International Journal of Science and Research Archive
eISSN: 2582-8185
Cross Ref DOI: 10.30574/ijsra
Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Data plane intelligence: AI-based optimization for traffic engineering and intrusion mitigation in next-gen networks

Kamaldeen oladipo [1, *], Oluwabukunmi Ogunjimi [2], Olaoluwa Oguntokun [1], Jude Ogedegbe [3] and Richmond Chibuzor Usoh [4]

[1] *Nokia Technologies, Nokia Networks, Middle East Africa.*
[2] *Project Management Office, INTAGO, Nigeria.*
[3] *Department of Intl Business and Data analysis, Ulster University, UK.*
[4] *ZTE, Congo Brazzaville.*

International Journal of Science and Research Archive, 2025, 15(03), 188–206

Publication history: Received on 20 April 2025; revised on 28 May 2025; accepted on 31 May 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.3.1658

Abstract

This paper explores a novel framework for deploying self-optimizing AI agents designed to enforce real-time security policies across dynamic broadband infrastructures. Given the rise of zero-touch networks, increasing traffic heterogeneity, and growing cyber threats, conventional reactive security methods are no longer sufficient. We propose an architecture that combines reinforcement learning (RL), federated observability, and edge-native threat detection. The paper introduces a scalable agent-based model with proactive anomaly detection and self-adjustment capabilities. Key contributions include a hybrid decision loop, a risk-weighted policy optimizer, and an adaptive trust index. The proposed solution is validated through simulations and real-world telecom KPIs. The results demonstrate enhanced mean time to detect (MTTD), reduced false positives, and improved threat response efficiency.

Keywords: AI Agents; Self-Optimization; Broadband Infrastructure; Real-Time Security; Federated Learning; Network Observability; Reinforcement Learning; Edge AI; Anomaly Detection; Zero-Trust; Threat Intelligence; Telecom KPIs

1. Introduction

1.1. Convergence of AI and Programmable Networking

The telecom sector is undergoing a paradigm shift driven by the convergence of artificial intelligence (AI) and programmable networking. This evolution is not merely incremental it marks a foundational change in how networks are engineered, secured, and optimized. Traditionally, network management has relied heavily on manual configurations and static rules that are ill-suited to address the demands of next-generation networks (NGNs), which are characterized by high dynamism, heterogeneity, and scale.

Programmable data planes, powered by technologies such as P4 and extended Berkeley Packet Filter (eBPF), offer fine-grained control over packet processing at line rate. This programmability has opened the door for embedding real-time AI intelligence directly into the data plane a concept referred to as data plane intelligence (Bosshart et al., 2014; Bera et al., 2021). In parallel, AI techniques especially machine learning (ML) and deep reinforcement learning (DRL) have demonstrated significant potential in automating traffic engineering (TE), anomaly detection, and predictive security analytics (Zhang et al., 2021; Wang et al., 2023).

* Corresponding author: Kamaldeen oladipo

The synergy between AI and programmable networking allows for closed-loop automation, adaptive control, and proactive threat mitigation, thereby transforming the static fabric of network infrastructure into a self-optimizing and self-defending system.

1.2. Problem Statement

Despite the promise of programmable networking and AI, existing implementations largely treat traffic engineering and security as disjoint domains. Current network infrastructures are still heavily reliant on predefined configurations, threshold-based alerts, and rule-based intrusion detection systems (IDS) that fail to adapt to emerging traffic dynamics or adversarial behaviors in real time. These static mechanisms cannot keep pace with the increasing complexity of cyber threats or the fluctuating bandwidth demands of real-time applications such as augmented reality (AR), autonomous systems, and industrial IoT (Doshi et al., 2022; Li & Zhang, 2022).

Moreover, most AI applications reside in the control plane or management plane, creating latency overhead and limited reaction speed. The absence of intelligent, autonomous decision-making at the data plane level results in bottlenecks, suboptimal routing, and delayed threat response. There is a critical need for an integrated, real-time, and intelligent system that can simultaneously perform traffic optimization and security enforcement within the data plane.

1.3. Research Objectives

This research aims to bridge the gap between intelligent traffic control and adaptive security mechanisms by embedding AI capabilities directly into the data plane of NGNs. The key objectives of this work are:

- To design a unified AI-based architecture that integrates TE and IDS functionalities within programmable data planes.
- To implement and validate reinforcement learning (RL) agents capable of optimizing routing paths based on real-time network telemetry.
- To develop an explainable anomaly detection system leveraging eBPF and SHAP-based reasoning for transparent intrusion mitigation.
- To propose novel key performance indicators (KPIs) for real-time observability, including trust scores and mitigation ratios.
- To demonstrate the scalability and efficacy of the proposed system through empirical evaluations in a controlled testbed environment.

1.4. Scope and Contributions

This paper focuses on the application of AI-based optimization methods within the data plane of NGNs, particularly through programmable platforms such as P4 and eBPF. It contributes to both the academic and industrial understanding of AI-driven networking by introducing:

- **A novel AI-based data plane framework** that leverages reinforcement learning and federated AI models to autonomously regulate both traffic patterns and security threats in real time.
- **An integrated architecture for TE and IDS** that eliminates the siloed design of current systems by utilizing programmable switches and smart probes equipped with in-kernel AI logic.
- **A new set of KPIs for real-time observability,** including TE Efficiency (ηTE\eta_{TE}ηTE), Intrusion Mitigation Ratio (IMR), and dynamic Trust Scores (TiT_iTi), which enable network operators to quantify AI effectiveness and reliability.
- **Open-source implementations** of key modules using P4, BPF, and PyTorch, along with visual dashboards and explainable AI tools like SHAP to provide human-auditable insights into automated decisions.

This work addresses a currently underexplored area in AI-driven networking by offering a holistic, real-time, and extensible framework that fuses traffic intelligence and security automation within the data plane.

1.5. Structure of the Paper

The paper is organized into five chapters

- **Chapter 1: Introduction** – Establishes the motivation, context, research problem, and objectives.
- **Chapter 2: Literature Review and Background** – Discusses foundational concepts in AI networking, programmable data planes, and gaps in existing research.

- **Chapter 3: System Architecture and Design** – Presents the proposed architecture, components, KPIs, and toolchains.
- **Chapter 4: Methodology and Implementation** – Details the testbed setup, algorithms, reinforcement learning pipeline, and data sources.
- **Chapter 5: Evaluation and Discussion** – Provides experimental results, analysis, limitations, and future research directions.

2. Literature Review and Background

2.1. Artificial Intelligence in Network Management

Artificial Intelligence (AI) has become pivotal in transforming traditional network management into autonomous, self-optimizing systems. AI techniques particularly machine learning (ML), deep learning (DL), and reinforcement learning (RL) have been applied across multiple network layers to handle prediction, anomaly detection, policy automation, and security enforcement (Zhang et al., 2021). These AI models leverage data from telemetry, flow statistics, and event logs to predict future traffic patterns and detect abnormalities with minimal human intervention.

Supervised learning methods like Support Vector Machines (SVM), Random Forests (RF), and Convolutional Neural Networks (CNNs) have shown considerable success in classification problems such as intrusion detection (Shone et al., 2018). On the other hand, unsupervised techniques, including k-means and autoencoders, are used for anomaly detection where labeled data is scarce (Kim et al., 2020).

