SmartEdge

Semantic Low-code Programming Tools for Edge Intelligence

This project is supported by the European Union's Horizon RIA research and innovation program under grant agreement No. 101092908

Deliverable D4.2

Second implementation of dynamic and secure swarm networking

Editor F. Cugini (CNIT)

Contributors R. Abu Bakar, F. Cugini, M. Mtiri, F. Paolucci (CNIT), J. J. Vegas Olmos, F. Alhamed (NVIDIA), N. Zilberman, H. Chen, D. Ding, P. Qian, C. Zheng (UOXF), P. Cudré-Mauroux, (FRIB), D.M. Nguyen, D. Le Phuoc, A. Le Tuan (TUB)

Version 1.0

Date 12 December 2024

Distribution PUBLIC (PU)

DISCLAIMER

REVISION HISTORY

Revision	*Date*	*Responsible*	*Comment*
0.1	*July 1, 2024*	*F. Cugini (CNIT)*	*ToC*
0.2	*September 10, 2024*	*F. Cugini (CNIT)*	*Partner contributions*
0.3	*October 31, 2024*	*F. Cugini (CNIT)*	*Revised version*
0.4	*November 30, 2024*	*F. Cugini (CNIT)*	*2nd Revised version*
1.0	*December 12, 2024*	*R. Wenning (ERCIM)*	*Quality Check*

LIST OF AUTHORS

Partner	*Name Surname*
UOXF	*Noa Zilberman, Peng Qian, Changgang Zheng, Damu Ding, Hongyi Chen*
CNIT	*Filippo Cugini, Mohamed Mtiri, Francesco Paolucci, Rana Abu Bakar*
NVIDIA	*Juan Josè Vegas Olmos, Faris Alhamed*
TUB	*Duc Manh Nguyen, Danh Le Phuoc, Anh Le Tuan*
FRIB	*Alberto Lerner, Philippe Cudré-Mauroux*

GLOSSARY

Acronym	***Description***
ACC	Accuracy
ACK	Acknowledgment
ACL	Access Control List
AP	Access Point
AUC	Area Under the ROC Curve
BMV2	Behavioral model version 2
BT	British Telecom
CDF	Cumulative distribution function
CNN	Convolutional Neural Networks
CP	Control Plane
CPU	Central processing unit
DDoS	Distributed denial-of-service
DDS	DDoS Detection System
DHCP	Dynamic Host Configuration Protocol
DL	Deep Learning
DP	Differential Privacy
DPDK	Data Plane Development Kit
DPI	Deep packet inspection
DPU	Data Processing Unit
DT	Decision Tree
FL	Federated Learning
FNR	False negative rate
FTG	Flow Traffic Graph
GNN	Graph neural network
GRU	Gated Recurrent Unit
ICMP	Internet Control Message Protocol
ID	Identifier
IDS	Intrusion detection and prevention system
ILP	Integer linear programming
INT	In-Band Telemetry
IoT	Internet of Things
ITS	Intelligent Transportation System
ISP	Internet service provider
KM	k-means
MAC	Media Access Control
ML	Machine Learning
M/A	Match-Action
NB	Naïve Bayes
NIC	Network Interface Card
NN	Neural Network
PDP	Programmable data plane
P4	Programming Protocol-independent Packet Processors
QoS	Quality of Service
RF	Random Forest
RSU	Road-side unit

RTT	Round Trip Time
SQL	Structured query language
SVM	Support Vector Machine
TCP	Transmission Control Protocol
UC	Use Case
TPR	True positive rate
UDP	User Datagram Protocol
URLLC	ultra-reliable low-latency communications
UUID	Universal Unique Identifier
VLAN	Virtual local area network
VNID	VXLAN Network Identifier
VXLAN	Virtual Extensible LAN
WP	Work Package
XGB	XGBoost

EXECUTIVE SUMMARY

Deliverable 4.2 is the second iteration of the Dynamic Swarm Networking technical Work Package. The Work Package combines three tasks: (1) Automatic discovery and dynamic network swarm formation (2) Embedded network security and isolation and (3) Hardware accelerated in-network operations.

These three intertwined tasks lead the way to enabling the networking layer of the SmartEdge project and for providing high performance and low latency operation through innovative network-programmability based solutions.

The requirements set in SmartEdge's global architecture have been taken into consideration, and an implementation that has been built upon the SmartEdge use cases, which adheres to the architecture of the proposed Work Package.

This deliverable presents the advancements and contributions of Work Package 4 (WP4), which focuses on the development and optimization of intelligent swarm systems. It covers critical areas such as swarm management, communication, security, and performance improvement, organized into six primary sections.

The document begins by outlining the objectives of WP4 and its contributions to the broader project. It highlights the relationships among key artifacts and provides updates on performance indicators, ensuring alignment with the project's strategic goals. A summary of the requirements addressed by WP4 follows, covering essential aspects such as swarm management, communication, identity management, policy and security, and application lifecycle management. Additional areas such as low-code programming and semantic integration are also discussed, along with a forward-looking plan for future Releases.

A significant portion of the deliverable is dedicated to mechanisms for automatic discovery and dynamic swarm formation. This includes innovations such as P4 programmable data planes, enabling flexible and efficient swarm networking, as well as detailed architectures and tools for managing node inclusion, controlled exits, and recovery. Preliminary results demonstrate the effectiveness of these approaches, supported by performance analysis.

The deliverable also focuses on embedded network security and isolation. It describes advanced techniques such as programmable access control for swarm intelligence, machine-learning-driven in-network defenses, and federated mechanisms for detecting and mitigating threats. Innovative solutions for DDoS detection, leveraging hierarchical graph neural networks and programmable switches, are also explored.

In the context of hardware-accelerated operations, the report showcases significant advancements, including the development of a secure authentication protocol (DPUAUT) integrating SmartNICs, and the design of intelligent roadside and edge units. These contributions, such as the Street Intersection Edge Unit (SINED) and its Bird's Eye View (BEV) capabilities, demonstrate the potential for hardware solutions to enhance swarm intelligence and situational awareness.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

1. Introduction and Objectives

1.1 Overview

Work Package 4 (WP4) focuses on dynamic and secure swarm networking, specifically addressing the communication among SmartEdge system components and fostering swarm intelligence within the network. This Work Package tackles three key networking aspects: automatic detection and formation of dynamic network swarms, integrated network security and isolation, and hardware-accelerated in-network processing. These areas are developed in three interconnected tasks. Structurally, the Work Package operates between the network's physical layer and the software application layer, essentially in the middle layer. It utilizes existing physical layer communication standards, such as 4G/5G, Wi-Fi and other wireless protocols, without modification. Leveraging network programmability, it innovates from the network layer (layer 3 of the OSI model) upwards. By transferring certain processes from computing resources into the network, the Work Package seeks to boost performance, reduce latency, improve security, and add new functionality.

WP4 is divided into three tasks, each aligned with one of the objectives. The tasks are:

T4.1. Automatic Discovery and Dynamic Network Swarm formation in near real time
T4.2. Embedded network security and isolation
T4.3. Hardware-accelerated in-network operations for context-aware networking

The three tasks and their relations are illustrated in Figure 1.1.

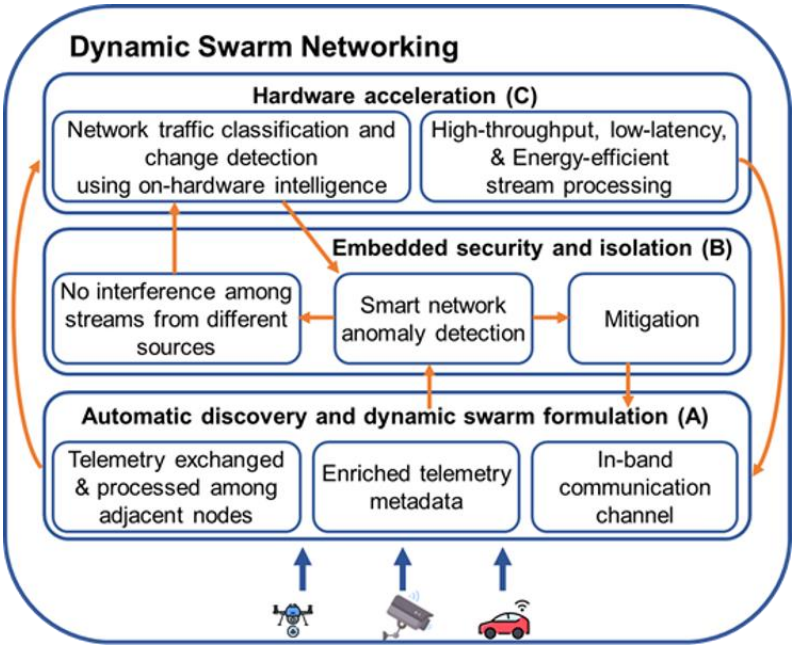


Figure 1.1: High Level Architecture of WP4

Task 4.1 (T4.1, marked A in the figure) is responsible for the formation of the swarm. It detects new nodes and decides if and how to add them to the swarm. An in-band communication channel will be employed that will enable innovation in intelligent network telemetry and its

processing. The elements designed as part of the tasks are tightly integrated, providing an important component in every swarm node.

Task 4.2 (T4.2, marked B in the figure) is responsible for embedded network security and isolation. The task innovates in moving network security intelligence into network devices, primarily using in-network machine learning (ML) for anomaly detection and mitigation. The task is responsible to mitigate detected threats and to isolate traffic streams from different sources, so that nodes outside the swarm cannot interfere with its operation. It does so by using a mix of software-based and hardware-based programmable network devices.

Task 4.3 (T4.3, marked C in the figure) relates to hardware-acceleration of in-network operation for stream processing and network security applications. This includes, for example, hardware based sensor fusion and sensor data pre-processing for data reduction.

The three tasks are intertwined, with information exchanged between the different parts of the Work Package. T4.1 shares both swarm membership and status information with other tasks, allowing isolation of communication within the swarm by T4.2. Task 4.1 also shares telemetry information, used as features in the ML based anomaly detection of T4.2. T4.2, in turn, reports anomalous nodes behavior, used to isolate malfunctioning nodes and exclude malicious ones from the swarm by T4.1. T4.3 enhances the functionality of both tasks by providing hardware acceleration, improving performance toward KPI goals, while stream processing acceleration supports the overall functionality of SmartEdge and intersects with WP5.

This document reports the status of the activities at M22 for all three tasks. In particular:

- Section 2 lists the requirements related to WP4. For each relevant requirement, it provides the current status of implementation within the SmartEdge Release 1.
- Section 3 details the implemented Release 1 solution for automatic discover and dynamic swarm formation, i.e. related to the aforementioned objective 1.
- Section 4 focuses on the design for embedded network security and isolation, related to the aforementioned objective 2.
- Section 5 discusses hardware accelerated in-network operations (objective 3).

1.2 OBJECTIVES

WP4 has three objectives:

1. *Automatic Discovery and Dynamic Network Swarm formation – providing the functionality required to create a swarm of SmartEdge devices, including adding, maintaining and removing nodes.*

In D4.1, the Automatic Discovery and Dynamic Network Swarm formation solution was designed. Furthermore, D4.1 reported on the development of telemetry components that are operational and will be integrated in Release 2.

In this Deliverable, D4.2 (Section 3), we report on the first implementation (Release 1) of the Automatic Discovery and Dynamic Network Swarm formation procedure. The implementation is discussed against the specific requirements provided by WP2 and reported in Section 2.

2. *Embedded network security and isolation – providing network security for nodes that are part of the swarm, including protection against common network attacks (e.g., DDoS) and detection and mitigation of anomalies and emerging attacks. Note this does not cover end-to-end security (e.g., application level). The tasks also isolate swarm communication from nodes outside a swarm.*

In D4.1, the following embedded network security and isolation solution were reported (they are considered final and part of Release 1):

- A Runtime Model Update for In-network Defense

In this deliverable D4.2, the following embedded network security and isolation solutions are reported (they significantly enhance the preliminary solution reported in D4.1):

- A framework for rapid prototyping of in-network anomaly detection across a range of network devices (section 4.2.1)
- A federated learning solution for attack detection and mitigation of attacks using IoT devices (section 4.2.3)
- A DDoS detection and mitigation mechanisms for programmable network switches
- DDoS attack mitigation based on graph neural networks (See Section 4.4)

This deliverable D4.2 also includes the following novel embedded network security and isolation solutions (not reported in D4.1):

- A hybrid in-network defense against security threats solution (section 4.2.2)
- A secure DDoS mitigation solution for programmable switches sketch vulnerabilities (section 4.5)

3. *Hardware-accelerated in-network operations – providing acceleration to SmartEdge functionality by offloading it to run within the network. This functionality will range from network security to acceleration of stream processing, while building upon emerging network-hardware technologies such as Data Processing Unit (DPUs).*

In D4.1, the following Hardware-accelerated in-network operations were reported (They are considered final and part of Release 1):

- Distributed In-network Operations
- Hardware-accelerated DDoS Detection using Convolutional Neural Networks (CNNs) on DPU

In this deliverable D4.2, the following hardware-accelerated in-network solutions are reported (they significantly enhance the preliminary solutions reported in D4.1):

- Secure Authentication Protocol with SmartNiC Integration for Trustworthy Communications in Intelligent Swarm Systems (See Section 5.1)
- Intelligent road -side unit (section 5.2)

This deliverable D4.2 also includes the following novel hardware-accelerated in-network solutions (not reported in D4.1):

- Street Intersection Edge Unit (Section 5.3): this is an extension of the hardware-accelerated LiDAR approach that was initially introduced in D4.1. (See Section 5.3.3 in D4.1).

1.3 SUMMARY OF WP4 ARTIFACTS

WP4 designed and it is implementing the following SmartEdge Artifacts:

A4.1 Access Point Manager

Description: The Access Point Manager (APM) is an artifact involved in the swarm formation. It is a software component running on each Access Point providing connectivity to Smart Nodes. The APM monitors the connection/disconnection of Smart Nodes and enables communication within the swarm only among Smart Nodes whose Swarm Joining Request was approved by the Swarm Coordinator. Details in Section 3.

Lead: CNIT

Development status: Release 1 of APM supports all expected swarm functionalities. Testing is in progress. In Release 2 the APM will have full support of the heartbeat.

A4.2 Swarm Coordinator

Description: The Swarm Coordinator (SC) is an artifact involved in the swarm formation. It operates at the network level. It is a software component interacting with one or more Access Point Managers. In particular, the Swarm Coordinator manages the P4 switches inside each access point, ensuring that flow rules are updated to authorize communication strictly between swarm node members only. The SC uses control messages to manage the behavior and communication of connected nodes through the network control plane, enforces access control, and communicates with the Orchestrator through the Adaptive Swarm Coordinator (A5.3.1) for determining whether smart nodes can join the swarm. Details in Section 3.

Lead: CNIT

Development status: Release 1 of SC supports all expected intra-swarm functionalities. In Release 2, the SC will be enhanced to support inter-swarm communications.

A4.3 Swarm-node Manager

Description: The Swarm Node Manager (SNM) is an artifact involved in the swarm formation. It operates at the network level. It is a software component running on each smart node between the application and network layer. It continuously monitors the network interfaces status of the smart nodes and manages VXLAN tunnels for secure intra-swarm communications. It initially receives the swarm configurations from the Access Point Manager to enable communication with the Swarm Coordinator. Then, it exchanges swarm control messages to the Swarm Coordinator. Details in Section 3.

Lead: CNIT

Development status: Release 1 of Swarm Node Manager supports all expected node functionalities. In Release 2, the Swarm Node Manager will be enhanced to support telemetry (implemented – see D4.1 – but not integrated yet) and have full support of the heartbeat.

A4.4 P4 Switch

Description: The P4 switch is an artifact involved in the swarm formation. Two programs are available, one to be executed at the Access Point, one to be executed at each Swarm Node. In

either case, this artifact manages the P4-based packet forwarding, guaranteeing isolation and communication restricted to authorized swarm members only, preventing unauthorized packet forwarding. P4 entries are received at the Access Point from the Access Point Manager and from the Swarm Coordinator. P4 entries at the swarm node are received from the Swarm Node Manager. Both P4 switches at the Access Point and Swarm Nodes can also receive flow rule entries through the P4 plugin developed within WP5 (Artifact A5.4.3.1). Details in Section 3.

Lead: CNIT

Development status: Release 1 supports all expected intra-swarm functionalities. In Release 2, the P4 Switch will be enhanced to support inter-swarm communications.

A4.5 Distributed Database for Network Information

Description: The Distributed Database for Network Information, also called Address Resolution Table (ART), is an artifact involved in the swarm formation. It serves as a component for storing and managing essential networking information about the swarm nodes, including their status, IP address associated with its Uniform Resource Identifier, etc. This information is used by the swarm coordinator to make networking configurations regarding node admission and to enforce access control policies. The ART database helps in maintaining an up-to-date record of the swarm's membership and activity at the network level. The ART database is continuously updated as nodes connect or disconnect from the swarm. Details in Section 3.

Lead: CNIT

Development status: Release 1 implementation relies on Cassandra database. A different technology may be adopted in Release 2.

A4.6 Programmable Access Control for Swarm Intelligence

Description: The Programmable Access Control leverages P4-programmed switches to enforce a flexible, dynamic Access Control List (ACL) for secure swarm communication. By embedding node identifiers (e.g, URIs) in packet headers, traffic from malicious nodes or different swarm is blocked, unapproved protocols are dropped, and VXLAN ensures swarm traffic isolation. Only swarm coordinators interact externally. Programmable controls enable runtime updates, enhancing security and adaptability. Successful test cases verified the correct blocking of unauthorized nodes and traffic isolation based on Swarm IDs. Details in section 4.

Lead: UOXF

Development status: Release 1 implementation includes tested P4 code allowing runtime modification of ACL list, and integrated with A4.4.

A4.7 Anomaly detection

Description: The anomaly detection artifact leverages an in-network machine learning (IN-ML) solution deployed on edge programmable devices to enable rapid attack detection and mitigation within IoT gateways and edge systems. By offloading ML inference to programmable data planes, it ensures swift, in-pipeline threat mitigation while maintaining high-speed network traffic. Key features seamless updates of ML models to address evolving threats, portability across platforms, resource efficiency and high detection rate, thus achieving flexible, and efficient defense against emerging attacks. Details in section 4.

Lead: UOXF

Development status: The framework implementation is completed and included in Release 1.

A4.8 Distributed in-network Operations

Description: The objective of distributed in-network computing in SmartEdge is to overcome resource constraints in programmable network devices by efficiently partitioning and distributing complex services across multiple devices. This ensures scalability for intelligent operations within device swarms while maintaining functionality, minimizing latency, and meeting network constraints. Details in section 5.

Lead: UOXF

Development status: The framework implementation is completed and included in Release 1.

A4.9 Intelligent road-size unit

Description: The intelligent RSU in transportation systems enables V2X communication and edge computing, reducing latency and network load. In a smart junction scenario, it integrates data from radars, cameras, LiDAR, and GPS for time-critical services like collision avoidance. Algorithms designed for multi-sensor fusion and deployment on FPGA hardware help to ensure extremely low processing latency. Key metrics include high precision, low processing latency per frame, and efficient memory use. Details in Section 4.

Lead: UOXF

Development status: Release 1 implementation relies on SORT algorithm for multi-sensor fusion, and it achieves 99.75% accuracy. Current software latency is 4-5 ms/frame. In Release 2, we anticipate hardware acceleration using Kria KR260 FPGA targeting 1 ms/frame processing.

A4.10 Street Intersection Edge Unit

Description: The objective of the Street Intersection Edge Unit is to pre-process LiDAR (i.e., point cloud) data on the edge of the network using hardware acceleration. Details are provided in section 5.3.

Lead: FRIB

Development status: The component is currently being implemented taking advantage of low-level, hardware acceleration leveraging an FPGA. It will be part or Release 2.

1.4 HIGH-LEVEL RELATION AMONG WP4 ARTIFACTS

WP4 artifacts can be classified in four main categories, as shown in Figure 1.2:

- Artifacts running in Swarm Nodes, providing secure swarm networking capability within each swarm node. This category includes the Swarm-node Manager (A4.3) and the P4 Switch (A4.4), and Street Intersection Edge Unit (A4.10).
- Artifacts running in Access Points or Road-Side Units, providing secure swarm networking capabilities in the whole network, i.e. enabling communication among all

swarm nodes. This category includes: Access Point Manager (A4.1), P4 Switch (A4.4), Programmable Access Control for Swarm Intelligence (A4.6), Anomaly detection (A4.7), Distributed in-network Operations (A4.8), and Intelligent road-size unit (A4.9).

- Artifacts providing network swarm coordination and network intelligence. These artifacts could be collocated with Access Points or Road-Side Units (as shown in the figure below), but could also be located on different/independent edge nodes. This category includes the Swarm Coordinator (A4.2),
- Artifacts storing relevant swarm information. This category includes the Distributed Database for Network Information (A4.5) also called ART.

The Artifact providing network swarm coordination and network intelligence, by leveraging the information stored in the distributed database, enables the proper configuration of all data plane network elements, i.e. the Access Points / Road Side Units and the Swarm nodes. Such elements exploit advanced network programmable solutions to guarantee effective forwarding, isolation, in-network processing, and embedded cyber-security.

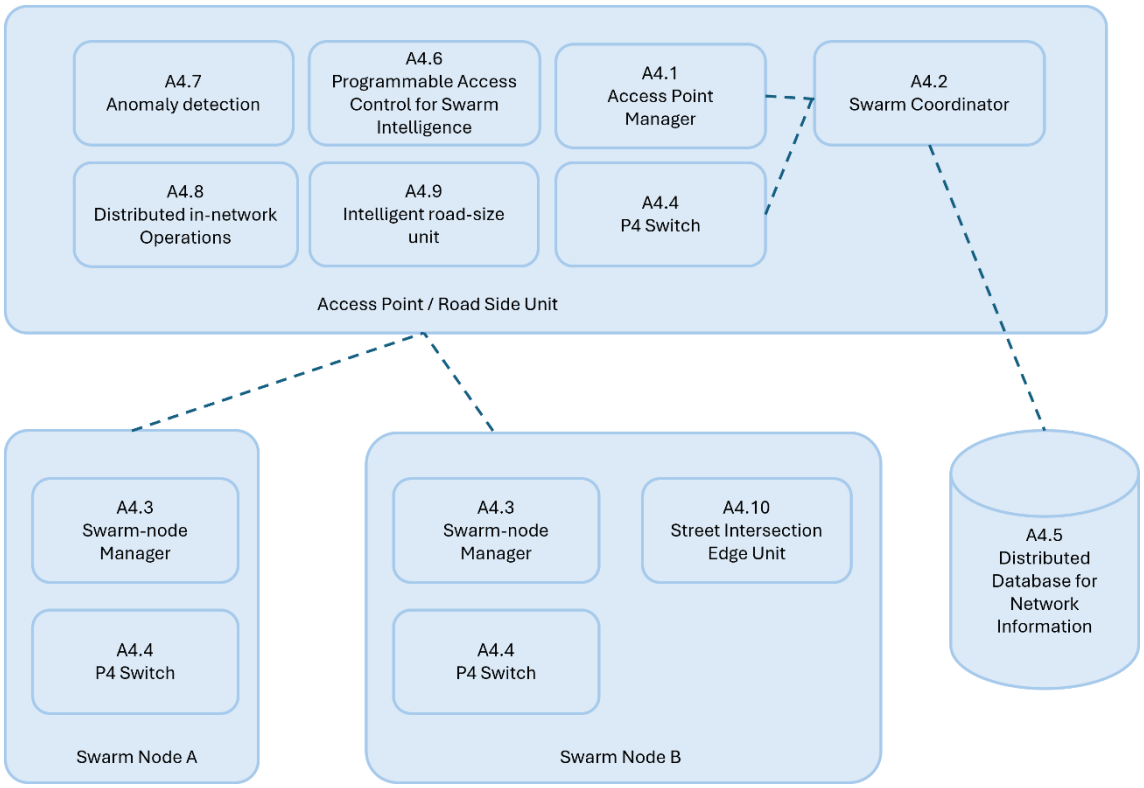


Figure 1.2: High level relation among WP4 artifacts. Dotted lines indicates main interactions among artifacts/nodes

1.5 Update on Key Performance Indicators

Although Release 1 of the WP4 SmartEdge solution is not yet adequate to verify the project objectives, it is important to monitor the progress towards the relevant KPIs.

The most relevant Key Performance Indicators (KPI) associated with WP4 are the following:

K3.1: time required to perform automatic discovery and dynamic network swarm formation, reduced by 60% thanks to HW-accelerated in-network operations compared to regular processor-based execution.

