International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Technical Report

(September 2025)

ITU-T Focus Group on AI Native for Telecommunication Networks

Proof-of-Concept activities

Summary

This Technical Report provides details on the Proof-of-Concept activities under ITU-T FG-AINN WG4. This report provides the technical summary of the activities done under PoC, and it covers the following:

- Requirements for the Proof of Concept, including Build-a-thon.
- Description of the Proof of Concept and results.
- Learnings from PoC development mentoring sessions, including Build-a-thon

NOTE - This document includes details of PoCs done under Build-a-thon 1.0, 2.0, and part of 3.0.

Keywords
AI agent, AI native networks, AI pipeline, Explainable AI, Intent, Knowledge Base, Proof of Concept, telemetry, use case.

Contributors: Preksha Shah
Indian Institute of Technology Bombay E-mail: 30005100@ee.iitb.ac.in
Mumbai, India
Liya Yuan
ZTE Corporation E-mail: yuan.liya@zte.com.cn
China

Table of contents

Technical Report

Proof of Concept Activities

1 **Scope**

This Technical Report summarizes the Proof of Concept (PoC) activities developed under the ITU-T FG-AINN initiative. It presents the technical details of AI-native network PoCs contributed in alignment with the AI-native defining characteristics—namely, architecture approaches for deep integration of AI, engagement of AI in all stages of the lifecycle, and AI itself as a core component. The report also includes outcomes from collaborative activities such as the Build-a-thon, and combined learnings from Build-a-thon mentoring sessions.
Each PoCs are mapped to (a) gaps, (b) use cases, and (c) architecture concepts.

2 **References**

[ITU-T Y.3102] ITU-T Recommendation Y.3102, "Framework of the IMT-2020 network," February 2018.
[ITU-T Y.3172] ITU-T Recommendation Y.3172, "Architectural framework for machine learning in future networks including IMT-2020".
[ITU-T Q.5001] ITU-T Recommendation Q.5001, "Framework for intelligence-aware functional architecture of future networks including IMT-2020," June 2020.

3 **Definitions**

None.

3.1 Terms defined elsewhere

None

3.2 Terms defined in this document

None

4 **Abbreviations**

AI: Artificial Intelligence
ANSP: AI-Native Service Platform
BTS: Base Transceiver Stations
CLI: Command Line Interface
CSI: Channel State Information
DQN: Deep-Q-Network
DWT: Discrete Wavelet Transform
E2E: End-to-end
ELA: Error Level Analysis
ETSI: European Telecommunications Standards Institute
FGAINN: Focus Group on AI Native for Telecommunication Networks
KB: Knowledge Base
KPI: Key Performance Indicator
LLM: Large Language Model
LSTM: Long Short-Term Memory
ML: Machine Learning
NLP: Natural Language Processing
NMS: Network Management Systems
ONOS: Open Network Operating System

RAG: Retrieval-Augmented Generation
RAN: Radio Access Network
RSSI: Received Signal Strength Indicator
SLA: Service Level Agreement
UE: User Equipment
URA: Uniform Rectangular Array
URLLC: Ultra-Reliable Low Latency Communication
QoS: Quality of Service
XAI: Explainable AI

5 Conventions

In this Technical Report, in alignment with the conventions of [Supplement 55 to ITU-T Y- series Recommendations] possible requirements which are derived from a given use case, are classified as follows:

The keywords "it is critical" indicate a possible requirement that would be necessary to be fulfilled (e.g., by an implementation) and enabled to provide the benefits of the use case. The keywords "it is expected" indicate a possible requirement which would be important but not absolutely necessary to be fulfilled (e.g., by an implementation). Thus, this possible requirement would not need to be enabled to provide complete benefits of the use case.

The keywords "it is of added value" indicate a possible requirement which would be optional to be fulfilled (e.g., by an implementation), without implying any sense of importance regarding its fulfilment. Thus, this possible requirement would not need to be enabled to provide complete benefits of the use case.

6 Introduction

The evolution of future communication networks is increasingly driven by the integration of machine learning (ML) and artificial intelligence (AI) capabilities across all layers of the network architecture. As outlined in ITU-T Recommendations [ITU-T Y.3102] and [ITU-T Y.3172] the development of intelligent, adaptive, and learning-enhanced network systems is central to achieving scalable, efficient, and context-aware service delivery in next-generation systems such as IMT-2020 and beyond.

The [ITU-T Y.3172] framework introduces a modular and functional architecture for incorporating ML in future networks, where ML pipelines support functions such as data collection, model training, inference, and feedback loops.

Building on these foundational principles, the current study explores the design and implementation of AI-native network functions and control loops that go beyond traditional automation. Rather than layering AI capabilities on top of existing management systems, AI-native networks embed intelligence directly into the fabric of network operations; interpreting user intents, coordinating multi-agent decision making, optimizing closed-loop control, and autonomously adjusting to dynamic network conditions. Shah P, et al. [b-Shah] carried out a gap analysis of existing frameworks in AI native networks.

The proof-of-concept (PoC) studies presented in this report focuses on demonstrating such AI-native capabilities across a diverse range of network scenarios. These includes distributed decision making, intent-based and feature-specific applications, emergency & resilience-centric AI inference systems, among others. Special emphasis is placed on leveraging standardized ML architecture components, ensuring that implementations are interoperable, reproducible, explainable, and aligned with the ITU-T Focus Group on AI Native for Telecommunication Networks (FGAINN) vision.

As AI-native networking continues to mature, these studies lay the groundwork for standard-compliant, reference-grade implementations that embody closed-loop autonomy, real-time learning, and AI-driven orchestration. According to [b-FG-AINN-O-08], AI-Native systems are defined by three delimiting characteristics: (i) architecture approaches for deep integration of AI, (ii) engagement of AI in all stages of the lifecycle, and (iii) AI itself as a core component. In the PoCs, these characteristics are realized by embedding AI into the system fabric through native architectural patterns (e.g., AI pipelines, service-based interfaces, and telemetry/data fabrics); ensuring AI participation end-to-end across design, deployment, orchestration, runtime operation, and evaluation of network components, functions, applications, and services; and elevating AI primitives to first-class resources within the control, data, and management planes. Through this mapping, the PoCs concretely demonstrate AI-native principles in practice.

The ITU-T FG-AINN organized three Build-a-thon challenges in 2025 to accelerate the development and validation of AI-native networking solutions. Build-a-thon 1.0, conducted in June 2025, saw participation from 23 registered teams, out of which 10 teams submitted final prototypes, and 6 winners were selected. To support the participants, six mentoring sessions were held from May to June 2025. Build-a-thon 2.0, held during June–July 2025, was supported by five dedicated mentoring sessions and concluded with the announcement of three winners. Build-a-thon 3.0 taking place through Aug – Oct 2025. A series of seven mentoring sessions were conducted and concludes with the announcement of winners.

The key outcomes from both Build-a-thon 1.0 and 2.0 include the demonstration of feasible and impactful AI-native solutions for telecom networks and societal applications, as well as validation of performance and functional requirements through executed test cases. The challenges also provided deep insights into integration challenges, design trade-offs, and the implementation of modular, interoperable architectures using AI pipelines and intent-based orchestration. Participants gained hands-on experience in prototyping and cross-disciplinary collaboration, while their contributions—ranging from use cases and architectural insights to PoC documentation and gap analysis—have enriched FG-AINN's technical work. These efforts have further strengthened the collaboration between academia, industry, and standardization bodies, supporting the evolution of future AI-native standards.

This Technical Report is organized as follows: Section 7 outlines the general requirements for the PoC, Section 8 provides the PoC description, Section 9 presents the PoC details, Section 10 discusses the combined learnings from Build-a-thon 1.0 and Build-a-thon 2.0 mentoring sessions, and Section 11 concludes with the bibliography.

7 General Requirements for the PoCs

This clause describes the requirements for the PoCs.

Requirement	Description
Gen-Build-a-thon-PoC-001	It is critical that PoC development activity, builds upon a key concept in FG AINN, especially aims to prove the concept practically with code, test setup and demo setup.
Gen-Build-a-thon-PoC-002	It is critical that PoC development activity create well- documented artifacts and opensource code.
Gen-Build-a-thon-PoC-003	It is critical that the maturity of the PoC is evaluated in test scenarios in the accompanying documentation.
Gen-Build-a-thon-PoC-004	It is critical that PoC demonstrates the feasibility (or lack of it) of specific architecture approaches.
Gen-Build-a-thon-PoC-005	It is critical that demonstration is focussed on a set of unique scenarios.
Gen-Build-a-thon-PoC-006	It is expected that participants submit other use case implementations in the same format as the reference examples.
Gen-Build-a-thon-PoC-007	It is expected that the PoC demonstrates the three delimiting characteristics of AI Native: Architecture approaches for deep integration of AI, engagement of AI in all stages of the lifecycle, and AI itself as a core component [b-FG-AINN-O-08].

8 PoC Description

The scope of PoCs collected during Build-a-thon 1.0 was oriented toward exploring and validating innovative concepts, emphasizing how AI-native technologies could be embedded into telecom networks while also delivering broader societal value. The scope of PoCs collected during Build-a-thon 2.0 extended this foundation by transforming preliminary ideas into mature realizations, featuring working prototypes, practical software artifacts, and live demonstrations that showcased their applicability in operational environments. The scope of PoCs conducted during Build-a-thon 3.0 is architecture-driven, focusing on designing and demonstrating scalable AI-native network frameworks and their integration across system components.

9 PoC Details

This section includes details of each PoC, including PoC title, delimiting characteristic, description, gaps addressed, test setup, code and data sets, simulated use cases, architectural concepts, demo and evaluations, PoC observation and discussions, conclusions, and future work.

PoC-001 to PoC-010 were contributed during Build-a-thon 1.0, which primarily served as an exploratory stage to study, validate, and refine a diverse set of forward-looking ideas on the integration of AI-native technologies. These early contributions emphasized conceptual soundness, feasibility analysis, and the identification of architectural touchpoints for embedding AI across the network lifecycle.

PoC-011 to PoC-014 were collected during Build-a-thon 2.0, where the focus shifted from ideation to realization. In this phase, initial concepts were elevated into robust and demonstrable prototypes, including code implementations, system integrations, and functional demonstrations. This progression showcased not only the technical viability of AI-native networking principles but also their potential for deployment in real-world scenarios, thereby bridging the gap between theory and practice.

Each PoC is tagged with its delimiting characteristic, like architecture approaches for deep integration of AI, engagement of AI in all stages of the lifecycle, and AI itself as a core component.

9.1.1 PoC-001 [b-FG-AINN-I-104_R1] AI-Native Proactive Network Slice Marketplace for Dynamic Service Ecosystems

9.1.2. Delimiting Characteristic

AI integrated services and AI pipelines itself as a core component – because this PoC integrates an AI agent as a core component which implements the service of slice selection.

9.1.3. Description

This PoC demonstrates an AI-native network slicing framework where Deep-Q-Network (DQN), Long Short-Term Memory (LSTM), and Large Language Model (LLM) agents enable intelligent, real-time decisions for vendor selection and slice optimization. It showcases integration with open5GS and supports structured data exchange for adaptive orchestration. The figure shows AI-Native Proactive Network Slice Marketplace.

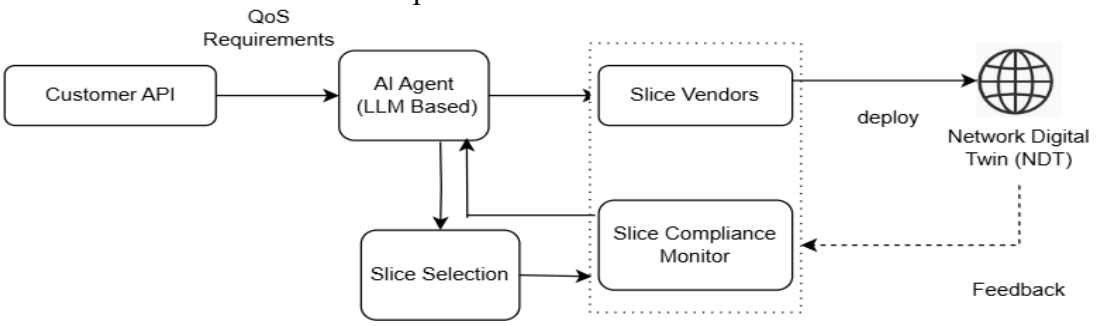

Fig. 9.1.1: AI-Native Proactive Network Slice Marketplace

9.1.4. Gaps Addressed

This tackles the absence of intelligent, data-driven orchestration in network slicing by introducing AI agents (e.g., DQN, LSTM, LLM) that automate decisions like vendor/resource selection and slice configuration based on real-time application requirements and traffic behavior something traditional static or rule-based systems fail to handle dynamically.

9.1.5. POCs Test Setup

This PoC is developed using Python-based simulation tools and Jupyter/Colab environments, optionally extendable to AWS EC2. The traffic simulator emulates variable load scenarios, including stadium surges and disaster responses, across >50 concurrent slice contexts. A supervised Random Forest classifier maps traffic characteristics (latency, bandwidth, device count) to slice types, while a DQN agent handles dynamic resource allocation. Vendor selection logic uses Open5GS-based emulators to return latency/jitter-based slice offers, and an LLM (e.g., Ollama) validates slice decisions against user intents. LSTM-based forecasting models anticipate traffic surges, feeding into a feedback loop that enables self-tuning of slice templates, vendor trust scoring, and retraining. Real-time metrics (latency, Service Level Agreement (SLA) violations, energy use) are logged and visualized using Matplotlib/Seaborn. The simulation environment includes a Key Performance Indicator (KPI) evaluation engine and a feedback module for continuous learning. Figure shows workflow of AI native proactive network slice marketplace.

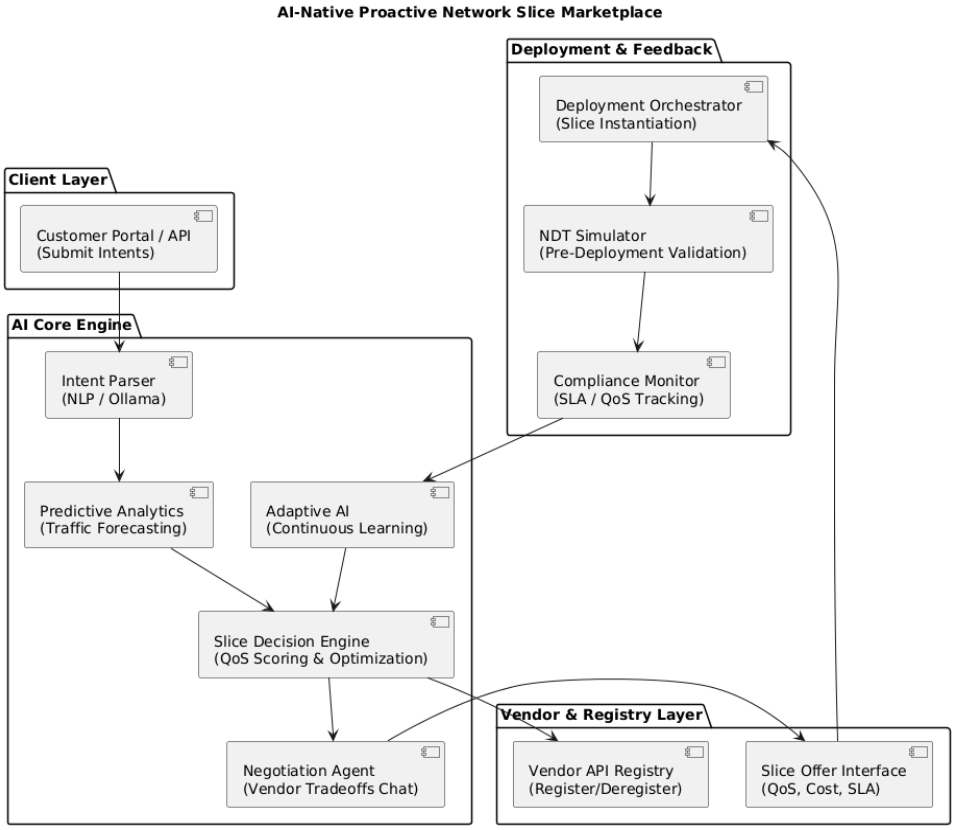

Fig. 9.1.2: Workflow of AI native proactive network slice marketplace

9.1.6. Code and Data Sets

A synthetic dataset generator produces multi-dimensional traffic traces (eMBB, URLLC, mMTC), augmented by public 5G datasets (e.g., https://www.osti.gov/biblio/1880776) to support realism. Code and data sets can be found here: https://github.com/Harsh067899/Build-A-Thon-SRM.

9.1.7. Simulated Use cases

AI system enables real-time classification of traffic into eMBB, URLLC, or mMTC slice types based on application demands. It dynamically selects the most suitable vendor slice template and adapts to varying traffic loads, including emergencies and public events. Key scenarios include smart city operations, emergency prioritization, and large-scale event management. This showcases proactive AI-Agent driven network slicing aligned with SLA, latency, and energy goals.

9.1.8. Architectural concepts

This PoC introduces an architecture where AI agents (DQN, LSTM, LLM) are modular components integrated alongside or within slice management functions (e.g., NSMF, NSSMF). These agents interact with orchestrators through structured interfaces (e.g., JSON-based APIs) and are capable of both inference and feedback handling, enabling real-time, adaptive slicing decisions. The architecture supports closed-loop control, telemetry-driven inputs, and application-aware outputs, marking a shift from static policy-driven orchestration to dynamic, AI-native management.

9.1.9. Demo and Evaluation

The demonstration covered five operational scenarios ranging from baseline benchmarking to emergency response and smart city integration. Real-time traffic simulations were used to assess the AI agent's ability to pre-allocate resources, classify slices dynamically, and handle SLA violations through reallocation. The success criteria focused on latency reduction, energy-aware resource usage, and improved adaptability under dynamic load conditions. Evaluation relied on real-time and post-simulation KPIs including SLA compliance, response time to traffic changes, and accuracy of slice classification and forecasting. Information exchange between application and model, including SLA constraints and traffic types, was logged in structured formats, enabling transparent KPI tracking. A component classifies high-level client intents (e.g., latency/bandwidth needs) into 3GPP-aligned slice

types (eMBB, URLLC, mMTC) using an ML model. This replaces static policy mapping with dynamic, learning-based slice interpretation — foundational to AI-native orchestration

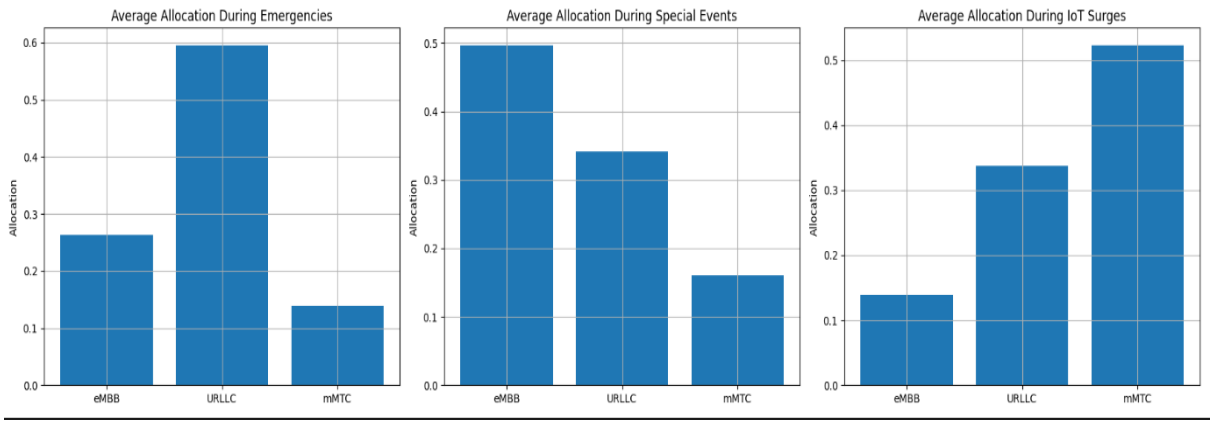

Fig. 9.1.3: Slice Allocation across use cases

A large language model (LLM) is used to evaluate and score vendor slice offerings based on soft constraints like SLA, provisioning time, security, and pricing. It enables semantic, multi-factor, and explainable slice decision-making beyond rigid rule-matching — a core AI-native advancement..

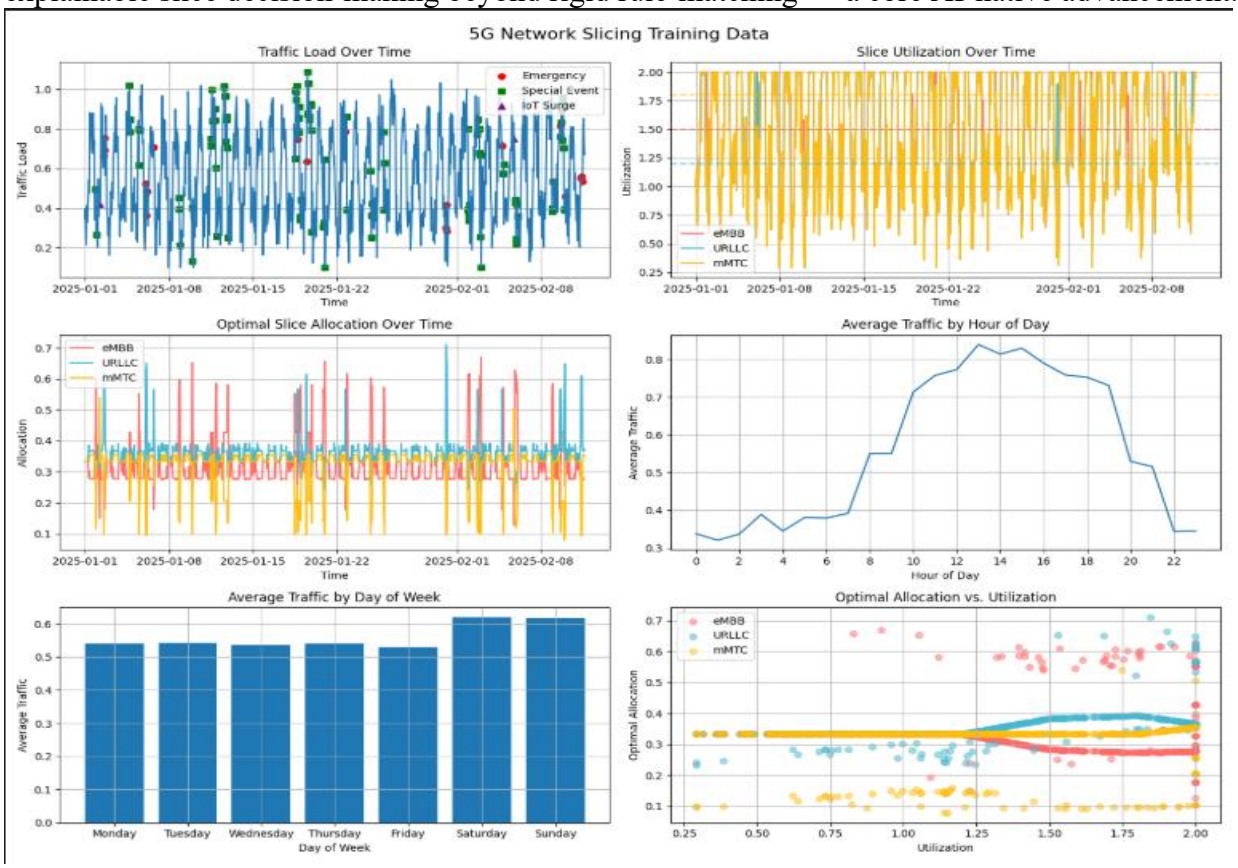

Fig. 9.1.4: Training data identification and visualization

An orchestrator component triggers vendor-side slice deployment automatically after scoring, without manual intervention. Fig. 9.1.4 demonstrates full intent-to-slice lifecycle automation — a step beyond 5G's NF-centric deployment model.