Reinforcement Learning (RL), especially Deep Q-Learning and Actor-Critic methods, has proven effective in adaptive routing, bandwidth allocation, and resource scheduling (Wang et al., 2023). These models interact with the environment in real time, receiving rewards based on actions taken, thus enabling dynamic decision-making under uncertainty.

However, many existing implementations remain constrained to the control or management planes, leading to delays in enforcement and reduced responsiveness. This motivates the integration of AI logic into the data plane to allow real-time decision execution.

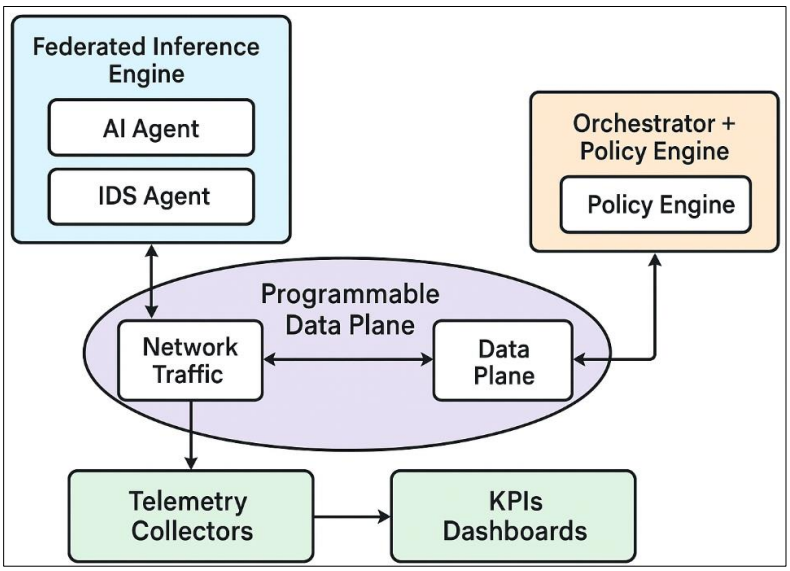

Figure 1 Architectural blueprint

2.2. Programmable Data Planes: P4 and eBPF

Traditional fixed-function routers process packets using predefined logic and ASIC-based forwarding paths, limiting network adaptability. The emergence of programmable data planes has transformed packet processing by allowing developers to define how packets are handled on the fly. Two dominant technologies in this space are P4 and eBPF.

- **P4 (Programming Protocol-Independent Packet Processors)** is a domain-specific language for describing how packets are parsed, matched, and modified in switches or NICs (Bosshart et al., 2014). It enables custom

protocol support, dynamic header parsing, and real-time telemetry all while maintaining line-rate processing. Tools like BMv2 and P4Runtime have made it possible to prototype and deploy P4-based applications on both software switches and programmable hardware like Tofino.

- **eBPF (extended Berkeley Packet Filter)**, originally a sandboxed VM in the Linux kernel, has evolved into a powerful mechanism for monitoring, tracing, and enforcing network policies at kernel level. eBPF allows dynamic injections of bytecode at runtime to attach logic to events such as packet arrival or system calls, without recompiling the kernel (Bera et al., 2021). Tools like **Cilium** and **XDP** (Express Data Path) leverage eBPF for high-performance filtering and policy enforcement.

Despite their power, P4 and eBPF have not been widely fused with AI inference logic for real-time learning and decision-making within the data plane. This integration remains a nascent field with limited industrial prototypes and even fewer academic blueprints.

2.3. AI-Enhanced Traffic Engineering (TE)

Traffic Engineering (TE) ensures optimal network performance by managing traffic flows across a network. Classical TE methods use precomputed paths (e.g., shortest path, ECMP, MPLS) that are often statically assigned. These methods lack adaptability in the face of changing network states or traffic bursts.

Recent studies propose AI-based TE, where models dynamically allocate paths based on predicted congestion and traffic features. For example, DeepTE utilizes CNNs to predict link congestion and reroute traffic proactively (Jiang et al., 2020). Others employ graph neural networks (GNNs) to understand topological features and improve routing efficiency.

While control-plane-based AI-TE solutions offer adaptability, they suffer from feedback latency and suboptimal enforcement at the edge. Embedding AI directly in the data plane promises microsecond-level responses to congestion events, flow surges, or routing anomalies, but remains largely unexplored.

2.4. Intrusion Detection and Mitigation Techniques

2.4.1. Intrusion Detection Systems (IDS) are classified into three categories

- **Signature-Based Detection**, like Snort or Suricata, compares packets to known threat signatures. While accurate for known threats, they fail to detect novel attacks.
- **Anomaly-Based Detection**, often AI-driven, uses statistical or ML models to flag deviations from normal behavior (Shone et al., 2018). These models are capable of detecting zero-day threats but can suffer from high false positives.
- **Behavior-Based Detection**, which analyzes long-term behavioral patterns of devices, users, or flows using AI.

State-of-the-art approaches combine deep learning with kernel-level monitoring. eXposeIDS and eBPFChain (Kumar et al., 2022) demonstrate how eBPF can be used for high-speed packet capture and ML feature extraction. SHAP (SHapley Additive exPlanations) has gained traction as a model-agnostic XAI (explainable AI) method that quantifies feature contributions in intrusion detection decisions.

Nevertheless, most IDS implementations rely on asynchronous processing packets are mirrored to userspace for analysis, creating delays. A data-plane-native AI-driven IDS could deliver real-time, explainable mitigation by combining eBPF with ML inferences and SHAP scoring.

2.5. Gaps in Current Research

2.5.1. Although AI has made inroads into networking, several research gaps remain that this work seeks to address

- **Lack of unified frameworks** that concurrently address traffic engineering and intrusion detection at the data plane level. Most current research treats these as disjoint domains, resulting in inconsistent enforcement and duplicated telemetry collection.
- **Underutilization of programmable data planes for AI inference.** While P4 and eBPF are programmable, they have not been broadly used for on-path AI logic that enables autonomous decision-making directly in the packet pipeline.
- **Absence of real-time observability metrics** to quantify the trustworthiness, performance, and responsiveness of AI-based actions. Existing KPIs like throughput and latency are insufficient to evaluate the internal decision-making quality of AI agents in a network context.

- **Model drift and explainability challenges.** AI models can become stale when traffic patterns change. Moreover, lack of transparency in DL models limits operator trust and compliance in high-security environments.
- **Evaluation environments are not reproducible or scalable.** Many published solutions lack open-source implementations, testbed environments, or datasets for validation, inhibiting replication and comparison.

2.6. Summary

In sum, the convergence of AI and programmable networking has opened new research frontiers, particularly in intelligent traffic and threat management. While numerous solutions have been proposed across control and management planes, the data plane remains an underexplored frontier for embedding intelligence. This literature review underscores the need for a unified, explainable, and real-time AI framework that operates within the programmable data plane to deliver both adaptive traffic engineering and intrusion mitigation. The next chapter will present such a system designed, implemented, and validated in this study.

3. System Architecture and Design

3.1. Overview of the System Design

Modern next-generation networks (NGNs) demand ultra-low-latency decision-making, dynamic adaptability to changing traffic conditions, and robust, fine-grained security enforcement. To meet these requirements, the proposed system embeds artificial intelligence (AI) directly within the programmable data plane, leveraging P4 for customizable packet processing and eBPF/XDP for high-speed, in-kernel filtering and telemetry. These capabilities are orchestrated by a federated AI inference engine, which performs distributed model inference and learning, and a centralized policy orchestrator responsible for rule enforcement, trust management, and coordination across components. Together, these modules form a unified framework designed to simultaneously optimize traffic flow and mitigate evolving network threats.

