ITUJournal
Future and evolving technologies

ITU Journal on Future and Evolving Technologies, Volume 7, Issue 2, June 2026

ARTICLE

Towards zero-touch network orchestration: An NWDAF-centered standards-compliant closed-loop approach

***Fatemeh Shafiei Ardestani*[1], *Noura Limam*[1], *Raouf Boutaba*[1]**

[1] *University of Waterloo, Canada*

Corresponding author: Fatemeh Shafiei Ardestani, f2shafie@uwaterloo.ca

The transition toward zero-touch and AI-native network operations requires closing the loop between data collection, analytics, and automated control across distributed network functions. While the 3GPP Network Data Analytics Function (NWDAF) provides a standardized foundation for network intelligence, its practical realization as a network automation enabler remains largely unexplored. In this paper, we present a standards-compliant framework for closed-loop network automation that elevates NWDAF from an analytics service to an active component in runtime orchestration. Our design integrates (i) real-time user-plane observability via a novel implementation of the User Plane Function (UPF) Event Exposure Service, (ii) actionable analytics with modular engines and Machine Learning (ML) lifecycle management, and (iii) automated action and policy enforcement through control-plane functions, forming a complete data analytics action pipeline.
We implement the proposed framework, integrate it with an open-source 5G core, and evaluate its end-to-end performance under realistic workloads. Our results show that fine-grained user-plane telemetry can be collected with minimal overhead while enabling responsive closed-loop control at runtime. Through two representative use cases, anomaly detection with automated mitigation and congestion-aware quality of service adaptation, we demonstrate that NWDAF-driven control can effectively translate analytics into actionable policies while preserving system scalability. This work provides the first comprehensive evidence that standards-aligned NWDAF can serve as a practical foundation for zero-touch network automation, bridging the gap between 3GPP-defined analytics and runtime orchestration.

Keywords: Closed-loop network orchestration, ML-driven analytics, NWDAF, observability, policy enforcement.

1. INTRODUCTION

The evolution of 5G and emerging 6G networks is driving a shift in network operations from manual configurations toward zero-touch, AI-native automation, a vision reinforced by initiatives such as the ETSI Zero-touch network and Service Management (ZSM) [1]. This transition is motivated by the growing scale, heterogeneity, and complexity of networks, where diverse services with conflicting requirements must be managed across distributed cloud-native infrastructures. In this setting, network orchestration must evolve from static provisioning and configuration toward continuous, closed-loop automation, in which network telemetry is collected, analyzed, and used to drive control decisions that are enforced at runtime.

The 3rd Generation Partnership Project (3GPP) introduced the Network Data Analytics Function (NWDAF) as a standardized mechanism to enable data-driven network operations and management. NWDAF provides a service-based interface for collecting network data, generating analytics, and exposing insights to other network, application, or management and orchestration functions (Fig. 1).

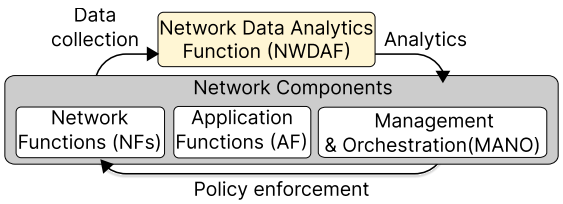


Figure 1 – NWDAF-enabled closed loop network automation

Existing work [2, 3, 4, 5, 6, 7] has explored isolated components of the data analytics action pipeline but has yet to integrate standards-compliant data collection, real-time analytics, and automated policy enforcement into a unified system. Specifically, prior work exhibits three limitations:

- **Lack of standards-compliant observability:** Most work relies on passive packet capture or indirect control-plane exports, rather than leveraging the 3GPP-defined Event Exposure Service (EES) for real-time, structured data collection.
- **Lack of analytics-to-action integration:** Existing solutions focus on generating analytics but stop short of translating them into automated control-plane actions, leaving the loop open.
- **Missing lifecycle management for analytics models:** The dynamic nature of network environments requires continuous model evolution, yet ML lifecycle management (MLOps) is largely absent from NWDAF implementations.

Prior work also offers limited quantitative evaluation of the resource costs of data collection, analytics generation, and reporting, as well as of NWDAF's effectiveness in closed-loop network orchestration.

This paper evaluates whether standards-compliant NWDAF can support practical closed-loop network orchestration, and identifies the mechanisms required to make this feasible. To this end, we design and implement an NWDAF-centered framework that combines standards-compliant data collection, analytics, and policy enforcement. We integrate our framework with a major open-source 5G core network implementation (Open5GS [8]), validate it through two use cases, and release our code as open-source [1]. Our key contributions are as follows:

- **A standards-compliant architecture for closed-loop network orchestration.** We design the first end-to-end framework that integrates 3GPP-compliant data collection, analytics generation, and policy enforcement into a unified closed-loop system. Our architecture operationalizes NWDAF beyond analytics, positioning it as an active component in runtime network control.
- **Realization of user-plane observability via UPF event exposure service.** We implement the first practical realization of the UPF Event Exposure Service (UPF EES), enabling fine-grained, per-flow telemetry collection directly from the user plane. We design a lightweight data collection pipeline that preserves packet processing performance while exposing rich observability for analytics-driven control.
- **Integration of analytics with automated policy enforcement.** We bridge the gap between NWDAF analytics and control-plane actions by extending SMF and PCF to act as NWDAF consumers. This enables fully automated closed-loop actuation, including session-level mitigation and runtime QoS adaptation, demonstrating how analytics can be translated into enforceable network policies.
- **ML lifecycle management for in-network analytics.** We implement a standards-aligned ML model provision service integrated with MLflow, enabling model discovery, versioning, and runtime provisioning. This provides a practical realization of MLOps within NWDAF-driven systems.
- **End-to-end evaluation on a 5G testbed.** We provide a detailed evaluation of the complete closed-loop pipeline, including data collection overhead, analytics performance, and actuation latency. Our results show that fine-grained telemetry can be collected with minimal overhead, while enabling responsive and scalable closed-loop control.