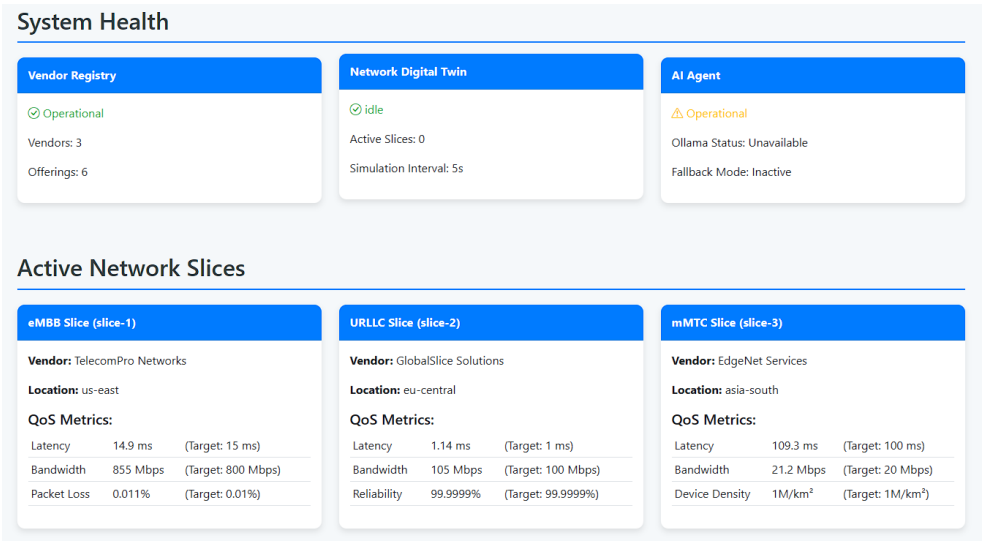

Fig. 9.1.5: System health and Active purposes

Simulated vendor agent deploys slice logic (Open5GS/UPF) and sends back a valid slice connection profile (AMF/SMF/DNN, QoS). This standardizes the slice provisioning return flow and prepares the UE connection interface — enabling multi-vendor interoperability.

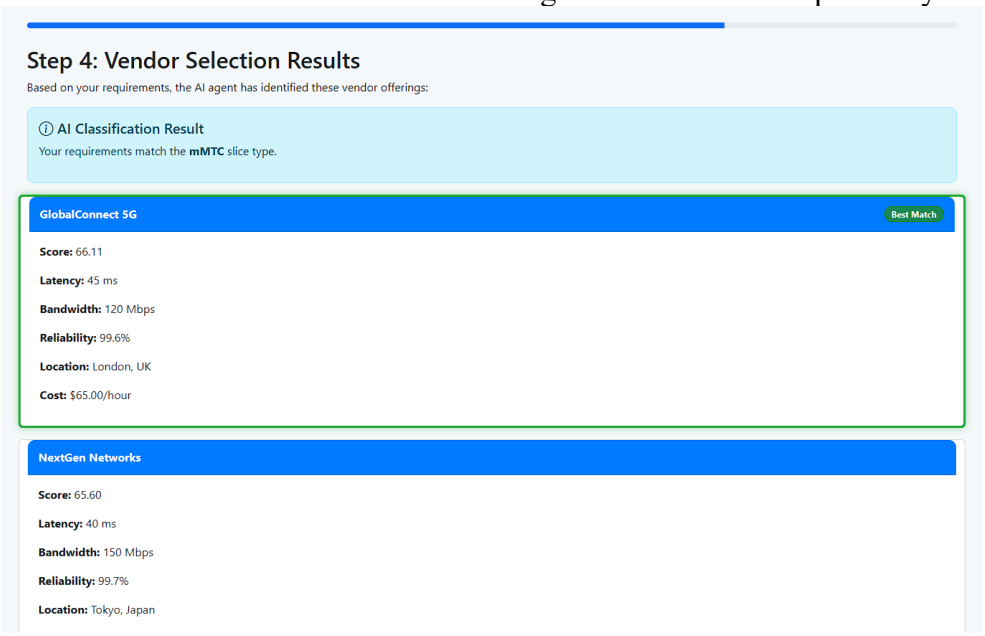

Fig. 9.1.6: Vendor selection and best score identification

Fig. 9.1.6 shows an API securely delivers the vendor-generated slice profile to the client or UE, enabling automatic session initiation. This replaces static NSSAI provisioning with dynamic, signed configuration delivery — fulfilling AI-native slice bootstrapping.

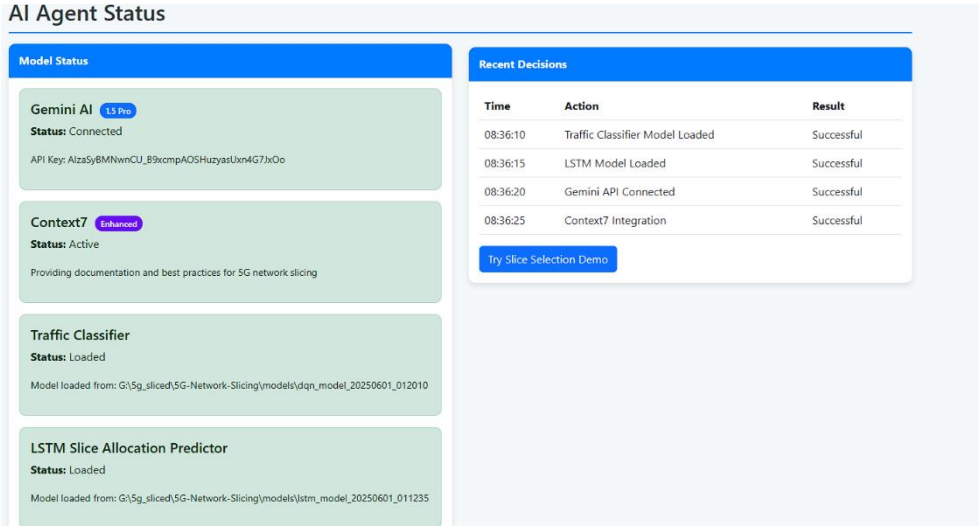

Fig. 9.1.7: AI Native agent status

Fig. 9.1.7 shows prometheus-based metrics simulate real-time latency, bandwidth, and availability monitoring for deployed slices. This validates compliance post-deployment and closes the loop between deployment and assurance — vital for trust-based networks.

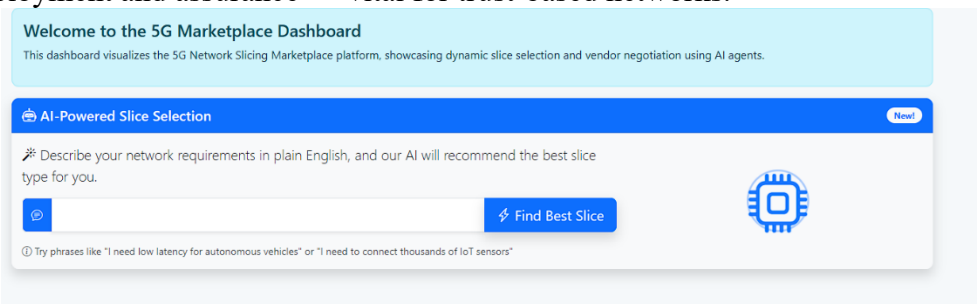

Fig. 9.1.8: Context based identification and slice allocation from natural language

Fig. 9.1.8 shows an AI agent adjusts each vendor's trust score based on real-time SLA violations or success metrics from the NDT. Implements continuous learning and self-regulation — a defining feature of AI-native networks per ITU/European Telecommunications Standards Institute (ETSI) standards.

Together, these results showcase a complete AI-Native Network Slice Marketplace architecture, where slice classification, vendor negotiation, deployment, monitoring, and adaptation are fully autonomous and learning-driven. This system advances beyond static 5G models and illustrates core building blocks for AI-native orchestration in 6G.

9.1.10. PoC Observation and discussions

This PoC successfully demonstrated integration of AI techniques like DQN (for vendor/resource selection), LSTM (for traffic prediction using open5GS), and LLM agents (for monitoring and reasoning via Ollama + Gemini). While the setup addressed key challenges around slice decision-making and validation using real traffic, one key suggestion was to focus more on the information exchange between the application and AI models, particularly in terms of data formats and schema consistency (e.g., JSON, time-series). It was also recommended to clearly identify the AI agent's position in the architecture, such as whether it resides within the NSSMF, NSMF, or as an external entity interacting via standard interfaces. Another critical insight was the need to study closed-loop monitoring and optimization, where feedback from systems (via LLM agents) is used to trigger real-time adaptations. Enhancing the AI inference endpoint (orchestrator) to support such structured input/output and feedback loops was also identified as a future enhancement.

9.1.11. Conclusion

PoC 1 effectively addresses the gap in dynamic slice management by introducing modular AI agents and defining their interaction with core functions. It sets the foundation for closed-loop, application-aware orchestration in future 6G networks.

9.1.12. Future Work

- Testing in a real time 5G environment

To validate the effectiveness and practical viability of the AI-Native Proactive Network Slice Marketplace, future research should focus on deployment and testing within an operational 5G network setting.

- Integrating Explainable AI

Incorporating Explainable AI (XAI) into the AI-Native Proactive Network Slice Marketplace is crucial to clarify why the AI models select certain network slice types in response to given traffic patterns or enterprise requirements.

9.2.1. PoC-002 [b-FG-AINN-I-136] Using AI to Reduce the 6G Standards Barrier for African Contributors

9.2.2. Delimiting Characteristic

Architecture approaches for deep integration of AI – because this PoC reuses specific architecture components from ITU-T Y.3061 and similar architectural components are used to deeply integrate AI (as an enabler for multi-agent systems).

9.2.3. Description

PoC-002 introduces an AI-native system to empower African stakeholders in the 6G standardisation process. It leverages a specialized LLM coordinated by four autonomous agents—responsible for gap detection, analysis, standards recommendation, and compliance evaluation—operating on a semantically indexed ITU dataset. This agent-driven pipeline automates and simplifies the process of identifying and addressing gaps in existing standards.

9.2.4. Gaps Addressed

This PoC addresses key gaps in automating standards intelligence through embedded, modular AI. It introduces application-aware processing of large standards corpora, autonomous agent lifecycle management, and low-latency inference with broad generalization. By operating on a semantically indexed dataset, it enables scalable, context-driven gap analysis. This architecture enhances inclusion and agility in the 6G standards ecosystem, especially for underrepresented regions.

9.2.5 POCs Test Setup

This PoC runs on a modular, multi-agent AI architecture built using frameworks like AutoGen or AnythingLLM, operating over a semantically indexed knowledge base created from ITU-T 6G standards. The base is stored in a Pinecone vector database, allowing fast contextual retrieval. Autonomous agents are assigned specialized tasks: gap detection (via Natural Language Processing (NLP) parsing), standards recommendation, compliance verification, and code generation. Documents are pre-processed with LLaMA 3.23B-based embeddings and chunked with metadata annotations (e.g., interface IDs, region tags). Each test phase document ingestion, gap detection, and regional filtering. These are tracked against functional success criteria such as correct parsing, identification of ambiguous sections, and relevance to African infrastructure needs. All gaps and proposed standards enhancements are logged with source excerpts, confidence scores, and region-specific tags, enabling full traceability for validation and audit. Figure shows AI-Native Multi-Agent Workflow for Gap Detection, Contribution Generation, and Compliance Validation in ITU-T 6G Standards.

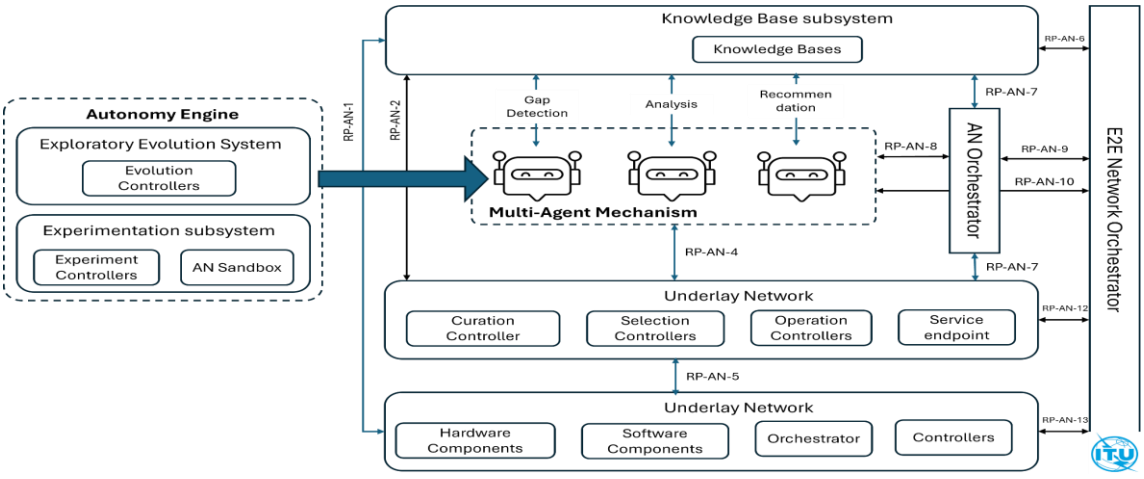

Fig. 9.2.1: AI-Native Multi-Agent Workflow for Gap Detection, Contribution Generation, and Compliance Validation in ITU-T 6G Standards

9.2.6. Code and Data Sets

A curated and semantically indexed corpus of ITU-T standards and related 6G technical documents. Code and data set available at https://github.com/AgabaEmbedded/Bridging-Standards-Gap.

9.2.7. Simulated Use cases

The system supports autonomous AI agents to analyze ITU-T 6G standards and identify specification gaps, particularly for developing regions. It semantically links abstract technical terms to local deployment challenges like power scarcity or rural infrastructure. Agents generate compliant proposal drafts and sample code aligned with standards evolution. This enables inclusive, AI-assisted contributions to global standardization efforts.

9.2.8. Architectural concepts

This PoC introduces a multi-agent AI architecture, where each autonomous agent performs a distinct function in the standards engagement lifecycle: gap detection, contextual analysis, recommendation, and compliance evaluation. The architecture is supported by a domain-specific LLM and a semantically indexed standards corpus, creating an end-to-end pipeline for AI-assisted, standards-aware collaboration.

9.2.9. Demo and Evaluation

The system was evaluated through three targeted tasks: semantic indexing of 6G standards, autonomous gap detection, and AI-assisted draft contributions. Each task was demonstrated using a curated document base, emphasizing the agent's ability to interpret, extract, and act upon regulatory and regional standardization needs. Controlled tests validated the completeness and accuracy of document parsing, relevance filtering for African network issues, and the precision of proposed enhancements. The evaluation focused on semantic search quality, the correctness of gap extraction, and alignment of generated proposals with known domain challenges.

9.2.10. PoC Observation and discussions

A key observation from PoC-002 is the need to focus more explicitly on the construction, structure, and utility of the knowledge base, which underpins the entire agent workflow. As a remote and the only foreign team, ensuring clarity and reproducibility of the knowledge base is especially critical for broader adoption and collaboration. It was suggested, notably by Marco, that the knowledge base itself could serve as a foundation for automated gap analysis across ITU standards, reinforcing its central role in enabling intelligent, AI-driven contributions to 6G standardisation. This highlights the importance of making the knowledge layer more transparent, accessible, and well-documented.

9.2.11. Conclusion

This PoC demonstrates an AI-native approach to democratizing 6G standards development through an agent-driven system powered by a semantically indexed ITU dataset. By automating gap detection, analysis, and compliance evaluation, it lowers the entry barrier for underrepresented regions, particularly in Africa. The modular design enables scalable, context-aware processing and sets the groundwork for broader integration of AI in standards workflows. This PoC marks a significant step toward inclusive, intelligent, and responsive 6G standardisation processes.

9.2.12. Future Work

- Scalability to Diverse Developing Regions: Future studies must prioritize the scalability and adaptability of 6G solutions to the highly diverse contexts within developing regions. Challenges include varied infrastructure maturity, significant language diversity, and fluctuating levels of technical capacity.
- AI Ethics: Critical attention is required for the ethical implications of deploying AI-driven 6G technologies. Future studies should address concerns related to data privacy, algorithmic bias, transparency, and accountability, particularly when AI models interact with sensitive user data and make decisions impacting connectivity and services.

9.3.1. PoC-003 [b-FG-AINN-I-128] Aionet Build-a-thon 2025: AIONETx

9.3.2. Delimiting Characteristic

AI itself as a core component – because the PoC is not just embedding AI into the architecture, but making AI the essential functional element that enables autonomous, on-device real-time control. Without the AI classifiers and predictors, the system would not work; it is fundamentally AI-driven rather than just architecturally integrated.

9.3.3. Description

PoC-003 is a lightweight, AI-native system that runs entirely on user devices to provide autonomous, real-time control of mobile data usage. It classifies traffic, predicts app behavior, and dynamically prioritizes bandwidth based on user-defined preferences and contextual factors like CPU load and packet flow without relying on centralized infrastructure.

9.3.4. Gaps Addressed

This PoC addresses key gaps in user-centric traffic control, edge-level autonomy, and infrastructure fairness. It fills the absence of application-aware, adaptive bandwidth management at the device level, especially in constrained or underserved environments. By decentralizing intelligence, it supports resilient operation under poor connectivity, and empowers users with informed control over data usage—contributing to responsible digital consumption.

9.3.5. POCs Test Setup

This PoC is deployed locally on a Linux environment simulating an edge device, operating without any cloud dependency. Scapy is used for real-time packet sniffing, psutil and subprocesses for CPU and process-level monitoring, and iptables/tc for active bandwidth shaping. A rule-based AI agent dynamically prioritizes flows based on CPU usage, packet type, port mapping, and user-defined bonuses. Unknown apps are logged, visualized on a Flask dashboard (with Chart.js and SweetAlert2), and handled through manual feedback stored in an encrypted SQLite database (AES-256). A heuristic priority engine references a live aionet_priorities.json config file, which is auto-reloaded every 30 seconds to reflect admin updates. The entire system—including data capture, classification, enforcement, and override is integrated and tested on a resource-constrained local stack, with simulation of network degradation using tc netem and real-time telemetry logs retained for policy validation and system tuning. Figure shows AIONET Workflow for Real-Time Traffic Monitoring, AI-Based Prioritization, and Traffic shaping.

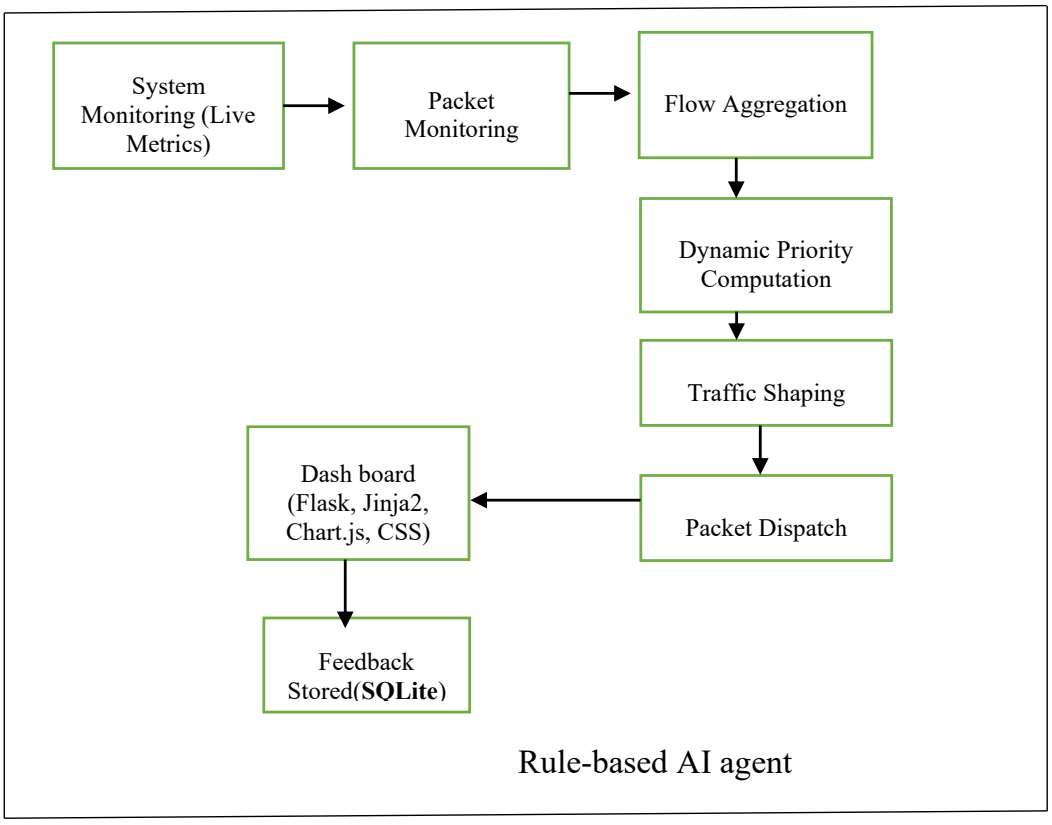

Fig. 9.3.1: AIONET Workflow for Real-Time Traffic Monitoring, AI-Based Prioritization, and Traffic shaping.

Collectively, the PoC setups in [1]-[3] implement a range of well-defined test environments, including edge-based execution, standards document analysis, and AI-driven network orchestration. Each setup emphasizes real-time telemetry collection, structured data logging, and system-level responsiveness, demonstrating the applicability of AI agents in enabling proactive, context-aware network management and standards support.

9.3.6. Code and Data Sets

Uses simulated real-time network traffic generated on the device for development and testing, capturing fields like timestamp, protocol, source/destination ports, and packet size. Code and data set available at https://github.com/Varsh-gr8/AIONET_ITU_build-a-thon2.0.git.

9.3.7. Simulated Use cases

In a constrained bandwidth scenario in this POC with concurrent Zoom, YouTube, and Google Drive traffic, AI agent in AIONET ensured high-priority access for Zoom while moderately throttling YouTube and delaying Drive sync. This behaviour, driven by real-time statistics and rule-based classification, validated the effectiveness of edge-intelligent traffic management.

9.3.8. Architectural concepts

It demonstrates key design directions for AI-native, user-empowered edge systems. It emphasizes on-device intelligence, real-time application-aware control, and autonomous bandwidth prioritization without relying on network-side orchestration. This direction aligns with the broader vision of decentralized, context-driven, and privacy-preserving AI-native networks, particularly for extending connectivity benefits to underserved users.

9.3.9. Demo and Evaluation

The PoC was demonstrated on a local edge device simulating real-world applications and network conditions. The AI agent monitored active traffic, assigned dynamic priorities, and enforced shaping policies in real time via a local dashboard. Demonstration steps included simulated bandwidth constraints, user overrides, and detection of unknown applications. The evaluation plan measured throughput capacity, latency overhead, dashboard responsiveness, and shaping accuracy. Success

criteria emphasized real-time adaptation, effectiveness of heuristic priority assignment, and offline operability, validating AIONET's suitability for privacy-sensitive, resource-constrained environments.