Baseline: perform automatic discovery and dynamic network swarm formation using software-based forwarding components. Forwarding time in the range between 100micorseconds and few milliseconds, depending on CPU performance and software and system overhead.

The SmartEdge Release 1 of the automatic discovery and dynamic network swarm formation solution has been implemented on software systems (e.g., P4 switch) running in wireless Access Points and Swarm Nodes. Work is described in Section 3. To demonstrate the KPI, the P4 program will be also implemented on a different HW target, such as a Tofino ASIC. This will allow us to demonstrate the expected time reduction when HW-accelerated solutions are adopted. Specific measurements will be implemented during Year 3 to demonstrate the KPI.

K3.2: Increase performance by enabling in-network processing to reduce end-to-end latency (by 50% or more, assuming propagation delay is not the dominant component) and reduce variability.

Baseline: perform processing and machine-learning inference using software-based components. Processing time in the range between 4-20ms for data fusion and >1ms for inference, depending on CPU performance and software and system overhead.

Partially achieved. As reported in D4.1, ultra-low latency, including <1µS on a switch, <1ms in container, and ~2ms on a Raspberry Pi has been demonstrated running an ML-driven in-network defense solution. To demonstrate the KPI for data fusion, hardware-accelerated solution is developed (section 5.2) and specific measurements will be implemented during Year 3 to demonstrate the KPI.

K3.3: Provide isolation to data streams and a distributed security barrier for smart anomaly detection (80% or more of events detected and mitigated within the network, with 97% accuracy / F1 score >90%, detection provided in 1µs by ML algorithms (e.g., decision tree) optimized to run on Deep Processing Unit (DPU), >100x faster wrt. CPU baseline.

Baseline: perform smart anomaly detection using public datasets and compare with CPU based state-of-the-art solutions.

Partially achieved. As reported in D4.1, ML-driven in-network defense solution achieves accuracy of >99% and F1>94% for known attacks and 90%-99% accuracy with F1>94% for new attacks. Microsecond-scale latency achieved on both Intel Tofino and AMD Alveo U280. Furthermore, relevant work targeting smart anomaly detection is described in Sections 4.2 “ML Driven In-Network Defense” and 5.1 “DPUAUT: Secure Authentication Protocol with SmartNiC Integration for Trustworthy Communications in Intelligent Swarm Systems”.

K3.4: Enable distributed operation of a large number of heterogeneous IoT devices and smart systems (1K devices or more handled by HW accelerators ~~DPU only~~, in-line with expected number of devices/vehicles to be considered in the SmartEdge scenarios) to achieve higher resilience.

Note, the original description of KPI3.4 has been updated to better reflect the whole project activities, i.e. expanding from “DPU only” to multiple types of HW accelerators (e.g., FPGA, P4 ASIC).

Baseline: 100 devices using software forwarding elements

Partially achieved. As reported in D4.1, distributed in-network computing across 1000 nodes has been successfully demonstrated, in-line with expected number of devices/vehicles to be considered in the SmartEdge scenarios.

In this document, a secure authentication solution for swarm systems has been designed and validated, specifically targeting DPU. The solution successfully supports >1k devices. Results are reported in Section 0, “DPUAUT: Secure Authentication Protocol with SmartNIC Integration for Trustworthy Communications in Intelligent Swarm Systems”.

In addition, to demonstrate the tracking of 1000 devices/vehicles, specific measurements will be implemented during Year 3 to demonstrate the KPI on the intelligent road side unit.

2. List of Requirements addressed by WP4

The requirements for WP4 addressing secure swarm networking are set by the overall architecture of the SmartEdge project as defined by WP2. In this document, we refer to the detailed list of requirements reported in Deliverable D2.2. In particular, we focus on the following requirements listed in D2.2, since they include most of the requirements that are revenant for WP4.

- Swarm Management
- Swarm Communication
- IDM and Policy & Security
- Application Lifecycle
- Hardware and Protocols

For each requirement we provide the first mapping with the WP4 artifacts/activities. Full/updated mapping of requirements will be provided by D6.1 while addressing the integration with artifacts provided by WP3 and WP5. D6.1 will also further clarify which are all the artifacts fulfilling each requirement.

To visually represent the development status, the following colors are used:

- Green: development is complete, and integration/testing/refinement is in progress within WP6
- Light blue: development either in progress or to be provided in Release 2. No issues expected.
- Red: development either in progress or to be provided in Release 2. Problems or delays are expected.
- Grey: requirement Not relevant for WP4 (i.e. addressed by WP3/WP5)

It is important to highlight that no critical delays are currently envisioned, i.e. none of the following requirements is highlighted in red.

2.1 Swarm Management

Requirements around swarm management deal with the creation and definition of swarms as well as the execution of recipes within the swarm. This includes, among other things, actions such as the addition and deletion of nodes from the swarm.

Table 2-1 Swarm Management list of requirements

ID / Ver.	Description / Status
SW-001 v1.1	A swarm node is an autonomous entity that can participate in a swarm. For example, a brown field device that can be incorporated into a swarm.
	Status: Implemented in SmartEdge Release 1 for what concerns swarm networking. Testing and integration in progress. In the context of WP4, each Swarm Node includes a specifically designed P4 program and control software (Artifact A4.3) managing the swarm communication and swarm networking

	operations, enabling the node to operate as an autonomous entity that can participate in a swarm.
SW-002 v1.1	A swarm smart-node is an autonomous entity that can actively participate in a swarm and must be able to support its basic functionality. For example, the swarm smart-node must have the processing environments to be able to host the SmartEdge swarm components.
	Status: Implemented in SmartEdge Release 1. Testing and integration in progress. (See SW-001)
SW-003 v1.1	A swarm coordinator is a swarm smart-node that must support the advanced functionality necessary to form and maintain a swarm. For example, the ability to onboard or offboard nodes to and from the swarm, or in UC5 to provide medical staff with such node that support coordination, holistic management, and monitoring of a group of patients represented by their swarm nodes.
	Status: Implemented in SmartEdge Release 1. Testing and integration in progress. In the context of WP4, a specific Artifact A4.2 has been defined and implemented to support the swarm coordinator capabilities of networking and communication. This artifact operates in conjunction with the Access Point Manager (A4.1) and P4 switch programs (A4.4) to enable the communication within the swarm.
SW-004 v1.1	A manifest swarm must consist of one or more swarm smart-nodes and at least one of those smart-nodes must be a swarm coordinator. Typically, the swarm will consist of two or more swarm smart-nodes, but during swarm formation or reformation there may only be one seed swarm smart-node that will enlist other nodes into the swarm, to perform the desired behavior.
	Status: Implemented in SmartEdge Release 1. Testing and integration in progress. Once a Swarm is created, it always has a swarm node serving as Swarm Coordinator (A4.2). The implemented swarm formation procedure is reported in Section 3.
SW-005 v1.1	A swarm should have three or more swarm coordinators to provide resilience if a swarm controller is lost.
	Status: The Swarm Coordinator (A4.2) operating at the network level is stateless and relies on the ART. The Swarm Coordinator communicates with Access Point Manager (A4.1) and Adaptive Swarm Coordinator. Both Access Point Manager and Adaptive Swarm Coordinator should be provided with a list of static IP. Currently, only a single IP is provided. Extension to support multiple IPs is straightforward given the stateless behavior. It will be available in Release 2.
SW-006 v1.1	A swarm must have an odd number of swarm coordinators to prevent the “split brain” scenario.
	Status: refer to SW-005
SW-007 v1.1	The swarm coordinator(s) must maintain the state of the swarm. For example, which nodes are part of the swarm.
	Status: Implemented in SmartEdge Release 1 in terms of networking and communication states. To comply with SW-005, the Swarm Coordinator (A4.2) is stateless. However, it relies on Address Resolution Table (Artifact A4.5) to

	maintain the state related to networking data. Currently the ART is implemented using a Cassandra database as a temporary solution.
SW-008 v1.2	Swarm smart-nodes must reach a consensus on any action that affects the swarm. For example, the swarm smart-nodes must agree on the offboarding of a new node from the swarm. If a smart-node (A) requests to leave the swarm, but another smart-node (B) is relying on it to perform some operation, then smart-node (B) must have the opportunity to request smart-node (A) to remain in the swarm. There may be exceptions to this rule, e.g., if all swarm coordinators agree to eject a node (C) from the swarm, then node (C) cannot block this action.
	Status: The consensus mechanism for smart-nodes operates at the application layer. This aspect is outside the domain of WP4. However, once a decision is reached, the application will communicate with the network layer via an API to either accept or reject the requested action from the smart-node. The API implementation is in progress. Involved artifact: A4.2.
SW-009 v1.1	Any swarm smart-node can abstain on any action proposed by another swarm smart-node. This may happen if the action does not directly affect the smart-node. Abstaining from proposed actions is desirable, as it reduces the decision-making process and possible action contentions.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SW-010 v1.1	The swarm must have a process to break any action contentions automatically between swarm smart-nodes, even if this is a random decision. Otherwise, external intervention would be required, which goes against the concept of a swarm. Taking random actions can be an effective mechanism to break out of repetitive behavior that continually fails.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SW-011 v1.2	A swarm smart-node (A) must have the right to leave a swarm if it chooses, but another swarm swarm-node (B), can request it to remain if it needs the assistance of smart-node (A). Smart-node (A) must then decide if it stays or leaves the swarm.
	Status: planned to be implemented in Release 2. Artifacts: A4.1-2-3
SW-012 v1.2	Selected data originated by one node may be provided to multiple swarms. For example, when a node is providing a sensor stream that could be of use to multiple swarms.
	Status: support of static data sources at the network level to be provided for Release 2 (see SC-001). This task will be handled by the P4 Runtime Plugin developed within WP5. An API needs to be developed between the Orchestrator and the Coordinator running the Plugin
SW-013 v1.2	A swarm coordinator should only belong to one swarm at a time.
	Status: Implemented in SmartEdge Release 1. Testing and integration in progress. The implementation does not allow a swarm coordinator (A4.2) to coordinate more than one swarm. Multiple instances of swarm coordinators must be created to handle multiple swarms.
SW-014	A protocol must exist to onboard a smart-node into a swarm.

v1.1	Status: Implemented in SmartEdge Release 1. See the swarm discovery section and related preliminary results in Section 3.
SW-015 v1.1	A protocol must exist for a smart-node to request to join a swarm.
	Status: Implemented in SmartEdge Release 1. The specifically designed Swarm Communication protocol includes a Join Message enabling a smart-node to request to join a swarm (See Section 3)
SW-016 v1.1	A protocol should exist for a swarm to broadcast a request for a smart-node with a specified set of capabilities to join the swarm. Based on the responses received, the swarm will select a candidate and onboard the smart-node. This feature is particularly important for when a swarm is missing a critical smart-node to enable an operation to be performed. It may be useful for the responding smart-nodes to offer a time limited option to join the swarm. When the option expires there is no obligation for them to join the swarm.
	Status: to be implemented for Release 2. The intelligence of the swarm to detect a missing node should be implemented at the application layer.
SW-017 v1.1	There may be cases where a device (A) in a swarm may object to a device (B) joining the swarm. For example, if device (A) has prior knowledge of the untrustworthiness of device (B). In this case device (B) is not allowed to join the swarm. There may be some limits or time constraints on device (A) objecting to device (B).
	Status: Planned to be implemented in Release 2
SW-018 v1.1	A swarm must be able to handle smart-nodes by mapping their data into the same coordinate system regardless of their movement. For example, for two smart-nodes in a swarm to cooperate about the physical movement of an object, they must the same special frame of reference, or be able to transform the other smart-nodes frame of reference into their own. The same holds true for temporal mappings, i.e., all collaborating smart-nodes must share a common concept of time.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SW-019 v1.1	A swarm must handle units disappearing without proper protocol due to network failure or similar external factors.
	Status: Implemented in SmartEdge Release 1 at the Access Point Manager (A4.1). A swarm-node can disconnect/ disappear without a proper protocol and the swarm coordinator will manage it properly (e.g., the IP revoked, removed in the ART, flow entries in P4 switches removed)
SW-020 v1.2	A standard protocol for accepting new smart-nodes to join a swarm must be defined. Deprecation note: Duplicates SW-001, SW-002
	Status: Implemented in SmartEdge Release 1. To be complemented by WP5 logics. See SW-001 and SW-002.
SW-021 v1.1	Swarm must always know what units, data sources and capabilities are available. This is similar to SW-007, however, swarm unit might still be present just unable to operate at full capacity, so this is a bit wider concept.

	Status: Implemented at the network level in SmartEdge Release 1. The Distributed Database for Network Information (Artifact A4.5), also called ART, includes a dedicated column to list the active swarm nodes in the swarm, together with their IP address. Currently, the ART is implemented using a Cassandra database as a temporary solution. Data sources (e.g., camera, sensors) that are not swarm nodes are listed in a static database.
SW-022 v1.1	The swarm smart-nodes and the swarm must have a universally unique identifier (e.g., UUID), encoded in the form of a URI. In many use cases when assigning a URI to a smart-node it is not possible to have a single controlling agency to issue the URI or provide an Internet (or even intranet) resolvable URL. Whilst UUIDs have no implicit meaning, but simply strings of hexadecimal characters, they do (for all intents and purposes) guarantee that the resource is uniquely identified.
	Status: Implemented at the network level in SmartEdge Release 1. The Distributed Database for Network Information (Artifact A4.5), also called ART, includes a dedicated column to list the UUID of each active swarm nodes in the swarm. This allows to correlate the actually available swarm nodes with their capabilities stored in the TDD. Full integration with TDD and related Project artifacts to be performed in WP6
SW-023 v1.1	A swarm can be the instantiation of a recipe. A recipe can exist before any nodes are members of the swarm, i.e., before it has been fully manifest. Such a recipe has an objective and ID, but no member nodes.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SW-024 v1.2	The swarm Federated is the engine that executes a queries/DSL-based workload (e.g primitive recipe) at runtime. It has to check that all capabilities required for the swarm functionality, defined in a query, are available and can be executed.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SW-025 v1.1	The swarm can be made up of homogeneous or heterogeneous swarm nodes and capabilities. The swarm node can be all of the same type and capabilities, or they can be of different types and capabilities.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SW-026 v1.1	In many cases the swarm coordinator and orchestrator are coresident on the same swarm smart-node, but they can reside on different nodes.
	Status: Implemented in SmartEdge Release 1. The artifacts of AP Manager (A4.1), Swarm Coordinator (A4.2), and ART table (A4.5) can be deployed either on the same or on different nodes since they are implemented as dockers and they communicate through API.
SW-027 v1.1	A cloud or brownfield system (i.e. can be part of a swarm and could be regarded as a type of swarm node with its own specific capabilities. This may be necessary if the swarm needs to communicate with legacy systems or use their resources. For example, to transfer data for long-term storage or deep data analysis. In this context, a brownfield system refers to a piece of equipment, hardware, or system that is already deployed and operational in an existing environment or infrastructure. It contrasts with greenfield devices or systems, which are new, built from scratch, and involve no prior constraints.

	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SW-028 v1.2	To keep latency low the Cloud should not take on the role of swarm coordinator or orchestrator. The Cloud should implement the SmartEdge stack but may be restricted to only certain swarm protocols. There may be challenges if the Cloud has too much control over the swarm. Deprecation Note: SmartEdge aims to develop tools that facilitate the use of edge computing.
	Deprecated.

2.2 SWARM COMMUNICATION

Requirements in the area of Swarm Communication deal with the communication withinSwarms. Thereby, minimum requirements for protocols to be used as well as latency requirements are set, and messages are defined which are necessary for the Swarm management.

Table 2-2 Swarm Communication Requirements

ID / Ver.	**Description / Status**
SC-001 v1.2	The swarm must be able to integrate brown field devices which cannot be extended with SmartEdge software (as required by the use cases).
	Status: The Swarm Coordinator (Artifact A4.2) / Access Point Manager (A4.1) installs flow rules to enable communication from/to data sources (i.e., static IP belonging to the non-swarm) to/from swarm nodes. To be implemented for Release 2
SC-002 v1.1	SmartEdge must provide a procedure so that a Swarm can communicate events about its state to swarm participants that registered for certain swarm events.
	Communications events are currently logged in a file. Communication of events will be enabled for Release 2.
SC-003 v1.1	It must be possible to register for application specific events that can be published by SmartEdge applications. The events must support application specific data if the application needs to communicate payload data.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SC-004 v1.1	It must be possible to set a time to live –TTL- window until an event is valid. After that it is considered outdated and can be removed.
	Status: Implemented in SmartEdge Release 1 for what concerns the networking aspects. At the network level, in case the configured number of heartbeats is missed, the swarm node is removed from the swarm by the Access Point Manager (A4.1)
SC-005	Events that are propagated by a swarm node should have a unique name space and ID so they will not conflict with application specific events.

v1.1	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SC-006 v1.1	Events should contain data about the creation source, time, and date.
	Status: Implemented in SmartEdge Release 1. Implemented as log files by the WP4 Artifacts (A4.1, A4.2, A4.3).
SC-007 v1.1	SmartEdge swarm events shall be only created by the swarm itself (and not by any application that uses SmartEdge).
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SC-008 v1.1	Internal swarm communications must be limited to only current nodes in the swarm, e.g., information about the swarm must not leak out of the swarm or be sent to nodes that have been offboarded from the swarm.
	Status: Implemented in SmartEdge Release 1. The P4 programs are enforced with flow rules that enable only a swarm node to communicate within the swarm. (A4.4)
SC-009 v1.2	The latency of controller status data should be less than 100 ms.
	Status: Implemented in SmartEdge Release 1. This KPI is achieved when pre-authorization in the joining phase is enabled. Testing and preliminary KPI validation in progress (A4.2)
SC-010 v1.1	A network device characteristics message must be implemented to obtain the current functional characteristics of a swarm device, both internal and external to the swarm. The network message is intended to be used during the runtime operation of the swarm, rather than at design time. Internally it is used to obtain detailed current information about other devices in the swarm, e.g., remaining battery life. Externally it is intended to locate devices that might be enlisted into the swarm, e.g., to provide functionality required byan application that the swarm does not currently have.
	Status: Implemented in SmartEdge Release 1. The Heartbeat is sent by each swarm node to the Access Point and it includes as payload the UUID and selected device characteristics. Internal: If a swarm belongs to a swarm, the selected device characteristics (e.g., battery life status) can be conveyed as well through the Heartbeat message. External: if a Swarm Node does not belong to any Swarm, it sends the Heartbeat to its Access Point. In this case, the UUID is used, in conjunction with the TDD, by Swarm Coordinators to locate nodes that might be enlisted in their swarm. (A4.3, A4.5)
SC-011 v1.1	A network swarm characteristics message must be implemented to obtain the current functional characteristics of a swarm by an external device. The network broadcast message is intended to be used during the runtime operation of the swarm, rather than at design time. It is used by external devices to find swarms in their locality and their characteristics.
	Status: to be implemented for Release 2. See SW-016

SC-012 v1.1	A network node join message must be implemented for a swarm to request an external smart-node to join a swarm, or an external smart-node requesting to join a swarm. This would typically follow a network node characteristics message, where the swarm has identified a suitable smart-node and is requesting it to join the swarm. Alternatively, it could be used by a smart-node that has discovered a swarm and wishes to join it.
	Status: Implemented in SmartEdge Release 1 with reference to the networking aspects. The implemented swarm joining procedure is detailed in Section 3. The external smart-node requesting to join a swarm will be provided with Release 2. (A4.3)
SC-013 v1.1	A network device leave message must be implemented for a swarm device to request to leave a swarm it is currently part of. This message must be broadcast to all other members of the swarm.
	Status: partially implemented in SmartEdge Release 1. The Swarm Communication leave message is implemented, but, currently, not broadcasted. When the swarm node leaves, it is removed from the ART. Swarm Nodes can read the ART to be aware of active nodes in the swarm. Currently, the leave message is not broadcast to all the other swarm members. (A4.2, A4.3)
SC-014 v1.1	A network node remain message must be implemented for a swarm smart-node to request a leaving smart-node to remain in the swarm as it needs its services.
	Status: partially implemented in Release 1. Current version enables the delivery of swarm messages. Current version implements the message handling of Join and Leave. Type Remain to be developed for Release 2.
SC-015 v1.1	A network protocol must exist to setup a data stream feed between devices in a swarm. This may be achieved through a publish and subscribe mechanism, or a streaming data protocol. For example, an AMR may wish to constantly receive video images from several ceiling cameras. There must also be a mechanism to stop the feed.
	Status: Partially implemented in SmartEdge Release 1. The P4 switch of any swarm node and AP can configured to support the forwarding of selected data streams. The pub-subscribe mechanisms to enforce such configurations to the P4 switches will be implemented in Release 2 in collaboration with WP3/5.
SC-016 v1.2	Latency of radar message data should be less than 100 ms.
	Status: sensor fusion within the network is currently under development and will be included in Release 2. Other components are developed under WP5.
SC-017 v1.2	Latency of camera message data and processing (AI operations) should be less than 500 ms.
	Status: under assessment.
SC-018 v1.2	Latency of all V2X message data should be less than 100 ms.
	Status: under assessment.

SC-019 v1.1	There has to be sanity checking of data coming from the streams. In particular, in UC-2 the long-term solution is to allow "outsiders" to send data from V2X capable vehicles, this has to be checked. In UC-5 each sensor data shall be checked against unique sensor properties set during the configuration by the authorised swarm admin.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SC-020 v1.1	There has to be a mechanism for handling messages arriving at varying levels of delay (due to processing and network delays)
	Not relevant for WP4 (i.e. addressed by WP3/WP5). This requirement mainly applies to the automotive use case. The reference scenario considers identifying the same car via different cameras and/or radar. Data fusion should be robust to processing and communication delays.
SC-021 v1.1	Referring to SC-20, the "too long" delays for messages should trigger event warning participants about possible issue.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SC-022 v1.1	A protocol must be implemented for smart-nodes in a swarm to exchange high-level semantic information about their environment. For example, a new mobile rack detected in the swarm's operational area, or an unexpected obstacle that had not previously been detected. This type of information exchange is not the same as low level sensor data exchanged between nodes in the swarm. Here the smart-node has synthesized the sensor data into a knowledge graph, see SCI-004, and is syndicating the high-level semantics.
	Not relevant for WP4 (i.e. addressed by WP3/WP5)
SC-023 v1.1	Each swarm unit as well as all the data streams and other capabilities have to be given a unique identifier (e.g., UUID).
	Status: Implemented in Release 1 with reference to networking aspects. Each swarm node is assigned a unique UUID (also embedded in the join swarm request). Related artifact: A4.3
SC-024 v1.1	Smart-nodes in a swarm must regularly exchange heartbeat messages to other smart-nodes in the swarm. Rather than each smart-node communicating to each other smart-node in the swarm, smart-nodes may syndicate recent heartbeat messages they have received. In this way other smart-nodes, and in particular the swarm coordinator(s), can assess the cohesion of the swarm without having to constantly communicate with other members of the swarm.
	Status: Implemented in SmartEdge Release 1. It has been implemented adopting the authentication and non-repudiation technique proposed by IMC. The heartbeat is sent every 1s (configurable) to the AP (unicast), which keeps track of active authenticated nodes through the ART. The ART can be read by any SN in the swarm. (A4.1, A4.3)
SC-025 v1.1	When a swarm coordinator sends a heartbeat message, the message should contain a hash of its swarm state. Smart-nodes can cache the hashes from the swarm coordinator(s), to identify if the swarm state changes. If swarm coordinator (A) detects that swarm coordinator (B)'s state has change, it will immediately initiate a swarm update to reconcile the two states.

	Status: to be implemented for Release 2.
SC-026 v1.1	Smart-node heartbeat messages should never be propagated outside of the swarm. Where a node is shared between several swarms or a Cloud is part of a swarm, they should not provide a bridge for heartbeat message.
	Status: Implemented in SmartEdge Release 1. All the inter swarm traffic is not propagated outside the swarm network. (A4.2)
SC-027 v1.1	Smart-nodes should be able to digitally sign events to achieve non-repudiation. The swarm infrastructure should provide support for registering a smart-node's public key. The PKI used should be compatible with low-power low-bandwidth devices
	Status: The current implementation uses WPA2-PSK for authentication. Non-repudiation or the events stored in the ART to be implemented for Release 2.
SC-028 v2.1	The SmartEdge environment should be aware of its network state and thus should detect a set of common anomalies.
	Status: Implemented in SmartEdge Release 1. The Access Point Manager is capable of detecting the sudden disconnection of a smart node and coordinator (network components) and it takes the necessary actions to update the network, the database, and the P4 switch rules to maintain accurate and secure network operations. (A4.1, A4.2, A4.5)
SC-029 v2.1	SmartEdge must provide time synchronization capabilities for sensor data of different sources. The accuracy must be milliseconds or better.
	Status: Implemented in SmartEdge Release 1 using PTP.
SC-030 v2.1	As SmartEdge applications are intended to run in a distributed manner and therefore generate network load, SmartEdge itself should keep its network load as low as necessary so as not to interfere with the applications.
	Status: broadcast messages at the network level are currently limited only to ARP (A4.4)

2.3 IDM AND POLICY & SECURITY

Requirements in the area of IDM and Policy & Security deal with privacy and security requirements within Swarms. Minimum requirements for data transport and content are defined.