This paper builds on and extends our preliminary work [9, 10].

2. BACKGROUND AND RELATED WORK

In this section, we review the 3GPP core network functions and components relevant to this work (Section 2.1). We then introduce NWDAF and the services it uses (Section 2.4) and, finally, discuss prior work on NWDAF (Section 2.5) to highlight the gap this paper addresses.

2.1 Overview of the 3GPP core network functions

To achieve the flexibility and agility required to support a wide range of communication services and a growing number of users, the mobile network adopts Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies [11, 12], decoupling network functionalities from hardware and enabling more flexible deployment on shared, distributed cloud-based infrastructure.

The core network implements critical functions such as authentication and mobility management, and steers user traffic to ensure proper routing and service quality. These responsibilities are fulfilled by Network Functions

[1] https://github.com/fatemeshafiee/closed-loop-nwdaf

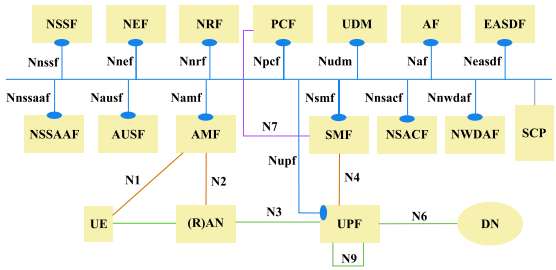


Figure 2 – Non-roaming 5G system architecture in service-based and reference point representation [13, 14]

(NFs), each serving a specific purpose. NFs are software applications with standardized interfaces [15] that can be independently deployed on virtual machines or containers within a service-based architecture, reflecting the softwarized and virtualized nature of the core network.

Fig. 2 illustrates the non-roaming 5G core network architecture in both service-based and reference point representations, as defined in the 3GPP technical specifications 23.501 [14], extended with NWDAF and its Nnwdaf service interface, as specified in [13].

Each NF provides its services to the other authorized consumers (e.g., other NFs) by exposing a service-based interface; these interfaces are named Nnf. For instance, Nupf (TS 23.501 clause 4.2.16 [14]) allows NFs such as NWDAF to access UPF services, including EES for data collection. The 5G system also has reference point interfaces; some of them are shown in Fig. 2. These interfaces are named with N followed by a number (e.g., N7, which connects PCF and SMF to deliver policy rules).

We provide an overview of some of the important and relevant responsibilities of selected NFs.

User Plane Function (UPF). The user plane function processes user-plane traffic flowing through the network. Users can send packets through the network after establishing Packet Data Unit (PDU) sessions. PDU sessions provide logical paths from User Equipment (UE) to gNB and to UPF. UPF connects to the RAN through the N3 interface and to the Data Network (DN) through the N6 interface, enabling end-to-end data delivery. It can also be connected to the other UPFs using the N9 interface. UPF enforces user-traffic policies and generates reports on user communications, including traffic usage and QoS measurements.

Session Management Function (SMF). The SMF is a control-plane function responsible for establishing, modifying, and releasing PDU sessions [14]. It controls the UPF's traffic detection, routing, and policy enforcement capabilities and can also use UPF traffic reports to support charging [14]. The control messages exchanged between the SMF and UPF follow the Packet Forwarding Control Protocol (PFCP) and are transmitted over the N4 interface.

Policy Control Function (PCF). The PCF oversees a range of network policies, including those governing PDU sessions and access and mobility management [16]. The PCF defines the Policy and Charging Control (PCC) rules that specify how network and charging policies should be applied to user traffic. These rules are provided to the SMF, which translates them into enforcement rules for the UPF. Each PCC rule includes the necessary information to identify the relevant traffic (e.g., via service data flow templates) and is associated with the policy and QoS treatment. These policies contain authorized QoS parameters, such as guaranteed and maximum bit rates for guaranteed bit-rate flows, as well as standardized QoS characteristics.

2.2 Quality of service framework

5G and beyond mobile networks are envisioned to support a wide range of services and applications, each with distinct QoS requirements. For instance, online gaming requires high responsiveness, autonomous vehicles rely on ultra-reliable low-latency communication for real-time decision-making, and massive machine-type communication must support large numbers of devices. Treating all flows equally would therefore be inefficient, motivating a QoS framework that differentiates between traffic types and supports each appropriately.

The QoS architecture defined by 3GPP for 5G systems [14] is flow-based. Multiple QoS flows can be defined within a PDU session to differentiate traffic treatment; each flow is identified by a QoS Flow ID (QFI), unique within the session. Each QoS flow is associated with a QoS profile, which includes parameters such as the 5G QoS Identifier (5QI) and Allocation and Retention Priority (ARP) that specify how packets are handled. 3GPP defines 5QI as a scalar value that indexes specific QoS characteristics of a network flow (e.g., resource type, priority, packet delay budget). Standardized 5QI values optimize signaling for common services (e.g., voice, video, IoT).