9.3.10. PoC Observation and discussions

Key observations from PoC-003 highlight the need to explicitly define the dataset structure, including packet fields (e.g., timestamp, protocol, source/destination ports, length), and demonstrate how it supports dynamic and unknown packet formats. The ability to classify, tag, and adapt to new traffic types is central to AIONET's AI-native claim, yet must be clarified with examples of policy formats, classification mechanisms, and how new rules are injected or updated at the edge. Additionally, deeper explanation of traffic shaping policy deployment across UE and UPF would strengthen the system's relevance to AI-native, decentralized network control. Addressing these would reinforce AIONET's potential to support real-time, evolving, user-driven networking at the edge.

9.3.11. Conclusion

It showcases a practical direction for AI-native networking by enabling autonomous, on-device traffic control and prioritization based on user context and preferences. It highlights how lightweight, decentralized intelligence can enhance digital inclusion and responsible consumption, especially in connectivity-constrained environments. While not proposing a full architecture, it offers a strong foundation for future exploration of user-driven AI in the non-radio domain.

9.3.12. Future Work

- Future Security Work:
    - Integrate Flask-Login for session-based dashboard protection.
    - Move to HTTPS with self-signed or CA-issued certificates.
    - Validate and restrict config updates with hash-based integrity checking.
    - Define a formal threat model for internal and external adversaries.
- Ethical AI Governance:

AIONET currently logs all traffic shapping and priority decisions, enabling retrospective audit of priority assignments. Although rule adaptation is automated based on user feedback, administrators are responsible for periodically reviewing logs via the dashboard to detect any unfair or biased prioritization patterns. This human-in-the-loop oversight ensures transparency and fairness in network traffic management, with plans to enhance automated bias detection in future updates.

- Future Study Directions

- Integration with Network Slicing

o Enable AIONET to interact with operator-side slice managers or SDN controllers for coordinated QoS.

- Knowledge Graph-Based Policies

o Use knowledge graphs to infer user intent and translate it into adaptive traffic control policies.

- Federated Edge Training

o Explore collaborative model training across devices using federated learning while maintaining privacy.

- Standardized Metadata Models

o Contribute to industry-wide standards for traffic/session metadata used in AI-native systems.

- Adaptive Threat Detection

o Integrate lightweight anomaly detection to identify misuse or hidden bandwidth drains.

- Mobile Optimization

o Optimize model and traffic shaping for mobile devices (Android/iOS) with varying app permissions and APIs.

- Explainable AI (XAI) Interface

- Design a dashboard that explains AI decisions using natural language or visual cues to improve transparency.

- Formal Intent-to-Policy Mapping
- Develop a standardized framework that translates user intent into enforceable network policies using AI + logic.

9.4.1. PoC-004 [b-FG-AINN-I-133] Build-a-thon 2025: NETSPEAK

9.4.2. Delimiting Characteristic

Engagement of AI in all stages of the lifecycle— because this PoC embeds AI not just for one function, but throughout the troubleshooting lifecycle; from interpreting complaints (input), to classifying intents, to generating configurations, and finally validating results in a closed loop. The AI is actively engaged end-to-end, enabling continuous improvement of the troubleshooting process.

9.4.3. Description

PoC-004 presents an AI-native troubleshooting assistant that interprets user complaints given in natural language, classifies them into network intents, and automatically generates and executes Command Line Interface (CLI) configurations. Using LLMs, Jinja2 templates, and tools like Netmiko and Ansible, the system supports multiple vendors (e.g., VyOS, Cisco) and performs dynamic validation via iperf and ping, creating a closed-loop, self-improving troubleshooting pipeline.

9.4.4. Gaps Addressed

This PoC addresses the lack of intelligent, intent-driven troubleshooting in network operations. Traditional systems rely on manual diagnosis or rigid, pre-defined templates. PoC 4 fills this gap by enabling context-aware automation, where user input in natural language leads to accurate classification, vendor-specific configuration, and validation without expert intervention.

9.4.5. POCs Test Setup

The PoC is deployed in an EVE-NG-based emulated testbed containing Cisco IOS, pfSense, and Juniper vSRX devices. A lightweight Flask-based orchestrator accepts natural language input from CLI or Web UI, which is classified, resolved, and mapped to Jinja2 templates before being executed via Netmiko. CLI outputs are validated, tested (e.g., using iperf and ping), and logged for continual improvement. The backend stack includes Python, PyTorch, Flask, Netmiko, and Ansible, enabling real-time inference, template rendering, and multi-vendor command execution. Test cases evaluate intent interpretation, CLI generation accuracy, rollback handling, and dynamic augmentation through documentation via RAG. Figure shows simulated network trouble shooting [4]

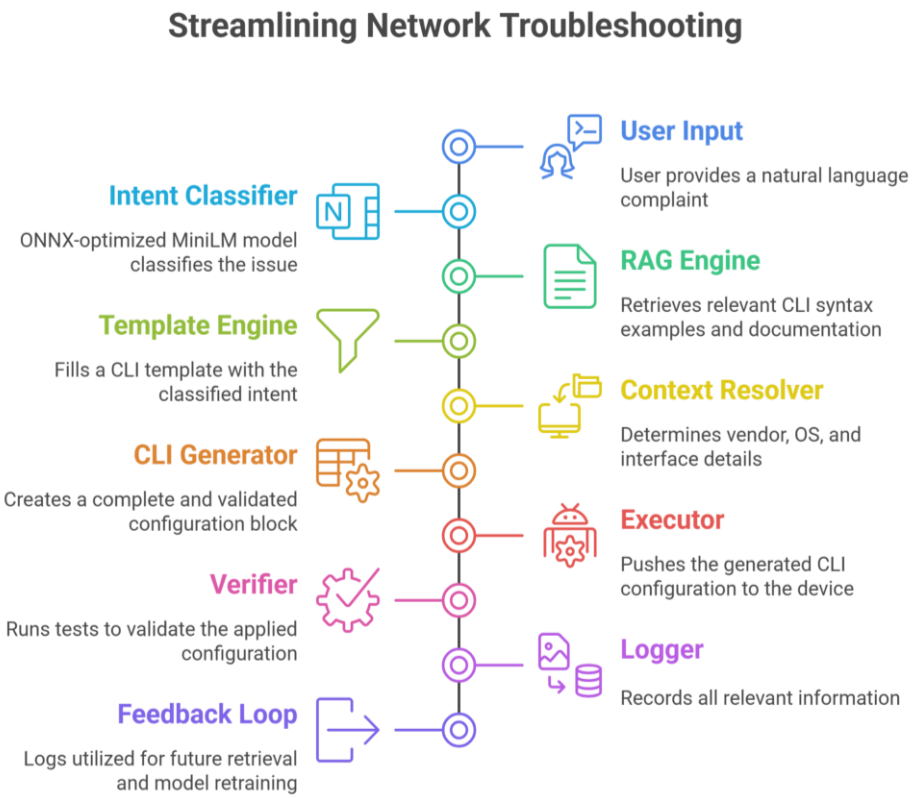

Fig. 9.4.1: Simulated network trouble shooting [4]

9.4.6. Code and Data Sets

The system uses a synthetic dataset of over 9,000 labeled complaints mapped to intent categories, CLI actions, and vendor contexts. Additionally, curated metadata files (e.g., parse.py, policy.json,

app_profiles.py) support structured inference and command generation. Code and data sets can be found via https://github.com/rajudhangar100/netspeak_ui.

9.4.7. Simulated Use cases

The chosen use case simulates a user request to increase link capacity during an IPL match in zone 1 from 7pm to 9pm. The user provides a natural language input, which is classified by the system into a structured intent including action (Increase Link Capacity), zone, time window, and service type (Video Streaming). The template engine loads a VyOS QoS policy, fills it with contextual parameters (e.g., eth1, video-stream, 800mbit), and the CLI generation module renders and validates the configuration block for execution—demonstrating real-time, intent-to-configuration automation for network optimization.

9.4.8. Architectural concepts

PoC-004 introduces an AI-native intent-to-action pipeline, comprising an NL parser, intent classifier, CLI template generator, and multi-vendor executor. It integrates real-time feedback and validation, enabling a self-improving, closed-loop system capable of handling diverse edge environments with minimal manual effort.

9.4.9. Demo and Evaluation

The PoC demonstrates AI-native, intent-driven automation through a user scenario requesting increased link capacity during an IPL match. A natural language input is classified using DistilBERT into a bandwidth_prioritisation intent. The system resolves the zone-to-interface mapping (e.g., zone 1 → GigabitEthernet0/1), selects the appropriate Jinja2 template, and generates the CLI block (bandwidth 1000000). The configuration is scheduled and deployed via Netmiko, with actions logged in MongoDB. Post-deployment ping and iperf tests verify success, and rollback is triggered if needed. The feedback loop stores outcomes for continuous learning, ensuring robustness, adaptability, and compliance with AI-native principles.

9.4.10. PoC Observation and discussions

PoC-004 demonstrates a promising AI-native approach to automated network troubleshooting, but several important areas for enhancement were identified. There is a need to systematically determine and isolate CLI changes, supported by a structured CLI command dataset and refined Jinja templates to improve accuracy across vendors like VyOS, Cisco, and Juniper. The relationship between network state, domain knowledge, and generated configurations must be made more explicit, including how intent is grounded in real-time network context. Additionally, future work should explore the development of a human intent dataset, clarify base models used for NL parsing, and address credential management and secure execution pipelines for automated CLI deployment. Finally, implementing a closed-loop feedback mechanism is essential for monitoring the impact of changes on KPIs.

9.4.11. Conclusion

This PoC demonstrates how AI-native techniques can automate complex, vendor-aware network troubleshooting tasks by bridging natural language input with executable configurations. It reduces dependence on human expertise, accelerates resolution, and lays the groundwork for intent-driven, adaptive network operations at the edge.

9.4.12. Future Work

- Model Adaptability: Ensure the AI model generalizes across diverse network environments and vendor-specific configurations.
- CLI Generation Accuracy: Prevent misconfigurations by refining dynamic template generation and validation methods.
- Reinforcement Learning Integration: Implement real-time feedback loops to optimize network adjustments continuously.
- Validation & Rollback Mechanisms: Develop robust safeguards to ensure deployed configurations do not destabilize the network.
- Multi-Cloud & Hybrid Network Support: Expand compatibility with cloud-based architectures and hybrid environments.

- Security & Privacy Considerations: Address vulnerabilities, adversarial attacks, and ethical concerns in AI-driven automation.
- Edge Execution Optimization: Enhance lightweight inference models for real-time execution on routers and microcontrollers.
- Scalability & Performance Improvements: Focus on reducing latency, improving classification accuracy, and automating network responses efficiently.

9.5.1. PoC-005 [b-FG-AINN-I-130] Build-a-thon 2025: Explainable AI-Native Intent-Based Self-Healing Network Orchestrator for Rural and Edge Telecom Infrastructure

9.5.2. Delimiting Characteristic

Engagement of AI in all stages of the lifecycle because the PoC applies AI across the entire fault management lifecycle: from intent interpretation, to anomaly detection, to explainability, to automated healing. Each stage depends on AI modules working in sequence within a coordinated pipeline, making lifecycle-wide AI engagement the defining characteristic.

9.5.3. Description

This PoC addresses the rural connectivity crisis by designing an AI-native fault management system for edge and resource-constrained telecom environments. It integrates intent-based operator input, LSTM-based anomaly detection, explainable outputs, and rule-based healing agents coordinated via a lightweight, modular pipeline. The system autonomously interprets faults, explains causes, and triggers corrective actions with minimal operator intervention, aiming to drastically reduce recovery times in rural infrastructure.

9.5.4. Gaps Addressed

This PoC addresses critical gaps in resilient, autonomous fault recovery for rural telecom networks. Traditional Network Management Systems (NMS) systems are reactive, centralized, and lack adaptability to edge environments. The proposed system overcomes these limitations by embedding AI-native capabilities directly at the edge, enabling proactive, explainable, and context-aware fault response in environments where manual intervention is slow or unavailable.

9.5.5. POCs Test Setup

The PoC is evaluated on an emulated 50-node rural telecom network, using synthetic datasets that reflect realistic outage and traffic patterns. Model training is conducted locally using Python (Scikit-learn/Keras), with LSTM-based anomaly detection and SHAP-based explainability. The multi-agent healing architecture simulates MCP-style coordination via Python FSM agents, while network flows are emulated in NS-3. Inference fallback is supported through a local RAG-style knowledge query layer, and additional orchestration and 5G core integration are achieved via xOpera and Open6GCore. The complete pipeline is planned for public release via the ITU AI/ML 5G Challenge GitHub. The following figure shows the pipeline of control in the test setup.

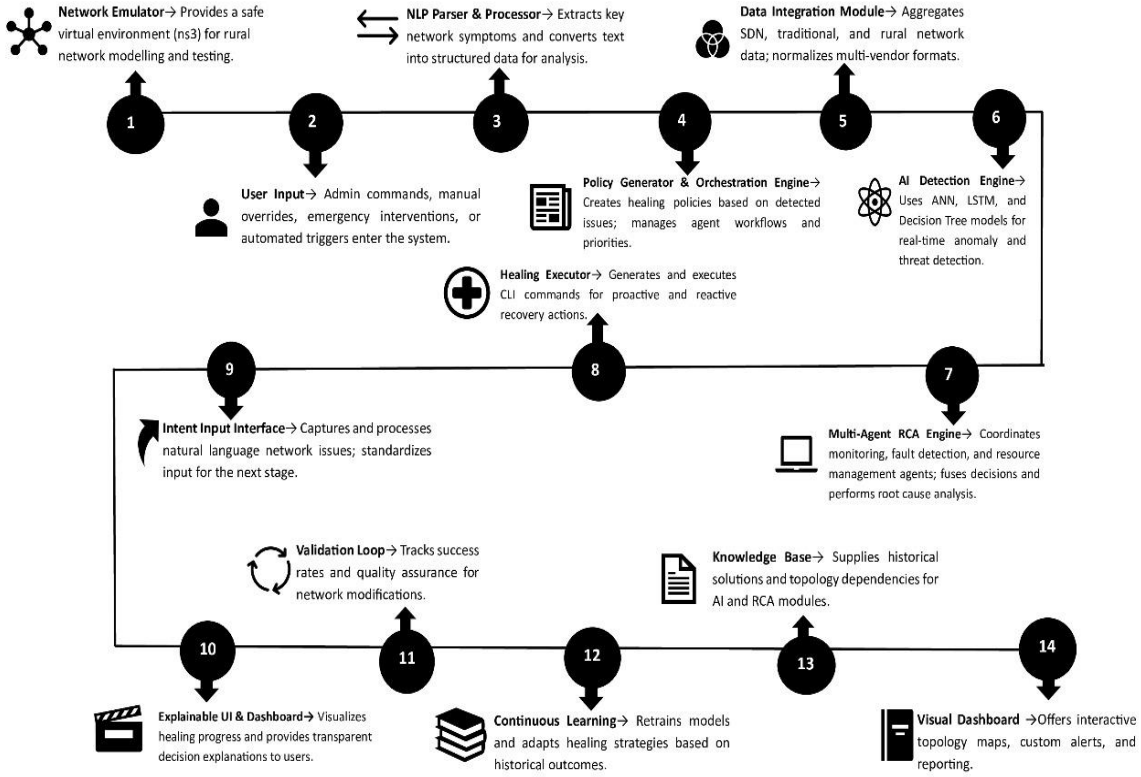

Fig. 9.5.1. The pipeline of control in the test setup.

9.5.6. Code and Data Sets

Synthetic time-series datasets simulating rural telecom metrics (throughput, latency, power quality, link status, etc.) are generated using NS-3 and Python. These include annotated anomaly traces for LSTM training (normal vs. degraded), curated edge-case failures (fiber cuts, voltage instability), and semantic intent examples for translation. Code and datasets can be found here: https://github.com/Rishi8520/rural_ai_selfhealing_net.

9.5.7. Simulated Use cases

This PoC validates AI-native fault detection and response through realistic edge network scenarios. In the fiber cut simulation, a fault is injected between critical nodes (e.g., CORE-0 and CORE-1). The LSTM-based anomaly detector identifies early degradation, explains the root cause using SHAP, and initiates automated self-healing via coordinated agents. In the power fluctuation scenario, voltage anomalies at nodes (e.g., DIST-2) are detected and proactively addressed using policy-based actions, with and without knowledge base support to compare recovery effectiveness. A third case tests intent translation, where an operator-defined policy (e.g., maintaining link uptime) is interpreted and propagated across agents. The presence of the MCP-style coordination layer significantly improves system responsiveness, demonstrating the impact of distributed intelligence and orchestration efficiency in rural networks.

Architectural concepts The PoC introduces a modular AI-native orchestration pipeline composed of:

- An intent interpretation layer that parses operator instructions.
- An LSTM-based anomaly detector for predictive fault detection and classification.
- A multi-agent coordination framework inspired by MCP (Multi-agent Control Plane) principles to trigger rule-based healing.
- A lightweight inference engine with knowledge-based fallback at the edge, enabling autonomy even under degraded connectivity.

The design emphasizes distributed intelligence, explainability, and fault resilience tailored to rural deployments

9.5.9. Demo and Evaluation

This PoC is is developed and tested on a 50-node simulated network using NS-3 to emulate rural broadband conditions and failure events such as fiber cuts and power fluctuations. The system operates through a sequence of coordinated autonomous agents, each handling a specific function in the fault management pipeline. The architecture includes a Monitor Agent for collecting real-time metrics, a Calculation Agent powered by LSTM models for detecting anomalies, a Healing Agent that uses Retrieval-Augmented Generation (RAG) and Gemini LLM to generate healing plans, and an MCP Agent that propagates decisions across agents. Actions are then executed by the Orchestration Agent using TOSCA templates and tools like xOpera, with communication managed through ZeroMQ sockets under a central orchestrator. This setup enables the system to autonomously process operator intents, detect issues using deep learning, provide SHAP-based explainability, and coordinate corrective actions across agents. Evaluation metrics include detection latency, recovery time, explanation fidelity, and intent propagation effectiveness, both with and without knowledge base or coordination mechanisms. The modular, distributed agent design allows flexible testing of AI-native orchestration capabilities under constrained edge conditions.

9.5.10. PoC Observation and discussions

This PoC presents an ambitious architecture, but observations suggest the need to narrow scope—particularly for the Build-a-thon 1.0 test case—to ensure implementability and clarity. The role of AI techniques (e.g., LLMs for healing and anomaly reasoning) and MCP/A2A coordination mechanisms requires further articulation, especially in terms of data flow and interaction timing. A clear mapping from NS-3 simulation (e.g., JSON outputs, node/link/path counts) to Monitoring Agent → MCP → Calculation Agent → A2A → Healing Agent is needed to assess real-time adaptability. In particular, the curation and schema design for RAG, and its use in Gemini-powered healing prompts, should be detailed alongside KB structure and YAML-based configuration plans. Focusing on a sample symptom and tracing its lifecycle—from anomaly detection to adaptive healing and reconfiguration—can illustrate the agent coordination and AI-native feedback loop. To strengthen the implementation pathway, it is recommended to prioritize fault isolation, explicitly document data/model requirements, and provide a layer-wise architecture diagram showing the before-and-after system state transitions.

9.5.11. Conclusion

This PoC demonstrates how AI-native orchestration can transform fault management in rural and edge networks. By combining anomaly detection, intent parsing, and agent-based healing in a modular, explainable framework, the system enables faster recovery, reduces dependence on central systems, and improves trust in autonomous operations. It lays the groundwork for scalable, fault-tolerant rural infrastructure aligned with the goals of digital inclusion and 6G resilience.

9.5.12. Future Work

While our current proof-of-concept demonstrates intent-driven anomaly detection and autonomous self-healing through explainable AI and lightweight agents, several open challenges remain for further research and deployment at scale:

1. Scalability of Agents in Heterogeneous Networks: Our current rule-based multi-agent system is effective in small to mid-sized topologies. Future work must explore how coordination and communication between agents scale in large, diverse rural deployments involving different vendor equipment and protocols.
2. Real-Time Constraints and Low-Latency Learning: Incorporating more complex ML models (e.g., transformer-based intent interpreters or reinforcement learning agents) may improve flexibility but introduces latency. Research is needed to balance explainability, speed, and predictive performance in real-time networks.
3. Interfacing with Real Network Controllers ((Software Defined Networking (SDN)/Network Function Virtualization (NFV)): Our orchestrator currently simulates healing actions. Integration with real-world network control systems (e.g., OpenDaylight, Open Network Operating System (ONOS)) will require protocol adaptation, security compliance, and stateful communication models.

4. Edge Model Optimization and Federated Learning: In future iterations, we aim to deploy distributed learning systems using federated or incremental learning to adapt models across geographically dispersed nodes while preserving data privacy and connectivity independence.
5. Intent Disambiguation and Natural Language Generalization: Currently, our intent parser is rule-based. A future goal is to integrate a fine-tuned language model for more flexible and user-friendly intent interpretation, especially for multilingual operators or ambiguous commands.
6. Resilience to Adversarial Events and False Positives: The system must be stress-tested under adversarial scenarios (e.g., spoofed telemetry, conflicting intents) to ensure robustness and prevent cascading failures due to incorrect healing triggers.

These open problems provide an opportunity for iterative enhancement of our framework toward a production-ready, standard-compliant AI-Native orchestration system suitable for rural and edge telecom networks.

9.6.1. PoC-006 [b-FG-AINN-I-135] Build-a-thon 2025: 5G Adaptive Signal Solutions for Rapid Transit and Aerial Operations

9.6.2. Delimiting Characteristic

AI itself as a core component because the adaptive beam steering fully depends on AI to learn mobility patterns and dynamically adjust antenna beams, making AI the essential element of the system's operation.

9.6.3. Description

This PoC proposes an AI-based adaptive beam steering mechanism designed for high-mobility scenarios such as rapid transit systems, fleet management, and drone connectivity. The system dynamically adjusts antenna beams by learning the motion of mobile stations using Received Signal Strength Indicator (RSSI) derived from Channel state Information (CSI), ensuring that user equipment (UE) remains within the optimal beam coverage (half-power beamwidth). The goal is to enhance link reliability, minimize signal loss, and sustain continuous 5G connectivity in fast-moving environments.

9.6.4. Gaps Addressed

Traditional beam steering methods are either slow to adapt or rely on pre-programmed patterns, which are insufficient for highly dynamic use cases like fast-moving vehicles or drones. This PoC addresses the gap by introducing real-time learning-based steering, enabling context-aware, low-latency beam tracking. It tackles the limitations of static or delayed beamforming and aligns with the needs of AI-based, mobility-optimized networks.

9.6.5. POCs Test Setup

The PoC is implemented using Sionna's AI-native Radio Access Network (RAN) simulation environment for PHY-layer modeling, with a Uniform Rectangular Array (URA) antenna and custom beamforming modules. An evolutionary algorithm explores diverse beam configurations across user positions and channel conditions, logging RSSI, CSI, and spatial parameters to generate a training dataset. This dataset is used to pre-train a reinforcement learning (RL) agent, which is later fine-tuned online using real-time feedback for adaptive beam tracking. The RL agent interacts with the RAN simulator through RESTful APIs, supporting near-real-time decision exchange. The API design reflects O-RAN architecture principles, with E2-like interfaces handling dynamic beam control and A1-like interfaces managing policy updates. Figure shows flowchart of method for adaptive steering for Rapid Transit and Aerial Operations.