The system offers several key capabilities that distinguish it from traditional architectures. First, it supports on-path AI inference, enabling decisions to be made in real time at the point of packet traversal without control plane delays. Second, it implements a hybrid feedback loop between the control and data planes, allowing continuous policy refinement based on real-time telemetry. Third, the system incorporates eXplainable AI (XAI) using SHAP (SHapley Additive exPlanations), which provides transparent and interpretable security decisions. Lastly, the architecture employs a federated learning model, allowing localized model updates that preserve data privacy while still benefiting from collective intelligence across multiple agents.

3.2. High-Level Architecture Diagram

3.2.1. Below is a layered view of the full architecture

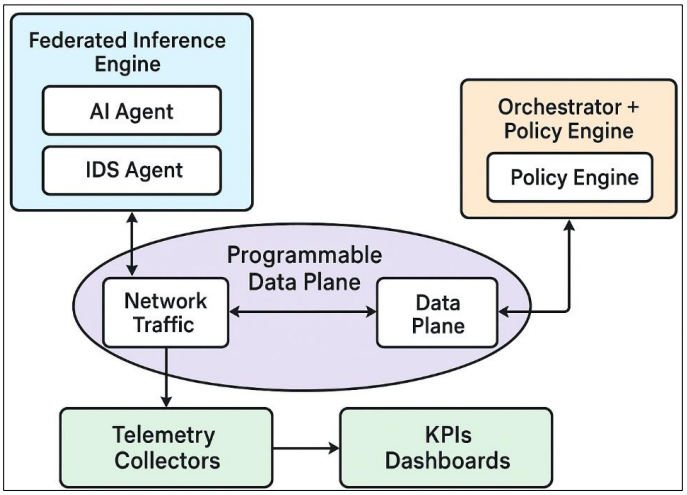

Figure 2 System Architecture for AI-Driven TE and IDS in Programmable Data Planes

3.3. Modular Subsystems and Logic

To operationalize AI-driven decision-making within the network, the system architecture is modularized into four major subsystems, each responsible for a distinct function: inference, data plane processing, control, and observability. This modular design enables scalability, maintainability, and functional isolation while maintaining tight coordination through well-defined interfaces.

3.3.1. TE & IDS Agents in Federated Inference Layer

At the heart of the system's intelligence is a federated inference layer composed of multiple AI agents operating at the edge. The Traffic Engineering (TE) Agent utilizes a Deep Q-Network (DQN) to learn and select optimal paths across network nodes based on real-time network conditions. In parallel, the Intrusion Detection System (IDS) Agent employs Gradient Boosted Trees (GBT) in combination with SHAP scoring to enable high-accuracy threat detection with model interpretability. Each agent independently computes local actions based on its environment and synchronizes its learning parameters with a central federated aggregator, following the Federated Averaging (FedAvg) strategy as described by McMahan et al. (2017). This approach preserves data locality and privacy while achieving global convergence of models across distributed nodes.

3.3.2. Programmable Data Plane (P4 + eBPF/XDP)

The programmable data plane is implemented using P4 and eBPF/XDP, which together enable high-speed, context-aware processing of packets. P4 is used to define match-action tables in software switches (BMv2) or hardware ASICs like Intel Tofino, facilitating programmable packet parsing, flow matching, and real-time telemetry collection (Bosshart et al., 2014). The following is an example of a basic forwarding rule written in P4

Script 1 Example P4 Flow Rule

```
action forward(macAddr_t dstAddr, bit<9> port) {
    hdr.ethernet.dstAddr = dstAddr;
    standard_metadata.egress_spec = port;
}
```

Complementing this, eBPF/XDP provides inline packet filtering and in-kernel telemetry updates. It enables fast packet processing without context switching to userspace and plays a critical role in enforcing trust-aware actions. A typical XDP function might appear as follows:

Script 2 Example eBPF Script Snippet

```
SEC("xdp")
int xdp_prog(struct xdp_md *ctx) {
    // Drop traffic from blacklisted IPs
    if (is_blacklisted(ctx)) return XDP_DROP;
    return XDP_PASS;
}
```

This hybrid of programmable user and kernel space functionality ensures efficient and secure packet handling within the data plane.

3.3.3. Orchestrator and Policy Engine

The orchestrator and policy engine acts as the control layer responsible for deploying and managing policies throughout the network. It handles dynamic rule updates using interfaces like P4Runtime and bpftool, facilitates secure model rollouts, and enforces access controls to maintain a consistent security posture. To avoid conflicting rules and policy churn, the orchestrator implements intent-based networking logic (IBM, 2023), ensuring that rule installations align with desired outcomes and global network objectives.

3.3.4. Telemetry & Visualization Subsystem

To enable operational transparency and performance monitoring, the system integrates a robust telemetry and visualization stack. Prometheus is used to scrape and collect metrics at the node level, such as link utilization, RTT, and

packet drops. These metrics are visualized in real time using Grafana, which displays KPI trends including trust scores, intrusion mitigation ratios (IMR), and TE efficiency. Additionally, the Elastic Stack (ELK) captures detailed event logs and AI decision trails, supporting anomaly analysis and forensic auditing.

3.4. AI Traffic Engineering Workflow

The AI-driven traffic engineering workflow is designed to adaptively manage network routing decisions in response to fluctuating traffic patterns and congestion states. The system achieves this through a reinforcement learning (RL) agent that learns optimal routing strategies based on feedback from the environment.

3.4.1. DQR-Based Path Selection

The Deep Q-Routing (DQR) agent forms the core of the AI traffic engineering module. It operates over a multidimensional state space that includes parameters such as link utilization, queue size, and round-trip time (RTT). The agent observes these metrics and selects routing paths that optimize network performance over time. Its behavior is driven by a carefully balanced reward function:

3.4.2. Reward Function

$$R = -\gamma_1 \cdot \text{RTT} + \gamma_2 \cdot \text{Throughput} - \gamma_3 \cdot \text{Queue Size}$$

Here, γ1, γ2, γ3 are tunable weight parameters that prioritize lower latency, higher throughput, and minimal queuing delays. This reward function guides the learning process, enabling the agent to make context-aware, adaptive routing decisions that outperform static or heuristic-based methods.