A QoS flow's resource type can be Guaranteed Bit Rate (GBR) or non-GBR. The QoS profile of a GBR flow includes fields such as Guaranteed Flow Bit Rate (GFBR), Maximum Flow Bit Rate (MFBR), and maximum packet loss rate. Aggregate bit rate parameters are used to enforce rate limitations, such as the Session Aggregate Maximum Bit Rate (Session-AMBR) and the UE Aggregate Maximum Bit Rate (UE-AMBR) [14]. As their names suggest, these parameters limit the total bit rate of non-GBR QoS flows: the Session-AMBR applies to all non-GBR flows within a single PDU session, while the

UE-AMBR applies across all PDU sessions of a UE.

2.3 Policy enforcement workflow

In the 5G system, session-level policy enforcement involves translating network policies into rules applied to each packet. This process demands coordinated interactions between network functions, primarily PCF, SMF, and UPF. The PCF is responsible for determining the appropriate network policies. The SMF, acting as a session orchestrator, converts these policies into rules that the user plane can enforce, and the UPF subsequently applies these rules to user traffic. This process enables control over traffic behavior based on operator policies, subscription data, network/Application Function (AF) inputs, and, where applicable, runtime analytics of network conditions. We discuss the three phases of the policy enforcement process in the following sections.

Phase 1: Policy decision and provisioning. The PCF can be preconfigured with the information it needs to consider when making network policy decisions. Additionally, it can obtain information from other functions, such as AFs, SMF, and NWDAF [17]. Using this information, the PCF derives session-management-related policies in the form of PCC rules and provisions them to, and subsequently updates and/or removes them from, the SMF via the session management Policy Control Service (Npcf_SMPolicyControl). In addition to identifying the traffic to which the rule applies, a PCC rule includes fields specific to the policy type, such as charging, QoS, or usage-monitoring control. The PCC rules can be dynamically provisioned to the SMF or SMF can be configured with predefined rules [17]. A session rule, as the name suggests, contains policies for a PDU session, such as authorized Session-AMBR and authorized default QoS.

Phase 2: QoS flow binding and N4 rule generation. The SMF coordinates enforcement by provisioning the UPF with rules for flow detection, packet routing, and authorized QoS treatment [16] in the form of Packet Detection Rule (PDR), Forwarding Action Rule (FAR), and QoS Enforcement Rule (QER). PDRs define a set of parameters that classify traffic flows; each PDR is associated with a FAR that specifies what the UPF should do with packets that match the PDR (e.g., forward, duplicate, or drop). Each PDR can also be accompanied by a QER that specifies the QoS enforcement required for those flows. When a PCF provides a PCC rule, the SMF should bind it to a QoS flow within a PDU session. There are QoS binding parameters that help with this procedure (e.g., 5QI). If a matching QoS flow exists, the SMF binds the PCC rule to it and processes it accordingly [16]. The SMF then converts the QoS requirements into QERs, associates them with the relevant PDRs, and sends them to the UPF through the N4 interface using the PFCP.

Phase 3: Per-packet policy enforcement at UPF. PDRs received by the UPF from the SMF contain a combination of packet-matching parameters and a precedence. When processing incoming packets, the UPF uses these parameters to identify and select the appropriate rule [18]. For each packet, the UPF should examine PDRs in order of precedence from highest to lowest and stop at the first matching PDR. Each PDR is associated with additional sets of rules that instruct the UPF on how to handle the packets described by the PDR. One of these rules is FAR; it tells the UPF how to process the packet (e.g., forward, drop, duplicate, buffer, ...). The PDR can be paired with QER, which specifies the UPF's QoS requirements for a matching QoS flow, such as its MBR and GBR.

2.4 NWDAF

NWDAF collects operational data from core functions and external sources and applies statistical methods or ML to derive actionable analytics. These analytics are subsequently exposed to authorized consumers as standardized services [13]. The 3GPP specification defines two logical functions within NWDAF: the Analytics Logical Function (AnLF) and the Model Training Logical Function (MTLF). An NWDAF instance may contain one or both. The AnLF is responsible for deriving analytics and exposing results to consumers, while the MTLF trains ML models and provisions them via services such as the ML model provision service, which AnLF instances use to discover and retrieve models matching their requirements. This specification also defines the services the NWDAF uses to operate. For example, the analytics subscription service enables NWDAF consumers to subscribe to various types of analytics and receive periodic notifications, while the analytics info service allows consumers to request a one-time report.

The 3GPP defines more than 20 analytics types that the NWDAF can provide to its consumers, covering a wide range of network conditions and operational objectives, including slice load, NF load, network performance, UE mobility, UE communication, abnormal UE behavior, QoS sustainability, user data congestion, and DN performance. The NWDAF supports two types of analytics output: statistics derived from past data and predictions for a future time period.

To obtain operational data, the NWDAF can subscribe to Event Exposure Services (EES) offered by other NFs. Among these NFs, the UPF plays an important role. It enables access to detailed user-plane telemetry such as throughput, volume, and QoS statistics at *per-flow* or *per-session granularity* [19]. The UPF EES allows consumers to define event type, measurement granularity, and re-

porting frequency, and provides standardized visibility of the data plane.

2.5 Prior work on NWDAF

We review relevant NWDAF-centered work, grouped by their primary focus. As Table 1 summarizes, prior work addresses the data analytics action pipeline only in isolated parts; no prior implementation realizes NWDAF as a 3GPP-compliant closed-loop orchestration primitive.