Table 2-3 IDM and Policy & Security list of requirement

ID / Ver.	Description / Status
IDM-001	The visibility of events must be restricted to application, user, "swarm", or public level roles according to the system groups controlled by the swarm admin. (A4.1,

v1.1	A4.2, A4.3)
	Status: Implemented in SmartEdge Release 1 in terms of networking visibility
IDM-002 v1.2	A swarm can be created within a private secure environment where all nodes belong to the same owner. Nodes of other owners cannot join the swarm. (A4.3)
	Status: Implemented in SmartEdge Release 1. The swarm is a private secure environment enabled by the standard Wi-Fi in UC3 and by V2X in UC2. Static configuration of swarm creation implemented in SmartEdge Release 1.
IDM-003 v1.1	Swarm participants must support timely secure software updates to fix security flaws as the flaws are discovered, and the software is revised to address them.
	Note: This requirement was moved to deprecated since product-ready level software is not originally part of the proposal where TR Level 5 must be achieved.
IDM-004 v1.2	Swarms must have the means to detect, monitor and mitigate cyber-attacks. This involves monitoring suspicious behavior and providing the means to isolate compromised devices and block hostile network traffic.
	Status: Partially implemented in SmartEdge Release 1. Basic security policies are implemented, such as swarm nodes authentication and P4 flow rules, which ensure that only permitted nodes can communicate with each other. Malicious nodes can also be blocked. Solution to detect malicious nodes is available but still not integrated and is included in Release 1.
IDM-005 v1.2	Swarms must report suspected cyber-attacks and system failures.
	Status: Implemented in SmartEdge Release 1. An in-network ML based support for anomaly detection is implemented.
IDM-006 v1.1	Swarms should support the means to securely transfer critical participant roles to ensure robust operation of the swarm as a whole, for instance, when a device fails, or is compromised in a cyber-attack, or when the device is scheduled for hardware replacement or a lengthy software update.
	Status: partially implemented, validation in progress. The networking artifacts (A4.1-A4.4) are implemented as stateless elements, relying on the ART (A4.5) to keep state information. This enables effective transfer of participant roles
IDM-007 v1.1	Identification of objects and actors and participants within the system by globally unique Identifiers, including the grouping of those into a swarm.
	Status: Implemented in Release 1 with reference to networking aspects. Each swarm node is assigned a unique UUID along with the component's name. Involved artifacts: A4.1-A4.5
IDM-008 v1.1	Semantic data protection expressions to allow for sophisticated data handling and usage control within the system and between the participants.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
IDM-009 v1.1	Semantic expression about security properties needed for special data handling requirements like encryption.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
	Semantic policy expressions to express deontic requirements.

IDM-010 v1.1	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
IDM-011 v1.1	Expression of situation specific conditions that trigger obfuscation or blurring of images and mechanisms to execute.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
IDM-012 v1.1	Data retention policy. For example, if a swarm is made up of smart-nodes from individuals or different organisations, and the smart-node leaves the swarm, there must be a limit on how long a swarm can retain that nodes data.
	Status: Implemented in SmartEdge Release 1 with reference to networking aspects. Swarm node info is currently not retained at the network level (e.g., IP address). Involved artifacts: A4.1-A4.5
IDM-013 v1.2	Encryption of data in transit using hardware acceleration. This is particularly important when data is being transported across a publicly accessible fabric, such as the Internet.
	Status: HW-acceleration implemented in DPU. DPUs are used within edge computing nodes, e.g. part of the backbone in the factory floor that also interconnects access points. The smart-nodes are connected to an access point using a pre-shared key (PSK) and WPA2 (Wi-Fi Protected Access 2) for authentication, the communication between the nodes and the access point (Iner swarm communication) is encrypted.
IDM-014 v1.1	Security and compliance to regulations related to the use case domain e.g., machinery, healthcare, etc.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
IDM-015 v1.1	Consent management system and compliance with metadata.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
IDM-016 v1.1	Methods for aggregation and differential privacy applied to streams.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
IDM-017 v1.1	Capability to automatically update edge unit's software with new versions which have to be signed by author's key. This is a more general requirement of IDM-003.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
IDM-018 v1.1	There has to be a mechanism for signing all the data sources and ensuring theauthenticity of data.
	Status: Implemented in SmartEdge Release 1 with reference to networking aspects: Two mechanisms in place: (1) an innovative lightweight technique proposed by IMC applied to HeartBeat messages. (2) Standard authentication applied to swarm network messages. Involved artifacts: A4.1 and A4.3
IDM-019 v1.1	Communications emanating from the swarm should appear to come from the swarm and not a specific device, i.e., the internal structure of the swarm should not be ascertainable from its external communications.

	Status: Swarm Nodes do not communicate outside the Swarm
IDM-020 v1.1	There has to be a mechanism for tagging privacy sensitive data streams (e.g., video stream is sensitive, radar data is not).
	Status: to be implemented for Release 2
IDM-021 v1.1	Mechanism for encrypting sensitive data streams (encryption) and making it available only to authorized users.
	Status: partially implemented for swarm control messages such as join and leave messages.
IDM-022 v1.2	The swarm, and particularly the swarm coordinator(s), should constantly monitor the behaviour of swarm nodes to look for aberrant behaviour, either due to malfunction or malicious intent. For example, a smart-node (A) may incorrectly syndicate to other smart-nodes in the swarm that it has received heartbeat messages from another smart-node (B), when it has not. Where practical, the swarm should try to corroborate swarm information from other smart-nodes, and isolate smart-nodes that are shown to be untrustworthy.
	Deprecation note: Maintaining trust in nodes was assessed as out of scope.

2.4 Application Lifecycle

Requirements in the area of application lifecycle deal with processes for the execution of SmartEdge applications (recipes) as well as supporting tools necessary for SmartEdge.

Table 2-4 Application lifecycle list of requirements

ID / Ver.	**Description / Status**
AL-001 v1.1	A swarm should be able to run more than one application at a time, providing they do not directly compete.
	Status: implemented in SmartEdge Release 1 with reference to networking aspects. Multiple applications can be supported within the swarm subject that they are coordinated by a single Swarm Coordinator (A4.2). From the WP4 networking perspective, multiple applications are treated as data flows and its forwarded is supported by P4 switches (A4.4) embedded within each Swarm Node.
AL-002 v1.1	Running an application should be coordinated by a designated swarm smart-node, called the orchestrator, but it may enlist other swarm smart-nodes into performing the task described by the application. For example, an AMR can only hold one product at a time, it needs to enlist a mobile rack to hold multiple products of this type. As this is also the functional objective of the mobile rack, there is no conflict of interest.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)

AL-003 v1.1	Technically a swarm should have at least one running application to continue to exist. If the swarm has no running application, it has no purpose and so should be dissolved by agreement from its constituent swarm devices. In practice there may be a small-time lag between the swarm completing its last task and starting the next. If the swarm dissolved immediately it might have to be reformed a little time later, which could incur an operational penalty. Instead, the swarm should exhibit some level of "stickiness" for a time, i.e., the swarm may continue to exist without an immediate purpose.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
AL-004 v1.1	If an application orchestration smart-node fails or leaves the swarm in the middle of the execution of an application, if possible, another smart-node in the swarm should take over as orchestrator.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
AL-005 v1.1	A swarm device may refuse to cooperate with an application orchestration device if this conflicts with a higher objective. For example, an AMR may enlist a mobile rack to take another product, but the rack may refuse if the product is of the wrong type, or the rack is full. In this case the AMR should recruit a new device (new rack) into the swarm so the task can be fulfilled.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
AL-006 v1.1	It may be possible to have more than one orchestrator running a different application in the same swarm at the same time.
	Status: Not relevant for WP4. The Coordination among multiple Coordination should enable only one Adaptive Swarm Coordinator to interact with the Network Swarm Coordinator
AL-007 v1.1	An orchestrator may have more than one task assigned at a time, but only one application task should run at a time. Therefore, the orchestrator should implement a task queue. The orchestrator will run the application task with the highest priority. If this application can't run because the prerequisites have not been met, then it runs the next highest priority application task. For example, if the AMR can't perform the highest priority task, to move a product from the conveyer to a rack, because there are no more spaces in the rack; the AMR may take a lower priority task to move the rack to its next processing stage and enlist into the swarm a new empty rack.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
AL-008 v1.1	A swarm device may orchestrate its own tasks, whilst collaborating with another orchestrator on its tasks. A device should only participate in one task at a time, and so must put all tasks, its own and other orchestrators, into a single task queue. This is particularly important for battery powered AMRs, who's highest priority task is to ensure they have sufficient power to complete the current task and get to a recharging station. This is also a reason why swarm devices have the right to leave a swarm.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
AL-009 v1.1	Where possible if a task cannot be completed or the application fails part way through, e.g., the path of an AMR is blocked by an unexpected obstacle, the orchestrator should try to autonomously recover from the incident, e.g., replan an obstacle free path. If it can't recover, it should request manual intervention.

	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
AL-010 v1.1	A tool must be implemented to take the image feeds from multiple cameras and triangulate the position and pose of the object based on the image, e.g., three images could be obtained by three the overhead cameras showing the same object from three different views, by combining the three images, the tool should be able to orient the object in 3D space.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
AL-011 v1.1	A tool must be implemented to classify known objects in the specific use case domain, but should also support the “unknown” classification, i.e., an object has been detected but can’t be classified with a reasonable degree of accuracy.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)
AL-012 v2.1	Recipes may influence each other. For example, if something went wrong during execution of a recipe another recipe may need to wait until the blocking state was solved.
	Status: Not relevant for WP4 (i.e. addressed by WP3/WP5)

2.5 HARDWARE AND PROTOCOLS

Here we list the requirements that are only relevant to WP4 in Hardware and Protocols deal with protocols and devices that must be supported by SmartEdge at least to be able to provide support for the five SmartEdge use cases.

Table 5. Hardware and Protocols list of requirements

ID / Ver.	Description / Status
HP-002 v1.1	Vehicles communicating via C-ITS/G5 devices and without SmartEdge SW stack should be able to connect to the Swarm as "dummy" devices.
	Status: Implemented (see SW-001 and SW-027) the Access Point through the Access Point Manager (A4.1) provide connectivity to such devices. (A4.1)
HP-013 v1.1	The SmartEdge components system must be able to run on ARM-based hardware and Ubuntu/Linux OS.
	Status: Implemented in WP4. Currently all WP4 scripts are tested and run in Ubuntu 20.04 and 22.04, and Debian 12 on RPi5 devices (ARM Arch) (Some software components need to be compiled for the specified CPU architecture)
HP-015 v1.1	The SmartEdge components system must be able to run on x86-based hardware and Ubuntu/Linux OS.
	Status: Implemented. Currently all WP4 scripts are tested and run in Ubuntu 20.04 and 22.04, and Debian 12 on x86 Arch (Some software components need to be compiled for the specified CPU architecture)
HP-018 v1.2	The swarm edge components of the SmartEdge toolchain must be deployable onto an underlying software framework, such as Kubernetes or another service framework. SmartEdge is a series of compatible tools, forming a toolchain, as such it will likely be deployed on top of a suitable software framework.
	Status: to be implemented in Release 2.

2.6 OTHER REQUIREMENTS

1. Low Code Programming

Requirements in the area of low code programming deal with possibilities to implement SmartEdge applications as easily as possible.

ID / Ver.	Description / Status
LC-004 v1.2	SmartEdge runtime should provide discovery functionality to discover available devices and their capabilities.
	Status: implemented in SmartEdge Release 1 with reference to networking aspects. The Access Point Manager (A4.1) supports discovery functionality.
LC-018 v1.1	Capability for receiving control messages from nodes and relaying them to appropriate participants. For example, we might have a process outside the

	SmartEdge connected to a node and receiving data, analysing it and creating events.
	Status: implemented in SmartEdge Release 1 with reference to networking aspects. The Access Point Manager (A4.1) and the P4 switches (A4.4) enable the exchange of control messages.

All other requirements are not relevant for WP4.

2. Continuous Semantic Integration

Requirements in the context of Continuous Semantic Integration deal with the provision of environmental data by means of long-lasting data streams. Specific requirements for creating semantic integration pipelines for data streams suitable both in scalable cloud environments and in resource-constrained edge environments. For example, video streams are to be describedsemantically in such a way that an application can react to contents in the video itself. Not relevant for WP4.

2.7 SUMMARY OF MOST RELEVANT REQUIREMENTS TO BE ADDRESSED IN RELEASE 2

The following requirements will be addressed in Release 2 plus any other new/additional requirement that might emerge during the integration process:

- SW-005, 006: Support of three or more swarm coordinators
- SW-007: Implemented at the network level (Address Resolution Table - ART) in SmartEdge Release 1. Currently, the ART is implemented using Cassandra database as a temporary solution. A different technology may be utilized (e.g., Redis)
- SW-008: API between Swarm Coordinator and Adaptive Swarm Coordinator
- SW-016: requests outside the swarm
    - Non-swarm implementation (no SC)
- SW-022: integration with TDD
- SW-012, SC-001: static data sources communicating with the Swarm Nodes
    - Available at the Swarm Coordinator
    - Supported by P4 Runtime Plugin developed within WP5
- SC-025: hashed swarm state within the heartbeat
- SC-027: digitally sign events (e.g., IP assigned by the AP manager) to achieve non-repudiation
- IDM-20: tagging privacy sensitive data streams
- AAL-004: support for orchestration smart-node dynamic replacement

3. AUTOMATIC DISCOVERY AND DYNAMIC NETWORK SWARM FORMATION

This section reports on the design and implementation (Release 1.0) of the innovative *SmartEdge Network Layer* solution, providing automatic discovery and dynamic network swarm formation.

3.1 P4 DATA PLANE PROGRAMMABILITY FOR SWARM FORMATION

To illustrate the mechanisms used for discovery and dynamic swarm formation we use Figure 3.1 that depicts a reference scenario of the SmartEdge Smart Factory Use Case. The swarm formation utilizes P4 language for the high-level behavioral description of the forwarding logic of the programmable network and data-plane layers. The data-plane programming capability using P4 Language is provided to the components in the smart factory, including mobile robots and racks that assume the role of Swarm Smart Nodes, and IoT/Wi-Fi edge gateways that assume the roles of Access Points and Swarm Coordinators. This allows for efficient and flexible processing of data streams generated by the devices that are participating in the swarm. Each group of components forms a swarm network characterized by its own subnet to ensure isolation and security, and that data streams generated from the many sources within the swarm are processed efficiently and within the time requirements. A swarm network in this scenario has the following major components:

1- Swarm smart nodes (SN), which are a group of devices that can be running different applications and collaborate to achieve the purpose of the swarm. These nodes implement the SmartEdge Network layer, and they run the software module called Smart Node Manager (Artifact A4.3).
2- Access Points, which are the swarm nodes that provide connectivity in the swarm area. The Access Points runs the AP Manager software module (Artifact A4.1), which handles the process of monitoring the connected smart nodes and assigns them IP addresses.
3- Swarm Coordinator (Artifact A4.2), which is a software module responsible for assigning the subnet to the swarm and handles the onboarding and offboarding of swarm smart nodes, it also keeps track of the state of the swarm by recording smart node data in the swarm database (A4.5).

Although the above description uses the smart factory use case to explain the role of data plane programmability in the swarm formation and communication, this description of the network layer applies the same to the other use cases suggested within SmartEdge, as the smart nodes can be either robots in a factory, vehicles on the road, or health monitors.

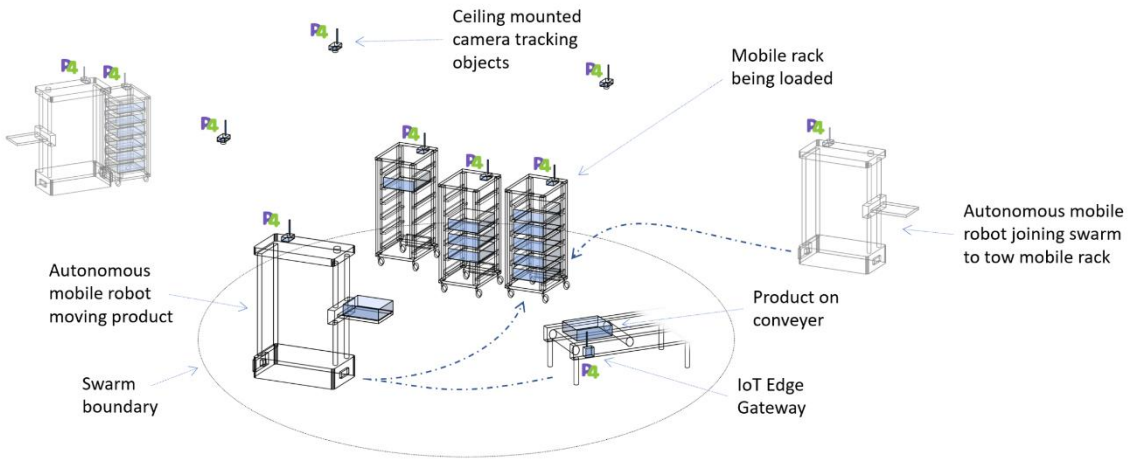


Figure 3.1 Smart factory utilizing P4 network layer swarm formation and management.

3.2 SWARM DISCOVERY AND FORMATION

Following the requirements of Swarm management and Swarm Communication to discover swarm nodes, including their capabilities listed in Knowledge Graph Repository including Thing Description Directory (TDD) (Artifact A3.3), and enlist them into the swarm intelligence system, the P4-based system is to be programmed as follows:

i. As new smart nodes join the swarm area, they connect to the wireless network established by the Access Points. In the case of Wi-Fi, this phase relies on traditional authentication procedures of Wi-Fi technology.
ii. The Access Point Manager then assigns an IP address to the connected smart node and establishes a VXLAN tunnel to enable the exchange of swarm communication messages (e.g., Join message towards the Swarm Coordinator).
iii. The Access Point Manager inserts the Swarm Node data into the ART, to make it discoverable by swarm coordinators. At this stage the P4 switch of the AP allows only communication to swarm coordinators, no other communication is possible with any other smart nodes.
iv. The Access Point Manager then establishes a VXLAN tunnel to the swarm smart node and informs the Smart Node Manager of the available swarms and coordinator IPs. This information can be used by the swarm smart node in case it needs to request to join a specific swarm.
v. The smart node can then proceed to request to join a particular swarm or can be requested to join by one of the swarm coordinators.
vi. After a node has joined a swarm, the swarm coordinator updates the swarm ART to indicate the presence of the new node. Then, it updates the P4 rules in the relevant access points to allow communication with other nodes in the swarm.

The automated swarm formation is based on a recipe, and is clarified in Figure 3.2. It is performed in the following way:

1- A request for recipe (defined in D3.2) execution is submitted to the Task Orchestrator (A5.3.2), the recipe defines the requirements of the application this includes the capabilities that the swarm nodes need to have to execute the application.
2- The orchestrator asks the coordinator to provide the necessary swarm nodes to execute the application listing the necessary capabilities and number of nodes.
3- The coordinator then checks the TDD to find the suitable nodes that satisfy the application requirements and that are available.
4- The coordinator then starts the onboarding process of the discovered node, and asks them to join the swarm.
5- A list of the assigned nodes is returned to the Orchestrator by the Swarm Coordinator.
6- The Orchestrator then verifies the received nodes consulting the Industrial Knowledge Graphs.
7- After the successful onboarding and verification of the nodes, the Orchestrator assigns the tasks to the swarm nodes.

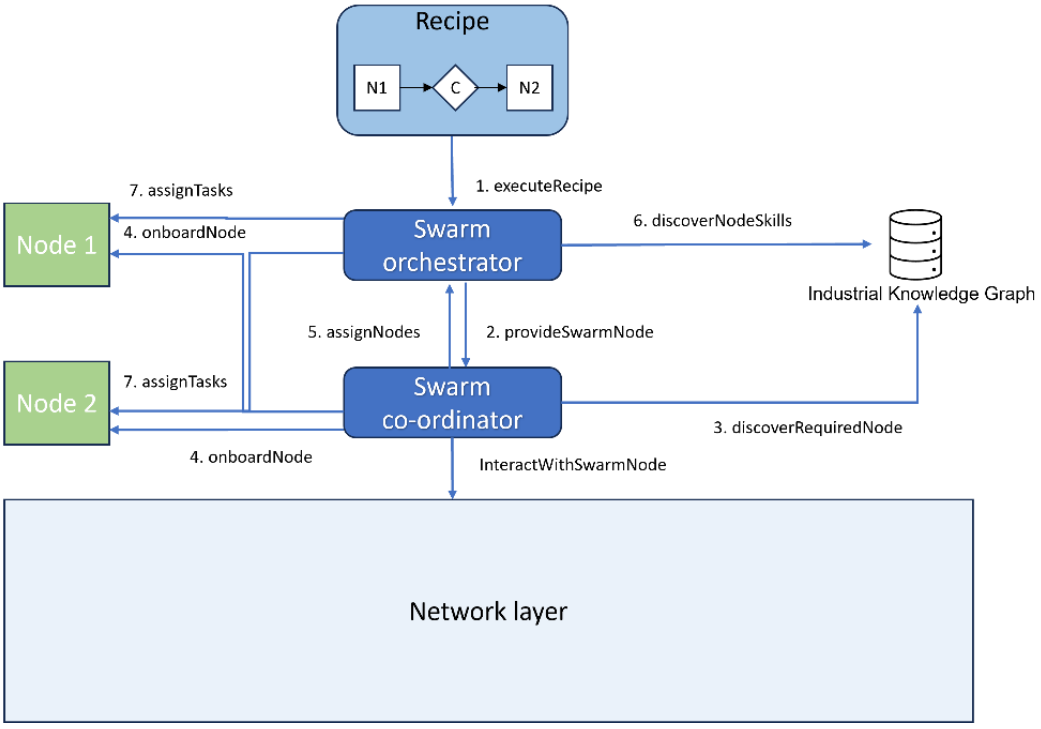


Figure 3.2 High-level Interaction between swarm orchestrator and swarm coordinator for the formation of the swarm [D3.1]

3.3 CONTROLLED SWARM EXIT

Nodes can leave the swarm in a controlled manner in one of the following ways:

1. A node sends a leave request to the swarm coordinator, this request is then processed by the swarm coordinator and is either accepted or rejected. In case the request is accepted, the node is removed from the swarm database, and the coordinator then removes the P4 entries in the APs to prevent routing traffic from or to the node that has just left.
2. The swarm coordinator sends a leave request to the smart nodes, when it is no longer needed by the application, then the node is removed from the database, and P4 routing entries related to the node are also removed from the APs.

3.4 UNEXPECTED SWARM EXIT AND RECOVERY

To enable the swarm intelligence system to automatically recover if a swarm device is unexpectedly lost, the following steps are taken:

1. The Access Point actively monitors the connected nodes, and when it detects a disconnection of one of the nodes from its wireless network, it updates the swarm database to indicate the absence of the disconnected node. In the case of Wi-Fi Access Points, disconnection is detected:
    a. as a Wi-Fi event in the underlying Linux Operating System (through linux daemon wpa_supplicant) when the device goes out of range or in case of an intended disconnection initiated by the node itself (the node sends a disconnection frame to the Access Point)
    b. by the absence of a configured number of heartbeats. In this case the node might remain connected to the Wi-Fi network. The heartbeats start after the swarm node has successfully joined the swarm. The absence of heartbeats might indicate a software crash within the SmartEdge stack.
2. The AP waits for a configurable amount of time, if the node has not reconnected, the AP deletes the routing entries from itself and from other APs and updates the swarm database to indicate that the node is no longer part of the swarm.

By defining and implementing suitable mechanisms to detect lost devices and automated error remediation routines using the P4 language and programmable network architecture, devices within the network can continue to work even in the presence of device loss. These implementations enable network-wide fault tolerance and resiliency, leading to a more reliable and efficient swarm intelligence system.