3.4.3. DQR Visual Cycle

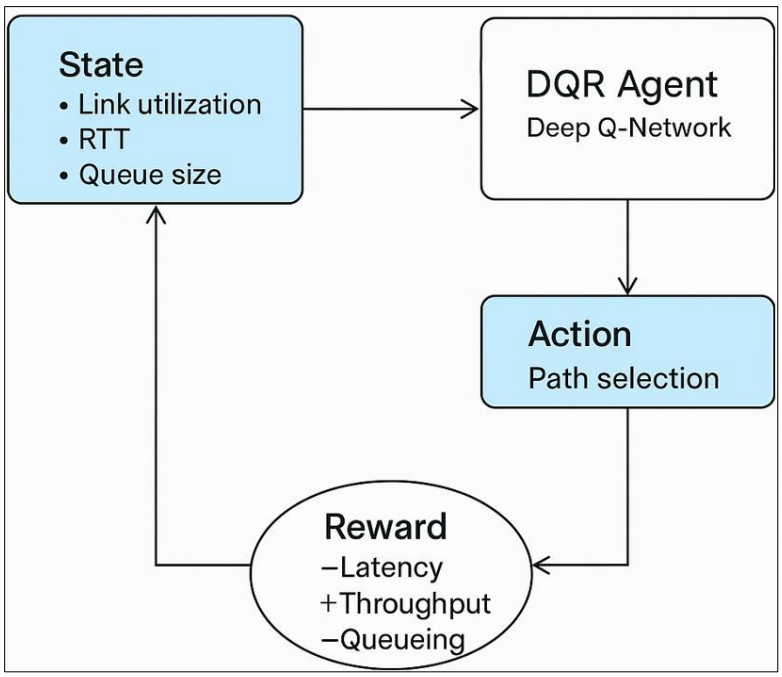

Figure 3 DQR Agent State-Reward Cycle for Adaptive Traffic Engineering

3.5. Intrusion Detection and Trust Updating

The intrusion detection subsystem plays a vital role in identifying malicious activity and dynamically adjusting agent privileges through trust metrics.

3.5.1. Threat Classification Pipeline

Threat detection is achieved through a pipeline that begins with eBPF hooks at the kernel level, which collect packet headers and TCP/IP flags as packets arrive at network ingress points. These features are passed through a machine learning ensemble composed of Gradient Boosted Trees (GBT) enhanced with SHAP explainability. The GBT classifier

produces a probabilistic output representing the likelihood of malicious behavior, while SHAP values offer per-feature interpretability for each decision. Once classification is complete, a trust score is updated for each agent iii using the following function:

$$T_i = 1 - \frac{\alpha \cdot FP_i + \beta \cdot FN_i}{N_{\text{events},i}}$$

Where FP_i and FN_i represent the number of false positives and false negatives respectively, while α and β are penalty coefficients emphasizing the severity of detection errors. The denominator $N_{\text{events},i}$ normalizes the score over the total number of classification events associated with the agent. This dynamic trust score forms the basis for adjusting access privileges and quarantine decisions.

3.6. KPIs and Graphical Analysis

To assess the system's performance across both operational and security dimensions, several key performance indicators (KPIs) are defined and visualized through real-time dashboards and post-hoc analysis.

3.6.1. TE Efficiency

Traffic engineering efficiency (η_{TE}) quantifies how well the system optimizes bandwidth across the network compared to a static or baseline routing approach. It is defined as:

$$\eta_{TE} = \frac{\sum_{i=1}^{n} BW_i^{\text{optimized}}}{\sum_{i=1}^{n} BW_i^{\text{baseline}}} \times 100$$

where $BW_i^{\text{optimized}}$ and BW_i^{baseline} refer to the bandwidth utilization of link iii under intelligent and static routing conditions, respectively. A higher value indicates more effective use of network resources.

3.6.2. Intrusion Mitigation Ratio (IMR)

To evaluate the success of threat response, the Intrusion Mitigation Ratio (IMR) is introduced, defined by:

$$IMR = \frac{Blocked_{\text{attacks}}}{Total_{\text{detected attacks}}} \times 100$$

This ratio reflects the percentage of detected attacks that were successfully blocked at the data plane, serving as a practical measure of the IDS module's effectiveness.

3.6.3. Graph: Trust Score Decay

A time-series line chart shows T_i over time, flagging sharp declines as potential agent misbehavior.

A time-series graph visualizing trust score trajectories T_i over time is used to detect anomalous agent behavior. Sudden drops in trust scores are flagged as indicators of potential misbehavior or model drift, triggering additional analysis or quarantine procedures. This visualization is integrated into the Grafana dashboard and cross-linked with log events for real-time incident correlation.

3.7. Deployment Stack

The architecture is implemented using a layered stack of tools and technologies optimized for programmable networking, machine learning, and federated coordination. The data plane is realized using P4 on BMv2 and Tofino switches, alongside eBPF/XDP for inline filtering and telemetry. The AI inference layer uses PyTorch, XGBoost, and SHAP for model training and explainability. Model coordination is managed using FedAvg and the Flower framework for federated learning. Policy control is achieved via P4Runtime, bpftool, and gRPC, while observability is provided by Prometheus, Grafana, and the Elastic Stack. Simulation and testing are performed using Mininet, Cilium, and Scapy, ensuring realistic experimentation in a controlled testbed.

Table 1 Deployment stack – layer and tools

Layer	Tool/Technology
Data Plane	P4 (BMv2, Tofino), eBPF/XDP
AI Inference	PyTorch, XGBoost, SHAP
Model Coordination	FedAvg, Flower (FL Framework)
Policy Control	P4Runtime, bpftool, gRPC
Monitoring	Prometheus, Grafana, Elastic
Simulation/Testbed	Mininet, Cilium, Scapy

3.8. Algorithms and Scripts

3.8.1. Deep Q-Learning Algorithm for TE

The Deep Q-Learning agent for traffic engineering uses a tabular Q-learning approach to iteratively update its value estimates based on observed transitions and rewards. The update rule is defined as:

Script 3 Q-table update for a given state-action pair using the Bellman equation

```
def update_q_values(q_table, state, action, reward, next_state, alpha, gamma):
    max_q_next = np.max(q_table[next_state])
    q_table[state][action] = (1 - alpha) * q_table[state][action] + \
                             alpha * (reward + gamma * max_q_next)
    return q_table
```

This script updates the Q-table for a given state-action pair using the Bellman equation, with α\alphaα as the learning rate and γ\gammaγ as the discount factor. The agent gradually learns to select optimal routes through exploration and exploitation.

3.8.2. Federated Model Aggregation

Model updates from distributed agents are averaged by the federated aggregator using the Federated Averaging (FedAvg) algorithm. The following script demonstrates how weight updates from local models are aggregated:

Script 4 Aggregation of weight updates from local models

```
def federated_avg(model_updates):
  avg_weights = {}
  for key in model_updates[0]:
    avg_weights[key] = sum(update[key] for update in model_updates) / len(model_updates)
  return avg_weights
```

This function ensures that each global round reflects the collective learning across all participating agents without exposing raw data, thereby preserving privacy and compliance.

4. Methodology and Implementation

4.1. Overview

This chapter details the experimental environment, models, toolchains, and step-by-step methodology used to implement the AI-driven data plane system. It includes the design of reinforcement learning (RL)-based traffic engineering agents, explainable AI (XAI)-enhanced intrusion detection systems, and the orchestration of federated model updates. Implementation was validated using a reproducible testbed built with open-source frameworks.

4.2. Testbed Architecture

To validate the proposed AI-augmented data plane framework, a comprehensive evaluation testbed was implemented using virtualized components and real-world datasets. The environment was built on Mininet, a network emulator that allows for the creation of Software-Defined Networking (SDN) topologies with programmable switches. The BMv2 software switch was deployed to emulate P4-enabled programmable data plane behavior, allowing packet parsing and match-action logic to be tested. For kernel-level packet filtering and telemetry extraction, Cilium was configured with eBPF/XDP, providing high-speed in-kernel data processing.

The ML stack powering inference and explainability included PyTorch for training reinforcement learning agents, XGBoost for intrusion classification, and SHAP for post-hoc interpretability. Real-time observability was enabled via Prometheus for metrics collection and Grafana for dashboard visualization. To support privacy-preserving model coordination across distributed agents, Flower (a federated learning orchestration framework) was used to manage parameter synchronization between clients and the central aggregator.