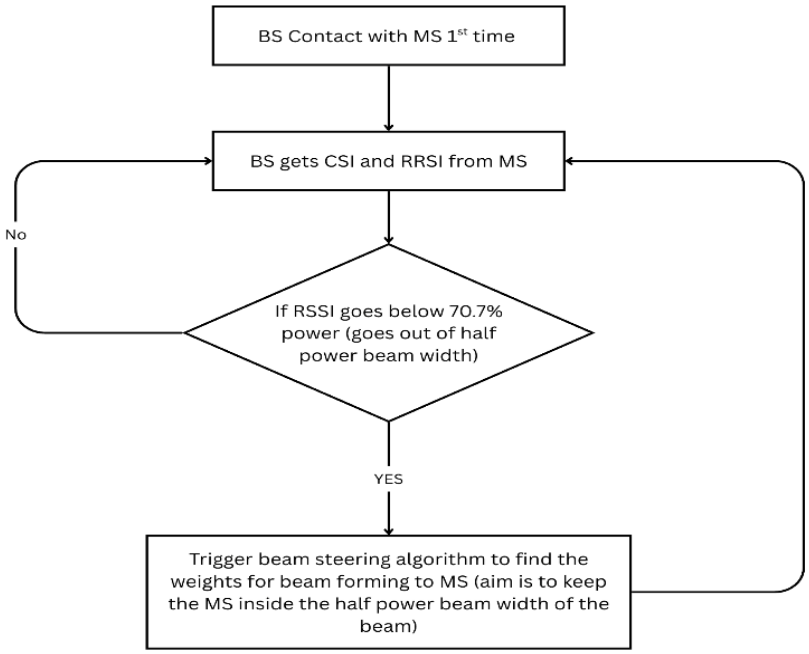

Fig. 9.6.1: Flowchart of method for adaptive steering for Rapid Transit and Aerial Operations

9.6.6. Code and Data Sets

A synthetic dataset was generated using the Sionna RAN simulator, capturing beam configurations, Channel State Information (CSI), Received Signal Strength Indicator (RSSI), and spatial parameters across varied UE positions and channel conditions. This dataset was used to pre-train the reinforcement learning agent, enabling it to learn an initial beamforming policy prior to online fine-tuning. The data generation was driven by evolutionary algorithms exploring the beam search space under mobility conditions. Code and data sets can be found here: https://github.com/syed-azim-git/Buildathon2.0-SSNECE.

9.6.7. Simulated Use cases

The simulation models a high-mobility wireless environment, such as urban transit or drone-based communication, where a base station equipped with a Uniform Rectangular Array (URA) antenna tracks and serves mobile UEs in motion. The UEs follow configurable trajectories, and their positions—as well as channel conditions—change over time. The simulator generates real-time CSI and RSSI measurements based on user movement, which are fed to the AI layer to dynamically adjust beamforming weights. This setup allows testing the system's ability to maintain signal quality and connection continuity under fast-changing spatial and channel conditions.

9.6.8. Architectural concepts

The architecture integrates:

- Motion learning models for trajectory prediction.
- RSSI-CSI feedback loops for real-time environment sensing.
- Dynamic beam alignment logic executed at the antenna array controller or baseband unit.
- An AI-based control layer that adapts beam parameters without centralized intervention.

9.6.9. Demo and Evaluation

To evaluate the performance and robustness of the proposed dynamic beam steering algorithms, we employed Sionna, an open-source, TensorFlow-based simulator developed by NVIDIA. Sionna provides a highly flexible and accurate PHY-layer simulation framework, supporting digital twin environments and advanced antenna modeling. Its capability to simulate realistic urban propagation scenarios, combined with custom beamforming modules, made it well-suited for testing adaptive beam steering strategies under diverse mobility and channel dynamics.

9.6.10. PoC Observation and discussions

The PoC highlights a strong integration between AI-driven beamforming and PHY-layer simulation; however, there is a need to abstract the Sionna-specific components to generalize the approach for RAN optimization across simulators. Clarifying the timing and placement of the genetic algorithm (GA) for dataset creation and the online reinforcement learning (RL) agent for real-time adaptation is important to improve reproducibility and modularity. The RESTful API interfaces between the AI layer and the RAN simulator should be documented explicitly to ensure interoperability—enabling replacement of Sionna with other tools while preserving functionality. Future enhancements could focus on dataset creation workflows, YAML-based simulation configuration, and support for user-defined BS/UE locations to enable broader experimentation. Drawing from the DeepSense 6G Challenge, the team may extend the framework to other radio-layer learning problems. Finally, formalizing the simulator's configurable parameters and their representation in YAML would help standardize testing across use cases and environments.

9.6.11. Conclusion

The PoC demonstrates how AI-based beam steering can improve 5G performance in highly mobile scenarios by minimizing signal degradation and handover interruptions. By using real-time signal metrics and learning models, it offers a responsive and context-aware solution that enhances throughput, reliability, and service continuity in dynamic environments laying the groundwork for resilient, intelligent transportation and emergency communication systems.

9.6.12. Future Work

- Beam steering option for UE tends to vary the speed of UE on the go.
- Energy Aware Decision making for Beam Steering

9.7.1. PoC-007 [b-FG-AINN-I-132] Build-a-thon 2025: Invisible Guard – Passive Wi-Fi Sensing for Smart Border & Building Surveillance

9.7.2. Delimiting characteristics

AI itself as a core component because the surveillance system relies entirely on AI models to interpret CSI variations and recognize human motion, making AI the fundamental driver of functionality rather than just an auxiliary tool.

9.7.3. Description

Invisible Guard is an AI-native, privacy-preserving surveillance solution that leverages passive Wi-Fi Channel State Information (CSI) to detect human motion and anomalies without relying on cameras, radar, or wearable devices. Using standard 802.11n/ac routers and compatible NICs (e.g., Intel 5300, Nexmon), the system captures fine-grained multipath variations in CSI caused by environmental disturbances. These are processed by lightweight temporal deep learning models (e.g., GRU or 1D-CNN) to recognize motion signatures such as walking, running, loitering, or intrusions. Designed for continuous, sensorless operation, the system is suited for resource-constrained, privacy-sensitive zones like borders, critical infrastructure, and homes.

9.7.4. Gaps Addressed

Invisible Guard addresses a key network-layer gap by embedding AI-native perception into the physical (PHY) layer using passive Wi-Fi CSI. It enables privacy-preserving, sensorless intrusion detection at the network edge—where traditional systems rely on explicit sensors or centralized analysis. This fills a critical void in AI-native architectures by supporting real-time, context-aware anomaly detection directly through ambient wireless signals, enhancing observability and resilience in distributed environments.

9.7.5. POCs Test Setup

The demonstration setup for Invisible Guard is entirely offline, designed to simulate real-time motion detection using pre-recorded Wi-Fi Channel State Information (CSI) logs. Rather than relying on physical hardware or live data collection, the system uses .pcap files containing CSI traces representing four human activity classes—such as walking, running, loitering, and intrusion—captured under controlled conditions. These files serve as the input for a structured preprocessing pipeline. First, raw CSI amplitude and phase values are parsed from the .pcap format. To reduce noise and enhance signal clarity, Discrete Wavelet Transform (DWT) is applied. The resulting clean signals are then segmented into fixed-length temporal windows and normalized.

Depending on the model type, the pipeline either extracts statistical features like mean, standard deviation, and entropy (for classical models like Random Forest), or directly passes time-series windows to deep learning models such as Bi-LSTM or GRU, which are better suited for capturing sequential dependencies in the data. These models are pre-trained and saved to disk, and during the demonstration, they are loaded into a Python-based Jupyter Notebook environment. Using libraries like TensorFlow and Scikit-learn, inference is performed on each input segment. The output includes the predicted activity class and an associated confidence score, which is dynamically visualized in the notebook interface. This setup offers a reproducible and privacy-preserving alternative to hardware-based surveillance, effectively showcasing AI-native anomaly detection through ambient wireless signals in a simulated but realistic setting. Figure shows flowchart of workflow.

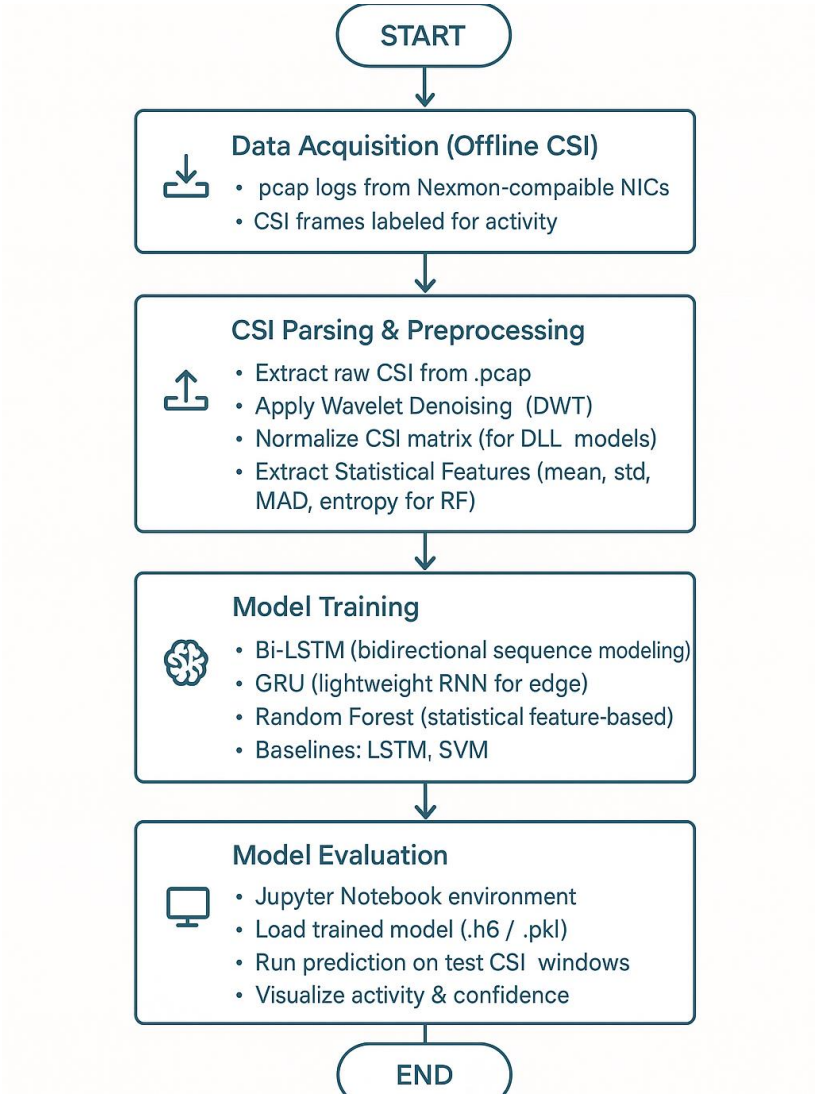

Fig. 9.7.1: Flowchart of workflow

9.7.6. Code and Data Sets

Uses an offline, simulated dataset derived from passive Wi-Fi Channel State Information (CSI) logs stored in .pcap format. The dataset includes four labeled human activity classes: Empty, Sitting, Standing, and Walking. Code and data sets can be found at https://github.com/himanshukhatri1511/ITU.

9.7.7. Simulated Use cases

The PoC simulates human activity detection (empty, sitting, standing, walking) using pre-recorded Wi-Fi CSI data. It mimics real-world intrusion scenarios in secure areas without using cameras or sensors, demonstrating AI-native, privacy-preserving surveillance via signal pattern changes.

9.7.8. Architectural concepts

Invisible Guard introduces an AI-native edge architecture where CSI data from Wi-Fi NICs is processed locally using lightweight deep learning models (1D-CNN/GRU) for real-time motion detection. It enables privacy-preserving, sensorless anomaly detection by embedding intelligence at the PHY layer, supporting distributed, context-aware surveillance without centralized sensors or cameras.

9.7.9. Demo and Evaluation

The demonstration and evaluation of Invisible Guard are conducted entirely within a Jupyter Notebook environment, showcasing a fully offline, hardware-free AI-native activity classification system. Pre-trained models—including Bi-LSTM, GRU, and Random Forest—are loaded from disk and applied to a test dataset derived from preprocessed Wi-Fi Channel State Information (CSI) logs

originally stored in .pcap format. These logs have been parsed and segmented into temporal windows, each labeled with one of four activity classes: Empty, Sitting, Standing, or Walking.
For each input window, the system performs inference and outputs a predicted activity class along with a confidence score, both of which are dynamically visualized in real-time through prediction traces and confusion matrices. This offline simulation allows complete reproducibility and systematic evaluation without the need for live CSI capture or physical testbeds. As you can see from the Result summary table below, the proposed models have achieved high classification accuracy, validating the effectiveness of offline CSI-based activity detection. The GRU model stands out with an accuracy of ~98.96%, offering the best balance between performance and computational efficiency. Bi-LSTM and LSTM also perform strongly, with accuracies of ~97.99% and ~97.85% respectively, demonstrating their capability to model complex temporal patterns. In contrast, traditional methods like Random Forest and SVM show significantly lower accuracy, reinforcing the advantage of deep learning approaches for this task.

Model	Accuracy (%)	Notes
GRU	~98.96%	Best trade-off between speed and accuracy
Bi-LSTM	~97.99%	Best for transitional and complex sequences
LSTM	~97.85%	Strong performance with engineered features
Random Forest	~72.85%	Baseline sequence model
SVM	~67.25%	High dependency on preprocessing

9.7.10. PoC Observation and discussions

This PoC presents a novel AI-native surveillance framework using passive Wi-Fi CSI for human activity detection. However, several critical aspects require refinement to align the PoC more closely with AI-native networking principles. First, the application layer model, human activity detection, must be explicitly mapped to the underlying RAN, indicating what information from the RAN (e.g., CSI, UE authentication data) is being utilized and how that supports inference services. Additionally, the PoC should clearly define its inference endpoints (e.g., alerts or system flags) and outline the network-level requirements or interactions necessary to support real-time detection and response.
The dataset used is offline and derived from .pcap CSI logs; this should be more explicitly stated along with the model types (Bi-LSTM, GRU) and their execution environment. The absence of information regarding hosting and training (e.g., cloud, edge, or on-device) leaves a gap in understanding the scalability and deployment feasibility. Moreover, while the project operates as a cooperative sensing scheme, its ISAC (Integrated Sensing and Communication) relevance could be further clarified by specifying how ambient UEs contribute to signal diversity or coverage. Finally, the need for a layered interface view clearly showing the app, AI, and network layers—remains unaddressed and would greatly aid in evaluating system integration and operational scope.

9.7.11. Conclusion

Introduces a CSI-powered HAR framework that demonstrates the viability of non-invasive, privacy-friendly, and hardware-free activity detection. Through the use of Bi-LSTM, GRU, and Random Forest, we achieve superior classification accuracy and computational flexibility across various model types.
By removing the need for live data capture and simulation, we ensure that our system remains portable, reproducible, and simple to deploy. The success of our models in static CSI environments confirms the potential for practical applications in smart homes, healthcare, and ambient IoT systems. Moreover, the active development of multi-user detection, embedded deployment, and transformer integration lays the groundwork for a new generation of real-time, scalable HAR systems rooted in signal intelligence.

9.7.12. Future Work

1. Multi-user HAR using multi-label classifiers, signal decomposition (ICA), and MIMO-based spatial differentiation.
2. Generalization to variable room layouts and device positions using transfer learning and domain adaptation.

3. GRU deployment on Raspberry Pi/ESP32, using quantized models with CSI from Nexmon firmware.
4. Transformer-based modeling to enhance long-range temporal pattern recognition via self-attention mechanisms.

9.8.1. PoC-008 [b-FG-AINN-I-131] An advanced emergency response platform

9.8.2. Delimiting Characteristic

Engagement of AI in all stages of the lifecycle, because AI is applied across the entire emergency response lifecycle—detecting anomalies, triggering alerts, optimizing ambulance routes, reconfiguring networks in real time, and restoring operations—showing continuous AI engagement from incident onset to resolution.

9.8.3. Description

This PoC introduces an AI-native emergency response system tailored for urban individuals, particularly those living alone. It orchestrates real-time health anomaly detection, emergency alerting, location tracking, ambulance dispatch, and continuous patient-health monitoring during transit. At the core, the system uses AI/ML algorithms for dynamic ambulance route optimization and real-time O-RAN network reconfiguration to guarantee end-to-end (E2E) Quality of Service (QoS) for emergency communication flows. Once the emergency is resolved, the system autonomously reverts network resources to normal operations.

9.8.4. Gaps Addressed

The PoC addresses the lack of real-time, intent-driven E2E QoS provisioning in emergency scenarios. Existing networks do not dynamically prioritize health-critical traffic or support AI-based decision-making for routing or RAN resource adaptation. The proposed system fills this gap through intelligent orchestration across layers, enabling ultra-reliable low-latency communication (URLLC) during healthcare emergencies.

9.8.5. POCs Test Setup

The test setup integrates O-RAN-based AI-native infrastructure with simulated healthcare and traffic environments. The OAIC platform is used to deploy a multi-container near-RT RIC running Dockerized xApps, each handling specific AI tasks. These xApps interact over E2AP/E2SM interfaces and leverage pretrained models exported via ONNX. The simulated radio access network uses ns-O-RAN, combining the ns-3 mmWave module with FlexRIC for dynamic control. UE mobility is emulated using SUMO with real-world maps from OpenStreetMap. The core network functionality is provided by Open5GS, completing the end-to-end network. Two AI pipelines—one for health emergency prediction (based on critical care datasets) and another for ambulance route optimization (trained on traffic flow data)—run within the xApps. All inference is performed locally, and emergency-related data is prioritized using a dedicated QoS management xApp.The figure given below is the sequence diagram that represents the workflow of AI-enabled emergency healthcare response using O-RAN architecture, xApps, MEC, and model repositories.

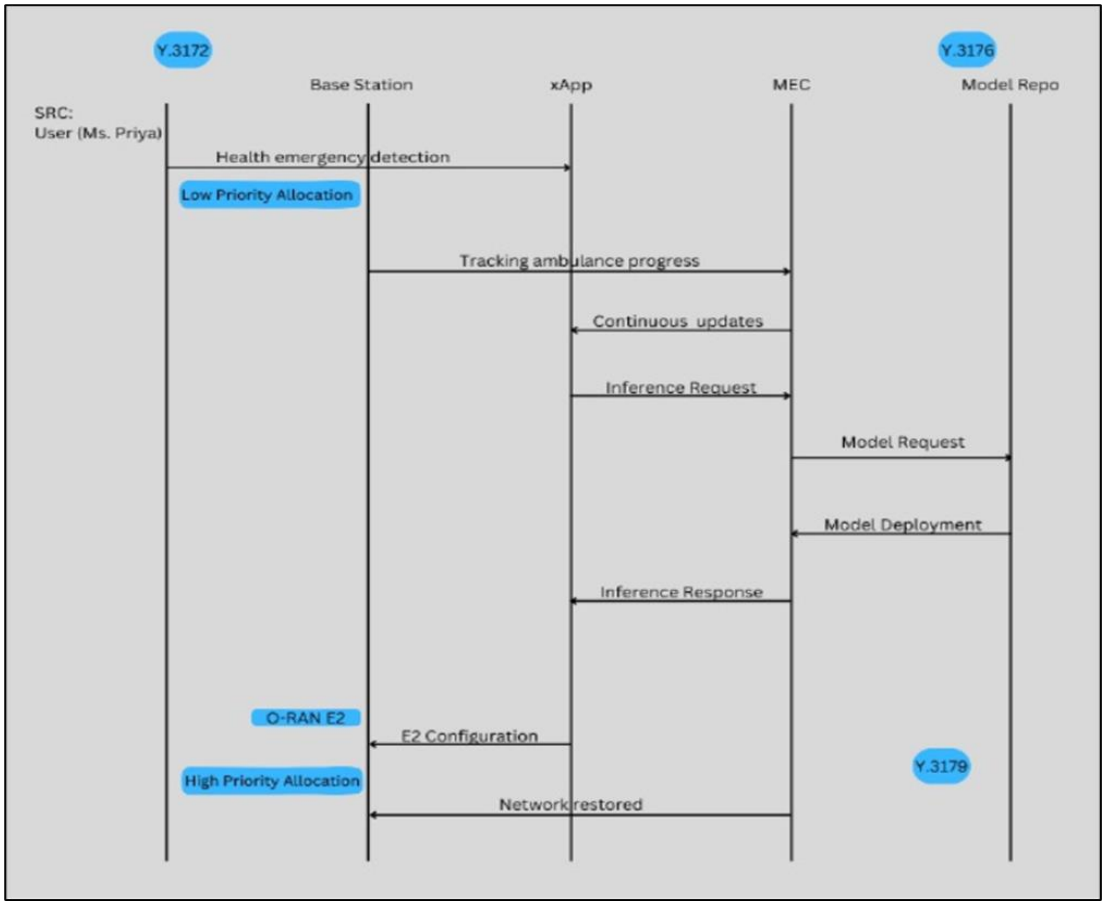

Fig. 9.8.1: sequence diagram that represents the workflow of AI-enabled emergency healthcare response using O-RAN architecture, xApps, MEC, and model repositories.

9.8.6. Code and Data Sets

The PoC utilizes a combination of real-world and synthetic datasets to support both health condition prediction and traffic-aware ambulance routing. For patient modeling, the Healthcare critical care Healthcare_dataset provides vital signs and diagnosis records used to train the health emergency classification model. For traffic simulation and route optimization, OpenStreetMap (OSM) is used to extract real-world road and infrastructure data, which is ingested by Synthetic Mobility Traces (SUMO generated) to generate synthetic mobility traces of ambulances and vehicles. These datasets collectively enable a realistic and context-aware AI-driven emergency response system. Code an ddata sets can be found here: https://github.com/shaista2509-sk/OMACS_EMERGENCY_HEALTHCARE_SYSTEM.git.

9.8.7. Simulated Use cases

The demonstrated use case simulates a real-time emergency health event triggered by a user via a mobile app. Upon pressing the emergency button, the system performs multiple coordinated actions: it analyzes the user's medical history to classify the type of emergency, locates the user, alerts nearby hospitals and ambulances, and identifies the fastest ambulance route using live traffic data. Simultaneously, the network is dynamically reconfigured via the RIC to ensure low-latency, high-priority communication for all emergency-related flows. As the UE moves (simulated via SUMO), the system handles handovers and maintains QoS guarantees. Once the patient is safely transferred and treated, the network autonomously de-prioritizes emergency traffic, restoring normal operation. This use case validates the effectiveness of AI-driven, intent-based orchestration in an O-RAN framework for critical healthcare services.

9.8.8. Architectural concepts

The architecture integrates the following key layers:

- Application Layer: User interface for emergency activation and real-time tracking.
- AI/ML Layer: Models for health anomaly detection and ambulance route optimization.

- O-RAN Network Control Layer: Dynamic E2E QoS enforcement and real-time network reconfiguration via RIC (RAN Intelligent Controller).
- Communication Layer: Continuous data relay between user device, ambulance, and hospital.

The system establishes closed-loop control from application intent (health emergency) to network-level prioritization and restoration.

9.8.9. Demo and Evaluation

The PoC demonstrates a real-time emergency health response scenario triggered by a user pressing an emergency button within a mobile app. This activates two AI models running as xApps on the OAIC/O-RAN platform. The first model, deployed via ONNX at the MEC, predicts the user's likely medical condition based on historical healthcare data and broadcasts alerts to hospitals and ambulances. In parallel, the second model identifies the most optimal ambulance route using live traffic data. The system employs an xApp for end-to-end QoS resource management to prioritize all emergency-related communication. The OAIC RIC simulator emulates user mobility and network conditions, enabling a realistic test of AI-native orchestration. As per initial tests, the RIC successfully provisions low-latency slices for critical telemetry (TC-1), while the AI-based routing model reduced ambulance ETA (TC-2) and maintained responsiveness under traffic load by scaling inference dynamically (TC-3). Final integration is ongoing, but each test case demonstrates the system’s potential to support life-saving, intelligent emergency interventions over AI-managed O-RAN infrastructure.