3.5 CATERING TO DIFFERENT TYPES OF NODES AND DATA FLOWS

In addition to smart nodes that adhere to the SmartEdge network layer specifications, there could be other non-smart devices that are incorporated into the swarm environment and that can provide important data vital to the proper functioning of the swarm. These kinds of non-smart devices might include cameras, sensors, radars, LiDARs, or any other type of device that can take a role in the swarm operation. Due to the dynamic nature of swarm networks, nodes can join and leave the swarm dynamically, so it is important to keep and updated state of the swarm and inform the nodes in the network of all the available data sources. The P4 programmable network layer manages the discovery operation and ensures that the up-to-date state of the swarm is kept, as any delays in the update of the swarm state can result in accidents or lead to idle time. As an example, and in reference to the scenario shown in Figure 3.1, the P4 implementation recognizes traffic generated from these data sources and assigns respective priorities. For example, location-tracking traffic generated from cameras monitoring the factory floor should be given higher priority than a humidity sensor, as the cameras are crucial to provide location information and to detect obstacles on the path of the moving robots, to prevent collisions and to reduce the possibility of human injuries.

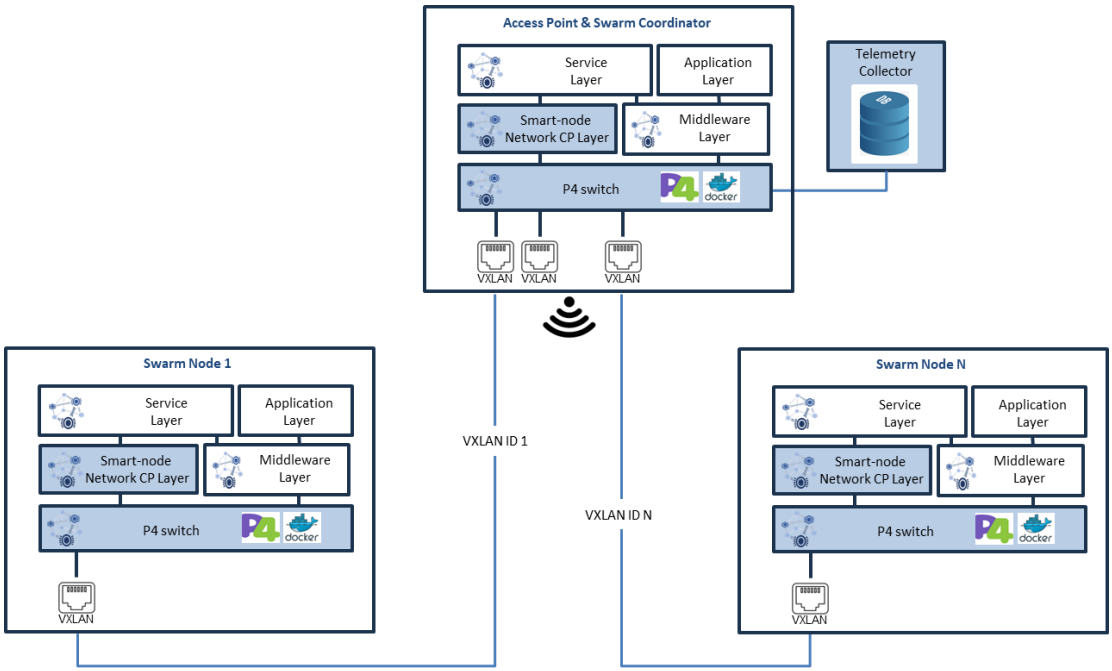


Figure 3.3 SmartEdge Node Architecture

3.6 ARCHITECTURE OF SMARTEDGE NETWORK LAYER

Figure 3.3 depicts the key functional components of each swarm node related to the innovative SmartEdge swarm networking technology: the P4 switch and the Smart-node Network Control Plane (CP) Layer.

The P4 switch forwards all communication messages. It leverages tunneling technology, such as VXLAN, to guarantee isolation. In addition, it inserts/extracts in-band telemetry to provide/retrieve detailed and timely information about the swarm node state. Furthermore, it implements optional reliable communication functionalities, such as 1+1 protection.

The Smart-node Network CP Layer manages the swarm formation from the networking perspective. It then adds, removes and updates forwarding rules on the P4 switch.

3.7 SWARM NETWORKING ARTIFACTS

In this section, we report on the key artifacts involved in swarm formation, outlining their roles and functions within the swarm network.

The SmartEdge innovative artifacts for swarm networking are:

A4.1	Access Point Manager
A4.2	Swarm Coordinator
A4.3	Swarm-node Manager
A4.4	P4 Switch
A4.5	Distributed Database for Network Information

Next figure illustrates the various components and their connections within the swarm network, highlighting the interactions between the access point, smart nodes, and the coordinator.

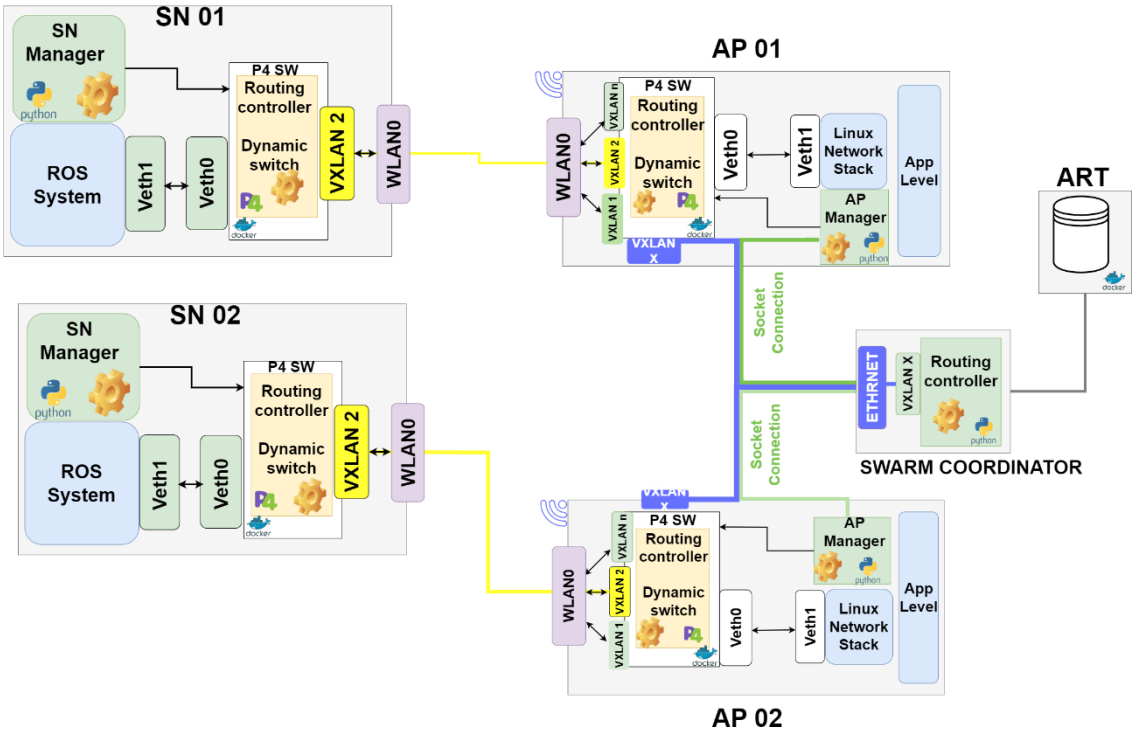


Figure 3.4 Swarm Network Components

A4.1 Access Point Manager

The Access Point Manager (APM) is implemented as a background program (Python script) running on the Access Point (AP). It continuously monitors the connection and disconnection of smart nodes, retrieving key information such as IP address, MAC address, and Swarm Node (SN) name. The APM creates a VXLAN interface for each newly connected smart node, attaches this interface to the P4 switch, and communicates the relevant VXLAN information to the connected node. The APM is responsible for managing the creation and deletion of these VXLAN interfaces in sync with the P4 switch updates.

Upon detecting a connection, the APM identifies when a smart node sends a join request to the swarm. This request, along with the node's details, is forwarded to the Swarm Coordinator, which then decides whether to accept or reject the node. Based on the Swarm Coordinator's decision, the APM receives the new P4 flow rules and updates the access point's switch accordingly.

Sub-component: Access Point

The Access Point serves as the primary interface for smart nodes to connect to the swarm. It hosts P4 switches, which are managed by the network coordinator to apply flow rules that ensure secure communication between swarm members. In addition, the access point provides the DHCP server and authentication server, facilitating the assignment of IP addresses and validating smart nodes before allowing them to join the network. It supports VXLAN interfaces for tunneling data between the coordinator and other network components. Under the coordinator's guidance, the access point enforces the P4 traffic flow rules, ensuring that communication is restricted to smart nodes within the same swarm, thereby maintaining secure and authorized interactions between them.

The current access point is built using a Raspberry Pi 5 running Ubuntu Linux. The Raspberry Pi operates as a Wi-Fi hotspot, managed through the Network Manager in Ubuntu. In addition to providing a wireless network, it runs a DHCP server for IP address assignment and uses WPA2-PSK password authentication to validate smart nodes attempting to join the wireless network.

A swarm can consist of multiple access points, each responsible for managing the connection and communication of smart nodes within its range.

A4.2 Swarm Coordinator

The Swarm Coordinator manages the P4 switches inside the access point(s), ensuring that flow rules are updated to authorize communication strictly between swarm node members only.

The SC also has read/write rights to a swarm ART that stores vital information and the status of the swarm's nodes.

The SC uses control messages to manage the behavior and communication of connected nodes through the network control plane, enforces access control, and communicates with the Adaptive Swarm Coordinator responsible for determining whether smart nodes can join the swarm.

A4.3 Swarm Node Manager

The Swarm Node Manager is a program (written in Python) running on the smart node between the application and network layer. It continuously monitors the network interfaces status of the smart nodes, retrieving key information such as assigned IP addresses and access point information. The Swarm Node Manager opens a listening socket after successfully connecting to the access point to receive the VXLAN tunnel and the coordinator network information. The Swarm Node Manager program creates and attaches the VXLAN interface to the P4 Switch, and sends a join request to the coordinator through the VXLAN tunnel.

The Swarm Node Manager is also responsible for monitoring the connection state of the node. It detects soft and hard disconnections and triggers the required functions to reset the node’s network configurations and initialize it to continue the functioning in a proper way.

A4.4 P4 Switch

The P4 switch is an artifact involved in the swarm formation. It plays the role of enforcing the inter swarm communication flow rules and controlling communication between smart nodes in the access point. Each access point is equipped with a P4 switch, managed by the Access Point Manager (A4.3) and also by the Swarm Coordinator (A4.2). When a smart node connects to the access point, the P4 switch is updated with specific flow rules that either allow or deny traffic based on the decision made by the Swarm Coordinator. These flow rules ensure that communication is restricted to authorized swarm members only, preventing unauthorized access or malicious activity within the swarm. Additionally, the P4 switch interacts with the VXLAN interfaces created for each connected smart node, allowing for the encapsulation of traffic between the access point and other swarm nodes. This enables secure and isolated communication between swarm nodes while maintaining flexibility in the network's traffic management. On the other hand, the Swarm Smart Node runs a P4 switch that can be configured by the Smart Node Manager (A4.3) with flow rules to enable communication to other nodes,

and to insert telemetry data within the packets in the form of In-band Network Telemetry (INT) for monitoring latencies and the state of the node.

A4.5 Distributed Database for Network Information – ART

The **ART** in the swarm serves as a component for storing and managing essential networking information about the swarm nodes. The ART stores vital information about each smart node, such as status and IP address associated with its UUID. This information is used by the swarm coordinator to make networking configurations regarding node admission and to enforce access control policies. The ART database helps in maintaining an up-to-date record of the swarm's membership and activity at the network level. The ART database is continuously updated as nodes connect or disconnect from the swarm.

The current ART database structure contains key values for each swarm node to track its status and connection details. These values include the **node_swarm_id**, a unique identifier for the node within the swarm; **last_update_timestamp**, which records the last time the node's information was updated; **node_current_ip**, the current IP address assigned to the node; and **node_swarm_status**, which is set to 0 during the joining process and updated to 1 once the node is officially accepted into the swarm. Additional fields include the **node_uuid**, the universally unique identifier of the node, along with the node's **swarm_ip** and **swarm_mac**, which are used to manage and authenticate communication within the swarm network.

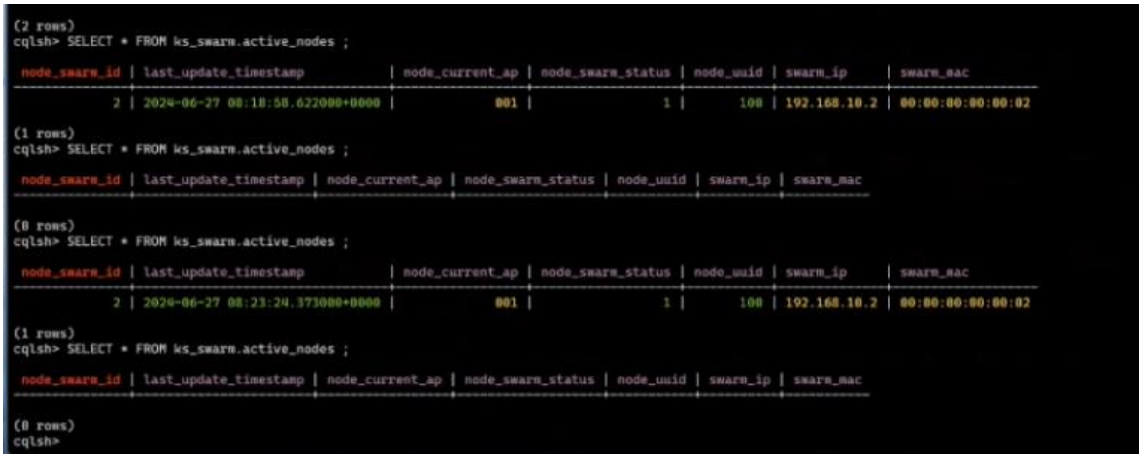


Figure 3.5 screenshot of the swarm ART

Table 3-1 ART fields

Swarm ID (32Bits)	SN UUID (64Bits)	SN ID in Swarm (32Bits)	SN IP (32Bits)	SN MAC (48Bits)	Node Swarm Status (8Bits)	Node Current AP (UUID) (64Bits)	Timestamp (64Bits)	RSA Signature (256 bytes)

3.8 PRELIMINARY RESULTS

To evaluate the system's performance, a series of measurements were conducted across the Access Point (AP), Swarm Coordinator, and Swarm Node (SN). These measurements were performed over 15 test runs, allowing for comprehensive analysis. The primary objectives of

these tests were to assess the system's efficiency, identify any potential bottlenecks, and provide insights into its operational performance under varying conditions.

Illustrative Sequence Diagram with Detailed Time Measurements

Figure 3.6 provides a comprehensive breakdown of the swarm management program, detailing how the system components including the SN, AP, and Coordinator communicate and work together to manage network operations.

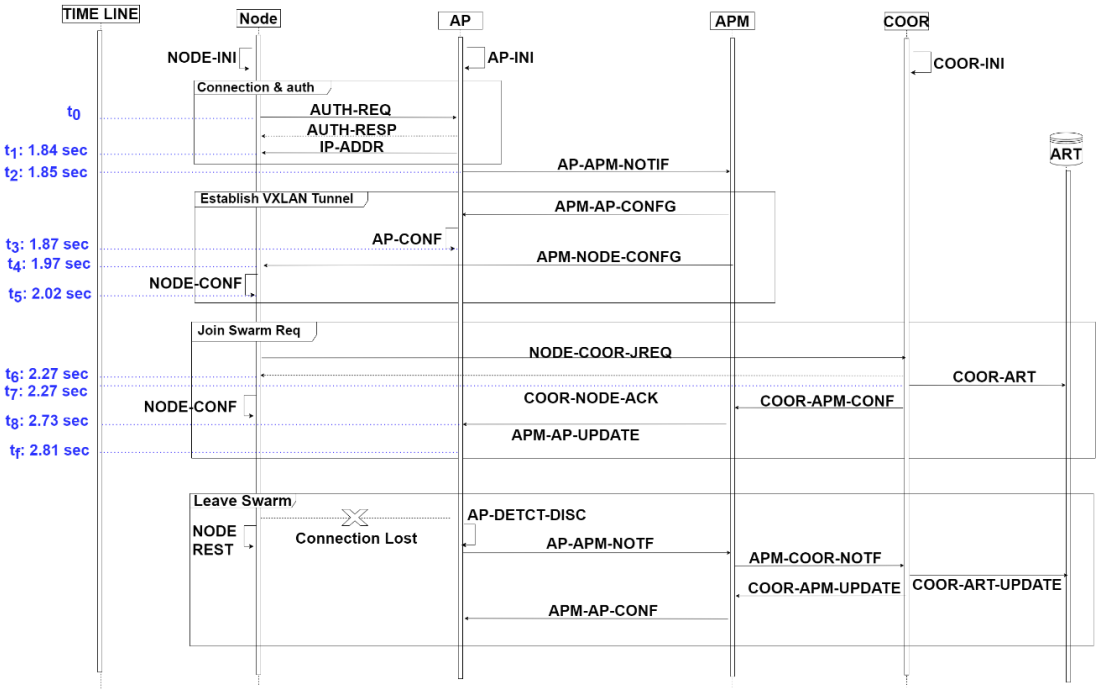


Figure 3.6 Sequence Diagram with Time Measurements

Table 3-2 Components Communications description

Name	Description
NODE-INI	The node powers on and enters a loop to scan for available wireless networks.
AP-INI	The Access Point initializes by activating the wireless hotspot, the P4 switch for network traffic management, and launching the AP Manager program to handle node connections, virtual network configurations, and communication with the swarm coordinator.

COOR-INI	Upon launching, the coordinator verifies the connection with the Access Point to ensure network communication is established, while simultaneously linking with the Cassandra database and P4 switch.
AUTH-REQ	Upon detecting an SSID that matches a pre-defined SSID, the node sends a connection request to the Access Point, providing a password for authentication.
AUTH-RESP	The Access Point verifies the password (Wi-Fi Protected Access 2 - Pre-Shared Key), and based on its accuracy, either accepts the request and grants network access or rejects it, denying the connection.
IP-ADDR	After successful authentication, the AP assigns an IP address to the node, enabling it to communicate within the network.
AP-APM-NOTIF	the AP uses a Wi-Fi monitoring tool that automatically detects new connections. As a result, when a new node successfully authenticates, the AP Manager is automatically notified without requiring an explicit message from the AP.
APM-AP-CONFG	The AP Manager creates a VXLAN (Virtual eXtensible LAN) tunnel specifically for the Node. Brings up the VXLAN interface, configures it and attaches it to the P4 switch to handle traffic for the Node.
AP-CONF	The AP configures its P4 switch (via the Thrift API) to forward packets destined for the Node's virtual IP/MAC addresses over the correct VXLAN interface.
APM-NODE-CONFG	The AP Manager sends the necessary network configuration to the newly connected node, including details such as the VXLAN ID, virtual IP and MAC addresses, and other parameters required for the node to integrate into the network.
NODE-CONFG	The node creates the VXLAN and virtual interfaces, configuring its network settings to establish communication within the virtual network managed by the AP Manager.
NODE-COOR-JREQ	After configuring the network, the Node sends a "Join Request" to the Coordinator containing its UUID, VXLAN ID, VIP, VMAC, and AP information.
COOR-ART	The Coordinator updates the database with the new node's information.
COOR-NODE-ACK	The Coordinator responds, allowing the Node to officially join the swarm network.
NODE-CONF	Upon receiving the acknowledgment from the **Coordinator** that the join request has been accepted, the **Node** pushes the newly assigned flow rules into the local **NIKSS switch**, enabling the routing and forwarding of traffic

	within the swarm network based on the configured virtual network parameters.
COOR-APM-CONF	The Coordinator notifies the AP Manager to update the forwarding rules to include this new node in the P4 switch.
APM-AP-UPDATE-P4	The AP Manager updates the relevant AP's broadcast ports and routing rules by sending commands to the P4 switch.
NODE-REST	When the Node physically disconnects from the AP (e.g., the Node powers off or moves out of range), it detects the disconnection and executes cleanup commands to remove the VXLAN interface, close TCP sockets, and reset network configurations.
AP-DETCT-DISC	The AP is continuously monitoring the wireless network for changes. When the Node physically disconnects from the AP the AP receives a "disconnected" event.
AP-APM-NOTF	Upon detecting the disconnection of the node, the AP Manager is automatically notified through the Wi-Fi monitoring tools, enabling it to begin the necessary cleanup process and remove the node's associated configurations from the network.
APM-COOR-NOTF	The AP Manager is automatically notified through the Wi-Fi monitoring tools and subsequently notifies the Coordinator.
COOR-ART-UPDATE	The Coordinator updates the database by deleting the disconnected node's information, ensuring that the swarm network's records remain accurate, and the node is fully removed from the system.
COOR-APM-UPDATE	The Coordinator sends commands to the AP Manager to remove any forwarding rules related to the Node. This prevents any other AP from trying to route traffic to a node that is no longer active.
APM-AP-CONF	The AP Manager removes the forwarding entries for the Node from the P4 switch, ensuring that the switch no longer routes packets to the Node. The AP Manager sends delete commands to the P4 switch via the Thrift API to remove any rules related to the Node's virtual IP (vIP) and virtual MAC (vMAC).

3. Measurements and Performance Analysis at the Access Point (AP)

To evaluate the system's performance, identify potential bottlenecks, and understand its operational efficiency, we conducted measurements across the Access Point (AP), Coordinator, and Swarm Node (SN) over 15 test runs. The tests involved all key components of the system: the Access Point Manager, Swarm Coordinator, Swarm Node Manager, P4 Switch, and Distributed Database. By utilizing Raspberry Pi 5 devices as test beds for both the AP and SN, we

crafted and implemented measurement scripts that calculate the exact timing of all processes between these components. These scripts output logs with detailed time measurements of the operations occurring between them. The preliminary measurements from these tests are presented in the table below.

Table 3-3 Measurement and Performance at the Access Point

Key Metrics	**Description**	**Average Time Duration**
Time taken to log client	The time it takes for the AP to log the connection event of a new node after successful authentication.	~0.01 sec
Create VXLAN and port	The time required for the AP to create a VXLAN and VETH interfaces and assign a port for the newly connected node.	~0.01 sec
Send VXLAN information	The APM sending the necessary VXLAN configuration information (virtual IP, MAC, and tunnel information) to the newly connected node.	~0.10 sec
Insert flow rules	The APM adding the necessary flow rules into the P4 switch, enabling the correct routing of traffic to and from the node within the swarm network.	~0.7 sec

The total average time for the AP to handle a new node is around 0.8 seconds, with few tests taking slightly more than 1 second. This delay is mainly introduced by APM-NODE-CONFG actions, which include establishing a TCP session from the Access Point Manager to the Swarm Node Manager, sending the VXLAN configuration and installing it in the Swarm Node). In Release 2 the implementation of APM-NODE-CONFG will be revised to reduce this time. Most variability comes from the send_vxlan_info step.

4. Measurements and Performance Analysis at the Coordinator

Table 3-4 Measurement and Performance at the coordinator

Key Metrics	Description	Average Time Duration
Update Database	updating the ART database with the new node's details (such as its virtual IP, MAC, and VXLAN ID) upon joining the swarm.	~0.002 sec
Update P4 switch	updating the P4 switch with new flow rules to ensure proper traffic forwarding for the newly added node.	~0.45 sec

The observations indicate that the performance of both database updates and P4 switch updates remained relatively stable throughout the tests. Notably, P4 switch updates account for the majority of the processing time in the Coordinator, consuming approximately 99% of the total execution time. Given that the time required for switch updates is expected to scale with

the size of the network, there is potential for further optimization, particularly as the number of nodes increases.

The Coordinator's role in updating the database and configuring the P4 switch is stable, with no major bottlenecks identified. P4 switch updates are the most time-consuming part, and further optimization may be necessary as the system scales with more nodes

5. Measurements and Performance Analysis at the SN

Table 3-5 Measurements at the Smart Node

Key Metrics	**Description**	**Average Time Duration**
Connection and receiving an IP	Time taken for the Node to connect to the network and receive an IP address from the AP (via DHCP).	~1.84 sec
Create VXLAN interface	Setting up a VXLAN interface after receiving the necessary configuration from the AP.	~0.014 sec
Insert flow rules	Inserting flow rules into its local switch (NIKSS switch) to ensure proper traffic handling.	~0.027 sec
Send Request and Receive Response	Sending a Join request to the Coordinator and receiving the acknowledgment.	~0.25 sec

The observations highlight significant variability in the time taken for the connection and IP assignment process, suggesting that DHCP-related factors may be influencing performance. For instance, one test run took significantly longer (~3.15 seconds) compared to other tests. Additionally, the "Send Request and Receive Response" step was nearly instantaneous. The overall average for this step is approximately 0.25 seconds, though it is impacted by these longer response times. In terms of total duration, the Node's overall process varied between ~1.58 to ~3.15 seconds, with an average duration of 2.14 seconds.