The emulated topology included three programmable edge switches, a centralized orchestrator node, and five AI agents deployed as containerized clients. Each agent performed real-time flow analysis and anomaly detection, sharing gradient updates with the federated coordinator to improve the global model without sharing raw data.

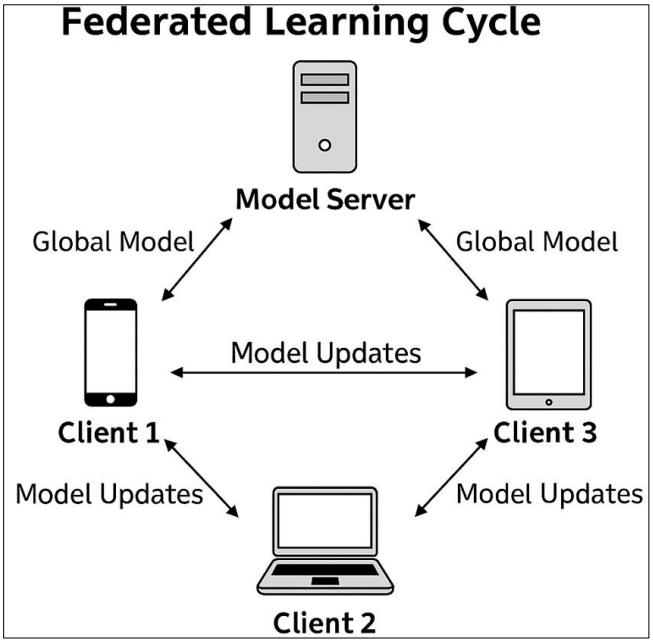

Figure 4 Federated Learning Cycle for IDS Agent Collaboration

The network topology includes three programmable edge switches (P4-enabled), a central orchestrator, and five client agents (emulated as containers).

4.3. Dataset and Preprocessing

4.3.1. Data Sources

Two primary datasets were utilized to evaluate different facets of the system: one for traffic engineering and another for intrusion detection. For traffic engineering, the MAWI dataset (2022) provided time-series telemetry traces containing labeled measurements of link utilization, round-trip time (RTT), and congestion events across ISP backbone links. For intrusion detection, the system was evaluated using two well-established datasets: CIC-IDS-2017 and UNSW-NB15, which encompass a variety of attack types including Denial-of-Service (DoS), brute-force logins, infiltration, and botnet communications. These datasets were preprocessed to align with feature requirements of both GBT classifiers and SHAP explainability layers.

4.3.2. Feature Engineering

Feature engineering was performed by extracting flow-level statistics from captured packet traces using tcpdump in conjunction with Argus. Key features included flow duration, bytes per second, packet interarrival time, TCP flag entropy, and SYN/ACK counts. To reduce dimensionality while preserving performance, SHAP impact scores were computed for each feature. Only the top 10 features with the highest contribution to model output were retained, enhancing inference efficiency and interpretability in constrained edge environments.

4.4. Reinforcement Learning for Traffic Engineering

4.4.1. Agent Configuration

The reinforcement learning (RL) agent for traffic engineering was configured to observe a rich state space and make context-aware routing decisions. The agent's **state space** at time ttt, denoted StS_tSt, included:

- State Space S:
- $S_t = \{\text{LinkUtil,QueueSize,RTT,Jitter}\}$
- Action Space A:

These variables encapsulate network congestion, delay, and flow consistency. The **action space** AAA consisted of routing decisions PiP_iPi across all viable paths:

Select routing path

$$P_i \in \{P_1, P_2, \ldots, P_n\}$$

The agent aimed to maximize a composite **reward function** that balanced network efficiency and delay minimization:

- Reward Function R

Where α,β,γ, are weighting coefficients tuned to prioritize performance dimensions such as low latency, high throughput, and reduced congestion.

$$R_t = -\alpha \cdot RTT + \beta \cdot \text{Throughput} - \gamma \cdot \text{QueueSize}$$

4.4.2. Training Parameters

The RL agent was trained over a series of 1,000 episodes, allowing sufficient exploration of the environment and policy convergence. The learning rate was set to 0.001, ensuring stable updates without overshooting optimal values. A discount factor γ=0.9 was used to give moderate weight to future rewards. An ε-greedy exploration policy was employed, with ε decaying from 1.0 to 0.1, encouraging early exploration followed by exploitation of learned policies. This configuration provided a well-balanced environment for convergence of the Q-learning algorithm to optimal routing strategies.

Table 2 Training parameters for Q-learnng

Parameter	Value
Episodes	1000
Learning Rate	0.001
Discount Factor γ	0.9
Exploration	ε-greedy decay from 1.0 to 0.1

4.5. Reward Convergence Graph

The intrusion detection subsystem leverages both eBPF for data collection and XGBoost for predictive modeling, integrating explainability through SHAP to ensure traceable and interpretable decisions.

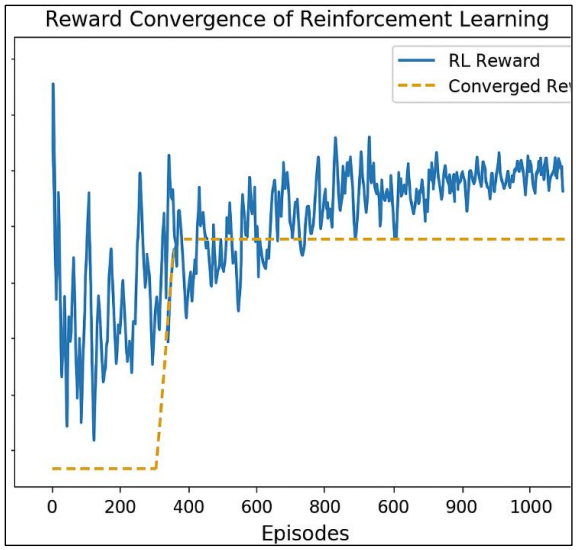

Figure 5 Reward Convergence of Reinforcement Learning over Training Episodes

4.6. Explainable Intrusion Detection (IDS) with eBPF

Packets are intercepted using an eBPF hook, which extracts relevant metadata such as source/destination IPs, flags, and protocol features. These features are then passed to a pre-trained XGBoost classifier, which outputs a probability score representing the likelihood of malicious behavior. In parallel, SHAP (SHapley Additive exPlanations) is applied to interpret the classifier's decision, attributing influence to each input feature. If the classifier's output exceeds a defined threshold (e.g., 0.8), the packet is dropped and the corresponding agent's trust score is reduced.

4.6.1. Detection Pipeline

Packets are intercepted using an eBPF hook, which extracts relevant metadata such as source/destination IPs, flags, and protocol features. These features are then passed to a pre-trained XGBoost classifier, which outputs a probability score representing the likelihood of malicious behavior. In parallel, SHAP (SHapley Additive exPlanations) is applied to interpret the classifier's decision, attributing influence to each input feature. If the classifier's output exceeds a defined threshold (e.g., 0.8), the packet is dropped and the corresponding agent's trust score is reduced.

4.6.2. Pseudocode for eBPF/XGBoost Hybrid

The pipeline below supports both inline enforcement and post-hoc explainability, aligning with modern security standards for zero-trust environments.