9.8.10. PoC Observation and discussions

The demo presents a compelling emergency response use case, but to reinforce its AI-native network integration, several aspects should be clarified. Specifically, the demo should explain how AI pipelines (e.g., ONNX models) are deployed at the edge with low latency, and what network enhancements (e.g., support for federated inference or container orchestration via TOSCA) are required. The need for cross-domain data sharing between healthcare, transport, and telecom—must be emphasized, showing how different xApps consume and act on diverse datasets. Additionally, clearer definition of UE app to network interfaces (e.g., how an emergency intent triggers slice creation or QoS adjustments) is needed. Finally, highlighting shared knowledge bases or application-aware policies would better demonstrate AI-native orchestration capabilities in a modular, scalable, and access-agnostic manner.

9.8.11. Conclusion

This PoC validates the feasibility of using AI-native orchestration to manage emergency healthcare workflows across user devices, ambulances, and hospitals. By ensuring fast anomaly response, dynamic resource allocation, and seamless E2E communication, it establishes a blueprint for intent-aware, QoS-guaranteed 6G emergency services. The demonstrated improvements in latency, throughput, and recovery time highlight the critical role of AI-integrated O-RAN in future public health infrastructures.

9.8.12. Future Work

1. Sharing sensitive patient data with MEC nodes or central servers poses privacy concerns. Future work should explore federated learning approaches, where models are trained across decentralized nodes (e.g., hospital servers) without moving raw patient data. This ensures compliance with data protection regulations like HIPAA or India’s DPDP Act.
2. In multi-region deployments, a single near-RT RIC may not suffice. Future research is needed on how to dynamically place, migrate, or replicate xApps across distributed RIC instances based on location, load, or priority of emergency events, without disrupting E2 QoS.

Clinicians and first responders require interpretable results when life-critical decisions are made by AI models. There is an open challenge in integrating explainable AI (XAI) techniques into your xApps, so they can provide reasoning for diagnoses or route suggestions.

9.9.1. PoC-009 [b-FG-AINN-I-134] Truth Shield: Real-Time AI-Powered Fake News Control System

9.9.2. Delimiting Characteristic

Architecture approaches for deep integration of AI because the system embeds AI through orchestrated pipelines and service-based architectural patterns, dynamically deploying multimodal AI tools and integrating verification services, which reflects a deep architectural integration of AI rather than a single-stage or isolated use of AI.

9.9.3. Description

Truth Shield is an AI-native, real-time misinformation detection and mitigation system designed for deployment during high-risk events like pandemics, political unrest, or national security incidents. It monitors digital communication across public platforms including WhatsApp groups, Telegram channels, Twitter feeds, and online news sources analyzing multimodal data (text and images) using AI pipelines. The system performs live cross-verification with official databases, OSINT sources, and trusted news APIs. Built on AI-native service orchestration principles, Truth Shield allows YAML or prompt-based use case configuration and dynamically deploys the necessary AI tools (e.g., GPT-based verification, CLIP for image-text alignment). It ensures proactive misinformation control through alerts to network operators and public warnings to users.

9.9.4 Gaps Addressed

Conventional network infrastructures lack the capacity to manage real-time, AI-driven content verification or prevent misinformation propagation. There is no native support in the network layer for triggering verification workflows, adjusting traffic priorities based on content risk, or coordinating mitigation actions across media platforms. Truth Shield addresses this gap by introducing AI-native hooks that integrate content analysis models directly with network functions. It prioritizes threat-related traffic using QoS management, informs operator systems via standardized APIs, and offers programmable interfaces for cross-domain data ingestion—filling the critical void in misinformation-aware network intelligence.

9.9.5. POCs Test Setup

The test setup for Truth Shield is designed as a modular, AI-native pipeline capable of handling real-time ingestion and analysis of multimodal content (text and images). The core infrastructure is built on a microservice architecture, where each agent—responsible for tasks like claim ingestion, preprocessing, classification, verification, and alerting—operates as a self-contained module, coordinated via a central orchestrator. The AI models (e.g., BERT, RoBERTa, CLIP) are trained and deployed on a private internal cloud with ONNX compatibility for efficient inference. Input claims are collected via WhatsApp using Twilio integration, processed through multiple AI agents, and finally responded to with a verified result. The entire system is deployed in a Dockerized environment and exposed via ngrok for remote demonstration. Additionally, a feedback loop allows users to respond to verdicts, enabling future improvements through model retraining and knowledge base refinement. Figure shows scene map.

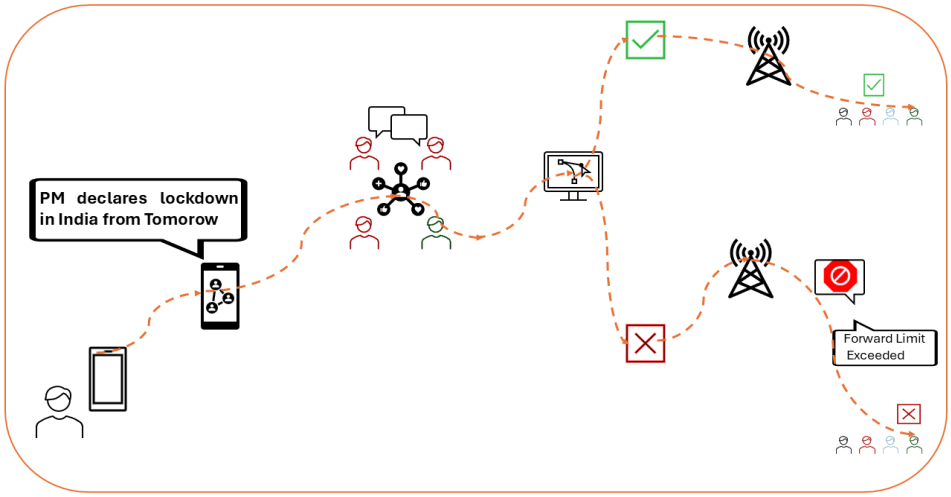

Fig. 9.9.1: Scene map

The following figure shows the pipeline of control in the test setup.

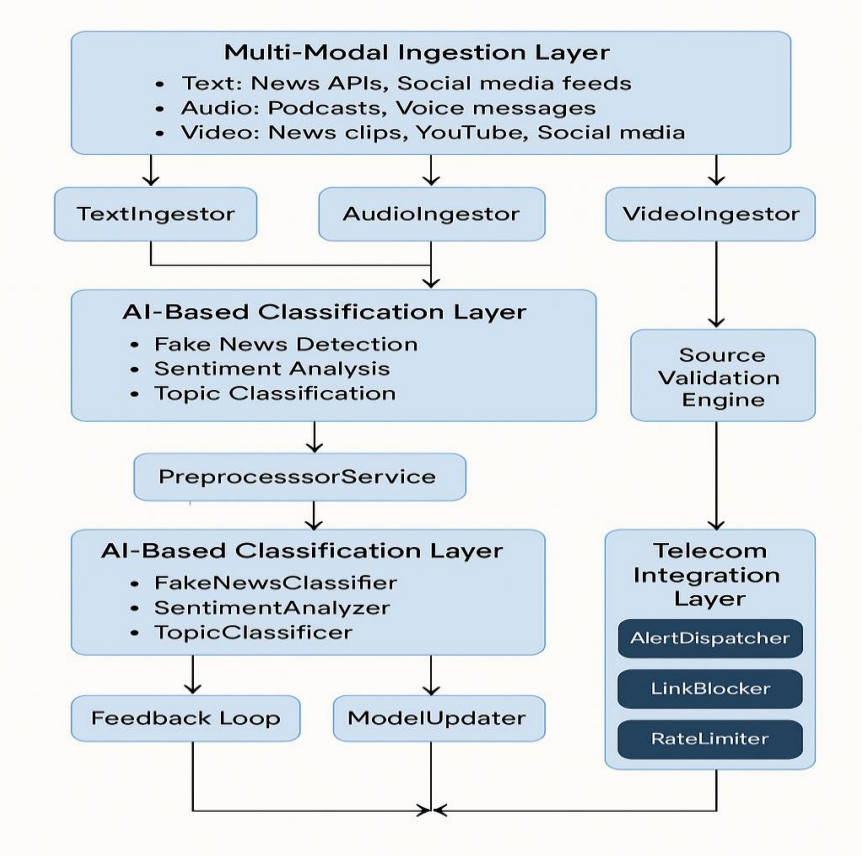

Fig. 9.9.2: The pipeline of control in the test setup.

9.9.6. Code and Data Sets

The dataset used in Truth Shield consists of real-time news and social media data feeds, collected via public APIs and telecom network taps. This data provides a continuous stream of multimedia content, such as text claims and images, which serves as input to the AI pipeline. The code and data sets can be found here: https://github.com/Eswarpnbs/Fake-News-Detection.

9.9.7. Simulated Use cases

In the simulated demonstration, a user sends a potentially false news claim via WhatsApp such as a viral text about a political event or an image implying violence. This input is captured by the Multi-Modal Ingestion Layer and passed through a real-time processing pipeline. The text is analyzed using a fine-tuned BERT model, while the image is checked for authenticity using OCR, CLIP-based similarity scoring, and image integrity tools like ELA (Error Level Analysis). The system cross-verifies the claim with reliable news sources and government databases using semantic similarity and source trust scoring. Based on the outcome, the user receives a verdict (Real or Fake) along with an explanation. If the claim is deemed highly misleading, the system is configured to auto-trigger a telecom alert, simulating an operator-level mitigation workflow. The goal is to demonstrate the end-to-end responsiveness and accuracy of the AI-native misinformation control platform in a fully virtualized testbed.

9.9.8. Architectural concepts

Truth Shield is structured around an AI-Native Service Platform (ANSP) architecture, comprising modular agents that interact over standard interfaces. It includes: (1) a Data Ingestion Agent for collecting user claims and media; (2) a Multimodal AI Agent equipped with image classifiers, text transformers, and cross-modal models (like CLIP) for detection; (3) a Knowledge Correlation Agent that references external verified datasets; and (4) a Policy Orchestrator that determines mitigation actions. These components are orchestrated through a YAML-based service definition, containerized using Docker, and deployed on AI-aware network edge infrastructure. The system supports

integration with telecom operator functions (e.g., via SBI or MEC APIs) to trigger fake news alerts or resource adjustments.

9.9.9. Demo and Evaluation

This PoC showcased real-time ingestion and analysis of user-submitted content—text or images—via a telecom-integrated interface (Twilio WhatsApp). The AI pipeline, composed of modular agents, processed the inputs and delivered verdicts (real, fake, or unverifiable) along with confidence scores and supporting evidence. The setup used pre-trained transformer models (BERT, RoBERTa), fine-tuned for regional relevance, and deployed via Docker microservices. Alerts were triggered based on classifier confidence, and user feedback was captured to refine source credibility. The system ran entirely over a simulated telecom environment and demonstrated accurate, low-latency misinformation detection in a reproducible, agent-based orchestration.

9.9.10. PoC Observation and discussions

The Truth Shield PoC effectively demonstrates the AI pipeline for multimodal misinformation detection but must better articulate its integration with the network infrastructure. One key observation is the lack of an explicit interface with network elements such as Content Delivery Networks (CDNs), which could be leveraged for model hosting, regional caching, and real-time inference services. The architecture should explore the use of semantic communications, enabling the network itself to participate in content verification workflows.

To enhance the AI-native character of the solution, there is a need to define interfaces for model transfer, data exchange, and federated learning, particularly toward CDN or edge computing layers. These models—BERT, CLIP, or other LLMs—can be fine-tuned centrally but hosted at the edge or CDN nodes to support low-latency inference.

Moreover, the proposal would benefit from treating "Inference-as-a-Service" as a native network function, where semantic verification becomes part of the communication pipeline. The demo should also visualize this link by showing how alerts or model decisions are routed via telecom or CDN pathways and clearly state what network capabilities are assumed or required (e.g., MEC support, low-latency data path, secure model hosting).

Concepts like Mixture-of-Experts, federated learning for regional model customization, and closed-loop monitoring should also be explored as potential AI-native extensions that would deepen the integration between AI agents and the underlying network fabric.

9.9.11. Conclusion

Truth Shield demonstrates the potential of AI-native networked systems to safeguard information integrity during critical events. By embedding content verification capabilities into the network layer, it transforms passive infrastructure into an active participant in misinformation control. The system promotes public trust, supports government and operator response mechanisms, and ensures resilience in digital communication ecosystems. With scalable AI pipelines and access-agnostic design, Truth Shield can be adapted to different regulatory, regional, or platform-specific needs serving as a blueprint for future AI-native digital governance tools.

9.9.12. Future Work

Building on the current capabilities of Truth Shield, future work will focus on designing a semantic communication protocol to allow more efficient and context-aware message exchanges between agents and systems. Additionally, the platform will be expanded to support multilingual and regional content, enabling detection and validation of misinformation across diverse languages and local dialects. Finally, we plan to integrate Truth Shield with government alert systems and emergency communication infrastructure, allowing verified responses to be disseminated at scale through official channels during critical events.

9.10.1. PoC-010 [b-FG-AINN-I-129] Build-a-thon 2025: Intelligent Network Traffic Capacity Prediction for BTS

9.10.2. Delimiting Characteristic

AI itself as a core component because the forecasting system fundamentally depends on deep learning models and an integrated LLM to perform prediction and reporting, making AI the indispensable driver of both network optimization and operator-facing insights.

9.10.3. Description

This PoC presents a real-time, AI-native system for forecasting cellular traffic at Base Transceiver Stations (BTS) using deep learning techniques. The system ingests historical BTS counter logs—representing real traffic activity over time—and applies hybrid neural architectures such as LSTM, GRU, and RNN to detect and predict short-term load fluctuations. To improve contextual sensitivity, the models are enhanced with engineered features such as day-of-week, holiday indicators, and peak/off-peak flags. Additionally, the Gemini LLM is integrated to convert predicted peaks into human-readable reports, aiding in operational planning.

The system is designed to run continuously with minimal latency and is optimized for deployment at the network edge (e.g., MEC or CDN layer). This enables telecom operators to proactively manage congestion, balance network load, and ensure optimal Quality of Service (QoS). The solution operates without the need for external sensors or traffic analyzers and aligns with the FG-AINN vision for embedded, intelligent network management.

9.10.4. Gaps Addressed

The current telecom infrastructure largely depends on reactive mechanisms and static thresholds for traffic load balancing, which are insufficient in handling dynamic, context-driven user behavior—such as sudden surges during festivals, emergencies, or location-specific events. There is a significant gap in integrating predictive intelligence within the RAN, particularly at the BTS level, where early detection of congestion could allow proactive action. Moreover, there is no embedded mechanism for contextual interpretation of traffic patterns, nor is there support for translating technical forecasts into operational insights in real time. Existing systems also lack interfaces for deploying AI models natively within the network, such as at the MEC or CDN layer, and offer limited support for real-time inference, semantic reporting, or closed-loop decision making. This PoC addresses these gaps by demonstrating an AI-native, edge-deployable forecasting pipeline that is adaptive, interpretable, and tightly integrated with network operations.

9.10.5. POCs Test Setup

The setup involved constructing a forecasting pipeline using historical BTS counter data enriched with contextual signals such as time-of-day, weekend flags, and event indicators (e.g., festivals). Where real air-interface data was not available, counters were synthetically generated using established mathematical relationships to emulate realistic transport and radio-layer traffic patterns. Signal preprocessing included windowed segmentation and normalization of traffic deltas. The models LSTM, GRU, and RNN were trained individually and also as ensemble sequences (e.g., LSTM → GRU → LSTM) with dropout regularization to prevent overfitting. A Gemini-based large language model (LLM) was added to automatically interpret the peak prediction output and generate user-readable traffic management reports. The system was deployed on a standard CPU setup with a Streamlit-based GUI allowing users to upload datasets, select models, tune hyperparameters, and visualize predictions dynamically. Fig. 9.10.1 shows user centric work flow for AI-based traffic prediction.

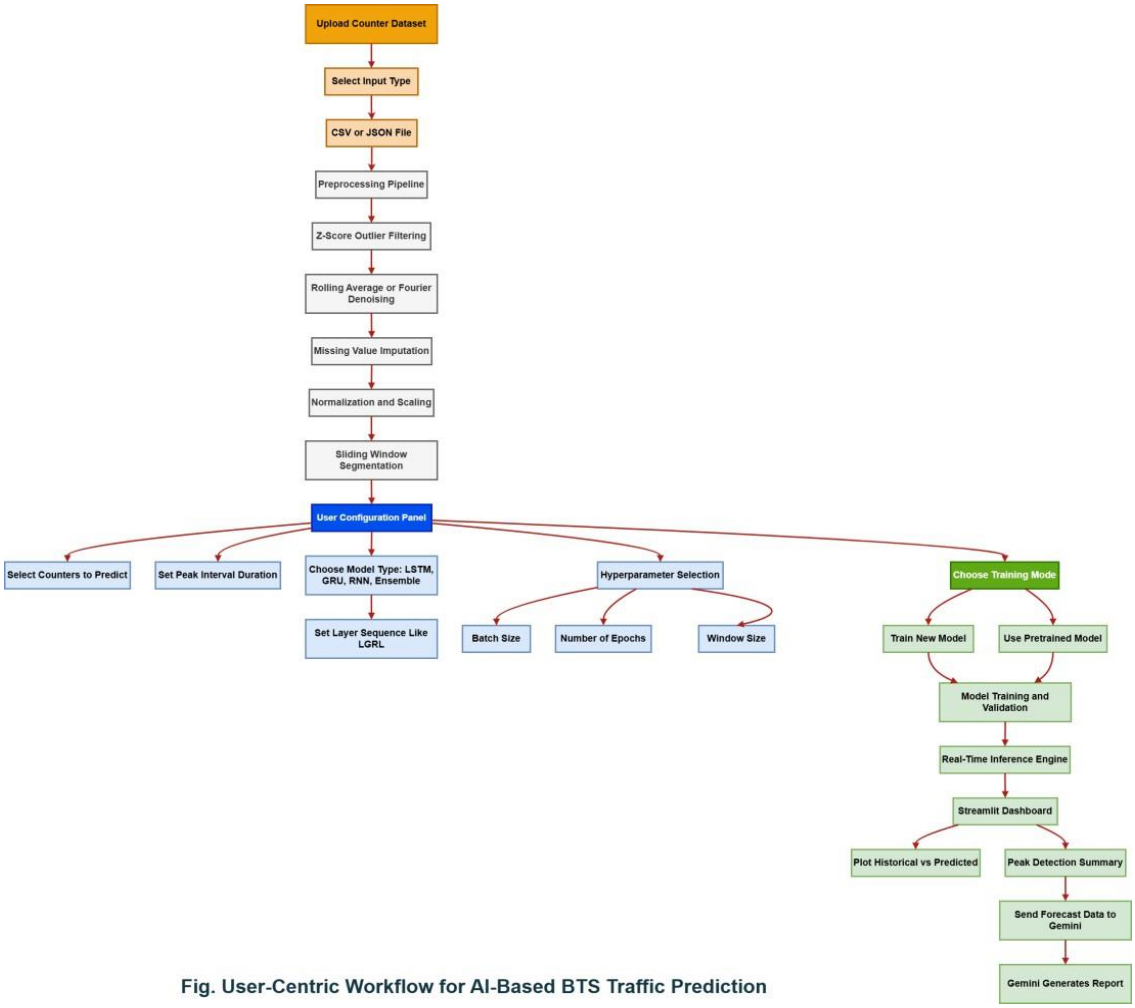

Fig. 9.10.1: User centric work flow for AI-based traffic prediction

Peak Detection Algorithm

Step 1: Data Import
The forecasted counter data is imported .
Step 2: Sliding Window Computation
A sliding window of size k was applied. For each data point $T[i]$, the maximum values in its left window $T[i-k:i]$ and right window $T[i+1:i+k+1]$ were computed.
Step 3: Significance Score Calculation
The significance score $S(i)$ for each point was calculated using: $$S(i) = \frac{\max(T[i-k:i]) + \max(T[i+1:i+k+1])}{2}$$
Step 4: Peak Condition Check
A data point $T[i]$ was considered a peak if: 1) $S(i) \geq \theta$ (threshold condition) 2) $T[i] > T[i-1]$ and $T[i] > T[i+1]$ (local maximum)

9.10.6. Data Sets

Two types of data were utilized:

1. Historical BTS Counter Data - Simulated hourly traffic logs reflecting voice/data sessions with contextual flags (e.g., weekend, holiday).
2. Air Interface Metrics - Synthetic counters mathematically generated using standard signal-level relationships, capturing dynamic wireless behavior under varying load conditions.

9.10.7. Simulated Use cases

The PoC simulated a real-world telecom scenario where Base Transceiver Station (BTS) traffic fluctuates based on time-of-day patterns, weekends, and special events like festivals. The setup

emulated both transport-layer counters and air-interface metrics to predict traffic congestion, enabling preemptive load balancing and QoS adjustments.

9.10.8. Architectural concepts

The architecture comprises a modular, edge-ready AI pipeline tailored for telecom-grade BTS traffic forecasting. Historical traffic counter data is ingested from the BTS and preprocessed to remove noise, normalize values, and extract engineered features such as time-of-day, day-of-week, and holiday indicators. This enriched time-series data is then fed into deep learning models—such as LSTM, GRU, and RNN—trained to detect temporal dependencies and forecast short-term network load. The output of these models is further processed using a Gemini-based language model that automatically converts peak traffic predictions into natural language reports for operational teams. All components are containerized and designed for deployment at the network edge using platforms like MEC or RIC-integrated xApps, ensuring low-latency inference. A real-time dashboard visualizes predictions and alerts, while the system operates continuously to support intelligent load balancing and proactive QoS management.

9.10.9. Demo and Evaluation

The demonstration simulated near real-time traffic prediction at a BTS, using either historical or synthetically generated input. Through the Streamlit dashboard, the system displayed the progression of traffic counter values, predicted near-future values, and highlighted anticipated peak intervals. Users could interactively explore model behavior by choosing between LSTM, GRU, or RNN models, including combinations, and adjust forecasting parameters such as window size and prediction steps. Key metrics like Symmetric Mean Absolute Percentage Error (SMAPE) were updated live to reflect model performance. The Gemini LLM produced explanatory summaries based on predicted peak patterns. The demo successfully showed that the system could anticipate upcoming load surges and generate actionable insights, all within a lightweight, edge-deployable environment running on commodity hardware, with forecast accuracy exceeding 90% on the test set.

Criterion	RNN	LSTM	GRU
Short-term spikes	Good baseline for rapid bursts.	Captures spikes reliably via input/output gates.	Captures spikes with fewer parameters.
Long-term seasonality	Struggles beyond few lags.	Excels at diurnal and weekly patterns.	Remembers moderate-length dependencies.
Training speed & footprint	Fastest to train; smallest memory.	Slowest to train; largest parameter set.	Middle ground in both speed and size.
Parameter complexity	Low complexity.	High complexity with 4 gates + cell state.	Moderate complexity with 2 gates.
Use case	High-frequency, short-horizon bursts.	Day/week cycle forecasting in telecom counters.	Real-time or edge forecasting with limited resources.