Control-plane monitoring. Chouman et al. [6] prototype NWDAF and integrate it with an open-source 5G core, connecting it to some network functions via standardized reference point interfaces (e.g., N34 with NSSF and N23 with PCF) and to others via custom ones. They capture control-plane traffic, extract statistics such as packet size and protocol distribution, and analyze interactions between the Binding Support Function (BSF) and the Network Repository Function (NRF). Manias et al. [3] extend this setup to further analyze NF interactions by filtering control-plane packets exchanged between network functions and deriving statistics such as average and maximum packet sizes per communication path. They also apply k-means clustering to classify traffic patterns. Both sets of work focus on control-plane monitoring and do not address UE communication traffic collected through the UPF EES or the use of NWDAF insights for closed-loop automation, leaving NWDAF's potential for intelligent and automated network orchestration and management insufficiently explored.

Predictive analytics The NWDAF implementation proposed in [4] collects data from the AMF and NRF via event exposure services. To supplement this, custom data collection methods were employed, including Google cAdvisor for resource utilization monitoring and scripts to capture traffic between UEs and the data network. NWDAF implementation provides analytics on UE mobility (location), UE communication (uplink/downlink volume), and NF load (CPU, memory, and overall usage). These analytics are stored in Prometheus and visualized with Grafana. For predictive modeling, the authors used linear regression, random forest, and decision tree regressors trained on tcpdump-collected traffic data to estimate network performance. NWDAF performance and resource usage were evaluated under single-user and ten-user scenarios to assess scalability. The stability of their proposed system was evaluated by running it for two minutes without failure. However, this implementation lacks UE communication data directly from the UPF EES, which limits its accuracy and compliance with 3GPP standards.

Anomaly detection. In [5], the authors implement an NWDAF for anomaly detection in private 5G networks by extending the Open5GS core to support limited AMF and SMF event exposure services (e.g., PDU session events and UES-IN-AREA-REPORT). A custom traffic monitor module collects UE communication data. The collected traffic and session data is used to train models that detect anomalies such as low throughput and abnormal PDU session patterns. However, the system does not respond to detected anomalies.

The NWDAF implementation proposed in [7] follows 3GPP specifications and adopts a microservice-based architecture. It gathers data from core NFs, a Virtualized Infrastructure Manager (VIM), and xApps, offering both control-plane and data-plane analytics. Their NWDAF provides analytics on network performance, UE communication, and NF load, as well as a mechanism for detecting abnormal UE behavior using an LSTM autoencoder to identify abnormal flows. The proposed NWDAF, however, does not directly collect from the UPF. Instead, traffic data is relayed from the UPF to the SMF via N4 and then to the NWDAF through Nsmf EES, an inefficient method, as direct UPF event exposure is supported by [19]. While these sets of work align with 3GPP analytics and anomaly-detection standards, they lack real-time closed-loop control and do not collect UE traffic data via a UPF EES.

Model lifecycle management. [22] explored integrating an AI-as-a-Service (AIaaS) platform with an NWDAF to enable ontology-based ML model discovery and provisioning. While this addresses interoperability, it does not address runtime model updates (e.g., proactively sending updated models to the consumer) or versioning. To address this, our design implements the 3GPP-defined ML model provision service using MLflow [23], enabling continuous model management.

Closed-loop automation. Only a few works considered closing the loop using NWDAF-provided analytics. In [20], NWDAF was integrated with an intent-based networking platform to trigger slice scaling or DDoS mitigation based on ML predictions. However, it is integrated into an open-source implementation of the Evolved Packet Core (EPC), which lacks standardized 5G functions and does not support event exposure services, hindering its integration with 5G and beyond core systems. [21] enabled AMF/SMF event generation for handover prediction but did not take any policy enforcement decisions based on the predictions. In contrast, our system extends SMF and PCF to subscribe to NWDAF analytics via the 3GPP-defined `Events Subscription` service, enabling automatic release of the user's PDU session or runtime adjustment of QoS, thereby achieving standards-compliant closed-loop control.

Table 1 – Comparison of NWDAF implementations

Prior work	Software stack	UPF Event Exposure Service	ML Provisioning Lifecycle Mgmt	Closed-Loop Orchestration	Focus area
[6]	Open5GS	✗	✗	✗	Control-plane packet statistics
[3]	Open5GS	✗	✗	✗	NF interaction clustering
[4]	Free5GC	✗	✗	✗	Performance prediction
[20]	OAI EPC	✗	✗	✓	Slice scaling; DDoS mitigation
[5]	Open5GS	✗	✗	✗	UE-level anomaly detection
[7]	OAI 5GC	✓*	✗	✗	Flow-level anomaly detection (SMF relay)
[21]	Free5GC	✗	✗	✗	AMF/SMF event exposure; mobility prediction
[22]	OAI 5GC	✗	✓	✗	Ontology-based ML model provisioning
Ours	**Open5GS**	✓	✓	✓	**Closed-loop network orchestration**

✓: Supported ✗: Not supported ✓*: Partial (Uses SMF relay for UPF data).

3. SYSTEM ARCHITECTURE AND DESIGN

Our goal is to enable closed-loop network orchestration, which requires continuous data collection, analytics over the collected data, and timely delivery of the resulting insights to the network functions responsible for policy enforcement. While 3GPP defines standardized services and procedures for each stage of this workflow, their integration into an end-to-end closed-loop system remains incomplete, as discussed in Section 2.