The Node's connection and IP retrieval process is the slowest and most variable, which could be affected by network conditions or DHCP performance.

4. EMBEDDED NETWORK SECURITY AND ISOLATION

Network security and traffic isolation are provided to SmartEdge nodes as part of Task T4.2. The task interacts both with Task T4.1, which creates and manages the swarm, and Task T4.3, which provides hardware acceleration. For example, network security solutions developed under Task T4.2 are accelerated under Task T4.3.

There are two parts to the embedded network security in SmartEdge:

- A novel P4 based layer of security that focuses on swarm requires and attends to threat vectors relevant to SmartEdge use cases.
- An innovative intelligent threat detection and mitigation solution that attends to emerging threats and anomalies.

The innovative solutions offered by SmartEdge are complemented by using industry-standard security mechanisms, such as the use of traffic encryption and authentication protocols. These are not developed by T4.2, and are integrated as part of WP6.

4.1 PROGRAMMABLE ACCESS CONTROL FOR SWARM INTELLIGENCE

SmartEdge leverages the P4 language to design a flexible and adjustable Access Control List (ACL) within its P4 switch, enhancing swarm intelligence capabilities. The method of operation leverages the fact that each node is assigned a distinct identifier, such as a UUID. This identifier is embedded in the packet header to enable exclusive communication among nodes within the same swarm. Any message originating from low-reputation nodes or utilizing unauthorized protocols are automatically discarded. Only the designated coordinator is permitted to interact with external entities, including nodes seeking to join or coordinators from other swarms. This structure effectively isolates network traffic streams for different swarms, ensuring secure and controlled communication. To elaborate, the functionality supported for programmable network control will include the following:

- Incoming traffic from malicious or low-reputation nodes is dropped. Such nodes are identified in a list of blocked nodes (blocklist), maintained by the swarm coordinator. The blocklist is advertised to all P4 switches in the coordinator's swarm. The malicious or low-reputation nodes are identified by UUID or other relevant identifiers, such as their IP address.
- Traffic using unapproved protocols is dropped, and only approved communications protocols are allowed within a swarm.
- To ensure traffic isolation between swarms, VXLAN is used with a unique VNID assigned to each swarm.
- For each incoming packet, the Swarm ID of both source and destination nodes are used, in addition to their roles. A message is forwarded only if the source and destination belong to the same swarm. An exception is the coordinator, which can communicate also with nodes outside the swarm (e.g., to admit new swarm nodes). This is in addition to the use of VXLAN.

- Only the coordinator can communicate with nodes requesting to join the swarm, until their onboarding is completed. The new nodes have restricted communications abilities, such as rate limiting (to prevent denial of service attacks on the coordinator).
- The access control functionality is programmable, enabling updates and extensions based on use case and deployment needs. New rules can be installed during runtime, enabling uninterrupted operation of the swarm.

The code for the programmable access control was included in the first Release and integrated with the swarm management components of T4.1.

Some test cases were designed and executed during the second year, such as:

- If a specified node is blocklisted, all packets originating from it are dropped by the node configured with the corresponding P4 code.
- If a specified node is blocklisted, all packets destined for it are dropped by the node configured with the corresponding P4 code.
- If a node was configured with a swarm ID that did not match the current Swarm ID, all packets it sent to nodes with the designated Swarm ID were dropped.
- If a node was configured with a swarm ID that did not match the designated Swarm, all packets from this node to the nodes with the designated Swarm ID were dropped.

All these test results were correct, and reproducible.

4.2 ML-DRIVEN IN-NETWORK DEFENSE

The high speed of the mobile and wireless networks used in SmartEdge pose challenges in unexpected emerging threats. In SmartEdge, gateways and P4-based switches can act as the first line of defense to prevent malicious traffic from propagating to other parts of the swarm.

In SmartEdge, in-network computing solutions are used for fast ML-based attack detection and mitigation. This includes both the development of a framework for rapid prototyping of in-network ML solutions and a solution for adding support and updating models for new attack vectors. Our novel in-network traffic analysis framework incorporates programmable data plane into IoT gateways, pioneering the utilization of in-network machine learning (ML) inference for fast mitigation. It facilitates continuous and seamless updates of in-network inference models within gateways. The solution was prototyped in P4 language on Raspberry Pi (representative of Use-Case 2 RSU) and Dell Edge Gateway (for Use-Case 3). With ML inference offloaded to gateway's data plane, our in-network approach achieves swift attack mitigation and lightweight deployment compared to prior ML-based solutions. Our evaluation results using three public datasets showed that we accurately detect and mitigate emerging attacks (>30% accuracy improvement and sub-millisecond mitigation time), without disrupting network traffic.

The support of ML driven in-network defense is built from multiple building blocks:

- A framework for rapid prototyping on in-network ML across targets. Reported in section 4.2.1
- A solution for runtime model updates without interrupting traffic. Reported in D4.1 and completed.

- A solution for distributing a ML model across multiple network devices, guaranteeing a model is correctly executed regardless of the path taken through the network. Reported in D4.1 and completed.
- A solution for a hybrid deployment of ML models to improve ML performance without compromising on system performance. Reported in section 4.2.2
- A solution for a federated learning deployment of ML models within the network. Initially reported in D4.1 and extended in in section 4.2.3

4.2.1 Rapid Prototyping of In-Network ML Models

To support ML-driven network defense, there is a need to easily develop ML models that are trained on recent data, use up to date ML models, and leverage programmable network devices available in the market.

As the following Figure 4.1 shows, even a single configuration change can require significant changes to a design, with hundreds of lines of code modified or added.

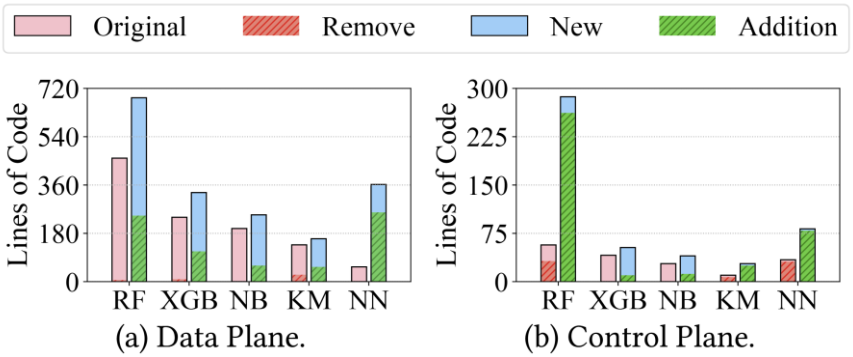


Figure 4.1 Lines of code changes because of a single setup change: RF - depth of 2 to 5, XGB - 2 to 6 trees, NB - 2 to 5 features, KM – v1model to TNA, NN - 2 to 5 layers. Source: [Zhe24-2]

This amount of manual work is unreasonable to support SmartEdge use cases. Therefore, there is a need for a framework enabling rapid prototyping of in-network ML inference solutions.

The goals of the framework would be as follows:

- Provide a general solution for mapping ML models to programmable network devices.
- Enable portability across network platforms.
- Support multiple use cases.
- Be resource efficient and co-exist with normal network functionality (P4 switch) without exhausting resources.
- Ease of use.

The developed framework, Planter, provides an end-to-end solution, taking a dataset and turning it into a tested deployment on a target device, as illustrated in the next Figure 4.2.

The operation of the workflow is as follows:

A dataset is provided by the use case owner for training purposes. Using a standard training framework, such as Pytorch or Scikit-learn, Planter trains the model, including parameter selection and hyper-parameter tuning. The trained model is then automatically mapped to a programmable network device, generating P4 code for the data plane and python code for the control plane. Together with this code, Planter generates code to test the generated solution

and bash scripts. The generated solution is first tested in a software environment to guarantee correct operation, before deploying on the target device. It can then be automatically tested again on the target.

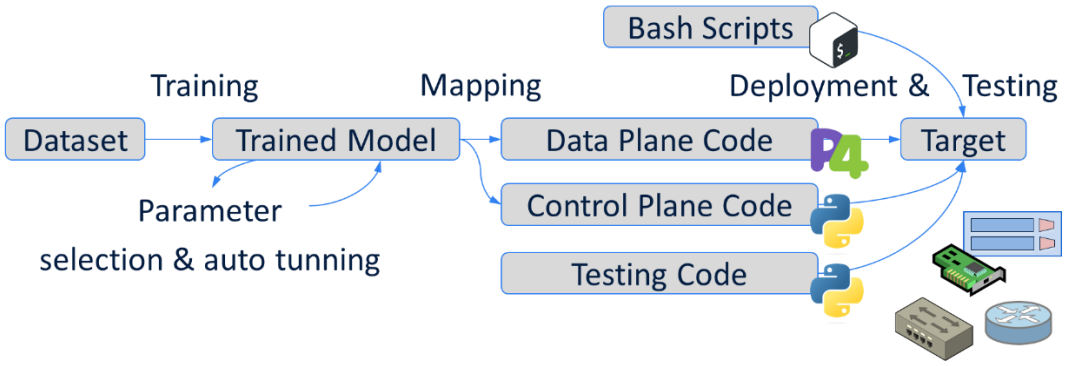


Figure 4.2 Planter Framework Operation Workflow.
Planter takes a dataset, trains a selected model on it, including parameter selection and tuning, and generates code for the data plane (in P4), the control plane (in python) and for testing (in python). It also generates scripts to deploy and test the trained model.

The Planter framework was implemented in 57k LOC in Python and is available at Planter's GitHub repository [Zhe24-3]. It supported targets including Intel Tofino and Tofino2 (switch-ASIC), AMD Alveo U280 (FPGA) over Open-NIC [Xil24], P4Pi-BMv2 and P4Pi-T4P4S on Raspberry Pi [Lak21], BMv2 on NVIDIA BlueField2 [Ida21] and BMv2. Pipeline architectures include v1model [Lar21], Intel TNA [Tna21], PSA [Psa17], AMD XSA [Xsa23] and NVIDIA spectrum. This enables easy portability of solutions across targets and architectures.

Planter is a modular framework, and in addition to supporting many targets and architectures, it also supports a wide range of ML models (e.g., ensemble models, NN, probabilistic models, dimension reduction and clustering) and different use cases.

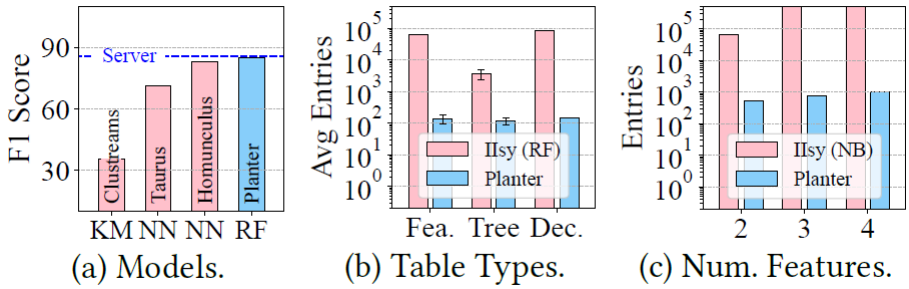


Figure 4.3 Comparison of Planter accuracy and table entries with State-of-the-Art. Source: [Zhe24-2]

Figure 4.3 highlights Planter's performance for in-network defense using the UNSW anomaly detection dataset [Nuo15]. In terms of accuracy, Planter achieves 84.88%, outperforming Clustreams [Roy21] (35.4%), Taurus [Tus22] (71.1%), and Homunculus [Tus23] (83.1%). Additionally, Planter models demonstrate reduced memory consumption and table entry requirements, efficiently minimizing table entry growth by converting multiplications to additions.

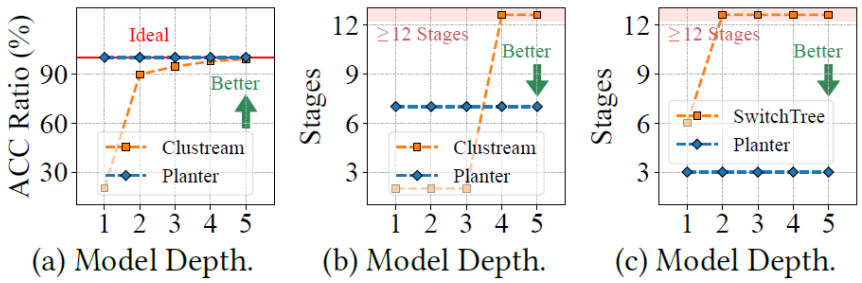


Figure 4.4 Comparison of Planter accuracy and stages used with State-of-the-Art. Source: [Zhe24-2]

In Figure 4.4, Planter demonstrates improved scalability over Clustreams [Roy21] and SwitchTree [Jon20], maintaining a constant number of stages for tree models and using parallel processing to avoid hardware limitations as model sizes grow. Tests on CICIDS dataset [Ima18] confirm Planter's high accuracy and efficiency across various model depth for in-network defense.

Table 4-1 Accuracy (ACC), resources and latency relative to switch.p4 for in-network ML defense using CICIDS and UNSW datasets. Using (S)mall, (M)edium, (L)arge and (H)uge models. Some models are not feasible (NF) on Tofino but are feasible (†) on Tofino2. Source: [Zhe24-2].

		Accuracy										Resource Performance										
		CICIDS				UNSW						UNSW										
		Switch (M)		Sklearn (M)		Switch (M)		Sklearn (M)		Server (H)		ACC (Switch)		Memory (%)			Latency (Relative)			Stages (Tofino)		
Work	Model	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	S	L	S	M	L	S	M	L	S	M	L
SwitchTree [50]	DT_{DM}	99.92	99.92	99.92	99.92	99.40	**94.53**	99.40	**94.53**	99.40	94.31	99.34	99.41	1.46	1.72	1.98	81.16	88.36	88.36	11	13†	15†
pForest [7]	RF_{DM}	99.80	99.79	99.80	99.79	99.38	94.44	99.38	94.44	99.42	94.51	99.25	99.39	7.71	14.01	NF	88.36	89.04	NF	11†	14†	NF
IIsy [91]	SVM	59.24	37.20	95.04	94.94	97.31	49.32	99.23	93.51	99.23	93.51	97.31	99.23	2.81	3.23	4.12	26.37	35.27	35.30	9	9	9
N3IC [76]	NN‡	92.09	92.00	99.96	99.96	98.33	85.68	99.25	93.67	99.25	93.68	98.33	97.50	NF	NF	NF	NF	NF	NF	NF	NF	NF
IIsy [91]	KM_{LB}	58.40	56.80	58.40	56.80	71.28	41.88	71.28	41.88	71.28	41.88	71.55	71.28	3.13	3.96	5.78	21.58	21.58	21.58	7	7	7
Clustreams [26]	KM_{EB}	56.92	55.75	58.40	56.80	72.69	42.37	71.28	41.88	71.28	41.88	77.21	71.30	0.40	3.16	NF	**19.52**	**19.52**	NF	2	2	NF
Planter	DT_{EB}	99.92	99.92	99.92	99.92	99.40	**94.53**	99.40	**94.53**	99.40	94.31	99.34	99.41	1.18	**1.34**	**1.34**	26.37	26.37	26.37	2	2	**2**
Planter	RF_{EB}	99.80	99.79	99.80	99.79	99.37	94.41	99.38	94.44	99.42	94.51	99.25	99.39	1.81	2.59	3.94	39.04	39.40	45.89	3	4	4
Planter	XGB	**99.98**	**99.98**	**99.98**	**99.98**	**99.42**	**94.53**	**99.42**	**94.53**	**99.43**	**94.59**	**99.40**	**99.45**	1.70	6.65	NF	33.22	45.78	NF	3	5	NF
Planter	NB	98.99	98.95	98.99	98.96	99.25	93.68	99.25	93.68	99.25	93.68	99.25	99.25	3.28	4.22	6.20	28.77	28.77	28.77	8	8	8
Planter	IF	44.89	35.35	37.90	31.08	84.86	58.90	63.83	45.07	86.33	55.05	81.74	NF	2.01	9.01	NF	36.30	43.33	NF	5	5	NF
Planter	KNN	69.33	60.63	99.38	99.36	87.51	31.55	99.30	93.17	99.30	93.17	78.24	92.73	**0.23**	1.89	21.01	20.74	20.74	22.22	**1**	**1**	5
Planter	PCA*	76.12	74.92	76.19	75.00	97.45	65.42	97.89	67.73	97.89	67.73	97.29	97.47	5.78	5.78	5.78	20.89	20.89	**20.89**	6	6	6
Planter	AE*	99.92	99.92	99.92	99.92	99.24	93.55	99.24	93.55	99.28	93.53	99.23	99.28	5.89	5.89	5.89	21.58	21.58	21.58	7	7	7

‡ NN is trained with PyTorch instead of Sklearn. **Bold** and underline indicate the top and second top performance among all models, respectively. * Results of PCA and AE are the accuracy of (S)mall DT using new features after dimensionality reduction. KM, PCA, and AE are unsupervised learning while others are supervised learning.

Table 4-1 highlights the effectiveness of the Planter framework in implementing various machine learning models on programmable switches for in-network ML defense, focusing on key metrics such as accuracy, memory usage, and inference efficiency. Key points and performance metrics include:

1. High Accuracy with Resource Efficiency: Planter's XGBoost (XGB) and Random Forest (RF) models achieve comparable or superior accuracy with lower memory consumption and optimized table entry usage compared to existing.
2. Model Flexibility and Optimization: Planter's models show strong scalability and performance by employing encode-based (EB) mapping techniques. Encode-Based (EB) Mapping is a highly efficient method for implementing ML inference in programmable data planes. By partitioning feature spaces and encoding regions, it reduces computational and memory demands and maintains consistent inference time.
3. Memory and Table Entry Management: Planter demonstrates better table entry efficiency for large models, preventing potential entry explosions seen in other

implementations, thus making it scalable and practical for hardware-constrained environments.

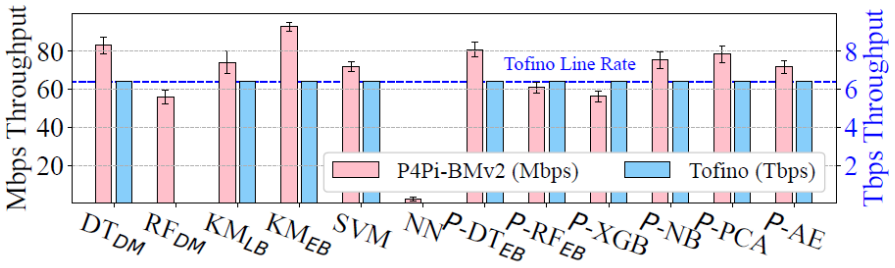


Figure 4.5 Throughput of ML algorithms for attack detection on Tofino (in Tbps) and P4Pi (in Mbps). Source: [Zhe24-2].

Figure 4.5 provides a throughput evaluation of Planter. It records the throughput of each in-network ML algorithm on a Tofino switch and P4Pi. The baseline throughput of basic forwarding is 6.4Tbps on Tofino and 94Mbps on P4Pi. On a Tofino switch, a full 6.4Tbps is achieved for all feasible models. On P4Pi, which essentially runs a software switch on a CPU, the results vary between models. Seven of the models achieve more than 80% of the baseline throughput.

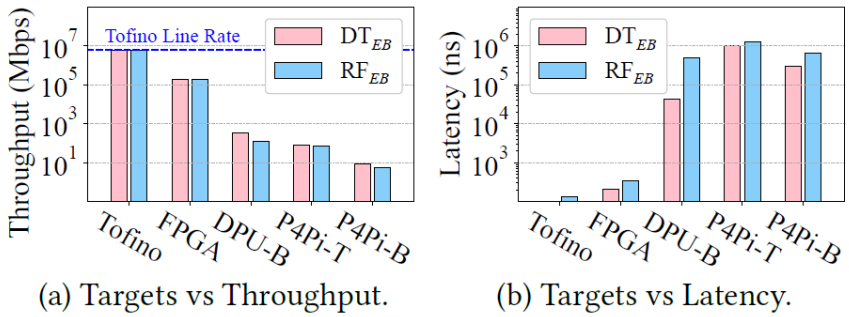


Figure 4.6 Throughput and latency of Encode-Based Decision Tree (DT_{EB}) and Encode-Based Random Forest (RF_{EB}) on different target devices. Source: [Zhe24-2].

Figure 4.6 provides a detailed comparison of throughput performance for Planter models on different types of target devices. On a 6.4Tbps switch-ASIC (Intel Tofino), Planter achieves full line rate, running models without dropping any packets. On an AMD Alveo U280 FPGA, it achieves the card line-rate, 100Gbps, without dropping traffic. On the NVIDIA DPU and P4Pi (Bmv2-based and T4P4S-based) it achieves 10's of Mbps to 100's of Mbps, depending on the model. Latency on all targets is exceptionally good, always achieving sub-millisecond latency on all targets except P4Pi, with microsecond-scale latency on Tofino and FPGA. This demonstrates Planter's efficiency in maintaining high throughput and low latency while embedding complex ML logic into the switch pipeline, showcasing its potential for high-performance, in-network data processing solutions.

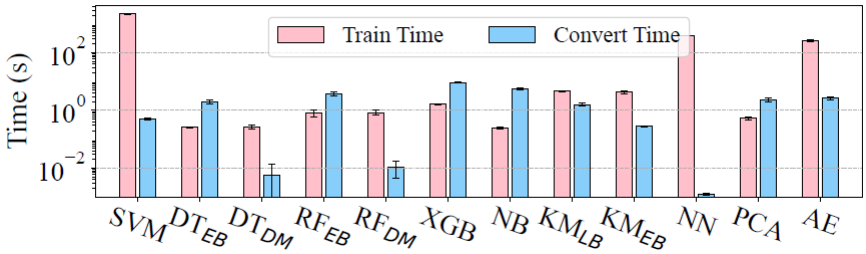


Figure 4.7: Algorithms' train & convert time (UNSW dataset). Source: [Zhe24-2].

Figure 4.7 depicts the measurement of the time required to train a model and convert the trained model to a network device. For a model using UNSW dataset under Anomaly Detection use case, most of the models' training time (except SVM, NN, and AE) and all of the models' conversion time are less than 10*s*, which shows Planter can prototype in-network ML fast.

This work was peer-reviewed and published in ACM SIGCOMM Computer Communication Reviews (CCR) [Zhe24-2]. It was presented in ACM SIGCOMM 2024 in the "Best of CCR" session, voted by CCR editors.

4.2.2 Hybrid In-Network Threat Defense

While in network ML is very efficient, the resource constraints of network devices don't allow running extremely large ML models. To attend to this challenge, SmartEdge builds upon a concept used in other resource-constrained applications domains: hybrid deployment. In SmartEdge, we develop a novel hybrid in-network threat defense solution called IIsy [Zhe24], combining a small model deployed within a network device, and a large model deployed in the backend.

The advantage of IIsy's hybrid deployments is that it benefits from the system performance of running an in-network model (low latency and high throughput), while also providing the high ML performance of a very large model running on a cluster.

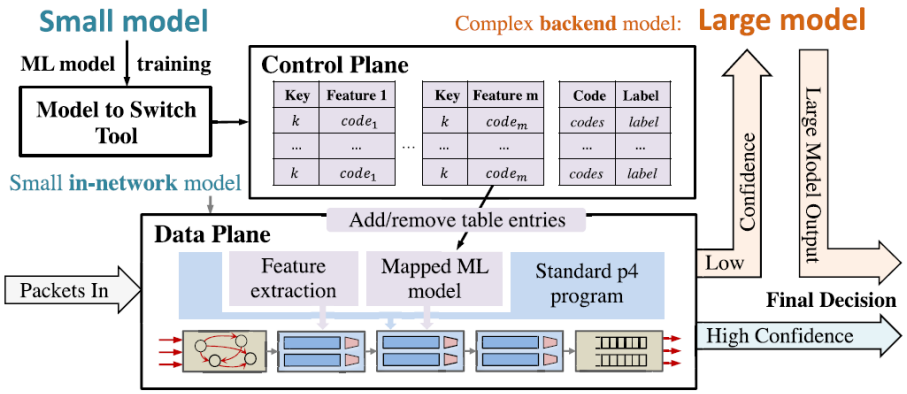


Figure 4.8: The high-level architecture of IIsy.

Figure 4.8 shows the high-level architecture of IIsy. The small in-network model handles the majority of classification tasks by leveraging the low-latency and high-throughput capabilities of

switches. When classification confidence is low or more complex processing is required, transactions are forwarded to the backend for further processing by the larger model. This hybrid deployment achieves near-optimal classification results while minimizing the load on backend resources, reducing latency, and enhancing the scalability of data-intensive applications.

