is a system as shown in Figure 3.

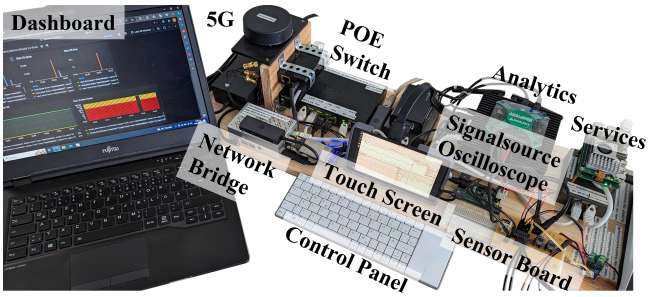

Fig. 3. Portable IoT-Platform for experimental cadaverine measurements.

III. Conclusion and Outlook

In this paper, we developed and presented a network architecture of a portable IoT platform. The platform supports local data acquisition and edge-based AI processing of sensor data. The focus of the application is on cadaverine-selective cantilever sensors for meat-freshness detection. The system can integrate multiple wired and wireless sensor types for experimental testing and measurements. The sensor data can be stored in a SQL and InfluxDB Database and visualized in real time on a Grafana dashboard. In future work, the systems networking performance and sensor-integration capabilities are further evaluated. Additionally, the performance and processability of the acquired data in the SQL database should be assessed in comparison with vector-database technologies such as Qdrant and time-series databases such as InfluxDB, in order to enable more advanced data presentation, clustering, and signal analysis.

References

[1] P. Rajak, A. Ganguly, S. Adhikary, and S. Bhattacharya, "Internet of things and smart sensors in agriculture: Scopes and challenges," *Journal of Agriculture and Food Research*, vol. 14, p. 100776, Dec. 2023.

[2] V. Zatsu, A. E. Shine, J. M. Tharakan, D. Peter, T. V. Ranganathan, S. S. Alotaibi, R. Mugabi, A. B. Muhsinah, M. Waseem, and G. A. Nayik, "Revolutionizing the food industry: The transformative power of artificial intelligence-a review," *Food Chemistry: X*, vol. 24, p. 101867, Dec. 2024.

[3] B. Sonkoly, D. Haja, B. Németh, M. Szalay, J. Czentye, R. Szabó, R. Ullah, B.-S. Kim, and L. Toka, "Scalable edge cloud platforms for

iot services," *Journal of Network and Computer Applications*, vol. 170, p. 102785, Nov. 2020.
[4] P. Ferrier, Y. Spethmann, B. Claussen, L. Nsubuga, T. L. Marcondes, S. Høegh, T. Heptaskin, C. Wiechmann, H.-G. Rubahn, and R. de Oliveira Hansen, "Application of a handheld electronic nose for real-time poultry freshness assessment," *Sensing and Bio-Sensing Research*, vol. 45, p. 100685, Aug. 2024.
[5] A. Aileen, A. D. Suwardi, and F. Prawiranata, "Wifi signal strength degradation over different building materials," *Engineering, MAthematics and Computer Science (EMACS) Journal*, vol. 3, no. 3, pp. 109–113, Oct. 2021.
[6] K. Adhinugraha, W. Rahayu, T. Hara, and D. Taniar, "Measuring fault tolerance in iot mesh networks using voronoi diagram," *Journal of Network and Computer Applications*, vol. 199, p. 103297, Mar. 2022.
[7] Y. Zhang, *Mobile Edge Computing for Beyond 5G/6G*. Springer International Publishing, Oct. 2021, pp. 37–45.
[8] S. Hauschild and H. Hellbrück, *Latency and Energy Consumption of Convolutional Neural Network Models from IoT Edge Perspective*. Springer International Publishing, 2022, pp. 385–396.