As a result, a key question remains: can a standards-compliant closed-loop system be realized in practice, and at what cost? In particular, it is unclear whether the required data collection, analytics, and enforcement pipeline can operate efficiently at the timescales needed for network control.

Answering this question requires not only integrating the full pipeline, but also validating the underlying mechanisms that enable it. For example, the 3GPP-defined UPF EES enables user-plane telemetry collection, but it remains unclear whether the exposed data is sufficient for meaningful analytics, and whether such fine-grained data collection can be achieved without degrading the UPF's packet-forwarding performance.

To this end, we integrate NWDAF with the 5G core and incorporate the components that act on its analytics, enabling evaluation of the full closed-loop pipeline. This allows us to quantify end-to-end efficiency and latency, from observing network conditions to generating analytics to enforcing the corresponding action.

We adopt a modular architecture building on the OAI NWDAF design and codebase [7, 24]. We design and implement (i) the NWDAF *ML Model Provision Service* (MLMPS) and (ii) *Analytics Engines*. We extend (iii) the NWDAF *Southbound Interface* (SBI) with an EES client to ingest UPF telemetry. We further develop (iv) the *UPF EES* for data generation and (v) an *NWDAF Subscriber* integrated into SMF and PCF, enabling them to act on NWDAF analytics. The OAI NWDAF *Northbound Interface* (NBI) is reused to deliver 3GPP-compliant reports to consumers.

Fig. 3 illustrates the resulting NWDAF-driven closed-loop architecture, which realizes network automation as a continuous data–analytics–action loop.

The workflow consists of three tightly integrated stages:

(1) Data collection. The NWDAF subscribes to UPF EES to obtain fine-grained, real-time telemetry from the user plane, providing visibility into per-flow and per-session behavior through standardized interfaces.

(2) Analytics. Collected data is processed by modular analytics engines using statistical methods or machine learning inference. Models are dynamically provisioned via the ML model provision service, enabling continuous adaptation to evolving network conditions.

(3) Action. Analytics results are exposed through the NWDAF northbound interface and consumed by control-plane functions such as the SMF and PCF. These functions translate analytics into enforcement actions, including session-level mitigation and QoS policy updates, thereby closing the loop.

This design elevates the NWDAF from a passive analytics provider to an active orchestration component, enabling automated, standards-compliant control across the network.

3.1 Data collection: UPF EES

Data collected at the UPF can provide fine-grained visibility into per-UE flows and usage patterns. This visibility is essential for analytics-driven orchestration across operational objectives such as QoS assurance and threat mitigation. According to the 3GPP specifications, the UPF EES is responsible for providing this data. However,

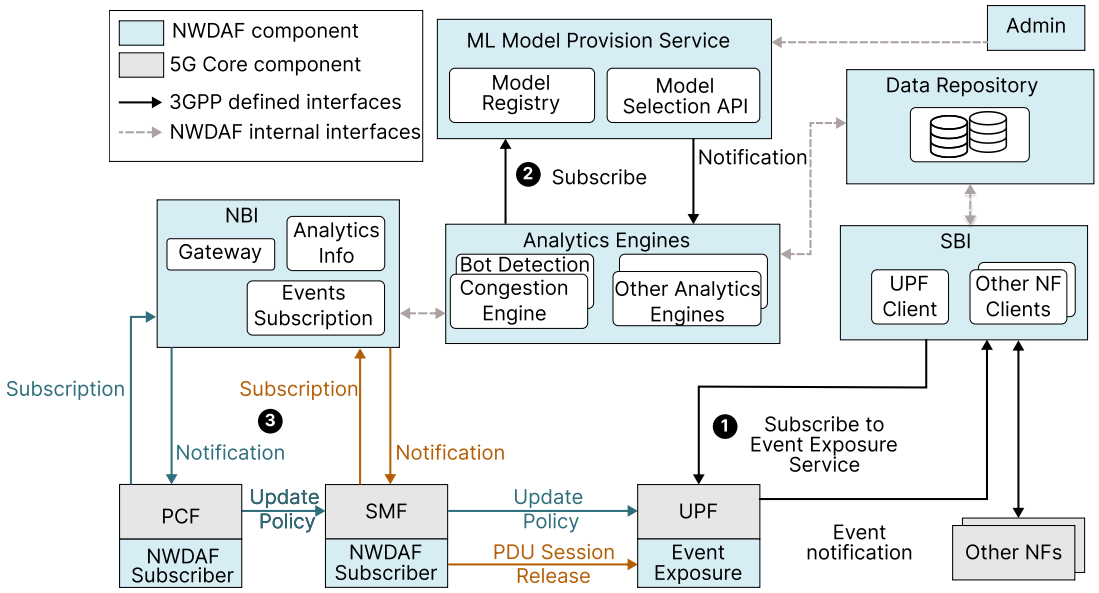


Figure 3 – NWDAF-driven closed-loop network orchestration architecture

no open-source implementation currently exists, leaving open questions about the effectiveness, resource usage, and latency impact of exposing fine-grained user-plane telemetry through this service. To address this gap, we implement and evaluate the first UPF EES, together with an NWDAF that consumes its reports for data collection.