The evaluation of Iisy used the UNSW-NB15 anomaly detection dataset, containing a mix of normal and malicious traffic. Intel Tofino switches were used for the in-network ML deployment.

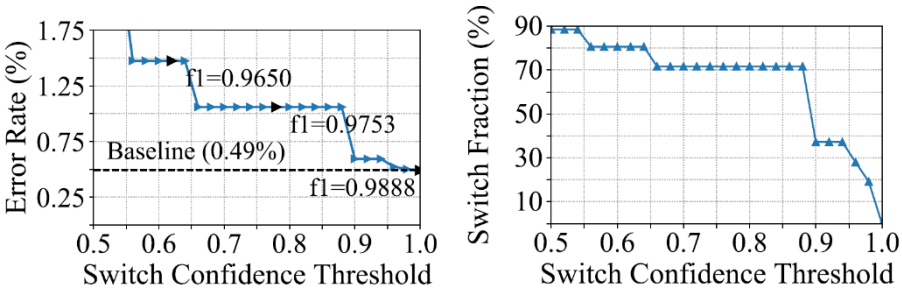


Figure 4.9: Anomaly detection in a hybrid deployment (Random Forest) -fraction of traffic handled by the switch and misclassification rate.

Figure 4.9 demonstrates how the Random Forest-based anomaly detection use case performs in terms of traffic offloading and misclassification rate at different switch classification confidence thresholds. At a confidence threshold of 0.7, the switch successfully handles 84.5% of the traffic, but with a misclassification rate of 1.03%, slightly higher than the baseline rate of 0.49%. Increasing the threshold improves accuracy but reduces the faction of traffic processed by the switch.

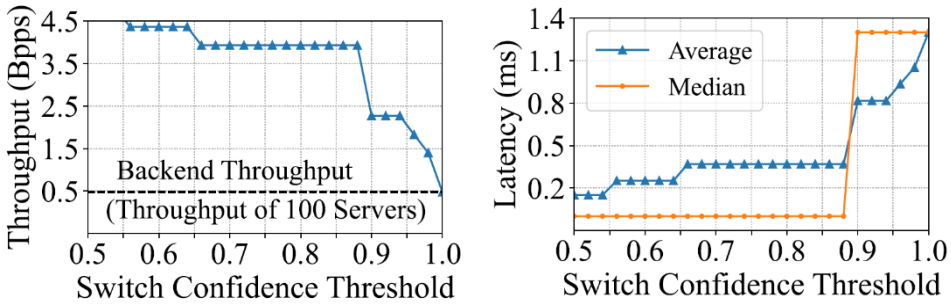


Figure 4.10: Throughput and latency of hybrid deployment in anomaly detection use case.

Figure 4.10 shows that in the hybrid deployment of anomaly detection, throughput increases as the fraction of traffic handled by the switch grows, while latency decreases correspondingly. With the same 0.7 confidence threshold, more than 3.5B messages can be processed per second, with a median latency of less than 0.1ms. This evaluation highlights how offloading traffic to the switch enhances overall system throughput and reduces latency.

The latency and throughput impact of different models were evaluated. For anomaly detection, latency increased by less than 5% compared to the baseline networking-only functionality, with throughput maintained at up to 6.4 Tbps. The IIsy framework is efficiently used under 9.3%

memory for anomaly detection, achieving line-rate processing and robust ML performance within programmable switch constraints.

This work was peer-reviewed and published in IEEE/ACM Transactions on Networking [Zhe24].

4.2.3 Federated In-network Defense

SmartEdge In-Network Defense solution attends to the challenges in scaling ML-based solutions at the edge, while also maintaining data privacy, using a federated learning based solution.

The basic architecture and design of the solution, named FLIP4, was provided in D4.1, and is omitted from this report.

There are three aspects of innovation in SmartEdge's federated learning-based defense:

1. It provides a lightweight learning process which is suitable for a distributed IoT setup.

2. It provides a secure data sharing process with low overhead, protecting against malicious intervention.

3. The solution creates a global model using local data and drives in-network inference at the edge.

The following Figure 4.11 illustrates the architecture of the solution:

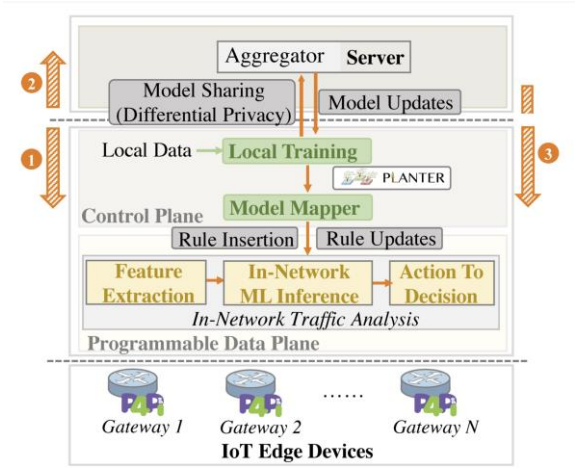


Figure 4.11: Federated Learning Framework Architecture. Source: [Zan24].

Implementation. Data plane code is written in P4 using v1model architecture and prototyped on Raspberry Pi using P4Pi-v.0.0.3. The controller and server functionality were implemented in Python. The prototype extended the design in FL algorithm [Mad22] and *Planter* [Zan23-1, Zhe24-2] with more ML training and inference functions. Network emulation was done using Mininet for multiple-node setup, with baseline results using offline training of neural networks in PyTorch and XGBoost using *sklearn*.

Network Setup. The deployment assumes a distributed network scenario where the network is configured with a limited number of connected gateways. A second network-edge scenario assumes a mobile edge network with edge computing capability is considered, processing traffic

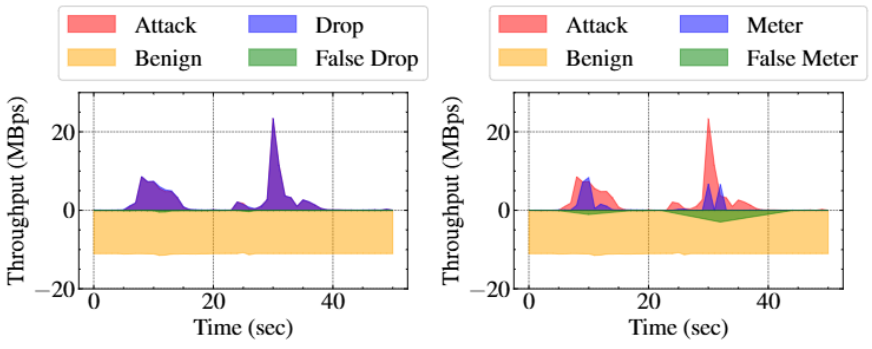
at Terabit-throughput. To simulate multiple-access gateways, a network topology is built in Mininet to the network edge, with multiple switches acting as gateways, which are connected to a remote server via switches. The number of devices in the topology varies across experiments.

Dataset. Three public datasets are used for the evaluation: IoT Sentinel [Mar17], CICIDS 2017 [Ima18], and CIC-AndMal2017 [Ara18].

Evaluation Metrics. The evaluation considers both machine learning performance metrics (e.g. accuracy, Receiver Operating Characteristic, Area Under the Curve) and system performance metrics (e.g., latency, throughput, resources).

Machine Learning Performance Results the evaluation compares FLIP4 using XGBoost with the State of the Art BNN [Qia20] and models such as neural networks, KNN and Naïve Bayes. Using the CICIDS 2017, we find that FLIP4 achieves a global accuracy of 0.919 for online operation and 0.988 for offline operation, compared with 0.9 and 0.92 (correspondingly) for BNN. On the IoT Sentinel dataset, FLIP4 achieves 0.962 online and 0.966 offline global accuracy, compared with 0.936 and 0.968 for BNN. We note that the accuracy is inversely affected by FLIP4's privacy measures, which are an operational choice. A sensitivity test found that setting the privacy parameter ϵ at 0.2 offers an optimal balance, minimizing accuracy loss. Compared with BNN [Qia20], FLIP4 achieves similar or slightly better performance for IoT identification as the number of gateways increases and performs significantly better for malware detection.

System Performance Results Data sharing volume using FLIP4 is 60% lower than BNN, for CICIDS 2017. Local training on gateways takes ~0.01s, with global model integration within ~0.71s and inserting updates to ML inference table taking 0.11s. Scalability evaluation, ranging from 2 to 10 gateways, shows stable attack detection performance, very close to global offline performance, for attack detection, IoT identification and malware detection, significantly better than using only a local model without information sharing. The measured latency of FLIP4 is around 2ms, compared with 10ms of state-of-the-art [Qia20]. The throughput on Raspberry Pi is 54Mbps, compared with 2Mbps of [Qia20], on the same setup.



(a) Drop-based mitigation (b) Metering-based mitigation

Figure 4.12: Federated learning based in-network traffic mitigation using FLIP4. Source: [Zan24].

Figure 4.12 demonstrates FLIP4's in-network traffic mitigation capabilities using either drop-based or metering-based actions. In drop-based mitigation, malicious traffic is immediately removed upon detection, providing fast response but potentially causing false positives to disrupt normal traffic. Metering-based mitigation moderates traffic rates for detected threats, reducing impact on overall throughput and minimizing service disruption from false alerts. This balanced approach ensures both security and service continuity in network operations.

This work was peer-reviewed and published in IFIP Networking 2023 Sec4IoT Workshop [Zan23-2] and ACM Transactions on Internet Technology [Zan24].

4.3 FTG-NET-E: A HIERARCHICAL ENSEMBLE GRAPH NEURAL NETWORK FOR DDOS ATTACK DETECTION

The rapid advancement of networking and computing technologies has brought about an unprecedented increase in cyber-attacks, posing severe threats to economic, political, and social stability. Among these, Distributed Denial of Service (DDoS) attacks stand out as particularly disruptive, overwhelming online services and rendering them inaccessible to legitimate users. As these attacks grow in frequency and sophistication, they pose a critical challenge to the integrity and availability of online resources. Traditional detection systems, though effective in controlled environments, often struggle to adapt to the dynamic and complex nature of real-world network traffic, necessitating the development of more robust and adaptable solutions. The persistent and evolving threat of DDoS attacks necessitates the creation of advanced detection mechanisms that can operate effectively in real-time environments. The limitations of conventional Machine Learning (ML) and Deep Learning (DL) approaches, particularly their reliance on static features and susceptibility to overfitting, highlight the need for innovative methodologies that can generalize across diverse network conditions. This project seeks to address these challenges by introducing a novel detection system that leverages Graph Neural Networks (GNNs) and ensemble learning. By focusing on the topological structure of network traffic and employing a hierarchical graph representation, the proposed system aims to enhance detection accuracy while reducing computational overhead. This approach not only promises to improve resilience against DDoS attacks but also paves the way for more adaptable and efficient cybersecurity solutions in increasingly complex digital landscapes using flow graph.

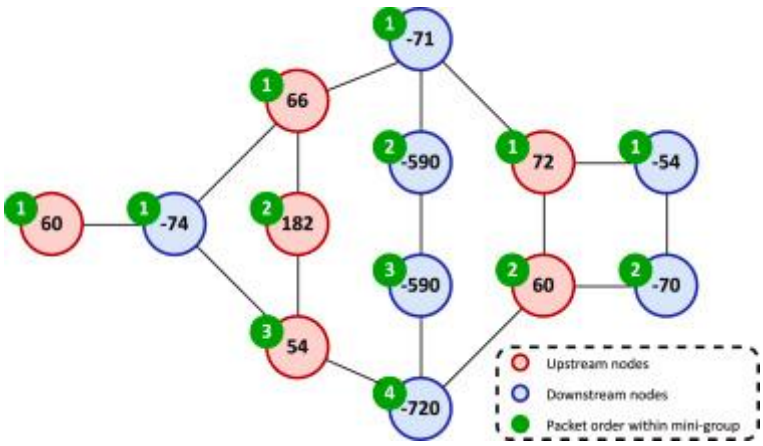


Figure 4.13: Flow Graph example: upstream packets are denoted by positive values, while downstream packets are represented by negative values. The sequence of mini-groups is organized from left to right.

Flow graphs are a vital component of our approach, which draws inspiration from the graph structure introduced in GraphDDoS but with key modifications to enhance its applicability. The

main difference is the use of time slots instead of node limits. In this structure, all packets exchanged between two endpoints within a given time slot are grouped, even if they belong to different network flows. Within each group, packets are organized chronologically and converted into nodes, where each node has packet length as its main characteristic. Focusing on packet length is strategic because it can detect common malicious patterns in network traffic, such as large packets often associated with DDoS attacks and small packets that indicate port scanning activity. Additionally, packet lengths can be easily obtained from network traffic data, facilitating efficient on-the-fly processing and enhancing model interpretability.

The flow diagram is further enriched by assigning direction flow values: upstream (client to server) packets are represented by positive values. In contrast, downstream (server to client) packets are represented by negative values. This difference allows for a more granular understanding of traffic directionality, which is critical for accurately detecting and classifying network behavior.

In the flowchart, consecutive packets sent by the same endpoint form a mini-group, and edge connections represent edges from neighboring nodes from the same mini-group. Additionally, a connection is made between the first packet of the minigroup and the first packet of the previous and subsequent minigroups, mimicking the connection pattern of the last packet within those groups. This structure provides a comprehensive view of packet relationships within a single flow, allowing the identification of anomalies that may indicate certain types of DDoS attacks. Additionally, it captures relationships between multiple flows involving the same endpoint, which is valuable for detecting bursty and periodic traffic patterns (common indicators of DDoS attacks).

Traffic graphs extend the concept of graphs by representing traffic over time throughout the network. For each time slot, each endpoint pair that appears in at least one packet of the traffic profile is converted into a graph node. The features of these nodes are derived from the output vectors of the flow GNN, making the flow graph a higher-level abstraction encompassing the interactions between multiple flows. If two nodes share the same source or destination IP address, an edge is added between them, creating a network topology that reflects real-world communication patterns. This traffic graph is particularly valuable for visualizing and analyzing the distribution of devices sending requests to the server and the specific behavior of individual flows. By encoding flow-level patterns into the flow-level topology, flow graphs provide a comprehensive view of the network, capturing the details of individual flows and the broader interactions between multiple flows. This dual perspective is critical for identifying complex attack patterns that may be absent when examining individual flows in isolation.

Feature engineering plays a crucial role in the effectiveness of our graph-based network traffic analysis approach. It involves extracting and selecting relevant attributes from raw network traffic data and then using these attributes to build nodes and edges in the traffic graph. Key attributes include flags such as URG, SYN, RST, PSH, FIN, ECE, and ACK, average packet size, streaming packets per second, destination port, and protocol information obtained from the packet header (e.g., IPv6, TCP, UDP) Traffic statistics.

These features were selected based on their relevance to intrusion detection and ability to capture a wide range of network behavior. We conducted feature selection to identify the most influential features and evaluate their contribution to the overall model performance. To ensure consistency in the learning process, features are scaled to maintain a uniform range and

distribution, preventing any feature with larger magnitudes from dominating the model. In addition, imputation techniques addressed missing values in the data set, ensuring the data were complete and available for analysis.

By combining these carefully designed features with our graph-based models, we can accurately detect and classify network traffic patterns, distinguishing benign from malicious activity with high accuracy. This comprehensive approach to feature engineering is critical to achieving robust and reliable DDoS detection in real-time network environments.

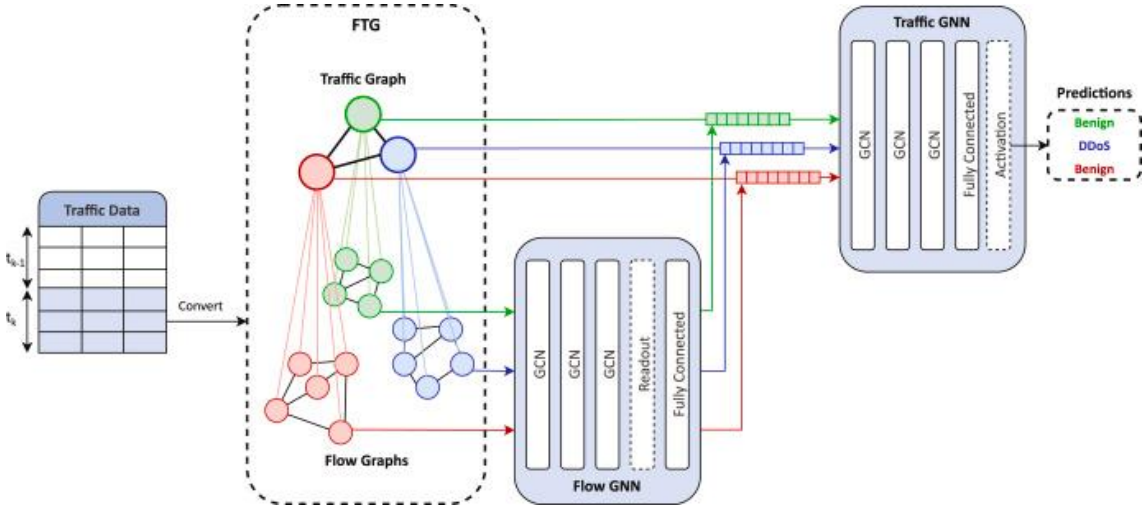


Figure 4.14: FTG-Net-E architecture. (1) "Traffic Data Preprocessing": Training on traffic data converted into flow graphs. (2) "Flow GNN" and "Traffic GNN": Learning representations of individual flow graphs and entire traffic graphs, respectively. (3) "Fully Connected Layer", gives the final predictions based on the Traffic graph network outputs as Benign and DDoS.

Figure 4.15 illustrates the confusion matrix for the FTG-Net-E model when trained with the best hyperparameters. The model achieved 235,640 true positives and 239,534 true negatives, with only 2,915 false positives and 29 false negatives. These results yield a good F1 score, which confirms the model's effectiveness in accurately classifying both positive and negative cases.

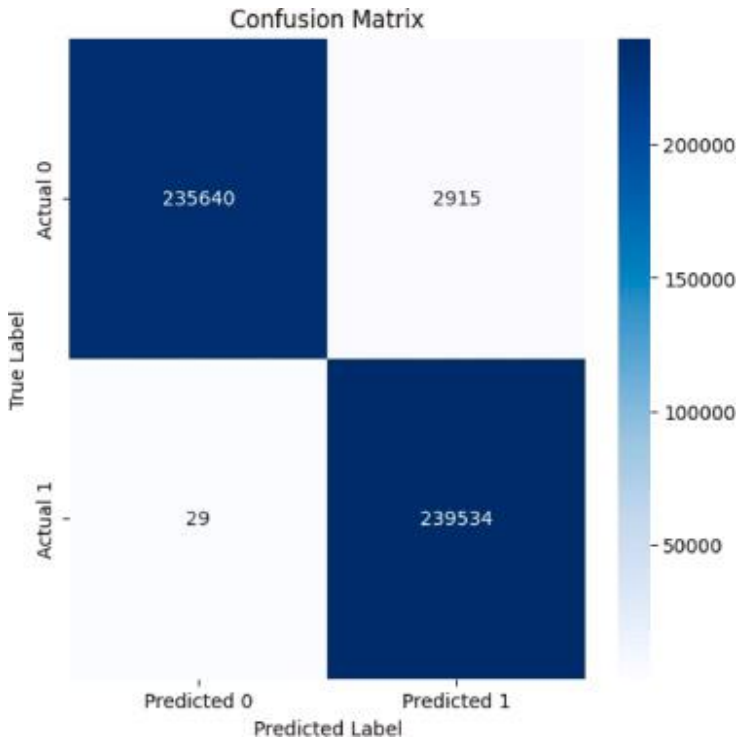


Figure 4.15: FTG-Net-E confusion matrix.

Figure 4.16 presents the performance metrics for the FTG-Net-E model, which achieved an F1 score of 0.9929, accuracy of 0.9967, precision of 0.9929, and recall of 1.0. These metrics indicate the model's capability to accurately predict both positive and negative instances while minimizing false positives and negatives, crucial for real-world applications.

Figure 4.17 shows the ROC curve for various hyperparameter configurations of the GNN models. The area under the curve (AUC) demonstrates the model's discriminative power across these configurations. The ensemble GNN models consistently achieved high AUC values, indicating their effectiveness in distinguishing between malicious and legitimate traffic. The consistent performance across different configurations underscores the robustness and generalizability of the proposed approach.

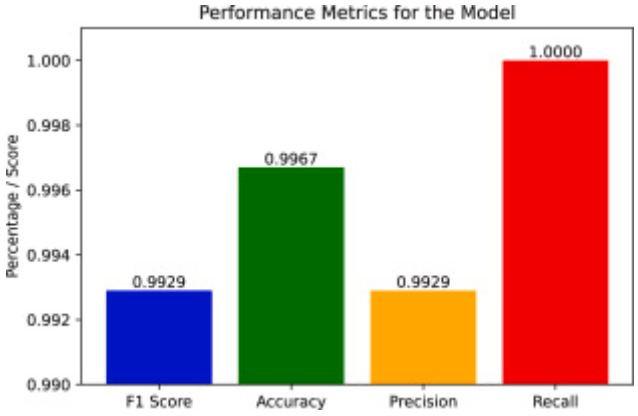


Figure 4.16: Performance metrics for the best FTG-Net-E model.

We assessed the FTG-Net-E model using 5-fold cross-validation, focusing on the impact of time slot size on accuracy and inference time. As shown in Figure 4.17 (a), the ensemble model

outperforms the simple GNN model in weighted F1-score across various time slot sizes. Figure 4.17 (b) highlights the average inference time for each model, showing that FTG-Net-E offers a better trade-off between accuracy and computational efficiency, particularly for time slot sizes up to 20.

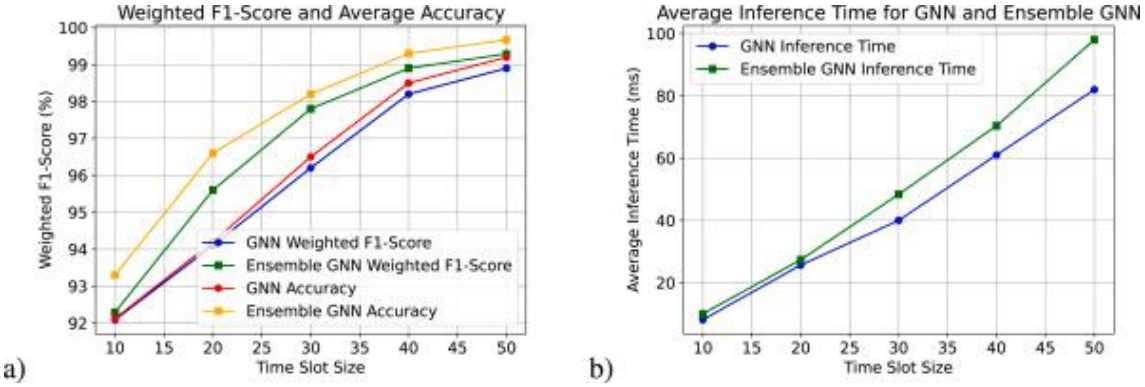


Figure 4.17: (a) Weighted F1-Score and accuracy for GNN and Ensemble GNN (b) Average inference time for GNN and Ensemble GNN models results using different time slot sizes.

This work was peer-reviewed and published in Computer Networks Journal [Bak24-1].

4.5 INDDoS+: Secure DDoS Detection Mechanism in Programmable Switches

Volumetric distributed Denial-of-Service (DDoS) attacks pose a significant threat to networks, as their goal is to deplete resources meant for legitimate users and disrupt network services. Recent years have seen DDoS detection mechanisms designed to operate directly within programmable switches, enabling efficient in-network threat detection [Din21]. To this end, current solutions use advanced data structures to monitor the number of connections targeting destination hosts. A destination host is flagged as a potential victim if there is a sudden spike in connections exceeding a predefined threshold.

However, emerging in-network DDoS detection mechanisms also mean new potential vulnerabilities. Specifically, we identified two new attack vectors on data structures within programmable network devices: inflating attacks and evasion attacks. In an inflating attack, a malicious actor does not launch a true DDoS attack against a destination. Instead, it tricks the in-switch data structure to indicate that there is a spike in the number of connections to a switch. In an evasion attack, the malicious actor launches a DDoS attack against a destination, but can hide their attack within the data structure without an indication that the number of connections has increased.

To address these weaknesses and mitigate the two new attacks, we designed INDDoS+ [Din24], using two BACON sketches with different hash functions. INDDoS+ is resilient against such attacks and can reliably detect DDoS attempts even when operating under limited hardware resources.