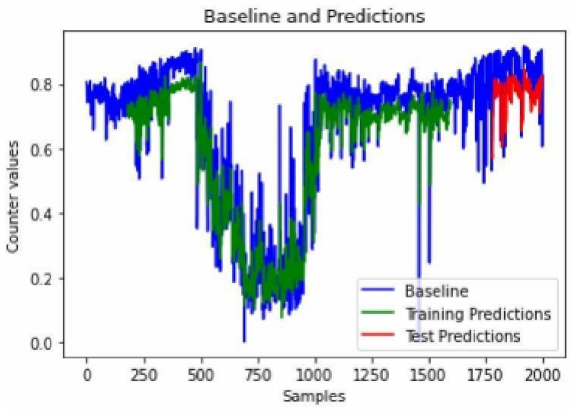

Fig. 9.10.2: Baseline and Prediction given by model

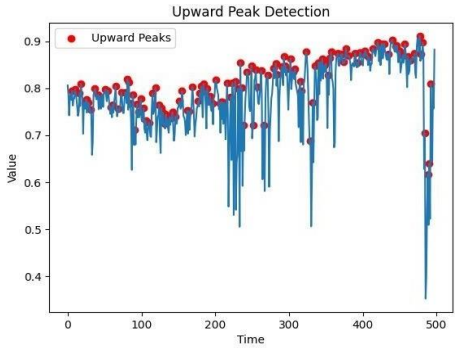

Fig. 9.10.3: Peak detection results

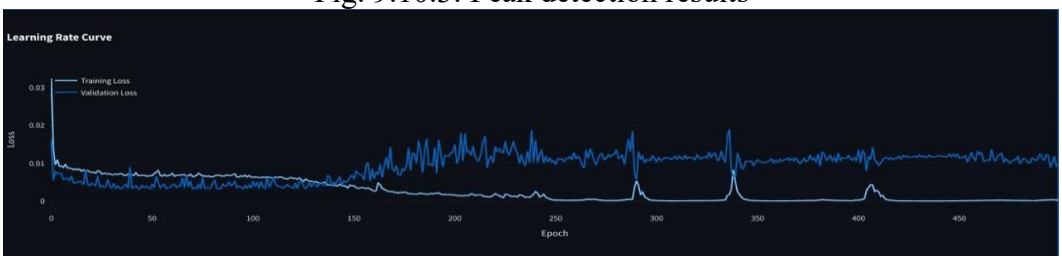

Fig. 9.10.4: Learning rate curve based on model training

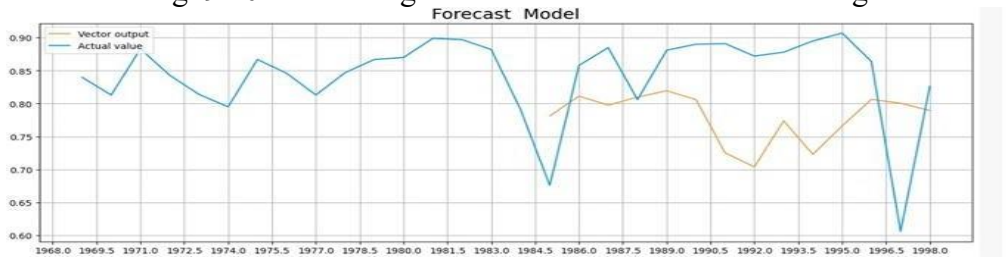

Fig. 9.10.5: Model Forecast result

Fig. 9.10.6: Streamlit-Based GUI for BTS Traffic Forecasting and Control

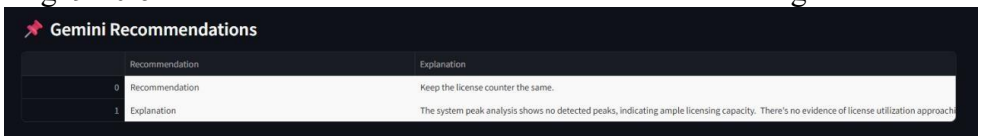

Fig. 9.10.7: Recommendations given by LLM based on peak prediction

9.10.10. PoC Observation and discussions

This PoC effectively demonstrates AI-native forecasting of BTS traffic but must further align with standardized datasets, such as those from ITU, to validate broader applicability and identify coverage gaps. The current approach should be complemented by a recorded video demo to support the real-time interface demonstration. The system processes dynamic inputs like time, packet counters, and contextual events (e.g., festivals), but future iterations must handle diverse NF data types and formats seamlessly. Additionally, adaptive model selection based on data characteristics—such as sequence length, variability, and burstiness—should be incorporated into a recommendation engine. This would enhance the PoC's intelligence in choosing between LSTM, GRU, or RNN under changing network scenarios, especially across air interface conditions.

9.10.11. Conclusion

The PoC demonstrates that AI-native time-series forecasting can replace static threshold-based approaches for BTS traffic prediction. By leveraging deep learning models and integrating contextual knowledge directly into the pipeline, the system offers highly accurate and timely insights. These predictions empower operators to act preemptively—optimizing network resources before congestion occurs—thus improving end-user experience and operational efficiency. Moreover, the use of Gemini for auto-reporting enhances interpretability and actionability, enabling seamless integration into NOC workflows.

9.10.12. Future Work

To enhance the current system, future efforts will focus on integrating additional contextual features such as weather conditions, major public events, and regional mobility patterns to further improve prediction accuracy. The pipeline will be extended to support real-time inference directly within telecom edge nodes, enabling network-aware adaptation through AI-driven RIC (RAN Intelligent Controller) xApps. We also plan to implement federated learning for distributed training across BTS sites while preserving data privacy. Moreover, expanding the model to multi-cell coordination scenarios will support intelligent load redistribution and handover optimization. Lastly, the use of semantic communication techniques for traffic classification and summarization will be explored to enable more intelligent, content-aware traffic control.

9.11.1. PoC-011 [b-FG-AINN-I-155] Explainable AI-Native Intent-Based Self-Healing Network Orchestrator for Rural and Edge Telecom Infrastructure

9.11.2. Delimiting characteristic

AI Agents, Intent-Based Self-Healing, Resilience-Focused Inference Systems, Distributed Multi-Agent Fault Management

9.11.3. Description

Architecture approaches for deep integration of AI because the PoC is centered on a scalable, multi-agent architecture where AI components (anomaly detection, explainability, RAG-based healing) are deeply integrated into the system fabric through agent coordination and knowledge-base coupling, emphasizing architectural design over single-function AI use.

9.11.4 Gaps Addressed

The PoC aligns fully with WG1 definitions of AI-native systems involving distributed decision-making, self-optimization, and resilience-centric inference. Our modular agents map to the WG1 definition of AI Agents, functioning independently to observe, decide, and act without human oversight. This PoC shows how decentralized inference can be operationalized using layered agents that collaborate asynchronously to detect faults, explain them, and heal them via intent-based workflows. The design eliminates reliance on static rule-based systems and enables dynamic adaptation to evolving network conditions, especially critical in resource-constrained rural deployments. By mapping definitions directly from WG1, this PoC clarifies how AI-native systems can be practically implemented for scalable and autonomous fault management.

9.11.5. POCs Test Setup

In this proposal, we use an emulated 50-node network and synthetic datasets derived from realistic rural telecom traffic and outage patterns. Model training is done using Python (Scikit-learn/Keras) on local computer subject to system compatibility and computational requirements. The agent architecture uses a lightweight rule-based coordination protocol, simulating MCP (Model Context Protocol) -style message exchanges. Explainability is demonstrated using SHAP. The system is orchestrated via simulated flows in NS-3 and Python FSM (Finite State Machine) -based agents. Knowledge used for inference and fallback is embedded via a local RAG-style query framework. The test setup for our implementation is customised to simulate a rural telecom infrastructure scenario. The setup includes the following components:

- Code generation model from manually constructed agent scripts (C++ & Python-based), with inference pipelines generated from LSTM + SHAP for anomaly detection and explainability.
- Service orchestrator from xOpera and Open6GCore, integrated with lightweight execution support for simulated TOSCA templates; orchestration flow is managed via an internal controller with optional hook to xOpera runtime for validation of remediation plans.
- Agent framework from custom multi-agent setup built in Python using ZeroMQ-based asynchronous communication between Monitor Agent, Calculation Agent, MCP Agent, and Healing Agent.
- Simulator from internal metric generator using time-series pattern replicators and noise injection scripts for rural network features (optionally extendable with NS-3).
- Datasets from synthetic datasets derived from rural network profiles with 18 input metrics per node, reflecting real-time behaviour (e.g., latency, packet loss, CPU, weather). These datasets replicate environmental, network, and hardware conditions common to edge deployments.

The setup enables isolated testing of fault scenarios (Fiber cut, power fluctuation), real-time metric streaming, model-based anomaly detection, explainability visualisation, and automated healing plan generation.

The Fig. 9.11.1 shows the pipeline of control in the test setup.

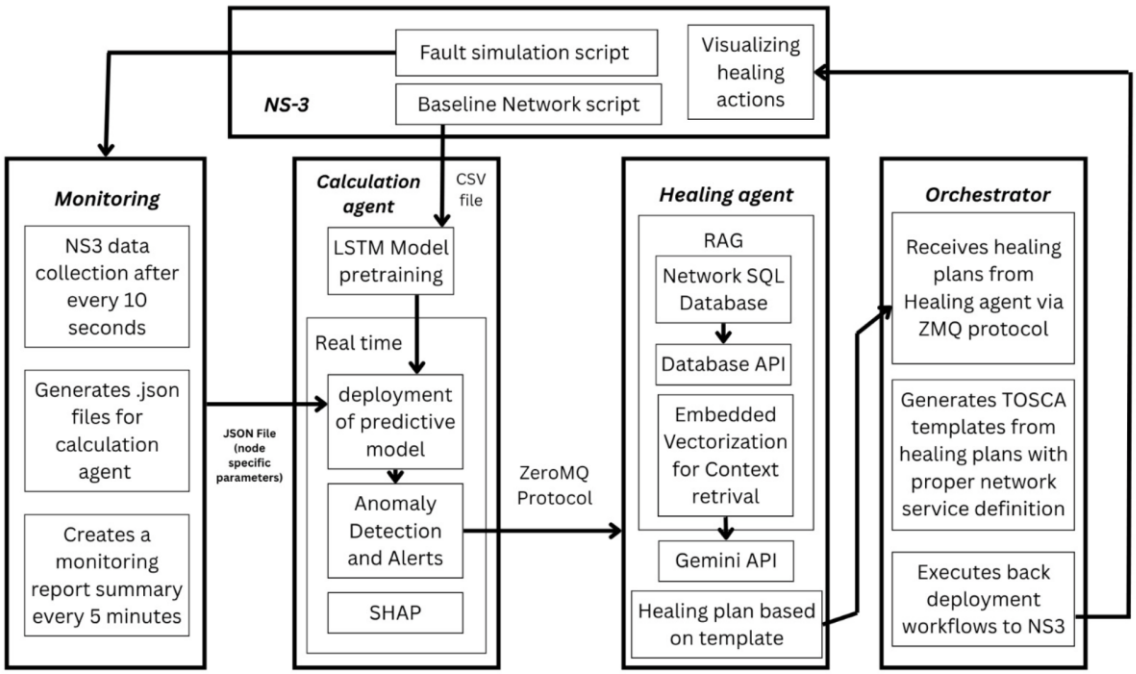

Fig. 9.11.1: The pipeline of control in the test setup.

9.11.6. Code and Data Sets

Synthetic time-series datasets from NS-3. Metrics include throughput, latency, link status, voltage. Annotated fault events: fiber cuts, power fluctuations. Labelled normal vs degraded sequences for LSTM training. Code and datasets can be found here: https://github.com/Rishi8520/rural_ai_selfhealing_net.

9.11.7. Simulated Use cases

Use Case 1 (TST-01): Fiber Cut — Gradual degradation leads to full disconnection. The NS-3 simulation injects progressive fault patterns on a fiber link, allowing the Calculation Agent to detect anomalies in advance using LSTM predictions. SHAP then explains the severity and likely root causes. This prompts the Healing Agent to initiate rerouting via a TOSCA-based self-healing workflow.

Use Case 2 (TST-02): Power Fluctuation — Voltage instability at a network node is simulated. Without the Knowledge Base (KB), the system is slow to react and lacks contextual healing guidance. With KB and RAG integration, the Healing Agent retrieves domain-specific tactics and executes stabilization actions. This comparison benchmarks how intelligence improves reaction speed and effectiveness.

Gaps Addressed:

- Static thresholding cannot anticipate gradual degradation or capture intermediate fault stages.
- Lack of real-time explainability slows decision-making and increases downtime.
- Without LLM-based reasoning, healing actions are rigid and unsuitable for edge-case anomalies.
- Current fault-handling mechanisms lack integration with operator intent or semantic remediation policies.

Necessity:

This approach enables proactive maintenance, real-time awareness, and autonomous coordination across agents, aligned with WG2's goals. It shows that AI-native systems can dynamically reason about complex fault states, adapt to contextual signals, and initiate healing without human intervention. Traditional rule-based systems are inadequate for evolving rural edge environments where fault patterns are unpredictable and policy response needs to be context-aware. Our system

improves fault isolation and ensures minimal false positives through layered agent collaboration and interpretable AI.

9.11.8. Architectural concepts

Key Enablers:

- Multi-agent architecture with isolated responsibilities (monitor, detect, heal)
- LSTM + SHAP-based inference loop
- ZeroMQ-based modular inter-agent protocol
- RAG-powered knowledge delivery for healing agents
- Orchestration via TOSCA/xOpera for intent translation

Modularity: Each agent can be improved independently, provided shared formats for anomaly alerts, healing actions, and orchestration templates are maintained. This enables harmonization across implementations and allows hybrid deployments.

Performance Improvements:

- Socket-based asynchronous execution improves concurrency.
- RAG/LLM reduces recovery latency by providing context-aware plans.

SHAP supports root-cause verification.

9.11.9. Demo and Evaluation

1. TST-01 – Fiber Cut: Detects via LSTM + SHAP. Self-healing workflow executed successfully.
2. TST-02 – Power Fluctuation: Benchmarked with vs. without KB. Measured improvements in detection latency and healing success.

Evaluation includes logs, timing benchmarks, and anomaly confidence scores.

9.11.10. PoC Observation and discussions

- The importance of context-driven healing (vs. static rules) was validated, especially through comparative tests with and without the Knowledge Base.
- Layer-wise logs improved explainability by tracing data flow through Monitor, Calculation, and Healing Agents, helping correlate symptoms to actions.
- Fault isolation steps were visualized with before-after comparison using structured outputs from SHAP explanations and ZeroMQ logs.
- Agents operated asynchronously but harmoniously via defined message schemas, validating the modular design.
- Comments from previous FG-AINN mentoring sessions influenced improvements in fault isolation layering, step-wise detection logic, and symptom tracing. We added internal before/after state snapshots at each layer (metric → anomaly → cause → remedy).
- Lessons learnt include the importance of defining shared input/output schemas early, allowing agents to evolve independently without losing interoperability.
- We also observed that without RAG + Gemini, agents struggle to resolve ambiguous faults, reinforcing the need for semantic context in healing.
- Discussions during development highlighted the value of performance benchmarking across modes (with/without KB) to validate the efficiency of AI-native healing.
- Finally, we iteratively improved the modularity by decoupling orchestration logic from decision-making, aiding maintainability and extensibility.

9.11.11. Conclusion

This PoC affirms that AI-native systems can transform fault management from reactive ticket-based processes into proactive, autonomous workflows. With minimal configuration, the system adapts to diverse rural scenarios.

9.11.12. Future Work

While our current proof-of-concept demonstrates intent-driven anomaly detection and autonomous self-healing through explainable AI and lightweight agents, several open challenges remain for further research and deployment at scale:

1. Scalability of Agents in Heterogeneous Networks: Our current rule-based multi-agent system is effective in small to mid-sized topologies. Future work must explore how coordination and communication between agents scale in large, diverse rural deployments involving different vendor equipment and protocols.
2. Real-Time Constraints and Low-Latency Learning: Incorporating more complex ML models (e.g., transformer-based intent interpreters or reinforcement learning agents) may improve flexibility but introduces latency. Research is needed to balance explainability, speed, and predictive performance in real-time networks.
3. Interfacing with Real Network Controllers ((Software Defined Networking (SDN)/Network Function Virtualization (NFV)): Our orchestrator currently simulates healing actions. Integration with real-world network control systems (e.g., OpenDaylight, Open Network Operating System (ONOS)) will require protocol adaptation, security compliance, and stateful communication models.
4. Edge Model Optimization and Federated Learning: In future iterations, we aim to deploy distributed learning systems using federated or incremental learning to adapt models across geographically dispersed nodes while preserving data privacy and connectivity independence.
5. Intent Disambiguation and Natural Language Generalization: Currently, our intent parser is rule-based. A future goal is to integrate a fine-tuned language model for more flexible and user-friendly intent interpretation, especially for multilingual operators or ambiguous commands.
6. Resilience to Adversarial Events and False Positives: The system must be stress-tested under adversarial scenarios (e.g., spoofed telemetry, conflicting intents) to ensure robustness and prevent cascading failures due to incorrect healing triggers.

These open problems provide an opportunity for iterative enhancement of our framework toward a production-ready, standard-compliant AI-Native orchestration system suitable for rural and edge telecom networks.

Along with these, the 3 additional test cases can also be implemented along with a robust frontend.

The following test cases are part of the proposed roadmap and will be demonstrated in future FG-AINN conferences or industry pilots:

- TST-03: Equipment Degradation

Simulates gradual performance decay (e.g., rising CPU temperature, error counters).

The goal is to detect degradation early and suggest preventive maintenance.

- TST-04: False Positive Suppression

Evaluates system robustness by injecting benign fluctuations (e.g., short latency spikes).

The system should not raise false alarms or trigger healing unless persistent patterns are observed.

- TST-05: Intent-Based Threshold Configuration

Simulates dynamic operator intents (e.g., "maintain uptime > 95%") and measures system-wide propagation via MCP.

9.12.1. PoC-012 [b-FG-AINN-I-156] Build-a-thon 2025: 5G Adaptive Signal Solutions for Rapid Transit and Aerial Operations

9.12.2. Delimiting Characteristic

AI itself as a core component because the PoC's novelty lies in an AI-based adaptive beam steering algorithm that learns motion patterns and adjusts connectivity in real time, making AI the essential functional element driving performance improvements.

10.12.3. Description

In this PoC, we aim to generate datasets from ray tracing simulators and study the effectiveness of the proposed adaptive beam steering algorithm in maintaining connectivity for moving devices, such as UAVs and emergency response vehicles, under various environmental conditions.

Specifically, the PoC will pave the way for generating datasets for beamforming applications, and our proposed algorithm performance ensures it meets the demands of dynamic environments while maintaining low computational complexity in real time.

At this stage, we hypothesise that our algorithm can learn various physical motion patterns of the mobile station and develop a sense of directional movement. Secondly, we hypothesise that in real-world 5G base station deployments, the worst-case time delay would be approximately 120 ms for getting CSI information from feedback from UE.

9.12.4 Gaps Addressed

Traditional beam steering methods are either slow to adapt or rely on pre-programmed patterns, which are insufficient for highly dynamic use cases like fast-moving vehicles or drones. This PoC addresses the gap by introducing real-time learning-based steering, enabling context-aware, low-latency beam tracking. It tackles the limitations of static or delayed beamforming and aligns with the needs of AI-based, mobility-optimized networks

10.12.5. POCs Test Setup

Sionna's AI-native simulation environment is used as a Ray Tracing simulator for PHY-layer modelling with integration of Uniform Rectangular Array (URA) antenna with custom beamforming modules. The scene will be loaded from OpenStreetMap based on the user's input. A config YAML file will be used by the user to define parameters and choose the settings and scenario. Evolutionary algorithms will be used to explore diverse beam configurations across varying user positions and channel conditions, logging beam configuration, Channel State information (RSSI), and spatial parameters.

Using this dataset, we then pre-train an agent, enabling it to learn an initial beamforming policy. Finally, the agent is fine-tuned online using real-time feedback, allowing it to adapt dynamically to user mobility and environment changes with faster convergence and improved beam precision.

9.12.6. Code and Data Sets

A synthetic dataset was generated using the Sionna RAN simulator, capturing beam configurations, Channel State Information (CSI), Received Signal Strength Indicator (RSSI), and spatial parameters across varied UE positions and channel conditions. This dataset was used to pre-train the reinforcement learning agent, enabling it to learn an initial beamforming policy prior to online fine-tuning. The data generation was driven by evolutionary algorithms exploring the beam search space under mobility conditions. Code and data sets can be found here: https://github.com/syed-azim-git/Buildathon2.0-SSNECE.

10.12.7. Simulated Use cases

The simulation models a high-mobility wireless environment, such as urban transit or drone-based communication, where a base station equipped with a Uniform Rectangular Array (URA) antenna tracks and serves mobile UEs in motion. The UEs follow configurable trajectories, and their positions—as well as channel conditions—change over time. The simulator generates real-time CSI and RSSI measurements based on user movement, which are fed to the AI layer to dynamically adjust beamforming weights. This setup allows testing the system's ability to maintain signal quality and connection continuity under fast-changing spatial and channel conditions.

9.12.8. Architectural concepts

The architecture integrates:

- Motion learning models for trajectory prediction.
- RSSI-CSI feedback loops for real-time environment sensing.
- Dynamic beam alignment logic executed at the antenna array controller or baseband unit.
- An AI-based control layer that adapts beam parameters without centralized intervention.

9.12.9. Demo and Evaluation

To evaluate the performance and robustness of dynamic beam steering algorithms, we utilized Sionna, a TensorFlow-based open-source simulator developed by NVIDIA. Sionna offers a flexible and highly accurate framework for simulating the physical layer of wireless communication systems and digital twin environment scenes.

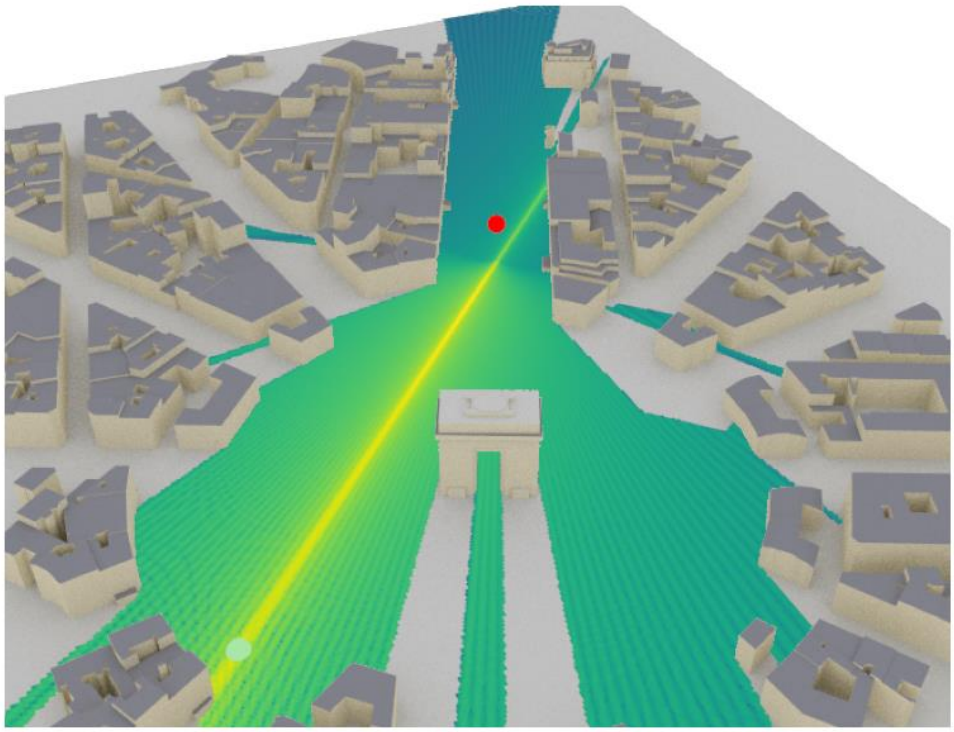

Fig. 9.12.1: Regional area simulation

A) Regional Area Simulation:

Fig. 9.12.1shows regional area simulation. To make our simulations more realistic and relevant, we plan to simulate wireless communication scenarios in our **local city environment**. While Sionna supports a few predefined urban scenes, we extend this by using **real-world building and street data** from **OpenStreetMap**.