3.1.1 Data collection workflow

NWDAF SBI is responsible for collecting data from various sources (e.g., SMF and UPF). The UPF client in this module sends a data collection subscription request to the UPF EES. In this request, the NWDAF can indicate which data it is interested in collecting via standard request fields such as event type (e.g., `usage-measures`), measurement type (e.g., `volume`), or measurement granularity (e.g., `per-flow`). In addition, it can indicate how it wants to receive data in the reporting mode fields, such as the frequency of data reports from the UPF (e.g., every second). This process is indicated as Step ❶ in Fig. 3.

After responding to the NWDAF's initial request, the UPF EES sends the data in periodic reports to the NWDAF, which then stores it in its own database. Each of these reports includes a timestamp field, enabling the NWDAF to analyze the network's historical behavior while monitoring its real-time status.

3.1.2 UPF event exposure service design

Encoder/decoder modules. All requests and responses must comply with the 3GPP-defined format. To ensure this, we implement dedicated encoder/decoder modules that convert between JSON and 3GPP-compliant structured data types.

Server module. The UPF EES must handle subscription requests, validate them, and send appropriate responses. In our design, the server module is responsible for these operations. It stores the subscription details, assigns a unique subscription ID, and sends a response back to the subscriber. The response includes the subscription ID and the details of the accepted subscription. If a subscriber requests events that are unavailable, the server omits them from the response.

Since subscribers may request different types of data, the server tracks each subscriber's requests so that the appropriate notifications can be generated. We use a HashMap, with each event as the key and the list of subscribers for that event as the value, enabling efficient lookup of relevant subscribers at the time of notification. To avoid collecting data when no consumer needs it, the server activates the data preparation module only after at least one subscription is registered. Because the subscriber list is shared with other modules, it is protected by a lock to ensure thread safety.

Data preparation module. The data preparation module is designed to efficiently collect data without compromising performance. Our goal is to collect data from packets passing through the UPF and use it to generate notifications for EES subscribers. However, the part of the UPF source code responsible for packet forwarding is a hot path because it is executed for every packet traversing the network; thus, its execution frequency increases with the number of packets. Additionally, the high packet rate at the UPF significantly exceeds the rate at which notifications are generated. This means a lightweight data collection approach is essential. To achieve this, we

separate data collection and notification generation into two modules.

Once activated by the server, the data preparation module receives each packet passing through the UPF. For every packet, it derives a hash key from the flow identifiers (source IP, destination IP, source port, and destination port), the PDU session ID, and the subscription ID, and uses this key to aggregate critical packet statistics such as volume and direction. The resulting per-key statistics are then shared with the client module, which generates notifications.

To prevent synchronization from blocking the fast path of packet processing, we use a lock-free Single-Producer Single-Consumer (SPSC) ring buffer between the data preparation and client modules instead of a shared lock. This allows the data preparation module to gather essential information during packet processing and defer the heavier computation until notification generation. Our evaluation (Section 6) confirms that this approach incurs limited CPU overhead without degrading UPF throughput.

Client module. The client module is responsible for periodically generating and sending notifications to each subscriber. Depending on the subscriber's request, our implementation can generate various notifications based on the parameters shown in Table 2.

Table 2 – Measurements supported by our UPF EES implementation

Parameter	Supported Values
Event Type	`"usage-measures"`, `"usage-trends"`, `"QoS-monitoring"`
Granularity	`"per-flow"`, `"per-session"`

Each subscriber defines its expected notification by selecting a combination of these features. The client module implements the processing logic needed to transform the basic flow statistics (from the data preparation module) into the specific notification formats requested by subscribers. To determine when to send notifications, the module compares the current time with the timestamp of the last sent report to check if the reporting period has elapsed.

Each measurement field (e.g., `volume`) in a notification report contains several parameters (an example is shown in Section 3.1.4). Each time a packet arrives, the data preparation module is triggered, while the server and client modules operate independently on separate threads. This configuration allows each module to operate simultaneously, preventing the loss of a subscription request or incoming packets.

3.1.3 QoS monitoring report

UPF EES supports QoS monitoring by reporting packet delay. According to the 3GPP specification, this service is expected to report the delay between the UE and the PDU Session Anchor (PSA) UPF. In this work, we focus on the UPF-side contribution to this metric and therefore instantiate the delay attribute as an internal UPF packet delay in microseconds. Incorporating the radio-side component to report the full UE–PSA UPF delay is left for future work. Specifically, we record the time the UPF receives a packet as the start timestamp and the time it forwards the packet as the end timestamp. The difference between these timestamps is treated as the per-packet internal delay. We then update the running average delay online by maintaining a packet counter and accumulated delay and recomputing the average after each packet arrival. This procedure is applied independently for both uplink and downlink traffic.

Our implementation supports delay reporting at both QoS flow (per QFI) and PDU session granularity. Delay statistics are collected per QoS flow by default; when a report is requested at the UE or session level, the UPF aggregates the per-QFI measurements associated with the corresponding session. Although the internal delay estimate is updated on a per-packet basis, the UPF EES sends notifications to subscribers (e.g., NWDAF) according to the configured reporting interval; i.e., delay values are reported as window-based summaries over the requested time window. We also further extend the measurement scope to include packet loss.