CAIDA 2018 [Cai18] trace was used in the evaluation of INDDoS+ as benign traffic, divided into short time intervals as described in [Din24]. Each time interval contains nearly 2.3 million packets and 60 thousand unique source IP addresses. It takes less than a 100 keys and attempts to fool a BACON sketch, and less than 10,000 packets to maliciously label all destinations as victims of an inflating attack.

For an evasion attack, we find that increased monitoring period (e.g., 5s instead of 1s) significantly increases the number of susceptible keys to hide an attack: increasing from 3456 to 17504 (out of 64K).

Table 4-2 COMPARISON OF DDOS VICTIM IDENTIFICATION PERFORMANCE BETWEEN INDDOS [Din21] AND INDDOS+

Strategy	BACON Sketch size ($d \times w \times m$)	# BACON Sketches	Recall	Precision	F1 score
INDDoS	$3 \times 1024 \times 1024$	1	0.96	0.99	0.97
INDDoS+	$3 \times 512 \times 1024$	2	0.96	0.79	0.86
INDDoS+	$3 \times 1024 \times 512$	2	0.17	0.96	0.28
INDDoS+	$3 \times 1024 \times 1024$	2	0.96	0.99	0.97

Table 4-2 (DDoS Victim Identification Performance Comparison) compares the performance of INDDoS and INDDoS+ in identifying DDoS victims under varying BACON Sketch sizes. The results demonstrate that INDDoS+ maintains high recall, precision, and F1 scores even when hardware resources are constrained, offering a robust and accurate detection capability. By minimizing resource overhead while maintaining reliable detection, INDDoS+ proves to be a highly effective solution for mitigating DDoS attacks.

This work was peer-reviewed and published in IEEE 25th International Conference on High Performance Switching and Routing (HPSR 2024).

5. HARDWARE-ACCELERATED IN-NETWORK OPERATIONS

5.1 DPUAUT: Secure Authentication Protocol with SmartNiC Integration for Trustworthy Communications in Intelligent Swarm Systems

The development of autonomous systems based on swarm intelligence in next-generation networks has extensively promoted the development of transportation and other key industries. These systems excel at performing complex tasks such as planning, real-time navigation, and tracking. However, their high processing and communication requirements on the underlying infrastructure may prevent them from reaching their full potential. Edge-centric systems have been introduced as a promising solution to alleviate these challenges. Deploying edge-centric servers and micro data centers close to clustered devices and leveraging data processing units (DPUs) such as Bluefield SmartNICs can alleviate load management issues, reduce communication latency, and optimize bandwidth utilization. While these advances increase the effectiveness and efficiency of autonomous systems, they also introduce new security vulnerabilities. The proximity of private cloud servers and wireless communication media increases the risk of security breaches. Attackers can exploit these systems to intercept, tamper with, or manipulate data, compromising the security and privacy of the devices involved. Therefore, developing secure and efficient authentication protocols is critical to protect the operation of cluster-based autonomous systems in edge-centric environments.

Swarm devices will become an integral part of next-generation networks because of their ability to handle large, distributed tasks efficiently. However, more than traditional cloud-based frameworks are required to meet these devices' needs due to their low latency and scalability requirements. Edge computing provides a viable solution to latency and scalability issues by bringing computing resources closer to devices. However, the shift to edge computing also brings new security threats, such as vulnerability to physical attacks, counterfeiting, and hardware tampering. Recent research has proposed unclonable function (PUF)-based authentication protocols to improve the security of edge computing environments. However, these protocols must often be revised to provide the strong physical security required for autonomous intelligent swarm systems (ISS). Ensuring the security and reliability of these systems involves secure and efficient authentication schemes. The challenge is to verify the physical identity of swarms of devices in different environments, including smart cities, industrial environments, and autonomous vehicle networks. With these challenges in mind, this deliverable proposes a novel authentication scheme designed for the autonomous International Space Station, addressing the limitations of existing protocols and providing a more secure and efficient solution.

Physically Unclonable Function (PUF) is a hardware security feature that generates a unique identification code based on the physical characteristics of the device. PUFs exploit the variations inherent in the manufacturing process to create device-specific fingerprints that are nearly impossible to copy or replicate. Leverage this uniqueness to enhance the security of the authentication protocol, making it difficult for attackers to copy or tamper with the device's security features. In the context of autonomous ISS, the PUF plays a key role in generating a

unique identification number for each device, which is then used to authenticate the device within the system. The proposed authentication protocol incorporates PUF to resist various forms of attacks, including physical tampering, side-channel attacks, and replay attacks. Given that clustered devices typically operate in resource-constrained environments, PUFs must also be lightweight and energy-efficient.

In this deliverable, the PUF is implemented in SmartNiC BlueField-2 hardware, consistent with the best practices outlined by Aitchison et al. They are working on integrating PUF into ARM TrustZone security technology. Implementation focuses on several key requirements:

I. Resistant to attacks: The PUF must be able to withstand physical attacks, machine learning attacks, and attempts to predict its response by observing its output patterns.
II. Ease of implementation: The PUF should be easy to integrate into SmartNiC BlueField-2 with minimal impact on the overall performance and cost of the device.
III. Latency improvements: PUF should help reduce latency in the authentication process, which is critical for on-the-fly operations in cluster-based systems.

DPU-based PUF operates by generating a unique response message (Ri) in response to a specific challenge message (CHi). The nature of the hardware ensures that each PUF responds uniquely, even when faced with the same challenges. The security of PUFs is further enhanced by ensuring that the Hamming distance between responses from different PUFs exceeds a predefined threshold, making it extremely unlikely that no two devices will produce similar outputs. This security mechanism is essential to ensure the reliability and uniqueness of the identity verification process, providing a solid foundation for protecting the autonomous International Space Station from a variety of attacks. PUF-based authentication protocols are designed to handle the dynamic and demanding environment in which clustered devices operate, ensuring security is maintained without compromising performance.

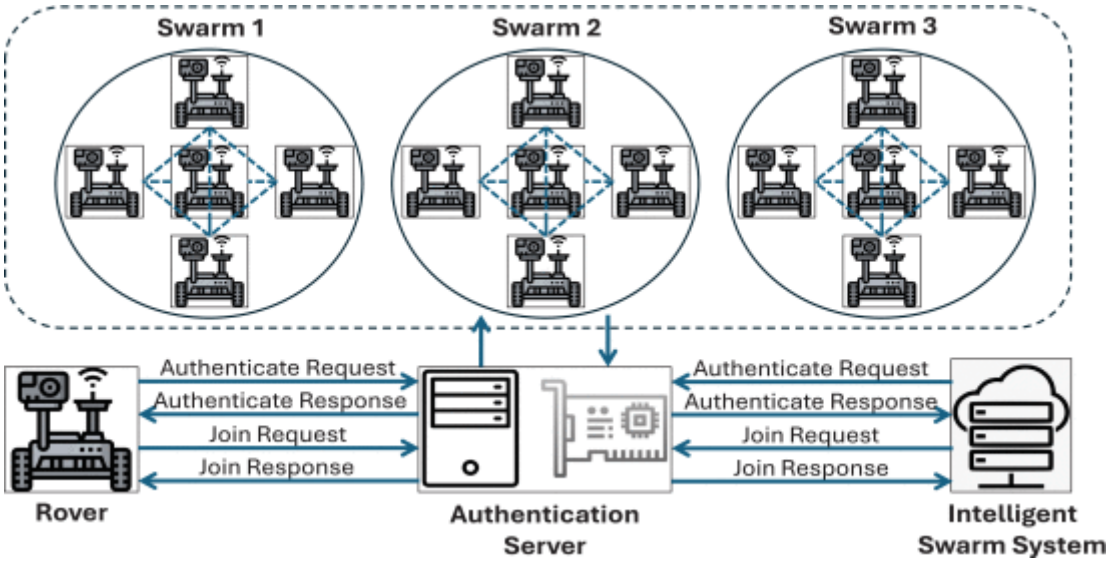


Figure 5.1: Autonomous swarm intelligence system.

The proposed protocol, DPUAUT (Secure Authentication Protocol with SmartNiC Integration for Trustworthy Communications in Intelligent Swarm Systems), is designed to enhance the security and reliability of communication within Intelligent Swarm Systems (ISS). By integrating Smart Network Interface Card (SmartNiC) technology, this protocol ensures that autonomous devices within the swarm can securely authenticate and communicate with each other. The protocol is

structured into four key phases: Initialization, Device Registration, Autonomous Swarm Devices Computing Server (ASDCS) Registration, and Authentication.

ISS-Initialization Phase

The ISS-Initialization phase is where the Primary Control Server (PCS) sets up the essential cryptographic parameters for the network. The PCS begins by selecting a Master Secret Key (MSK) randomly from a prime number set to ensure its confidentiality. Along with this, a one-way hash function is chosen. The MSK is securely stored within the PCS, while the hash function is made public for use by all system components during cryptographic operations.

Device Registration Phase

During the Device Registration phase, each autonomous swarm device must register with the PCS to establish its identity within the network. The process starts when the device sends its identity *ID* as a registration request to the PCS. The PCS verifies whether *the ID* is unique; if it is not, the PCS prompts the device to select a different identity.

After a valid registration request is received, the PCS generates a challenge message and sends it to the device. The device then uses its Physical Unclonable Function (PUF) to generate a response message, which is computed as by applying the PUF function to the challenge message. This response is sent back to the PCS.

The PCS then verifies the digital signature of *the device*. If valid, the PCS assigns a temporary identity to the device and computes a key using the hash function. The PCS securely stores all related registration data and sends the pair (key, temporary ID) back to the device, which stores it in its Trusted Personal Device (TPD) for future authentication use.

ASDCS Registration Phase

In the Autonomous Swarm Devices Computing Server (ASDCS) Registration phase, the ASDCS undergoes a secure registration process with the PCS to establish its identity. This process involves the exchange of unique identifiers and pseudo identities, the generation of a challenge message, and the use of digital signatures to ensure authenticity and integrity.

Upon receiving the registration request, the PCS computes a digital signature over the message using its private key. The PCS then sends this signed message to the device, which verifies it using the corresponding public key. If the verification is successful, the PCS computes a secret key, securely stores the mapping of the key against the ID, and transmits the encrypted pair (key, ID) to the device for use in future authentication.

Authentication Phase

The Authentication phase is crucial for establishing a secure communication channel between an autonomous swarm device and the ASDCS. The process begins with the *Authentication Server for Device* Interface (ASDI) sending an authentication request to *ASDCS*, including a unique identifier (UID) and a challenge message. Upon receiving the request, *ASDCS* verifies the authenticity and its challenge response. If verified, *ASDCS* sends a verification message back to

ASDI, allowing both devices to derive a shared key for secure communication. This key is generated using the secret key shared with the PCS and other relevant cryptographic data.

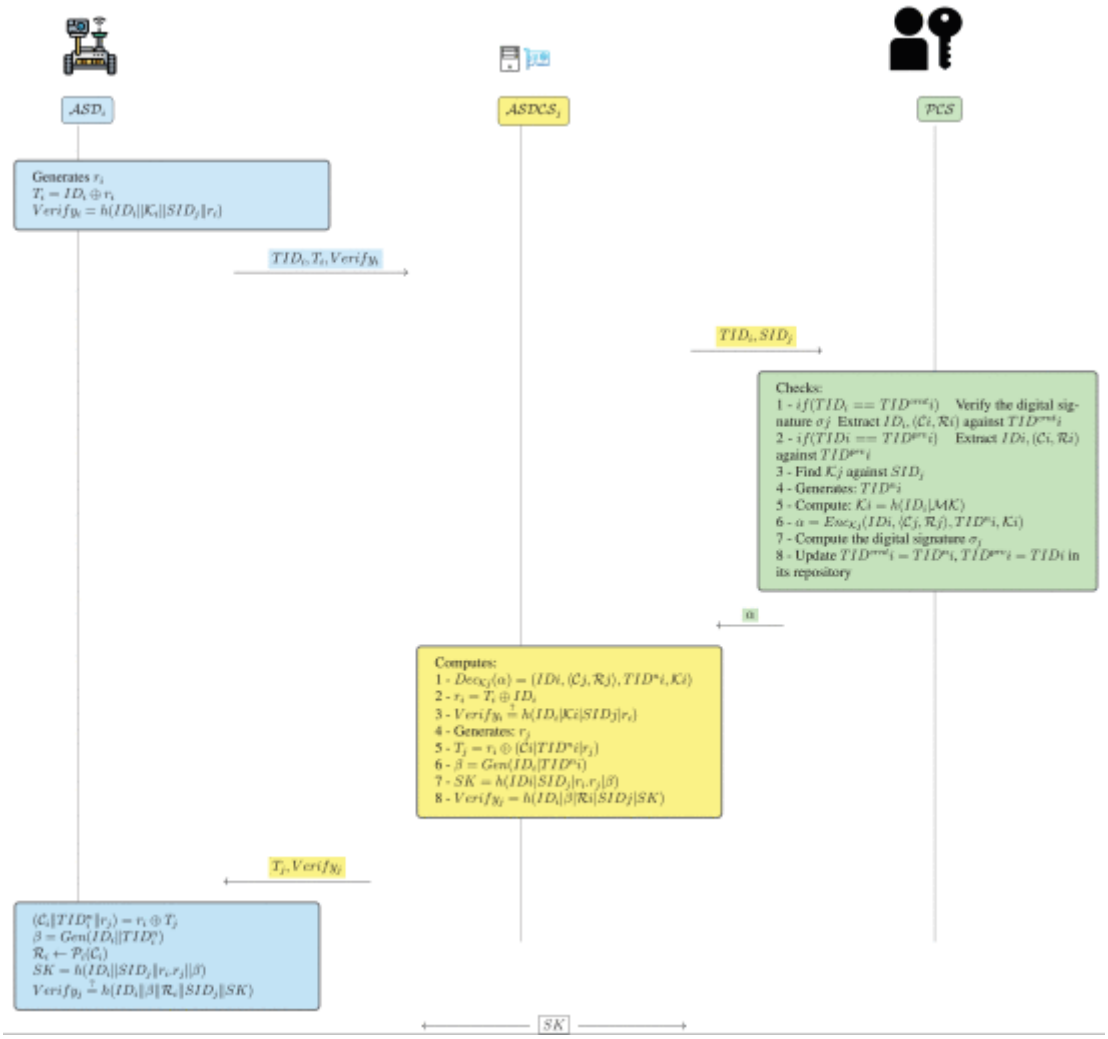


Figure 5.2: Autonomous swarm intelligence system.

Key steps in this phase include the generation of random nonces, the computation of digital signatures, the exchange of cryptographic credentials, and the derivation of a mutually agreed-upon session key. The integrity of the session is continuously verified, and any tampering results in the immediate termination of the session to prevent security breaches.

The formal description of the whole procedure is reported in [Bak24-2].

Performance and Security Features Evaluation

In this section, we provide a comprehensive analysis of the performance and security features of our proposed Secure Authentication Protocol with SmartNiC Integration for Trustworthy Communications in Intelligent Swarm Systems (DPUAUT). We compare our protocol with several state-of-the-art protocols, focusing on key performance metrics and security features.

Table 5-1 Evaluation of Authentication Schemes Based on These Security Properties A1: Mutual Authentication A2: Device Anonymity, A3: Provide Perfect Forward Secrecy, A4: Replay Attack, A5: Key Agreement, A6: Device

Impersonation Attack, A7: Withstand De-Synchronization Attack, A8: Formal Security Analysis, A9: Informal Security Analysis

Schemes	Security Properties								
	A1	A2	A3	A4	A5	A6	A7	A8	A9
[14]	✓	✓	✓	✓	✓	✓	×	✓	×
[17]	✓	✓	×	✓	✓	✓	×	×	×
[18]	✓	✓	×	✓	✓	×	✓	✓	×
[22]	✓	✓	✓	✓	✓	✓	×	✓	✓
[20]	✓	×	×	×	×	✓	×	✓	×
[24]	✓	✓	✓	×	✓	✓	×	✓	×
[25]	✓	✓	✓	✓	✓	✓	×	✓	✓
[26]	✓	✓	×	✓	✓	✓	×	✓	✓
DPUAUT	✓	✓	✓	✓	✓	✓	✓	✓	✓

Computation Overhead

We evaluated the computation overhead of our proposed protocol and compared it with various state-of-the-art protocols. The results, summarized in Table 5.2, show that our protocol exhibits lower computation overhead. Figure 5.3 visually compares the computation time for our protocol and the state-of-the-art protocols. Our protocol demonstrates significantly lower computation overhead at both ends (ASDi and PCSj) compared to other protocols.

Table 5-2 Analysis of Computation and Communication Overheads

Protocols	$\mathcal{ASD}_i$ Side	$\mathcal{ASDCS}_j$ Side	Aggregated Computation Overhead	Aggregated Communication Overhead
DPUAUT	$3T_h \approx 6.06$ ms	$3T_h + 1T_{e/d} \approx 0.01$ ms	6.161 ms	1052 bits
[22]	$3T_h + 1T_{e/d} \approx 8.888$ ms	$3T_h + 1T_{e/d} \approx 0.373$ ms	9.261 ms	2145 bits
[18]	$5T_h \approx 11.755$ ms	$4T_h \approx 0.392$ ms	12.147 ms	1568 bits
[17]	$6T_h \approx 14.106$ ms	$6T_h \approx 0.588$ ms	14.694 ms	1600 bits
[14]	$9T_h + 2T_{pm} \approx 24.405$ ms	$6T_h + 1T_b + 1T_{pm} \approx 1.756$ ms	26.161 ms	1664 bits

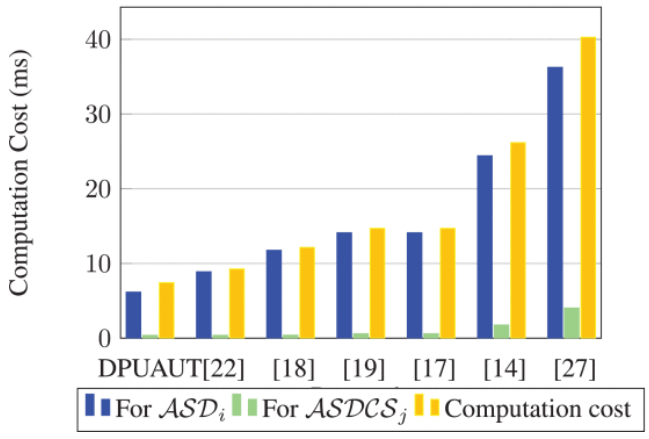


Figure 5.3: Computation cost comparison.

Communication Overhead

Communication overhead refers to the amount of data exchanged between entities during the execution of the protocol. We compared the communication overhead of our protocol with that of state-of-the-art protocols.

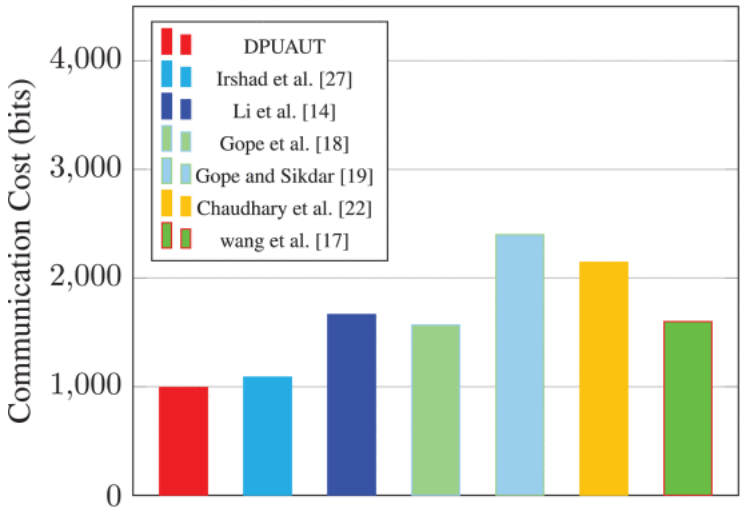


Figure 5.4: Communication cost comparison.

The results are illustrated in Figure 5.4, showing that our proposed protocol outperforms existing protocols in terms of communication overhead. The proposed protocol shows significant improvements in both computation and communication overhead compared to existing state-of-the-art protocols. The results underscore the efficiency and lightweight nature of our solution, making it a promising choice for secure authentication in autonomous vehicle swarm systems.

This work was peer reviewed and published in the IEEE Access Journal [Bak24-2].

5.2 Intelligent Roadside Unit

Background

A pivotal component within the framework of intelligent transportation systems (ITS) such as in UC-2, is the roadside unit (RSU), a multi-functional device strategically positioned alongside roadways. RSU facilitates vehicle-to-everything (V2X) communication, enabling vehicles, infrastructures, and the cloud to exchange crucial information with each other. Nevertheless, its primary function is not limited to being a wireless communication device that provides connectivity support to passing vehicles; rather, it is to serve as an intelligent computation, memory, and networking hub.

Such intelligent RSU offers more task-offloading options for complex ITS scenarios. Compared to computing on the vehicle side, intelligent RSUs naturally facilitate vehicle-to-vehicle and vehicle-to-infrastructure cooperations, cutting down the redundant sensors and repetitive computation across different vehicles; compared to computing on the traditional edge cloud servers, intelligent RSUs decrease latency and network traffic, shortening the data flow pathway.

One of the foremost advantages of utilizing the RSUs for computational tasks is the reduction of round-trip latency, as these tasks no longer need to access the cloud. In time-critical scenarios such as collision avoidance and intersection management, round-trip latency should be minimized to at most a few tens of milliseconds and even smaller for the computation time at

RSU. The demand for low-latency computation at the RSU underscores the need for acceleration to fully achieve the potential of intelligent RSUs.

SmartEdge Intelligent RSU

Within the context of SmartEdge Use Case 2, this scenario is centered on the smart junction scenario, encompassing multiple intersections. In this scenario, stationary radars, cameras, and controllable traffic lights are deployed on the roadside, all connected to the RSU. Three categories of vehicles are involved: ordinary vehicles, dummy vehicles, and smart vehicles. Dummy vehicles can send information to the RSU, but do not respond to RSU's messages. Smart vehicles are equipped with cameras, LiDARs, GPS, and wireless communication, enabling bidirectional data exchange with the RSU. This dynamic setting forms the background for our development of an intelligent RSU.

SmartEdge will accelerate sensor fusion and collaborative perception at the RSU to enable the provision of time-critical services for ITS. It will initially focus on accelerating the integration of multi-modal and multi-agent data at the RSU. This integration involves combining raw or processed radar data from the roadside, raw or processed LiDAR data from smart vehicles, as well as GPS information from smart and dummy vehicles to create a comprehensive data repository. Smart vehicles can query this dynamic repository to enhance their situational awareness and the RSU itself can utilize it as well. The integration of camera data is considered a stretched goal for future consideration, and the primary focus lies in accelerating this data fusion process.

SmartEdge will leverage sensor fusion and collaborative perception results to develop time-critical and resource-intensive services such as collision avoidance. This will be achieved, for example, by controlling traffic lights and other infrastructures, as well as communicating with swarm's nodes. The aim is to harness the capabilities of intelligent RSUs to improve transportation safety and efficiency.

Scenario

As Figure 5.5 shows, the project begins with a simplified scenario where sensors pre-process raw data and return only the object ID, bounding boxes, and kinetic information according to settings in the Smart Junction test area in Mobility Lab Helsinki. The scenario involves two primary data sources: (1) the vehicle transmits its location and kinetic data via V2X communication, collected through onboard sensors such as GPS; (2) additional sensors like cameras, LiDARs, and radars, installed on the vehicle or roadside, come with built-in object detection and tracking functions.

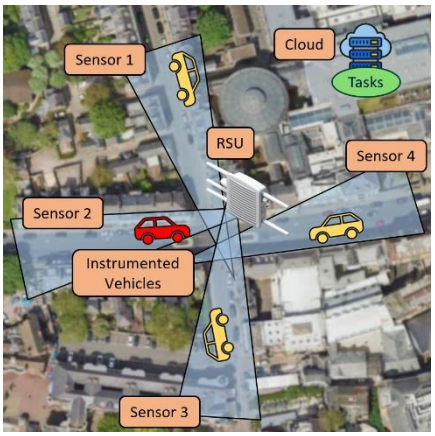


Figure 5.5: Starting scenario for hardware-accelerated intelligent RSU

One key challenge is that each sensor independently tracks objects and assigns unique IDs, which may result in multiple IDs for the same object across different sensors. The first task of this research is to associate these IDs to accurately match the same object, followed by fusing data from multiple sensors to update the object's trajectory, enhancing both precision and robustness.

Typically, sensor data is collected by the RSU and forwarded to a cloud server or MEC for processing. To reduce the round-trip latency, this research proposes offloading the ID association and sensor fusion directly onto an FPGA board attached to the RSU, enabling faster, low-latency processing.