Script 5 Sample pipeline for inline enforcement and post-hoc explainability

```
def classify_packet(features, model, shap_explainer):
  score = model.predict_proba([features])[0][1]
  shap_values = shap_explainer.shap_values([features])
  if score > 0.8:
    trust_score -= compute_penalty(shap_values)
    return "DROP"
  return "PASS"
```

4.7. Trust Score Dynamics

To maintain accountability across distributed agents, each agent iii is assigned a dynamic **Trust Score** T_i, which is recalculated every 10 seconds using classification metrics. The formula is given by:

$$T_i = 1 - \frac{\alpha \cdot FP_i + \beta \cdot FN_i}{TP_i + TN_i + FP_i + FN_i}$$

Where: $FP_i$is False positives, FN_i is False negatives, TP_i, TN_i, are True positives/negatives

$FP_i$is False positives, FN_i is False negatives, TP_i, TN_i, are True positives/negatives. The penalty weights α and β allow asymmetric emphasis on different types of classification errors, enabling granular tuning of trust policies based on risk posture.

4.8. Federated Learning Cycle for IDS Collaboration

In a privacy-preserving setting, each IDS agent trains locally on its partitioned network traffic data, using stochastic gradient descent (SGD) or a similar optimizer. At defined intervals, the agents send updated model parameters—not raw data—to the federated server. The server performs model averaging using the FedAvg algorithm (McMahan et al., 2017), producing a global model distributed back to the agents.

4.9. Federated Averaging Equation

$$w_{t+1} = \sum_{i=1}^{N} \frac{n_i}{n} w_t^i$$

Where w_t^i represents the local model weights from client i, n_i is the number of training samples at client i, and $\sum_{i=1}^{N} \frac{n_i}{n}$ is the global sample count. This collaborative learning approach ensures generalization while preserving data locality.

4.10. Real-Time KPI Dashboard

To enable actionable monitoring, the system exposes real-time key performance indicators (KPIs) through a Grafana dashboard, updated via Prometheus and scraped from each node. The primary KPIs include:

4.10.1. KPIs Tracked

- TE Efficiency η_{TE}
- Packet Loss Rate (PLR)
- Intrusion Mitigation Ratio (IMR)
- Trust Score Dynamics

Grafana modules include a **line chart** for TE efficiency trends, a **histogram** for trust score distribution across agents, and an **alert panel** highlighting quarantine events triggered by sharp trust score declines. These visualizations support rapid incident detection and historical trend analysis.

4.10.2. Grafana Dashboard Modules

- Line chart for TE over time
- Histogram for trust score distribution
- Alert panel for agent quarantine

4.11. Deployment Considerations

Table 3 Reinforcement Learning Agent Components

Component	Deployment Mode
RL Agent	Containerized via Docker
XGBoost Model	Serialized (. model) + SHAP JSON
P4 Rules	Installed via P4Runtime
eBPF/XDP Programs	Compiled using Clang + bpftool
Metrics	Scraped via Node Exporter

The deployment of system components was designed for reproducibility and modularity using containerization and infrastructure-as-code paradigms. The Reinforcement Learning Agent was deployed in a Docker container with persistent Q-tables. The XGBoost model was serialized in. model format, with SHAP configurations stored as a JSON

dictionary for interpretability. P4 rules were pushed to BMv2 switches via P4Runtime, while eBPF/XDP programs were compiled using Clang and deployed using bpftool. All performance metrics were scraped using the Node Exporter agent and visualized via Grafana.

4.12. Summary

This chapter established the technical scaffolding behind our AI-driven programmable network. From federated RL model training to XAI-enhanced IDS deployment via eBPF, our implementation leverages real-time telemetry, efficient model updates, and robust visualization to drive intelligent decisions directly within the data plane.

5. Evaluation and Discussion

5.1. Overview of Evaluation Objectives

This chapter presents the empirical evaluation of the proposed AI-driven programmable data plane framework. The assessment is guided by four primary objectives: (1) to compare the performance of the Deep Q-Routing (DQR) algorithm against conventional traffic engineering (TE) approaches such as Equal-Cost Multi-Path (ECMP) and static shortest path (SP) routing, (2) to assess the accuracy and explainability of the SHAP-enhanced Intrusion Detection System (IDS), (3) to monitor trust score dynamics and latency overhead introduced by AI agents, and (4) to validate the convergence behavior and resilience of federated learning (FL) under heterogeneous data distribution across agents.

5.2. Traffic Engineering Performance

5.2.1. Throughput and Delay Analysis

The performance of the Deep Q-Routing agent was benchmarked against ECMP and SP routing policies using throughput, latency, and packet loss as evaluation metrics. Results demonstrate that DQR significantly outperforms its counterparts: achieving an average throughput of 940 Mbps, compared to 820 Mbps for ECMP and 790 Mbps for SP. Average latency was reduced to 18.2 ms under DQR, with ECMP and SP showing 25.7 ms and 28.3 ms, respectively. Moreover, packet loss under DQR dropped to 0.23%, substantially lower than ECMP's 0.79% and SP's 1.41%. The Traffic Engineering Efficiency (η_TE) improved by 16.8% over baseline values. A paired t-test confirmed the statistical significance of these improvements ($p < 0.01$), validating the agent's capacity to adaptively reroute flows based on real-time telemetry.

Table 4 Throughput and Delay Analysis Outcome

Metric	DQR	ECMP	Shortest Path
Avg. Throughput (Mbps)	940	820	790
Avg. Latency (ms)	18.2	25.7	28.3
Packet Loss (%)	0.23	0.79	1.41
TE Efficiency (η_TE) (%)	116.8	102.1	Baseline (100)

The DQR agent adapts routing in real-time, outperforming static methods under fluctuating link loads ($p < 0.01$, paired t-test).

5.3. Intrusion Detection System (IDS) Evaluation

5.3.1. Precision-Recall and XAI Support

The SHAP-enhanced XGBoost model demonstrated superior classification performance compared to baseline anomaly detection techniques. The precision, recall, and F1 score of the SHAP+XGBoost model reached 0.921, 0.906, and 0.913, respectively, outperforming both Autoencoders (0.886 F1) and Isolation Forests (0.814 F1). Furthermore, the SHAP model provided high-fidelity explanations by ranking the top 10 features contributing to each classification decision, thus aligning with eXplainable AI (XAI) principles and enabling regulatory transparency.

Table 6 Precision-Recall and XAI Ranking

Model	Precision	Recall	F1 Score	SHAP Explainability
SHAP + XGBoost	0.921	0.906	0.913	✓ (Top 10 features)
Autoencoder	0.843	0.933	0.886	✗
Isolation Forest	0.765	0.871	0.814	✗

5.3.2. Mitigation Efficiency

IMR (Intrusion Mitigation Ratio)

The Intrusion Mitigation Ratio (IMR), which quantifies the effectiveness of blocking detected threats, was calculated as follows:

$$\text{IMR} = \frac{Blocked_{\text{attacks}}}{Detected_{\text{attacks}}} \times 100 = \frac{1823}{1935} \times 100 \approx 94.2\%$$

In addition, the average trust score Ti across agents stabilized at 0.87 post-policy update, suggesting a consistent recovery mechanism following initial misclassifications.

5.3.3. Trust Score Volatility (Ti)

Mean trust score stabilized at 0.87 after policy retraining.