3.1.4 UPF EES notifications

When the subscriber selects user data usage measures as the event type, it can choose between volume measurement and throughput measurement as the measurement type. The corresponding data is then delivered in the volumeMeasurement or throughputMeasurement field of the notification, respectively. On the other hand, the subscriber can ask for user data usage trends and choose throughput measurement. In this case, the throughput statistics measurement is filled. In all cases, only the field the subscriber asked for has a value and all the other fields are empty. A sample notification is shown in Listing 1. Listings 2, 3, and 4 show sample values for each measurement field.

```
{"eventType": "USER_DATA_USAGE_MEASURES",
  "ueIpv4Addr": "10.42.0.2",
  "snssai": {"sst": 2, "sd": "00002"},
  "timeStamp": "2025-03-27T18:03:49Z",
  "startTime": "2025-03-27T18:00:59Z",
  "userDataUsageMeasurements": [{
    "flowInfo": {
      "packFiltId" : {
        "SrcIp" : "10.42.0.2",
        "DstIp" : "142.***.***.***"
```

```
      },
      "fDir" :  "BIDIRECTIONAL"
    },
    "volumeMeasurement": { ... },
    "throughputMeasurement": { ... },
    "throughputStatisticsMeasurement": { ... }
  }]
}
```

Listing 1 – Sample user data usage measures notification per flow

```
    "volumeMeasurement":{
    "totalVolume": "1000B",
    "ulVolume":"624B",
    "dlVolume":"376B",
    "totalNbOfPackets":15,
    "ulNbOfPackets":8,
    "dlNbOfPackets":7}
```

Listing 2 – Sample volume measurement notification

```
    "throughputMeasurement":{
    "ulThroughput": "1344bps",
    "dlThroughput": "1344bps",
    "ulPacketThroughput": "2pps",
    "dlPacketThroughput": "2pps"}
```

Listing 3 – Sample throughput measurement notification

```
    "throughputStatisticsMeasurement":{
    "ulAverageThroughput": "1319bps",
    "dlAverageThroughput": "1319bps",
    "ulPeakThroughput": "1344bps",
    "dlPeakThroughPut": "1344bps",
    "ulAveragePacketThroughput": "1pps",
    "dlAveragePacketThroughput": "1pps",
    "ulPeakPacketThroughput": "2pps",
    "dlPeakPacketThroughput": "2pps"}
```

Listing 4 – Sample throughput statistics measurement notification

Listings 5 and 6 show a sample per-flow QoS monitoring notification and its measurement fields, respectively.

```
{"eventType": "QOS_MONITORING",
  "ueIpv4Addr": "10.42.0.20",
  "snssai": {"sst": 2, "sd": "00002"},
  "timeStamp": "2026-05-01T18:53:40Z",
  "startTime": "2026-05-01T18:52:40Z",
    "QosMonitoringMeasurements"  : [{ ... }]
}
```

Listing 5 – Sample QoS monitoring notification per flow

```
  "QosMonitoringMeasurements"  : [{
   "qosFlow" : {
     "packFiltId" : {
       "SeId" : 3179,
       "SrcIp" : "10.42.0.20",
       "DstIp" : "142.***.***.***",
       "SrcPort" : 0,
       "DstPort" : 0},
     "qfi" : 1,
     "fDir" :  "BIDIRECTIONAL" },
   "dlPacketDelay" : 97.0,
   "ulPacketDelay" : 114.0,
   "rtrPacketDelay" : 211,
   "dlAveThroughput" :  "516.92bps" ,
   "ulAveThroughput" :  "516.92bps"
}]
```

Listing 6 – Sample QoS monitoring measurement per flow

3.2 Analytics architecture

Each analytics engine periodically analyzes the data in the NWDAF's database, leveraging an ML model or a statistical approach. If an engine wants to use an ML model, it must send a subscription request to the MLMPS, specifying the necessary fields, such as the report type (e.g., `ABNORMAL-BEHAVIOR`), along with filters to help the service identify a suitable model. After the engine accepts the request, MLMPS sends a URL to the engine for running inferences on the ML model. Each engine sends its analytics results to subscribers via the northbound interface. This is shown as Step ❷ in Fig. 3. Below, we detail the newly introduced NWDAF modules and their roles.

3.2.1 Integration with UPF EES

To leverage the data provided by the UPF EES, we implement a UPF client within the SBI module of the NWDAF. Upon NWDAF deployment, the UPF client initiates the data collection process by subscribing to the UPF EES and actively listening for notifications. Each received event is processed using this module and stored in the NWDAF database.

3.2.2 Analytics engines

The NWDAF is introduced to support various types of analytics, each one handled by a dedicated analytics engine. These engines retrieve relevant data from the database, perform analytics (using statistical methods or ML models), and output the results. The NBI module will package this output into the appropriate notification format and send it to the NWDAF consumer. We applied minor modifications to the NBI module to enable its integration with our engines.

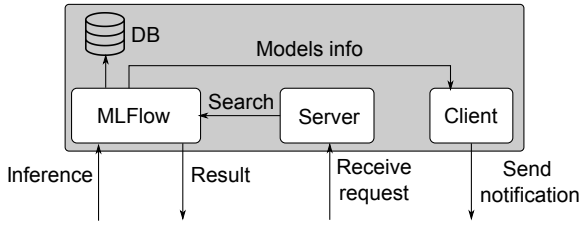


Figure 4 – ML model provision service

Table 3 – Components of NWDAF-based closed-loop network orchestration framework

Component	Role	Key Capabilities
UPF Event Exposure Service (EES)	Data collection	Real-time per-flow and per-session telemetry; lock-free data aggregation; generation of 3GPP-compliant notifications.
NWDAF Southbound Interface (SBI)	Data ingestion	Subscribes to UPF EES; continuously collects telemetry; stores raw measurements in MongoDB for downstream analytics.
NWDAF Analytics Engine	Data analysis	Fetches data from the database; performs statistical or ML-based analytics; generates analytics reports in 3GPP formats.
NWDAF ML Model Provision Service (MLMPS)	Model management	Integrates with MLflow for model registry and version control; manages model lifecycle; provisions models dynamically to analytics engines.
NWDAF Northbound Interface (NBI)	Report Delivery	Packages analytics results; delivers 3GPP-compliant notifications to subscribed consumers.
NWDAF Subscriber	Policy Enforcement	Subscribes to NWDAF analytics; triggers automated policy actions (e.g., PDU session termination).