This allows us to create a simulation environment that closely mirrors the **actual layout and structure** of our chosen region. As a result, our beamforming algorithms can be tested under **realistic propagation conditions**, improving their effectiveness and applicability to real-world deployments.

B) Genetic Algorithm-Based Beam Optimization

Fig. 9.12.2. shows flow chart of Genetic Algorithm-Based Beam Optimization. At the core, a **modified Genetic Algorithm (GA)** that iteratively searches for the optimal beamforming weight vector to maximize **Received Signal Strength Indicator (RSSI)**. Unlike conventional GA approaches that operate over a static search space, our implementation features a **dynamic search space** that evolves based on feedback from prior iterations. This allows the algorithm to:

- Focus its search on promising beam directions
- Adapt to fast-changing channel conditions

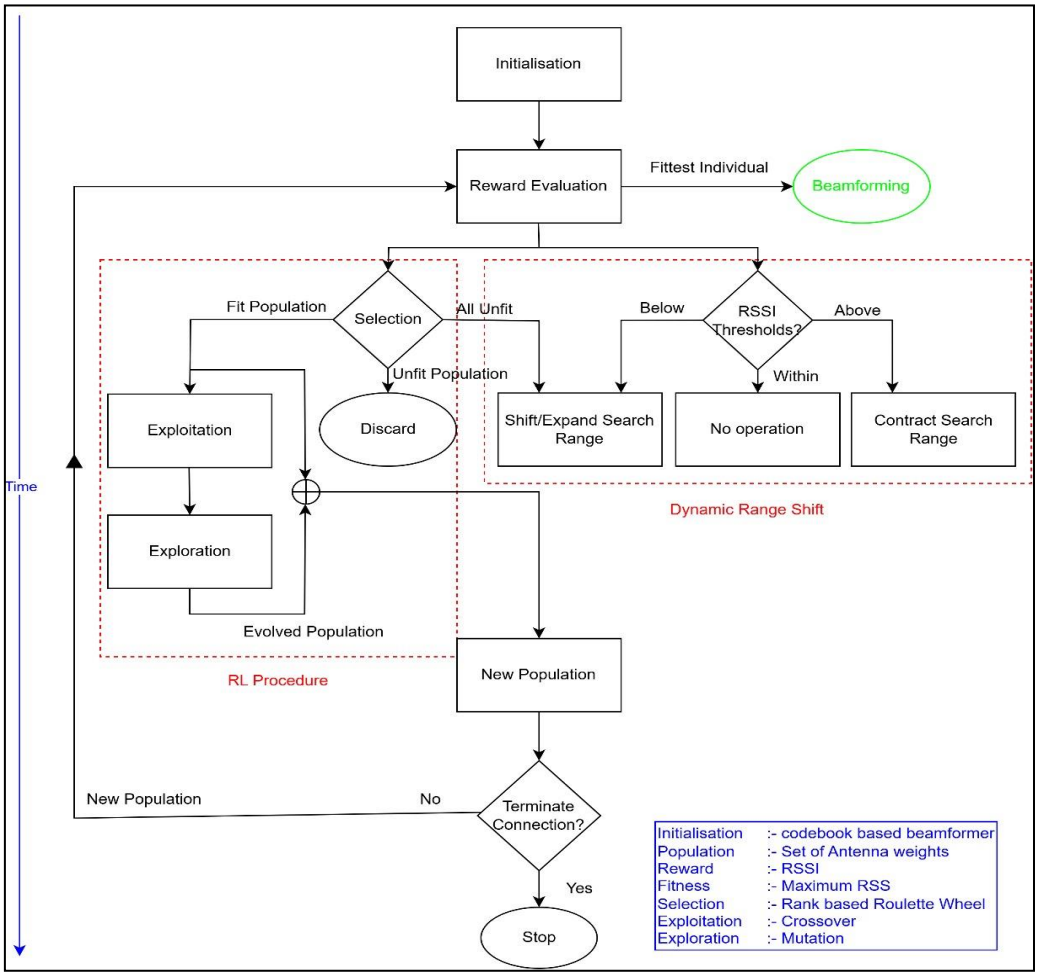

Fig. 9.12.2: Flow chart of Genetic Algorithm-Based Beam Optimization

C) **Key Features of Our GA Variant**

- **Feedback-Driven Evolution**: Each generation evaluates candidate solutions by applying the corresponding beam weights and measuring the RSSI at the UE.
- **Search Space Adaptation**: The GA narrows or shifts the search bounds based on previously high-performing beam directions, effectively "tracking" the best beam over time.
- **Realistic Evaluation Loop**: Each candidate beam is evaluated using Sionna's ray-traced propagation and MIMO system modelling, providing realistic feedback for selection pressure.

D) AI Native Implementation

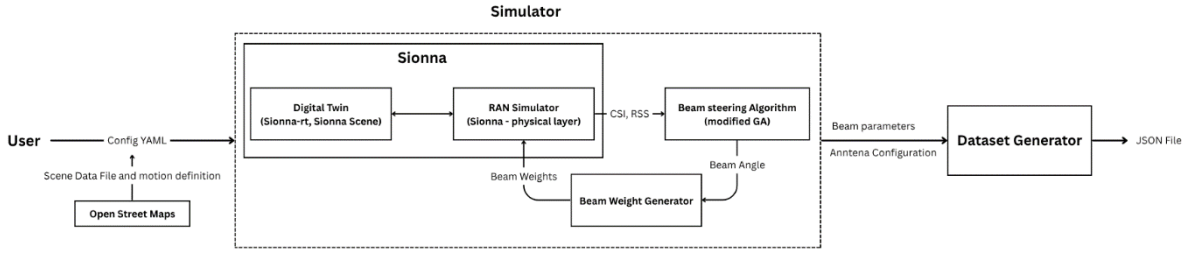

Fig. 9.12.3 shows AI native Implementation.

E) Dataset Generator

The Dataset Generator simulates the process of beam selection between a base station (BS) and a moving receiver (MS) by performing a grid-based beam search at multiple receiver locations. It is designed to create a comprehensive dataset that can be used for analyzing beamforming strategies, visualizing search spaces, or training machine learning models for beam prediction.
The generator first defines the metadata for the base station. This includes the 3D position of the BS, its orientation (which direction it is facing), and the dimensions of its antenna array (number of rows and columns). This metadata is constant for the entire dataset and describes the transmitter's physical configuration. It is saved once in the JSON file so that all receiver positions share the same BS context.
Next, the generator sweeps through a series of receiver positions (frames), simulating the movement of the receiver, such as a vehicle or pedestrian, along a path. For each frame, the receiver's 3D position and orientation are recorded. This captures how the user equipment (UE) moves in space relative to the fixed base station, providing realistic variation across the dataset. At every receiver position, the generator performs a beam search by trying various possible combinations of elevation (θ) and azimuth (φ) angles. For each beam direction tested, it records the received signal strength (RSS). The best beam, which yields the highest RSS, is saved separately for easy reference, along with its corresponding theta and phi angles. This best-beam data is crucial for evaluating beam selection algorithms, as it tells us the optimal direction at each receiver location.
In addition to the best beam, the generator also saves the entire search space for each frame. This includes all the theta and phi angles tried during the search and their associated RSS values. By storing this complete grid of results, the dataset preserves detailed information about how signal strength varies across the entire beamforming space. This is essential for reconstructing beamforming heatmaps, analyzing the performance of the search strategy, or training models that learn from the full distribution of possible beams. Finally, the generator also saves Channel State Information (CSI) for each receiver position. CSI contains detailed parameters of the radio channel, such as path delays, angles of arrival and departure, and complex gains. This rich data enables more advanced channel modeling and simulation, supporting the development of realistic communication system prototypes.
The resulting JSON file has a clear structure, which includes a single metadata section describing the base station setup and a frames section that contains an entry for each receiver position. Each frame entry includes the receiver's position and orientation, the best beam details, the complete search space data, and the channel state information. This organized format ensures the dataset is self-contained and easy to use for various research and development tasks in wireless communications.

9.12.10. PoC Observation and discussions

The PoC highlights a strong integration between AI-driven beamforming and PHY-layer simulation; however, there is a need to abstract the Sionna-specific components to generalize the approach for RAN optimization across simulators. Clarifying the timing and placement of the genetic algorithm (GA) for dataset creation and the online reinforcement learning (RL) agent for real-time adaptation is important to improve reproducibility and modularity. The RESTful API interfaces between the AI layer and the RAN simulator should be documented explicitly to ensure interoperability—enabling replacement of Sionna with other tools while preserving functionality. Future enhancements could focus on dataset creation workflows, YAML-based simulation configuration, and support for user-defined BS/UE locations to enable broader experimentation. Drawing from the DeepSense 6G Challenge, the team may extend the framework to other radio-layer learning problems. Finally, formalizing the simulator's configurable parameters and their representation in YAML would help standardize testing across use cases and environments.

9.12.11. Conclusion

The PoC demonstrates how AI-based beam steering can improve 5G performance in highly mobile scenarios by minimizing signal degradation and handover interruptions. By using real-time signal metrics and learning models, it offers a responsive and context-aware solution that enhances throughput, reliability, and service continuity in dynamic environments laying the groundwork for resilient, intelligent transportation and emergency communication systems.

9.12.12. Future Work

- An algorithm to understand the pattern of motion of UE to better predict the beam parameters
- Energy Aware Decision making for Beam Steering

9.13.1. PoC-013 [b-FG-AINN-I-157] Bridging the Standardization Gap in Next-Generation Telecommunications

9.13.2. Delimiting Characteristic

Architecture approaches for deep integration of AI because the PoC emphasizes an agentic knowledge base and coordinated multi-agent pipeline (gap detection, analysis, recommendation, compliance), showing how AI is deeply integrated into the system architecture to lower the standards participation barrier.

9.13.3. Description

In this Build-a-thon 2.0, our objective is to build and demonstrate a functional AI-native solution with the goal of significantly reducing the 6G standards barrier for contributors.

Specifically, we will:

1. Design and demonstrate a functional AI-native system centred on an agentic knowledge base, composed of autonomous agents tasked with identifying and addressing deficiencies within ITU-T 6G standards relevant to developing regions.
2. Deploy the Gap Detection Agent to autonomously scan curated ITU documentation and highlight issues such as network instability and energy inefficiency; subsequently, the Gap Analysis Agent will evaluate the technical and regional significance of these findings.
3. Employ the Recommendation Agent to generate technically sound, standards-aligned proposals addressing the identified gaps, followed by the Compliance Agent, which ensures that all outputs conform to ITU-T requirements and maintain consistency with established norms.
4. Showcase the complete pipeline from autonomous gap detection to the generation of validated contributions—thereby reducing the technical entry barrier and empowering broader African participation in the evolution of 6G standards.

At this point, we hypothesise that with the solution of integrating an autonomously identifying AI agent on our designed knowledge base, we can generate highly relevant and accurate technical contributions, thereby substantially lowering the entry barrier for African contributors to participate in 6G standards development.

9.13.4 Gaps Addressed

This PoC addresses key gaps in automating standards intelligence through embedded, modular AI. It introduces application-aware processing of large standards corpora, autonomous agent lifecycle management, and low-latency inference with broad generalization. By operating on a semantically indexed dataset, it enables scalable, context-driven gap analysis. This architecture enhances inclusion and agility in the 6G standards ecosystem, especially for underrepresented regions

9.13.5. POCs Test Setup

The test setup consists of a modular, AI-native system built around a multi-agent architecture, designed to autonomously analyse 6G standards, generate compliant contributions, and produce executable code samples. The system components are as follows:

- Multi-Agent Mechanism: A team of autonomous agents, built using the Langchain framework, collaborates to perform key tasks across the standardisation pipeline. This workflow is designed to be methodical, moving from a broad overview to specific, verifiable gaps. It leverages a multi-agent approach, where each agent has a specialized function. These agents include:
  - Agent 1: The Cartographer

  Maps the thematic landscape of ITU documents by performing topic modelling and summarisation to provide a high-level overview, guided by user input.
  - Agent 2: The Analyst

  Conducts a detailed comparative analysis of selected topics to uncover inconsistencies, outdated standards, or underexplored areas.
  - Agent 3: The Hypothesiser

  Transforms analytical findings into clear, testable hypotheses by generating probing questions about potential gaps.
  - Agent 4: The Verifier

Tests each hypothesis through targeted semantic search and evidence gathering to either confirm or refute the proposed gaps.

- Agent 5: The Synthesiser

Compiles the findings into a structured, prioritised report detailing the gaps, their supporting evidence, and potential implications.

- Agent 6:Training Agent: Generates simplified explainers and educational content to support new African contributors in understanding and using the system.

9.13.6. Code and Datasets

A curated and semantically indexed corpus of ITU-T standards and related 6G technical documents. Code and data set available at https://github.com/AgabaEmbedded/Bridging-Standards-Gap.

9.13.7. Simulated Use cases

The system supports autonomous AI agents to analyze ITU-T 6G standards and identify specification gaps, particularly for developing regions. It semantically links abstract technical terms to local deployment challenges like power scarcity or rural infrastructure. Agents generate compliant proposal drafts and sample code aligned with standards evolution. This enables inclusive, AI-assisted contributions to global standardization efforts.

9.13.8. Architectural concepts

This PoC introduces a multi-agent AI architecture, where each autonomous agent performs a distinct function in the standards engagement lifecycle: gap detection, contextual analysis, recommendation, and compliance evaluation. The architecture is supported by a domain-specific LLM and a semantically indexed standards corpus, creating an end-to-end pipeline for AI-assisted, standards-aware collaboration.

Functional requirements are as below:

REQ1: It is critical that the system perform NLP parsing, preprocessing across modalities, and semantic embedding on ITU 6G standard documents to create a structured and searchable knowledge base.

REQ2: It is critical that the knowledge base is well-structured to optimize search and accessibility.

REQ3: It is critical that an autonomous gap detection AI agent analyse the semantically indexed ITU knowledge base to identify underspecified, missing, or insufficiently detailed gaps relevant to 6G standards.

REQ4: It is expected that a gap analysis agent shall contextualize and evaluate identified gaps based on their technical importance and regional relevance, especially focusing on issues pertinent to African and developing regions.

REQ5: It is expected that the standards recommendation agent shall generate preliminary proposals and enhancements addressing the identified gaps within the 6G standard specifications.

REQ6: The system shall provide the potential standard contributor with clear, AI-generated insights and knowledge to assist in formulating innovative and standards-compliant contributions for 6G standardization.

Requirements for this type of application:

- Source of data: ITU standard material
- Models: Generative AI, Llama 3.23B, Mistral 7B, Imagebind, Jina embedding-V4
- Vector Database: Pinecone Vector Database
- Policies: Text generation based on prompt

9.13.9. Demo and Evaluation

TEST-1: Knowledge Base Loading and Parsing

Objective:

Validate that the AI agent can successfully load, parse, and semantically embed ITU-T 6G standard documents from the curated knowledge base.

Process:

- The agent accesses the knowledge base.
- Confirms the completeness and correctness of the loaded data structure for downstream processing.

Success Criteria:

- All relevant ITU-T documents are loaded without error.
- Semantic embeddings reflect accurate document content and structure.
- Metadata such as section titles, interface definitions, and regional tags are correctly extracted.

TEST-2: Gap Detection via Agentic Analysis

Objective:

Evaluate the AI agent's ability to detect gaps in the 6G standards by identifying incomplete sections, underspecified interfaces, and region-specific challenges.

Process:

- The agent scans the embedded knowledge base using keyword extraction, summarization, and embedding similarity.

It flags:

- Incomplete or missing sections of the standard.
- Underspecified or ambiguous interface definitions.
- Technical issues linked to African network scenarios (e.g., rural base stations).

Success Criteria:

- The agent outputs a list of candidate gaps with relevant excerpts and metadata.
- Detected gaps align with known problem areas from domain experts.

9.13.10. PoC Observation and discussions

The PoC demonstrated that the multi-agent AI system could effectively load, parse, and semantically embed standard documents into a structured and searchable knowledge base. Metadata such as section titles and interface definitions were accurately extracted, and the system operated efficiently across all test inputs. Gap detection agents successfully identified missing or underspecified sections, including region-specific challenges relevant to Africa and developing areas. The system provided clear, traceable outputs with relevant excerpts, enabling explainable insights. While text processing was robust, multimodal content like diagrams needs further refinement. Overall, the PoC validated the system's ability to support standards development through AI-driven analysis, contextualization, and recommendations.

9.13.11. Conclusion

This PoC demonstrates multi-AI agent based AI-native approach to democratizing 6G standards development through an agent-driven system powered by a semantically indexed ITU dataset. By automating gap detection, analysis, and compliance evaluation, it lowers the entry barrier for underrepresented regions, particularly in Africa. The modular design enables scalable, context-aware processing and sets the groundwork for broader integration of AI in standards workflows. This PoC marks a significant step toward inclusive, intelligent, and responsive 6G standardisation processes

9.13.12. Future Work

Scalability to Diverse Developing Regions: Future studies must prioritize the scalability and adaptability of 6G solutions to the highly diverse contexts within developing regions. Challenges include varied infrastructure maturity, significant language diversity, and fluctuating levels of technical capacity.

AI Ethics: Critical attention is required for the ethical implications of deploying AI-driven 6G technologies. Future studies should address concerns related to data privacy, algorithmic bias, transparency, and accountability, particularly when AI models interact with sensitive user data and make decisions impacting connectivity and services.

9.14.1. PoC-011 [b-FG-AINN-I-158] AIONET- AI-native Intent-Oriented Network Edge Tuner

9.14.2. Delimiting Characteristic

AI itself as a core component because the PoC makes AI the core decision-making engine for real-time traffic classification, priority inference, and shaping at the UE, with AI logic (rule-based inference) directly driving system autonomy and user-centric control.

9.14.3. Description

The proposed Proof of Concept (PoC), AIONET, implements an AI-native, rule-based edge traffic controller that performs real-time traffic classification, priority inference, and bandwidth shaping directly at the user equipment (UE). Operating as a lightweight agent, AIONET uses application-level metrics—such as CPU usage, flow statistics, and encrypted traffic detection—to infer per-application priorities dynamically. It supports programmable policy injection through user override mechanisms and provides feedback-based shaping without dependency on centralized infrastructure.

Key Components:

- Traffic Classification & Monitoring: Using Scapy and psutil, the system identifies active flows, maps ports to running processes, and calculates traffic statistics per flow.
- Priority Computation Engine: Based on flow bandwidth, packet counts, CPU usage, and encrypted flow flags, the system computes a composite score used to assign priorities (high/medium/low).
- Dynamic Bandwidth Shaping: Leveraging the pyroute2 library for kernel-based HTB shaping or pytun for user-space TUN shaping fallback, bandwidth limits are enforced per flow using adaptive logic.
- Policy Injection Interface: A Flask-based UI/dashboard allows users to manually override computed priorities. These inputs are recorded and reflected dynamically in shaping behavior without restarting the service.
- Unknown Traffic Handling: Flows not associated with known applications are tagged as "unknown" and shaped based on packet timing behavior. Their signatures are logged into a local database for inspection or future policy updates.
- Data Persistence: SQLite is used to log metrics, feedback, and unknown flows, enabling auditability and offline analysis.

Evaluation:

The system is tested in a live Linux environment, where shaping behavior is validated using tools like iftop, custom packet injections, and interactive bandwidth stress tests. The responsiveness to user-injected policies and unknown traffic detection is demonstrated through the dashboard interface.

Scope & Relevance:

The PoC directly addresses objectives outlined in the Terms of Reference for AI-native networking by providing:

- A programmable, autonomous, and user-centric shaping engine
- Functional operation in the non-radio domain
- Alignment with digital equity goals through fair bandwidth distribution

This approach exemplifies distributed intelligence, intent-based policy enforcement, and AI-native edge control without requiring ML model inference or central orchestration.

9.14.4 Gaps Addressed

Gaps Addressed:

1. Lack of Real-Time, Application-Aware Traffic Control at the UE:

o Gap: Traditional AI-native concepts have focused on network core or centralized orchestration, leaving user equipment (UE) with limited intelligent control.

o Addressed by AIONET: Implements fine-grained traffic monitoring and shaping on the UE using process-level statistics and flow dynamics, demonstrating the feasibility and value of decentralized, device-side intelligence.

2. Insufficient Support for Programmable Policy Injection:

o Gap: Current network policies are static or require backend orchestration.

o Addressed by AIONET: Enables dynamic, user-driven policy injection via feedback through a Flask dashboard, allowing priority override and real-time impact on traffic shaping logic.

3. No Handling Strategy for Unknown or Emerging Applications:

o Gap: Networks struggle to handle unclassified or new traffic formats effectively.

o Addressed by AIONET: Logs unknown flow signatures, assigns tentative priorities using flow interval heuristics, and provides a user feedback loop to later reclassify them—paving the way for semi-autonomous policy learning.

4. Missing Lightweight Implementations Suitable for Digital Inclusion Scenarios:

o Gap: AI-native solutions often assume high compute availability and overlook edge-constrained environments.

o Addressed by AIONET: Designed as a lightweight, rule-based daemon using Scapy, SQLite, and optional kernel hooks, making it deployable on low-end user devices without central coordination.

5. Underdeveloped User-Feedback Integration in Edge Intelligence:

o Gap: User-centric decision feedback is rarely integrated at the UE.

o Addressed by AIONET: The feedback loop affects real-time prioritization and persists across sessions, demonstrating adaptive intelligence rooted in user intent and contextual awareness.

9.14.5. POCs Test Setup

This PoC is deployed as a standalone Python-based agent (AIONET) on a Linux-based user equipment (UE), simulating an edge device in a real-world mobile network. The test environment is designed to evaluate real-time traffic classification, dynamic bandwidth shaping, and policy injection using the following components:

1. Hardware & OS

- Device: x86 Linux laptop
- OS: Ubuntu 20.04 LTS (system with tc, pyroute2, and TUN support)
- Kernel: Linux kernel ≥ 5.x (for HTB queuing)

2.Software Stack

Layer	Components Used
Traffic Sniffing	Scapy's sniff() in passive mode
Flow Analysis	Custom FlowAnalyzer using deque & interval metrics
Process Mapping:	psutil → app name ↔ port ↔ CPU usage
Shaping Control:	pyroute2 for kernel-based HTB OR pytun for fallback
Policy DB	SQLite (aionet1.db) with app_priorities, unknown_apps, feedback, app_metrics
Web UI	Flask with /metrics dashboard, override form, and custom packet sender
Threads	Background threads for CPU monitoring, port mapping, and packet sniffing

3. Traffic Generation

• Real Traffic: Zoom, YouTube, Chrome, file downloads, etc., run in parallel

• Synthetic Packets: Sent via /send_custom_packet Flask endpoint (custom UDP-based packet class MyPacket)

• Unknown Apps: Any traffic not mapped via psutil is logged in unknown_apps

4. Bandwidth Shaping Setup

- HTB Classes are created per-app using pyroute2 on interface ifb0
- Rate limits are based on dynamic priority score computed from:
- Application CPU %
- Flow bandwidth (kbps)
- Packet count
- Encryption flags
- User-defined bonuses
- UN fallback is used if pyroute2 or tc kernel interface fails
- Shaping updates are immediately visible in iftop or tc -s qdisc

Evaluation Metrics

- Accuracy of flow classification (known vs. unknown)
- Response to user override via web dashboard
- Effectiveness of shaping (cap observed via iftop / tc)
- Overhead on CPU usage by AIONET process
- Persistence & learning (e.g., unknown apps accumulate over time, user overrides persist in DB).