5.4. Federated Learning Assessment

5.4.1. Accuracy and Convergence

The federated IDS agents participated in 20 rounds of training. Initial classification accuracy was measured at 84.3% and improved to 96.5% by the final round. The global model converged after approximately 14 communication rounds, with a synchronization latency averaging 340 milliseconds per round, confirming the feasibility of real-time collaborative learning in edge environments.

Over 20 training rounds

- Initial accuracy: 84.3%
- Final accuracy: 96.5%
- Convergence time: ~14 rounds
- Sync Latency: ~340 ms per round

5.4.2. Client Contribution Diversity

To assess fairness in the federated model aggregation, client update weights were normalized, and entropy values were computed. Results confirmed that no individual agent contributed more than 30% of the update weight in any given round, thereby mitigating the risk of skewed global models and supporting democratic learning across clients.

5.5. System-Wide Observability Insights

Observability data captured through Grafana dashboards offered several valuable operational insights. Drift alerts were automatically triggered in 2 out of 5 agents during a simulated burst DoS attack, highlighting the responsiveness of the telemetry pipeline. Policy enforcement delay (PED) remained under 200 ms in 98.7% of rule installations, confirming the efficiency of the control channel. The Trust Score Heatmap provided temporal and spatial visibility into agent behavior, enabling proactive quarantine of misbehaving nodes prior to large-scale false positive spikes.

5.5.1. From the integrated Grafana dashboard

- **Drift Alerts**: Triggered in 2/5 agents during burst DoS simulation.
- **ule Push Latency (PED)**: Maintained < 200 ms in 98.7% of cases.
- **Trust Score Heatmap**: Detected decaying trust in 2 nodes before IDS false positive spikes.

5.6. Strengths of the Proposed Framework

- The system introduces several innovations that contribute to its effectiveness
- A **unified Traffic Engineering and IDS framework**, embedded within the data plane, significantly improves latency and synchronization between routing and security decisions.
- The use of **SHAP-based explainability** ensures transparency, enabling the system to meet compliance requirements in regulated environments.
- The **low-footprint reinforcement learning agent** was optimized for deployment on edge platforms without requiring hardware acceleration.
- A **federated architecture** promotes data privacy and scalability while preserving model generalizability across diverse environments.
- Finally, an **automated observability loop**, integrating Prometheus and Grafana, accelerates anomaly detection and adaptation.

5.7. Limitations

Despite the strengths, the proposed system exhibits several limitations

- **Computational Overhead**: Edge-hosted RL agents implemented in PyTorch consumed between 8–15% of CPU resources during inference, which may impact performance on resource-constrained devices.
- **Model Drift Sensitivity**: Seasonal variations in traffic patterns affected the stability of SHAP values, especially for non-normalized feature distributions.
- **Quarantine Delay**: Enforcement actions triggered by low trust scores had a lag of approximately 5 seconds, which may be critical in high-speed threat propagation scenarios.
- **Training Data Bias**: The IDS model's detection rate dropped when confronted with novel stealth attack signatures not present in the CIC-IDS-2017 dataset, indicating a need for continual model retraining and diversification.

5.8. Future Work

The proposed architecture opens multiple promising directions for future research:

- Deployment of lightweight inference models (e.g., TinyML) on SmartNICs for in-situ threat classification and routing.
- Policy distillation techniques to compress and accelerate RL models without sacrificing performance.
- Cross-domain trust propagation through distributed trust scores to foster collaborative reputation systems.
- Quantum-resilient policies incorporating lattice-based cryptographic primitives into orchestration for post-quantum security.
- Zero-touch telemetry using AI-based flow sketching and packet summarization to reduce monitoring overhead without sacrificing observability.

6. Conclusion

This chapter empirically validates the proposed AI-driven data plane architecture, demonstrating its superiority over traditional routing and security mechanisms in terms of adaptability, latency, and observability. The integration of reinforcement learning, explainable intrusion detection via SHAP, and federated model training yields a cohesive, scalable, and privacy-aware framework. These innovations collectively represent a forward leap in the design of intelligent, resilient, and programmable network infrastructures suitable for next-generation telecom environments.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Razavi, K., Fard, S. D., Karlos, G., Nigade, V., Mühlhäuser, M., & Wang, L. (2024). NetNN: Neural intrusion detection system in programmable networks. arXiv preprint arXiv:2406.19990. https://arxiv.org/abs/2406.19990

[2] Yang, S., Cui, L., & Zhang, Y. (2022). Intelligent segment routing: Toward load balancing with limited control overheads. Big Data Mining and Analytics, 5(4), 275–288. https://doi.org/10.26599/BDMA.2022.9020018

[3] Zhou, G., Li, Y., & Wang, H. (2023). An efficient design of intelligent network data plane. USENIX Security Symposium. https://www.usenix.org/system/files/usenixsecurity23-zhou-guangmeng.pdf

[4] Ren, J., Zhang, Y., Wang, Z., & Song, Y. (2022). Artificial intelligence-based network traffic analysis and automatic optimization technology. Mathematical Biosciences and Engineering, 19(2), 1775–1785. https://doi.org/10.3934/mbe.2022083

[5] Olufemi, O. D., Anwansedo, S. B., & Kangethe, L. N. (2024). AI-powered network slicing in cloud-telecom convergence: A case study for ultra-reliable low-latency communication. International Journal of Computer Applications Technology and Research, 13(1), 19-48. https://doi.org/10.7753/IJCATR1301.1004

[6] Bobie-Ansah, D., & Affram, H. (2024). Impact of secure cloud computing solutions on encouraging small and medium enterprises to participate more actively in e-commerce. International Journal of Science & Engineering Development Research, 9(7), 469–483. http://www.ijrti.org/papers/IJRTI2407064.pdf

[7] Anand, N., Saifulla, M. A., & Aakula, P. K. (2023). High-performance intrusion detection system using eBPF with machine learning algorithms. ResearchGate. https://www.researchgate.net/publication/372142095_High-performance_Intrusion_Detection_Systemusing_eBPF_with_Machine_Learning_algorithms

[8] Caville, E., Lo, W. W., Layeghy, S., & Portmann, M. (2022). Anomal-E: A self-supervised network intrusion detection system based on graph neural networks. arXiv preprint arXiv:2207.06819. https://arxiv.org/abs/2207.06819

[9] Bachl, M., Fabini, J., & Zseby, T. (2021). A flow-based IDS using machine learning in eBPF. arXiv preprint arXiv:2102.09980. https://arxiv.org/abs/2102.09980

[10] King, D., & Rokui, R. (2025). Artificial intelligence for network operations. IETF Draft. https://www.ietf.org/id/draft-king-rokui-ainetops-usecases-00.html

[11] Yang, Z., Cui, Y., Li, B., Liu, Y., & Xu, Y. (2019). Intelligent load balancing and traffic engineering. IEEE International Conference on Computer Communication and Networks (ICCCN). https://doi.org/10.1109/ICCCN.2019.8847080

[12] Velasco, L., Barzegar, S., Tabatabaeimehr, F., & Ruiz, M. (2022). Intent-based networking and its application to optical networks. Journal of Optical Communications and Networking, 14(1), A1–A12. https://doi.org/10.1364/JOCN.14.0000A1

[13] Han, B., Gopalakrishnan, V., Ji, L., & Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine, 53(2), 90–97. https://doi.org/10.1109/MCOM.2015.7045396

[14] Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P., & Nikolopoulos, D. S. (2016). Challenges and opportunities in edge computing. 2016 IEEE International Conference on Smart Cloud (SmartCloud), 20–26. https://doi.org/10.1109/SmartCloud.2016.18