Software Algorithm

The software algorithm developed for this starting scenario is based on the Simple Online and Realtime Tracking (SORT) algorithm. The Simple Online and Realtime Tracking (SORT) algorithm is a lightweight, efficient tracking algorithm designed for multi-object tracking in real-time applications. SORT uses Kalman Filters to predict the future states of tracked objects, such as their positions and velocities, based on previous observations. These predictions are then matched with newly detected objects using a Hungarian Algorithm for data association, which minimizes the difference between the predicted and detected object states. The simplicity of SORT allows it to operate in real-time while maintaining a high level of accuracy in object tracking, making it ideal for tasks involving multiple sensors and dynamic environments like those in ITS. The following figure presents the key steps of the algorithm.

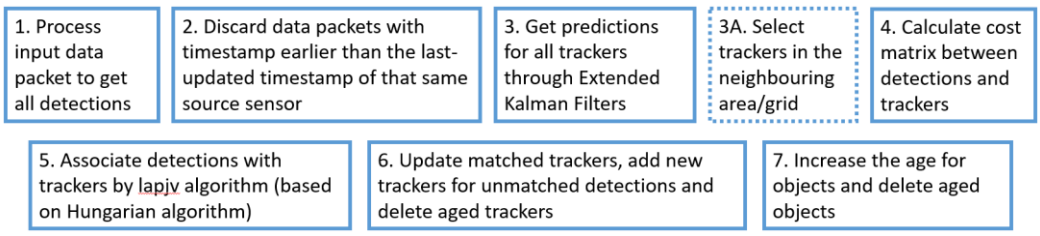


Figure 5.6: Software algorithm for hardware-accelerated intelligent RSU

Evaluation

A major challenge in evaluating our algorithm was the lack of suitable public datasets with detailed ground truth data. To address this, as Figure 5.7 displays, we generated a custom dataset by combining trajectory data from the NGSIM and Waymo Motion datasets with virtual sensor models. These sensors, each with a defined detection range, simulated real-world sensors like radars. At each time step, we checked if objects, based on their trajectories, were within a sensor's detection range, and if so, the sensor assigned an ID and added noise to simulate real-world conditions.

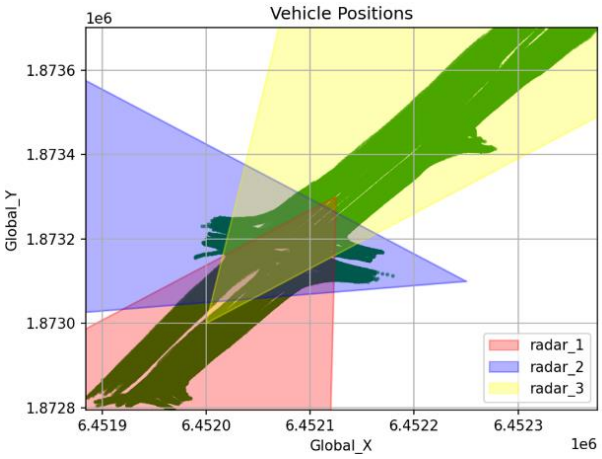


Figure 5.7: Virtual radar ranges generated on trajectory dataset

The algorithm then processed these data frames, performing ID association and sensor fusion. We compared the output against the original trajectory data to assess accuracy in object matching and trajectory reconstruction. This simulation-based method allowed for rigorous testing in a controlled environment.

The algorithm was evaluated for latency, accuracy, and memory usage. On a laptop, it achieved a latency of 4-5 ms/frame, increasing to 20 ms/frame on a Raspberry Pi, falling short of the 1 ms/frame target. However, accuracy was near to state-of-the-art: 99.75% on the NGSIM dataset and 99.6% on Waymo. Memory usage was minimal, mostly consumed by EKF predictions. As shown in Figure 5.8, the CPU profiling identified key areas for optimization, though the software alone was insufficient to achieve the low-latency requirements, reinforcing the need for hardware acceleration.

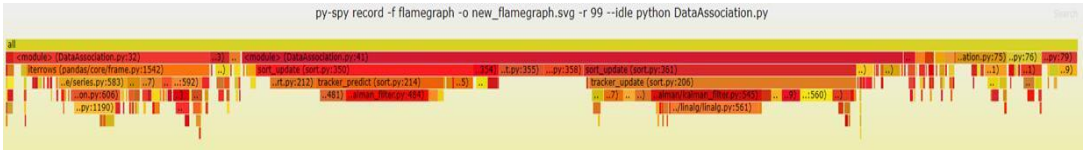


Figure 5.8: CPU profiling results for hardware-accelerated intelligent RSU

Hardware Design

To meet the latency goals, hardware acceleration is planned using the Kria KR260 FPGA. The Kria KR260 is an adaptive, edge computing platform that integrates an FPGA, designed for applications requiring high performance and low latency. FPGAs are ideal for hardware

acceleration because of their ability to execute customized parallel processing tasks, which significantly reduce computational delays compared to traditional CPUs. Common techniques such as pipelining, which allows multiple operations to be executed concurrently, and parallelism, which enables simultaneous processing of multiple data streams, are used to optimize sensor fusion tasks in this project. These features make FPGAs highly suitable for real-time, data-intensive applications like those in ITS.

The FPGA is designed to offload computation tasks from the RSU, taking over the heavy-lifting of sensor fusion and ID association. The main advantage of using FPGA is its ability to perform parallel processing through customized circuits, significantly reducing the time required to execute multiple sensor fusion operations simultaneously. The goal for the FPGA is to bring the processing time down to 1 ms/frame. Currently, the Verilog design for the sensor fusion algorithm is undergoing simulation and has yet to be synthesized and implemented on the FPGA board.

This early work was peer reviewed and presented at the MobiUK 2024 [Che24].

5.3 Street Intersection Edge Unit (SINED)

The use of LiDAR in the intersection RSUs creates a unique opportunity. These devices naturally combine visual and positional information that would otherwise need to be captured via a fusion of camera and radar sensors. As an added benefit, the data generated by LiDARs, also called "point clouds"**,** provide features enough to distinguish between different vehicle types while not allowing privacy-sensitive features–faces or license plates–to be distinguished. The point cloud, however, needs to be processed to make vehicle recognition and further applications possible. Moreover, this processing must occur within the RSU to avoid lengthy transmission times and in order to take prompt actions, as Use Case 2 requires. In this context, we designed and started implementing a solution to process the point cloud data on an edge device. We are designing a system to efficiently extract features from the LiDAR stream (e.g., bounding boxes of moving objects along with their velocity and reflexivity). The features can feed an object recognition machine learning model that determines the vehicle type of each moving object [Lin18]. Our emphasis is to deliver a low-code, declarative but generic language to express how to compute statistical, geometrical, and temporal features out of point cloud data and its corresponding runtime. Focusing on a language rather than pre-selected and hard-coded features allows the flexibility to cover many traffic model scenarios beyond Use Case 2 in the future.

The high-level architecture of our solution, named Street Intersection Edge Unit (SINED) is described in Figure 5.9 below. A hardware-accelerated LiDAR Processing Unit receives data from connected LiDAR devices. The processing unit is able to process raw point cloud data received from the LiDAR devices, and can be programmed through a low-code Point Cloud Manipulation Language (PCML). PCML is declarative and is compiled into low-level code. Various policies can be implemented in software in this way.

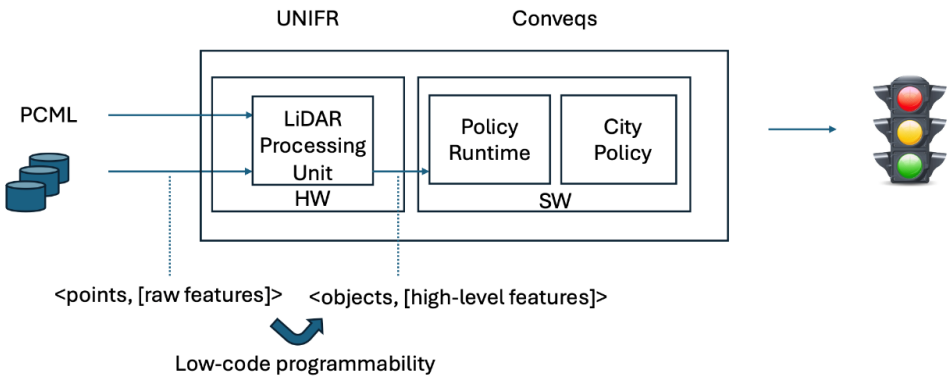


Figure 5.9: High-level architecture of LiDAR processing acceleration: our Point Cloud Manipulation Language (PCML) is a low-code, declarative language supporting hardware-accelerated processing of raw LiDAR data at the edge. Using PCML and our dedicated runtime, several processing (and policies in the context of UC2) can be easily implemented in software.

In practice, PCML expressions are executed by a dedicated runtime. This runtime must be adapted to the computational restrictions of the RSU. Unfortunately, the CPU capacity in that unit is not adequate to deliver the results on time, given the amount of processing necessary to extract useful information from the network stream produced by the LiDAR. Therefore, instead of generating a software binary from the features specifications, we provided a runtime implemented on an FPGA within the RSU unit that is capable of evaluating the latter. Because the number of features to extract is low, the FPGA area necessary for this runtime should be adequate for the power and physical dimensions of the RSU unit. We are currently experimenting with several low-power Xilinx FPGA units that are compatible with the restrictions.

5.3.1 PCML Design

After discussing with the use-case owner, we decided to focus on filtering, detection and classification of 3D objects (represented as parts of point clouds) in terms of basic primitives. Advanced analytics pipelines are typically handled by deep learning algorithms nowadays. In the context of LiDAR processing acceleration, we hence focus on computing the input representation for such pipelines efficiently and with limited resources (as the processing will take place on the edge of the network). In addition, we would like our language and runtime to be as generic as possible to support a broad range of ML models and use-cases. Figure 5.10 illustrates this point by showing various representations generated from raw LiDAR data using PCML that can be directly used by downstream ML algorithms.

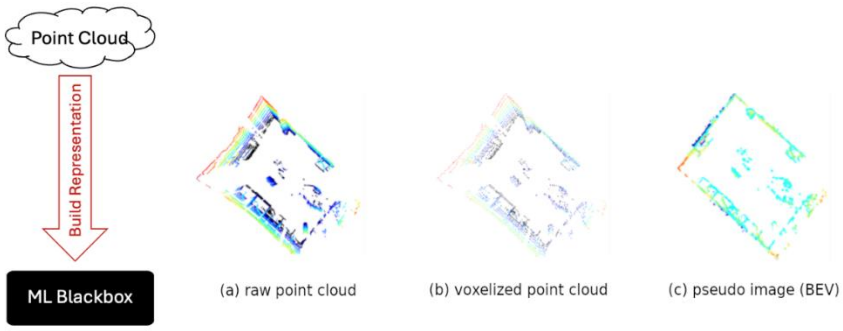


Figure 5.10: Building the input representation of ML pipelines by processing point clouds into voxels or by generating a bird's-eye view (BEV) from the raw data

We formalize our data and language as follows. We represent point clouds as multidimensional structures, where each frame is mapped onto a distinct matrix. Columns in those matrices hold simple features as output by the LiDAR, i.e., spherical coordinates (radius, azimuth, elevation) + reflection intensity. Figure 5.11(left) gives an illustration of our point cloud representation.

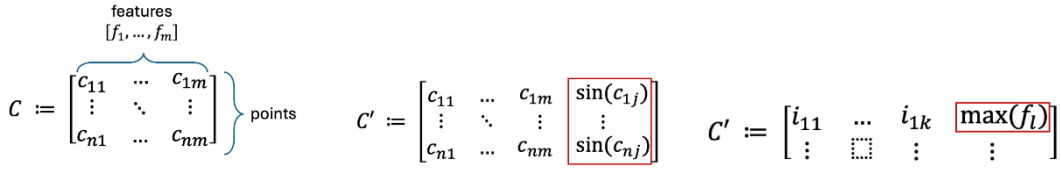


Figure 5.11: Formalism for point cloud data representation (left), and some of its processing: building a new representation by extracting new features (middle), and by grouping points from previous features (right)

Subsequently, two different kinds of processes can be applied on this base representation to add new features through PCML:

1. By deriving a new feature based on unary or binary functions that are supported by our runtime (see for example Figure 5.11, middle). The functions are applied on the values taken from another feature in the matrix. Both feature extraction and annotation can for example be supported through this feature.
2. By grouping points based on a *set* of features and aggregating values through a dedicated aggregation function (see for example Figure 5.11, right). This works similarly to aggregation (on multisets) in relational algebra / SQL. This changes the semantics of the point cloud values and allows to derive complex features or new point cloud representations (see below for a concrete example).

The operations currently supported by our runtime for unary, binary, and aggregation operations are listed in Figure 5.12.

Scope		Functions	
Unary	$\mathbb{R} \to \mathbb{R}$	sine	$\sin$
		cosine	$\cos$
		arcsine	$\arcsin$
		logarithm	$\log$
		square root	$\sqrt{\cdot}$
		floor	$\lfloor \cdot \rfloor$
Binary	$\mathbb{R}^2 \to \mathbb{R}$	addition	$+$
		subtraction	$-$
		multiplication	$\cdot$
		division	$\div$
		two-argument arctangent	$\arctan2(\cdot, \cdot)$

Aggregation	$\mathbb{R}^* \to \mathbb{R}$	counting	$\text{count}(*)$
		minimal value	$\min(\cdot)$
		maximal value	$\max(\cdot)$
		mean value	$\text{mean}(\cdot)$
		variance of an attribute	$\text{var}(\cdot)$

Figure 5.12: the unary, binary and aggregation functions currently supported by our hardware-accelerated PCML runtime.

5.3.2 Bird's Eye View (BEV) Derivation

The first use-case we implemented is the derivation of a Bird's Eye View (BEV), directly from the raw LiDAR data streamed at the edge and processed by our dedicated RSU. BEV derivation allows us to generate a 2D map-like representation from raw 3D LiDAR data, vastly simplifying subsequent processing and visualization of the data. More formally, we created the BEV by defining a regular grid structure on the x and y planes (forming "pillars" as Figure 5.13(left) depicts). We then build a representation for each pillar based on all included points, e.g., by extracting statistical information from all relevant values. Finally, we map the pillar representation to the pixels of a BEV pseudo-image.

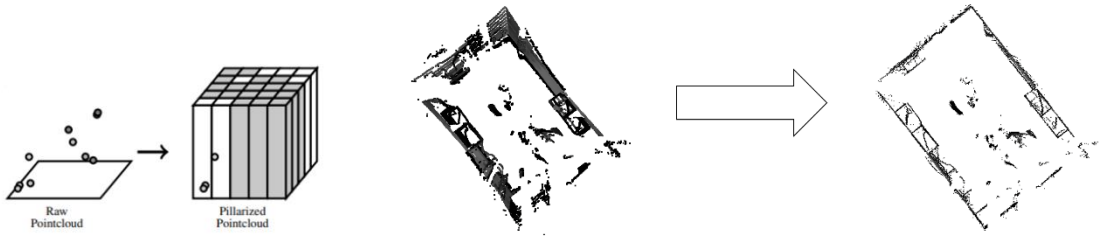


Figure 5.13: Bird's Eye Vies (BEV) derivation: we define a regular grid on the x-y planes ('pillars'), build a representation for each pillar based on the included points, and map pillar aggregates to the pixels of a resulting BEV representation.

Mor formally, the derivation unfolds as follows. We take as input a LiDAR point cloud frame $[x,y,z,\ I]$ with n points, each being described by Cartesian coordinates plus some reflection intensity. We compute 2D 'pixel index' for each point:

$$[x, y, z, I] \to [x, y, z, I, x_{2D}, y_{2D}] \quad \text{with} \quad x_{2D} = \left\lfloor \frac{x - x_{min}}{c_x} \right\rfloor, \qquad y_{2D} = \left\lfloor \frac{y - y_{min}}{c_y} \right\rfloor$$

where x_{min}, y_{min} are minimal values of x/y coordinate respectively (given by the sensor's field of view and used to obtain positive pixel coordinates) and c_x, c_y are the length & width of the pillars used to derive the BEV.

We then compute 'pixel indices' for each point; we group points by pixel indices and aggregate information across each group as follows:

$$[x, y, z, I, x_{2D}, y_{2D}] \rightarrow [x_{2D}, y_{2D}, \text{mean}(I), \max(z)]$$

.

This results in a BEV outcome with two channels: (mean reflection intensity, z coordinate of the highest point in the pillar), as depicted in Figure 5.13(right).

We plan to demonstrate some of these features at SIGMOD 2025.

6. CONCLUSIONS

This deliverable has highlighted significant progress in the development of intelligent swarm systems under Work Package 4 (WP4), specifically focusing on networking aspects. By addressing the requirements provided by WP2 in D2.2, WP4 has advanced the project's vision of secure, scalable, and high-performance swarm intelligence.

The implementation of automated discovery mechanisms and dynamic swarm formation techniques, supported by P4-programmable data planes, has demonstrated potential in enabling adaptive swarm behaviors. While preliminary performance analyses indicate encouraging results, further validation in diverse scenarios will be necessary to fully assess their robustness and scalability.

In network security, the exploration of in-network defenses, such as machine learning-driven threat detection and federated security frameworks, represents a step toward addressing the complexities of securing distributed systems. Methods like DDoS detection using neural networks and programmable switches offer promising results but require additional testing to ensure reliability and efficiency under real-world conditions.

Hardware-accelerated operations have shown potential to enhance system performance and scalability. Developments such as secure authentication protocols with SmartNICs and edge solutions like the Street Intersection Edge Unit (SINED) and intelligent road side unit (RSU) demonstrate progress toward integrating hardware innovations into swarm systems.

This deliverable highlights the progress made to achieve Release 1.0 of WP4 artifacts and solutions. The findings provide a basis for ongoing work in the subsequent Release 2.0, which will focus on addressing remaining open requirements, refining existing solutions, and expanding system capabilities to meet the project’s goals.

REFERENCES

[Ara18] Arash Habibi Lashkari, Andi Fitriah A. Kadir, Laya Taheri, and Ali A. Ghorbani. 2018. Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification. In 2018 International Carnahan Conference on Security Technology (ICCST).

[Bar23] L. Barsellotti, L. De Marinis, F. Cugini, F. Paolucci, "FTG-Net: Hierarchical Flow-To-Traffic Graph Neural Network for DDoS Attack Detection", HPSR Conference 2023.

[Bak24-1] Bakar, Rana Abu, et al. "FTG-Net-E: A hierarchical ensemble graph neural network for DDoS attack detection." Computer Networks 250 (2024): 110508.

[Bak24-2] Bakar, Rana Abu, et al. "DPUAUT: Secure Authentication Protocol with SmartNiC Integration for Trustworthy Communications in Intelligent Swarm Systems." IEEE Access (2024).

[Cai18] CAIDA UCSD Anonymized Internet Traces Dataset -[passive-2018], "http://www.caida.org/data/passive/passive dataset.xml," 2018.

[Che24] H. Chen, and N. Zilberman, "Hardware-Accelerated Intelligent Roadside Unit," Sixth UK Mobile, Wearable and Ubiquitous Systems Research Symposium, 2024.

[Din21] Ding, D., Savi, M., Pederzolli, F., Campanella, M., & Siracusa, D. (2021). In-network volumetric DDoS victim identification using programmable commodity switches. *IEEE Transactions on Network and Service Management*, *18*(2), 1191-1202.

[Din24] Ding, D, O Kesgin, and N Zilberman. 2024. "INDDoS+: Secure DDoS Detection Mechanism in Programmable Switches." In IEEE 25th International Conference on High Performance Switching and Routing (HPSR 2024).

[Ida21] Idan Burstein. 2021. Nvidia Data Center Processing Unit (DPU) Architecture. In 2021 IEEE Hot Chips 33 Symposium (HCS). 1–20. https://doi.org/10.1109/HCS52781.2021.9567066

[Lak21] S. Laki, R. Stoyanov, D. Kis, R. Soule, P. Voros, and N. Zilberman, ¨ "P4Pi: P4 on Raspberry Pi for networking education," SIGCOMM Comput. Commun. Rev., vol. 51, no. 3, p. 17–21, jul 2021.

[Ima18] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In ICISSP 18. 108–116.

[Jon20] Jong-Hyouk Lee and Kamal Singh. 2020. SwitchTree: In-Network Computing and Traffic Analyses with Random Forests. Neural Computing and Applications (2020), 1–12.]

[Lar21] Larry Peterson, Carmelo Cascone, Brian O'Connor, Thomas Vachuska, and Bruce Davie. 2021. Software-Defined Networks: A Systems Approach. Systems Approach, LLC. https://sdn.systemsapproach.org

[Lin18] Z. Lin, M. Hashimoto, K. Takigawa, and K. Takahashi, "Vehicle and Pedestrian Recognition Using Multilayer Lidar based on Support Vector Machine", 25th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Stuttgart, Germany, 2018.

[Mad22] S. Maddock, G. Cormode, T. Wang, C. Maple, and S. Jha, “Federated boosted decision trees with differential privacy,” in CCS ’22, 2022.

[Mar17] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). 2177–2184

[Nou15] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection systems (UNSW-NB15 Network Data Set). In 2015 military communications and information systems conference (MilCIS). IEEE, 1–6

[Psa17] The P4.org Architecture Working Group. 2017. P4_16 PSA Specification (v1.1). (2017). https://p4.org/p4-spec/docs/PSA-v1.1.0.html

[Qia20] Qiaofeng Qin, Konstantinos Poularakis, Kin K Leung, and Leandros Tassiulas. 2020. Line-speed and scalable intrusion detection at the network edge via federated learning. In IFIP Networking. IEEE, 352–360

[Roy21] Roy Friedman, Or Goaz, and Ori Rottenstreich. 2021. Clustreams: Data Plane Clustering. In Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR) (SOSR ’21). Association for Computing Machinery, New York, NY, USA, 101–107. https://doi.org/10.1145/3482898.3483356

[Tna21] Intel. 2021. P4_16 IntelR Tofino™ Native Architecture – Public Version. (2021). https://github.com/barefootnetworks/Open-Tofino/blob/master/PUBLIC_Tof ino-Native-Arch-Document.pdf

[Tus22] Tushar Swamy, Alexander Rucker, Muhammad Shahbaz, Ishan Gaur, and Kunle Olukotun. 2022. Taurus: A Data Plane Architecture for Per-Packet ML. In Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 1099–1114.

[Tus23] Tushar Swamy, Annus Zulfiqar, Luigi Nardi, Muhammad Shahbaz, and Kunle Olukotun. 2023. Homunculus: Auto-Generating Efficient Data-Plane ML Pipelines for Datacenter Networks. In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3. 329–342.

[Xil24] Xilinx. Accessed on 04/15/2023. Xilinx OpenNIC Shell. https://github.com/Xil inx/open-nic. (Accessed on 04/15/2023).

[Xsa23] Xilinx. Accessed on 04/16/2023. Vitis Networking P4 User Guide. https://docs.x ilinx.com/r/en-US/ug1308-vitis-p4-user-guide/Target-Architecture. (Accessed on 04/16/2023).

[Zan23-1] M. Zang, C. Zheng, L. Dittmann, and N. Zilberman, “Towards Continuous Threat Defense: In-Network Traffic Analysis for IoT Gateways,” IEEE Internet of Things Journal, 2023.

[Zan23-2] M. Zang, C. Zheng, T. Koziak, N. Zilberman, and L. Dittmann, "Federated Learning-Based In-Network Traffic Analysis on IoT Edge", IFIP Networking 2023 Sec4IoT Workshop, June 2023.

[Zan24] M. Zang, C. Zheng, T. Koziak, N. Zilberman, and L. Dittmann "Federated In-Network Machine Learning for Privacy-Preserving IoT Traffic Analysis", ACM Transactions on Internet Technology, 2024

[Zhe24] C. Zheng, Z. Xiong, T. T Bui, S. Kaupmees, R. Bensoussane, A. Bernabeu, S. Vargaftik, Y. Ben-Itzhak, and N. Zilberman, “IIsy: Hybrid In-Network Classification Using Programmable Switches,” IEEE Transactions on Networking, 2024.

[Zhe24-2] Zheng, Changgang, Mingyuan Zang, Xinpeng Hong, Liam Perreault, Riyad Bensoussane, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. "Planter: Rapid prototyping of in-network machine learning inference." *ACM SIGCOMM Computer Communication Review* 54, no. 1 (2024): 2-21.

[Zhe24-3] Changgang Zheng , Mingyuan Zang, Xinpeng Hong, Liam Perreault, Riyad Bensoussane, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. 2024. Planter’s GitHub Repository. https://github.com/In-Network-Machine-Learning/Planter. (2024).