9.14.6. Code and Data Sets

Code and data set available at https://github.com/Varsh-gr8/AIONET_ITU_build-a-thon2.0.git. These datasets are primarily generated at runtime and stored in SQLite (aionet1.db) or loaded from JSON configuration. They support traffic classification, policy injection, flow monitoring, and shaping decisions.

1. app_priorities(app_name TEXT PRIMARY KEY, base_priority TEXT, user_override TEXT, last_updated TEXT)

2.feedback(id INTEGER PRIMARY KEY, app_name TEXT, current_priority TEXT, override_priority TEXT, timestamp TEXT)

3.unknown_apps(id INTEGER PRIMARY KEY, app_signature TEXT UNIQUE, first_seen TEXT, last_seen TEXT, assigned_priority TEXT)

4.app_metrics(id INTEGER PRIMARY KEY, app_name TEXT, cpu_percent REAL, bandwidth REAL, priority TEXT, timestamp TEXT)

aionet_priorities.json (Contains static bonus weights for scoring and fixed port-based shaping rules)

9.14.7. Simulated Use cases

1. Dynamic Bandwidth Shaping for Real-time Conferencing (Zoom, Teams)

Scenario: A user is in a Zoom call while background apps like Chrome and YouTube are active.

Simulation:

- AIONET detects Zoom traffic by port and process mapping.
- FlowAnalyzer tracks packet intervals and bandwidth.
- High CPU and frequent packets raise Zoom's computed score.
- apply_bw_shaping() boosts bandwidth dynamically for Zoom (HTB or TUN).
- User feedback confirms priority as correct (or manually overrides via /override).

Outcome: Zoom receives consistently prioritized bandwidth without manual config.

2. Unknown Application Detection and Logging

Scenario: A new or unsupported application begins transmitting packets.

Simulation:

- Packets are identified by Scapy but not mapped in port_to_app.
- AIONET logs signature in the unknown_apps table.
- A default shaping strategy is applied (high if fast interval, else medium).

Outcome: Unknown flows are not ignored and can be reviewed via dashboard for user override purposes.

3. User Policy Injection via Feedback

Scenario: A user notices YouTube is buffering and wants it prioritized.

Simulation:

- User override with app=YouTube, priority =high.
- AIONET updates user_override in app_priorities.
- All future shaping checks respect this override.

Outcome: Priority changes take effect immediately without restarting the system.

5. Custom Packet Format Injection for Resilience Test

Scenario: Admin wants to test system behavior using non-standard packets.

Simulation:

- User sends packets using /send_custom_packet Flask route.
- Custom packet format MyPacket is injected into the system using Scapy.

- AIONET handles it like any unknown flow and applies default shaping.

Outcome: System can tolerate non-standard/experimental traffic without failure.

6. Monitoring Flow Behavior & Bandwidth Score Adaptation

Scenario: A user starts multiple downloads in Chrome.

Simulation:

- FlowAnalyzer records high throughput and frequent packets.
- Chrome gets boosted bandwidth based on CPU usage and traffic stats.
- Metrics logged into app_metrics and served via /metrics.

Outcome: The system adapts passively to load, optimizing without needing user intervention.

7. Fallback to User-space Shaping (TUN-based)

Scenario: The device lacks kernel support for tc / ifb0.

Simulation:

- IPRoute is not available or fails; AIONET uses _ensure_tun_shaper() instead.
- TunTapDevice (aionet0) simulates traffic queuing in user-space.

Outcome: Bandwidth control still functions without requiring privileged kernel support.

9.14.8. Architectural concepts

Fig. 9.14.1 shows Code flow and Fig. 9.14.2 shows Basic Flow.

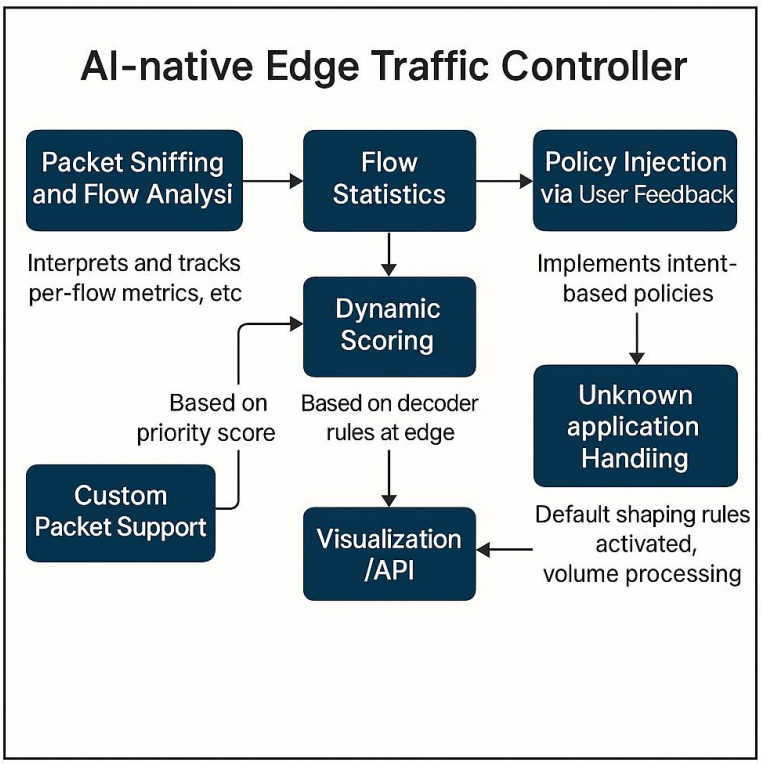

Fig. 9.14.1: Code flow

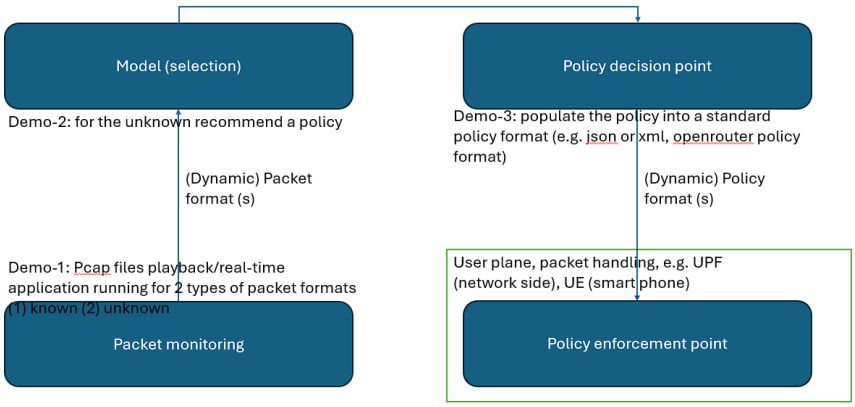

Fig. 9.14.2: Basic flow

9.14.9. Demo and Evaluation

Demo 1: PCAP Playback/Real-time running applications for Known and Unknown Packet Formats

- Setup: Use Real-time running applications/ Scapy or tcpreplay to inject .pcap files with both known application traffic (e.g., Zoom, Chrome) and unknown/custom packet formats (using the MyPacket class).
- Objective: Show that known traffic is prioritized based on real-time CPU, bandwidth, encryption, and past behavior. Unknown traffic is logged with auto-assigned default shaping rules (e.g., based on flow interval).
- Evidence:
  - Real-time classification via /metrics endpoint.
  - SQLite logging in unknown_apps table.
  - Visual shaping effect seen in logs/verified using iptables/tc

Demo 2: Policy Recommendation for Unknown Applications

- Setup: Introduce unknown traffic (e.g., via send_custom_packet() endpoint).
- Objective: Demonstrate how the system:
  - Detects unclassified flows.
  - Assigns shaping behavior (medium/high) based on interval and frequency.
  - Exposes visibility on dashboard for admin/operator to review and change the shaping by overriding the priority of the application. (from which model learns)
- User Feedback Loop:
  - Admin manually sets new priority via /override.
  - Reflects instantly in shaping behaviour.
- Evaluation Metric:
  - Time taken to log → identify → override.
  - Change in shaping confirmed using tc -s qdisc or API.

Demo 3: Standard Policy Injection and Enforcement

- Setup: Simulate a JSON-based update in aionet_priorities.json (e.g., bumping YouTube bonus to 0.8).
- Objective:
  - Demonstrate programmable policy injection.
  - Use a file watcher (optional thread) to reload bonuses without restarting the system.
- Outcome:
  - Bonus affects score → triggers change in computed priority.
  - Triggers HTB class update (visible in logs and system interface).

System Evaluation Criteria:

Aspect	Method
Traffic Classification	Evaluate packet parsing, protocol tagging, flow construction accuracy

Priority Scoring Accuracy	Correlate app usage, CPU stats, and shaping behavior
Bandwidth Shaping Efficacy	Use iftop and tc to verify traffic capping
Unknown Packet Handling	Monitor entries in unknown_apps, shaping defaults
Policy Adaptability	Change bonus/user override and observe system response
Resilience	Fallback to pytun if kernel shaper is unavailable
User Visibility	Dashboard /metrics response fidelity and UI responsiveness

9.14.10. PoC Observation and discussions

Motivation and Focus:

The PoC aimed to create an AI-native, lightweight, and autonomous traffic controller that operates at the User Equipment (UE) level without relying on a central backend.

Unknown Packet Handling:

To address feedback regarding handling previously unseen traffic, the system detects and logs unknown flows, using flow characteristics like packet interval and volume to assign a default priority (e.g., “high” if interval is low).

Dual-mode Bandwidth Shaping:

o Implemented both kernel-level shaping using pyroute2 (HTB via tc on ifb0) and

o A user-space fallback using pytun with a token bucket rate-limiter.

This ensured functionality even when ifb0 was unavailable.

AI-driven Priority Scoring:

- Developed a scoring model based on:
- CPU usage of the application (via top)
- Packet count and bandwidth of the flow
- Encryption flag (TLS detection on ports 443, 8443)
- Application-specific bonus weights
- This scoring is used to compute dynamic traffic priority.

User Feedback & Policy Injection:

o Users can override computed priorities through a Flask-based dashboard (/override endpoint).

o Overrides are persisted in SQLite and immediately affect shaping.

o This mechanism acts as a programmable, intent-based policy injection point—addressing one of the main comments.

Policy Configuration Reloads:

o While the system uses aionet_priorities.json to configure bonuses and port rules,

o I learned from feedback that adding a watcher thread to dynamically reload this file could improve policy responsiveness (though runtime override already exists).

Separation of Policy Decision & Enforcement:

- The system clearly separates:
- Decision logic (priority computation and user overrides)
- Enforcement (actual bandwidth capping via HTB or TUN)
- This aligns with FG-AINN architectural expectations.

Security Considerations:

o Current implementation flags encrypted flows but lacks deep packet inspection or robust anomaly detection.

o These were acknowledged as future areas of improvement.

Learning & Takeaways:

- Gained experience in implementing AI-native principles like:
- Autonomous, local inference
- Decentralized policy control
- Dynamic adaptation based on system and traffic context

- Understood how to balance simplicity with functionality, especially in resource-constrained edge environments.

9.14.11. Conclusion

This PoC successfully demonstrates an AI-native, lightweight, and autonomous traffic control system operating entirely at the user equipment (UE) level. By integrating real-time flow analysis, dynamic scoring based on application context, user-driven policy overrides, and dual-mode bandwidth shaping, it delivers decentralized, programmable network management without backend dependency. The architecture aligns with the objectives of FG-AINN by showcasing non-radio domain intelligence, flow-level adaptability, and user-centric control. This work lays the foundation for future enhancements in security, collaborative inference, and resilience—offering a concrete step toward AI-native network standardization.

9.14.12. Future Work

• Security and Trust Mechanisms

The current AIONET prototype lacks integrated encryption validation, secure update channels, and trust verification for user feedback. Future work will explore lightweight cryptographic checks and attestation mechanisms for policy integrity and safe override operations.

• Collaborative Inference across Devices

AIONET operates in isolation per device. Introducing cooperative, distributed learning among user equipment (UE) could enhance collective awareness of emerging traffic patterns or malicious activity, especially in dense edge environments.

• Scalability for Multi-User/Enterprise Scenarios

The solution is currently UE-centric. Scaling the framework for shared networks (e.g., hotspots, campus, or enterprise LANs) will require new abstractions for multi-user traffic shaping, role-based policy hierarchies, and per-user flow visibility.

• Fine-Grained Application Identification

The reliance on port-to-process mapping is coarse and can miss traffic from multiplexed or containerized applications. Integration of DPI-lite (Deep Packet Inspection) or machine learning-based traffic fingerprinting is a potential enhancement.

• Context-Aware Shaping Beyond CPU and Bandwidth

AIONET uses CPU usage, packet rate, and encryption flags. Future versions could integrate additional context like screen activity, user foreground focus, battery status, or even predicted intent to refine shaping decisions.

• Formal Policy Language Support

While current overrides are JSON-based and ad hoc, the introduction of a lightweight, declarative intent language for policies (similar to P4 or OpenIntent) would make the system more expressive and user-friendly.

• Long-Term Traffic Learning and Auto-Tuning

Adding support for learning from historical flows could allow AIONET to auto-tune scoring weights and bonuses over time using reinforcement learning or clustering methods.

• Resilience and Fault Tolerance

Currently, if a shaping module crashes or interfaces reset, the logic doesn’t automatically recover. Adding watchdogs, state persistence, and auto-recovery scripts would enhance robustness.

• Testing on Diverse Network Conditions

Extensive evaluation across varying wireless conditions (e.g., 5G, poor LTE, congested Wi-Fi) and integration with emulators (like ns-3) will provide deeper insights into system behavior under stress.

• Standardization Alignment

Future iterations should better align with FG-AINN WG1 deliverables, including mappings to AI-native architectural components and contribution to reusable templates for other implementers.

10. Combined Learnings from Build-a-thon 1.0 and Build-a-thon 2.0 Mentoring Sessions

The mentoring sessions across all participating PoCs revealed several critical themes that are essential for maturing AI-native network solutions and aligning them with real-world deployment constraints and research expectations.

Below is a summary of technical observations into actionable learnings that cut across AI modeling, system design, operational integration, and future alignment with standards and scalable architectures.

1) Clarity of Data Structures and Model Inputs

Across PoCs, there was a clear need to explicitly define data schemas—such as packet-level fields (e.g., timestamp, protocol, source/destination ports, packet length), CLI command parameters, JSON structures from simulators like NS3, and YAML templates for healing workflows. Projects were encouraged to demonstrate how their systems handle dynamic or unknown data formats, particularly when supporting real-time, user-driven scenarios. The inclusion of sample data formats, validation pipelines, and data parsing techniques is essential to communicate system robustness.

2) AI Model Selection, Integration, and Hosting

PoCs showcased diverse AI techniques (DQN, LSTM, Bi-LSTM, LLM agents), but mentors emphasized the importance of explaining where and how models are hosted, trained, and invoked. For example, if an LSTM model is used in Open5GS or if a DQN selects vendors, it must be clear what inputs it takes, what outputs it produces, and what network state or features are required. Some teams were advised to explore inference-as-a-service models, semantic reasoning, and hosting in CDN or MEC layers, depending on the use case (e.g., real-time healthcare, cooperative ISAC, fault healing).

3) System Interfaces and Layer-wise Architecture

A common gap was insufficient detail on how different components interact—such as how the application layer sends requirements to the AI model, how the AI layer makes decisions, and how the network layer enforces them. PoCs were urged to present layer-wise diagrams, interface schemas (e.g., REST, gRPC, NeIF, Os-Ma-Nfvo), and detailed before/after workflows to clarify orchestration, healing, or monitoring actions. For example, CLI execution workflows need to specify how changes are determined, mapped, and validated, what base models are used for intent parsing, and how domain/network knowledge is structured.

4) Closed-loop Automation and Fault Management

Several PoCs (especially related to slice management and healing agents) were encouraged to demonstrate closed-loop feedback systems with monitoring, reasoning, and actuation cycles. Projects like healorch_agent.py and others implementing NS3 simulation with real-time anomaly detection were advised to show clear workflows: from symptom detection → anomaly alert → healing plan generation → config action via CLI/YAML. Further, creating fault isolation logic, defining KPI thresholds, and developing domain-specific knowledge bases (KBs) were considered essential for making systems operationally meaningful.

5) Security, Credentials, and Execution Environment

For PoCs that execute network configuration (e.g., CLI push to VyOS/Cisco/Juniper), security considerations like credential management, access control, and trusted execution environments were largely underdefined. Teams were reminded to specify how login, authentication, and permission models are managed, especially in remote or multi-tenant environments.

6) Use Case Depth and Deployment Focus

Mentors emphasized the need for realistic scoping—particularly for PoCs targeting specific deployment contexts like Delhi. Projects were advised to focus on one use case end-to-end (e.g., human activity detection, slice resource scaling, ISAC for emergencies) and map the application's needs to network slice parameters, AI inference points, and orchestration enhancements. Clear documentation of what will be implemented, what is out of scope, and what AI technique is demonstrated helps in evaluation.

7) Knowledge Base and ITU Standards Alignment

Knowledge curation and sharing were recognized as a foundation for system scalability and remote team collaboration. The idea of building a central knowledge base—including CLI templates, known faults and fixes, model versions, RAG schemas, and standard mappings—was strongly recommended. Additionally, PoCs were encouraged to conduct gap analyses against ITU standards and public datasets (e.g., traffic datasets) to align with international frameworks and contribute back insights.

8) Advanced Themes: Semantic Communication, Federated Learning

Emerging techniques like semantic communications, federated learning, and mixture of experts were proposed as future enhancements. These approaches could support model collaboration across domains, reduce redundancy, and enable privacy-preserving analytics. For example, Cellucast and SPV were guided to explore semantic verification, regional inference, and RAN-aware learning, provided interface and resource requirements from the network are clearly defined.

11. Bibliography

[b-Shah] Shah. P et al. "Topics and sample submissions for Build-a-thon 2025", ITU Focus Group on AI Native for Telecommunication Networks (FGAINN) https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-094-R1.docx.

[b-FG-AINN-O-08] Xiaomi An et al., "Concept building for artificial intelligence native for telecommunication networks", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/output/FG-AINN-O-08.zip.

[b-FG-AINN-I-104_R1] Priyadarsini K et al., "AI-Native Proactive Network Slice Marketplace for Dynamic Service Ecosystems", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-104-R1.docx.

[b-FG-AINN-I-136] A. Israel et al. "Build-a-thon 2025: Using AI to Reduce the 6G Standards Barrier for African Contributors", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-136.docx.

[b-FG-AINN-I-128] C.Varshah and N. Chakraborty "Build-a-thon 2025: AIONETx", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-128.docx.

[b-FG-AINN-I-133] N. P. Mohare et al. "Build-a-thon 2025: NETSPEAK", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-133.docx. truth

[b-FG-AINN-I-135] S. Azim et al. "Build-a-thon 2025: 5G Adaptive Signal Solutions for Rapid Transit and Aerial Operations", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-135.docx.

[b-FG-AINN-I-130] A.Nishtala et al. "Build-a-thon 2025: Explainable AI-Native Intent-Based Self-Healing Network Orchestrator for Rural and Edge Telecom Infrastructure", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-130.docx.

[b-FG-AINN-I-132] K. Singh et al. "Build-a-thon 2025: Invisible Guard – Passive Wi-Fi Sensing for Smart Border & Building Surveillance", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-132.docx.

[b-FG-AINN-I-131] S. S. ZABEEN et al. "Build-a-thon 2025: SRM UNIVERSITY-AP", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-131.docx.

[b-FG-AINN-I-134] Bhavana et al. "Build-a-thon 2025: Truth Shield: Real-Time AI-Powered Fake News Control System", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-134.docx.

[b-FG-AINN-I-129] Nethravathi K A et al. "Build-a-thon 2025: Intelligent Network Traffic Capacity Prediction for BTS", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-129.docx.

[b-FG-AINN-I-155] A. Nishtala et al. "Build-a-thon 2025: Explainable AI-Native Intent-Based Self-Healing Network Orchestrator for Rural and Edge Telecom Infrastructure", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-155.docx.

[b-FG-AINN-I-156] S. Azim et al. "Build-a-thon 2025: 5G Adaptive Signal Solutions for Rapid Transit and Aerial Operations" , https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-156.docx.

[b-FG-AINN-I-157] A. Israel et al., "Bridging the Standardization Gap in Next-Generation Telecommunications", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-157.docx.

[b-FG-AINN-I-158] C. Varshah et al. "AIONET- AI-native Intent-Oriented Network Edge Tuner", https://extranet.itu.int/sites/itu-t/focusgroups/ainn/input/FG-AINN-I-158.docx.

Annex I

A.13 justification for proposed new ITU-T TR.POC-AINN "Technical Report - Proof-of-Concept activities for AI Native Networks"

Question:	Q20/13	**Proposed new ITU-T Technical report**	Tashkent, Oct 28-Nov 6	
Reference and title:	ITU-T TR.POC-AINN "Technical Report - Proof-of-Concept activities for AI Native Networks"			
Base text:	-		**Timing:**	2027-03
Editor(s):	Preksha Shah, IIT Bombay, email: 30005100@iitb.ac.in Liya Yuan, ZTE, email: yuan.liya@zte.com.cn		**Approval process:**	Agreement

Purpose and scope (Define what this document will address and its intent or objectives in order to indicate the limits of its applicability):

On the basis of the progress made by ITU-T FG AINN WG4, including PoCs and outcomes from collaborative activities such as the Build-a-thon, this technical report focuses on the following aspects:

- Considerations on general requirements and expectations from the PoCs, including demonstrating the delimiting characteristics of AI Native, proving the concept practically with code, test setup and demo setup, and evaluation on test scenarios;
- PoC studies focusing on demonstrating AI-native capabilities across a diverse range of network scenarios; PoCs are mapped to (a) gaps, (b) use cases, and (c) architecture concepts.
- Combined learnings from mentoring sessions held for Build-a-thon.

Summary: This Technical Report provides details on the Proof-of-Concept activities under ITU-T FG-AINN WG4. This report provides the technical summary of the activities done under PoC, and it covers the following:

- Requirements for the Proof of Concept, including Build-a-thon.
- Description of the Proof of Concept and results.
- Learnings from PoC development mentoring sessions, including Build-a-thon

Relations to ITU-T Recommendations or other documents (approved or under development):

ITU-T Y.3102, ITU-T Y.3172, YSTR.AN-PoC

Liaisons with other study groups or with other standards bodies:

ITU-T SG2, SG11, SG16, IEEE, ETSI ENI, ETSI ZSM, TMF, 3GPP, NGMN

Supporting members that are committing to contributing actively to the work item:

IIT Bombay, ZTE