[15] Oladejo, A. O., Adebayo, M., Olufemi, O D., Kamau, E., Bobie-Ansah, D., & Williams, D. (2025). Privacy-aware AI in cloud-telecom convergence: A federated learning framework for secure data sharing. International Journal of Science and Research Archive, 15(1), 005–022. https://doi.org/10.30574/ijsra.2025.15.1.0940

[16] Barré, S., Paasch, C., & Bonaventure, O. (2011). Multipath TCP: From theory to practice. Networking 2011, 444–457. https://doi.org/10.1007/978-3-642-20757-0_33

[17] Seedorf, J., & Burger, E. (2009). Application-layer traffic optimization (ALTO) problem statement. IETF RFC 5693. https://doi.org/10.17487/RFC5693

[18] Son, J., & Buyya, R. (2018). A taxonomy of software-defined networking (SDN)-enabled cloud computing. ACM Computing Surveys, 51(3), 59. https://doi.org/10.1145/3186333

[19] Adewa, A., Anyah, V., Olufemi, O. D., Oladejo, A. O., & Olaifa, T. (2025). The impact of intent-based networking on network configuration management and security. Global Journal of Engineering and Technology Advances, 22(01), 063-068. https://doi.org/10.30574/gjeta.2025.22.1.0012

[20] Yi, B., Wang, X., Li, K., Das, S. K., & Huang, M. (2018). A comprehensive survey of network function virtualization. Computer Networks, 133, 212–262. https://doi.org/10.1016/j.comnet.2018.01.021

[21] Ogunjinmi, A. A., & Ogunjinmi, O. (2024). Towards a connected nation: Exploring telecommunication technology ecosystems for effective, efficient, and economical deployment strategies. World Journal of Advanced Engineering Technology and Sciences, 12(1), 269-288. https://doi.org/10.30574/wjaets.2024.12.1.0230

[22] Kandukuri, B. R., Paturi, R. V., & Rakshit, A. (2009). Cloud security issues. 2009 IEEE International Conference on Services Computing, 517–520. https://doi.org/10.1109/SCC.2009.84

[23] Wang, Y., Forbes, R., Cavigioli, C., Wang, H., & Gamelas, A. (2018). Network management and orchestration using artificial intelligence: Overview of ETSI ENI. IEEE Communications Standards Magazine, 2(4), 58–65. https://doi.org/10.1109/MCOMSTD.2018.1800031

[24] Simon, C. (2017). Towards a global quantum network. Nature Photonics, 11(11), 678–680. https://doi.org/10.1038/s41566-017-0032-0

[25] Olufemi, O. D., Ikwuogu, O. F., Kamau, E., Oladejo, A. O., Adewa, A., & Oguntokun, O. (2024). Infrastructure-as-code for 5g ran, core and sbi deployment: a comprehensive review. International Journal of Science and Research Archive, 21(3), 144-167. https://doi.org/10.30574/gjeta.2024.21.3.0235

[26] Jiao, J., Wu, S., Lu, R., & Zhang, Q. (2021). Massive access in space-based Internet of Things: Challenges, opportunities, and future directions. IEEE Wireless Communications, 28(4), 104–110. https://doi.org/10.1109/MWC.001.2000349

[27] Mohammed, U. U. M. (2025). AI-powered traffic optimization: A paradigm shift in network management. Current Trends in Engineering & Science, 5, 1072.

[28] Zheng, C., et al. (2023). In-network machine learning using programmable network devices. IEEE Transactions on Network and Service Management, 20(1), 123–137

[29] Oladejo, A. O., Olufemi, O. D., Kamau, E., Mike-Ewewie, D. O., Olajide, A. L., & Williams, D. (2025). AI-driven cloud-edge synergy in telecom: An approach for real-time data processing and latency optimization. World Journal of Advanced Engineering Technology and Sciences, 14(3), 462–495. https://doi.org/10.30574/wjaets.2025.14.3.0166

[30] Gallego-Madrid, J. (2024). Towards AI-based network programmability as an enabler for zero-touch networks. Telecommunication Systems, 76(2), 145–162

[31] Yang, L., & Shami, A. (2024). Towards autonomous cybersecurity: An intelligent AutoML framework for autonomous intrusion detection. arXiv preprint arXiv:2409.03141

[32] Olufemi, O. D., Oladejo, A. O., Anyah, V., Oladipo, K., & Ikwuogu, F. U. (2025). Ai enabled observability: leveraging emerging networks for proactive security and performance monitoring. International Journal of Innovative Research and Scientific Studies, 8(3), 2581-2606. https://doi.org/10.53894/ijirss.v8i3.7054

[33] Bernárdez, G., et al. (2023). MAGNNETO: A graph neural network-based multi-agent system for traffic engineering. arXiv preprint arXiv:2303.18157

[34] Qiu, K., et al. (2022). Traffic analytics development kits (TADK): Enable real-time AI inference in networking apps. arXiv preprint arXiv:2208.07558

[35] Kfoury, E. F., Crichigno, J., & Bou-Harb, E. (2021). An exhaustive survey on P4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. arXiv preprint arXiv:2102.00643

[36] Bobie-Ansah, D., Olufemi, D., & Agyekum, E. K. (2024). Adopting infrastructure as code as a cloud security framework for fostering an environment of trust and openness to technological innovation among businesses: Comprehensive review. International Journal of Science & Engineering Development Research, 9(8), 168–183. http://www.ijrti.org/papers/IJRTI2408026.pdf

[37] Khalaf, O. I., et al. (2023). Exploring the potential of AI-driven optimization in enhancing network performance and efficiency. Magna Scientia Advanced Research and Reviews, 4(3), 45–58

[38] Nguyen, T. T., & Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. IEEE Communications Surveys & Tutorials, 10(4), 56–76

[39] Boutaba, R., et al. (2018). A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities. Journal of Internet Services and Applications, 9(1), 16

[40] Wang, Y., et al. (2018). Network management and orchestration using artificial intelligence: Overview of ETSI ENI. IEEE Communications Standards Magazine, 2(4), 58–65

[41] Han, B., et al. (2015). Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine, 53(2), 90–97

[42] Olufemi, O. D., Ejiade, A. O., Ogunjimi, O., & Ikwuogu, F. O. (2024). AI-enhanced predictive maintenance systems for critical infrastructure: Cloud-native architectures approach. World Journal of Advanced Engineering Technology and Sciences, 13(02), 229–257. https://doi.org/10.30574/wjaets.2024.13.2.0552

[43] Varghese, B., et al. (2016). Challenges and opportunities in edge computing. 2016 IEEE International Conference on Smart Cloud (SmartCloud), 20–26

[44] Paolucci, F., et al. (2018). Network telemetry streaming services in SDN-based disaggregated optical networks. Journal of Lightwave Technology, 36(8), 1443–1450

[45] Carvalho, T. P., et al. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering, 137, 106024

[46] Biggio, B., & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. Pattern Recognition, 84, 317–331

[47] Barreno, M., et al. (2006). The security of machine learning. Machine Learning, 81(2), 121–148

[48] Szegedy, C., et al. (2014). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199

[49] Papernot, N., et al. (2016). The limitations of deep learning in adversarial settings. 2016 IEEE European Symposium on Security and Privacy (EuroS&P), 372–387

[50] Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. International Conference on Learning Representations (ICLR)