3.2.3 ML model provision service

In line with the 3GPP specification, this service can receive subscription requests from the server module and provide information about requested ML models via the client module. We integrate MLflow [23] into this service for model management and versioning. Each analytics engine can subscribe to MLMPS and request an ML model with specific features in the format defined by the 3GPP specification. The service queries the MLflow registry to determine whether a suitable model is available.

The MLMPS periodically queries the MLflow registry at each subscriber's requested reporting period and notifies the subscriber of the latest matching model. In addition to implementing the core MLMPS APIs, we developed complementary MLflow-facing endpoints for logging new models and updating existing ones; we refer to these as the admin APIs.

3.3 Closing the loop: action and policy enforcement

The final step in closed-loop automation is to deliver the analytics generated by the NWDAF to the network functions that take action. This step is implemented via the NWDAF northbound interface, which packages the analytics engine's output into 3GPP-compliant reports and notifications and delivers them to subscribed consumer NFs.

3.3.1 Enabling NWDAF consumers

Different consumer NFs, including those responsible for network orchestration, should be able to subscribe to the NWDAF and receive periodic notifications. Such a subscription request includes parameters such as the requested analytics event and the desired reporting period. In addition to parsing NWDAF notifications, consumers should implement the logic needed to translate NWDAF-provided analytics into appropriate control actions. They may use NWDAF analytics in different ways depending on their operational role. In some cases, the analytics may trigger session-level actions, while in others, they may lead to policy adaptation at flow or session granularity. As a result, the consumer-specific interpretation of analytics and the corresponding enforcement logic are not fixed by the framework itself, but are implemented within the subscribing NF.

In this work, we demonstrate this action stage through two instantiations of the proposed framework. Step ❸ in Fig. 3 illustrates these two actions. In Section 4, we extend the SMF to consume abnormal behavior analytics and trigger the release of suspicious PDU sessions as a mitigation action. This is shown in orange in Fig. 3. In Section 5, we extend PCF to consume user data congestion analytics and adapt QoS-related policy parameters at runtime. The different steps of this action are shown in blue in Fig. 3. These two case studies show that the same NWDAF framework can support distinct operational objectives and control tasks while preserving a common standards-compliant pipeline for data collection, analytics generation, and report delivery.

3.3.2 Enabling rate enforcement at UPF

While PCF determines rate parameters, the UPF is responsible for enforcing them. As described in Section 5, we extend the PCF to operate as an NWDAF consumer and adapt these parameters according to the network condition. This, in turn, requires runtime rate enforcement at the UPF in the packet-processing path. To enable this capability, we implement a token-bucket mechanism that supports per-flow GBR and MBR enforcement and per-session AMBR enforcement.

When a QER is created at the UPF, the UPF initializes token buckets based on the QER's GBR and MBR parameters. For GBR QERs, separate token buckets are maintained for uplink and downlink GBR, as well as for the excess MBR portion above the GBR in each direction. The token generation rate is derived from the configured bit rate, and the bucket capacity is set to 1.2 times that rate to allow limited short-term burst tolerance. In addition, the SMF reports the Session-AMBR through the QER associated with the default QoS flow, where the GBR is set to zero, and the MBR is set to the AMBR. We also maintain a shared token bucket for all non-GBR flows within the same session.

When a packet arrives, the UPF first matches it to a PDR, then identifies the QER to enforce. For GBR flows, the QER associated with the matched PDR is used. For non-GBR flows, the UPF uses the QER associated with the default QoS flow so that all non-GBR traffic in the same PDU session consumes tokens from the same Session-AMBR bucket. For GBR flows, the packet is first checked against the GBR bucket. If the GBR bucket cannot admit the packet, the UPF then checks the excess MBR bucket. If neither bucket has sufficient tokens, the packet is dropped. For non-GBR flows, the packet is checked against the shared session-level token bucket keyed by the session ID. When the UPF receives a QER update with new bit rate values, the per-QER token buckets are updated accordingly. For session-level AMBR enforcement, the shared bucket rate and capacity are refreshed from the updated default QER during packet processing. This enables the PCF to change the rates by updating the PCC rules.

Although the rate enforcement logic is implemented for both uplink and downlink directions, enforcing uplink QERs for GBR flows requires RAN support to match packets accordingly. In our testbed, uplink packets arrive at the UPF with the default QFI and therefore match the default uplink PDR. As a result, uplink GBR enforcement could not be evaluated, and uplink traffic is instead policed using the session-level uplink AMBR.

Fig. 5 shows AMBR rate enforcement in the downlink direction. The downlink AMBR is set to 15 Mbps. As shown in the figure, the first non-GBR flow initially sends traffic at 17 Mbps. Although a short burst is allowed, its throughput is eventually capped at about 15 Mbps. When the second non-GBR flow joins, the first flow's throughput decreases as the AMBR is shared between the two flows. Finally, when the first flow leaves, the second flow's throughput increases to 15 Mbps.

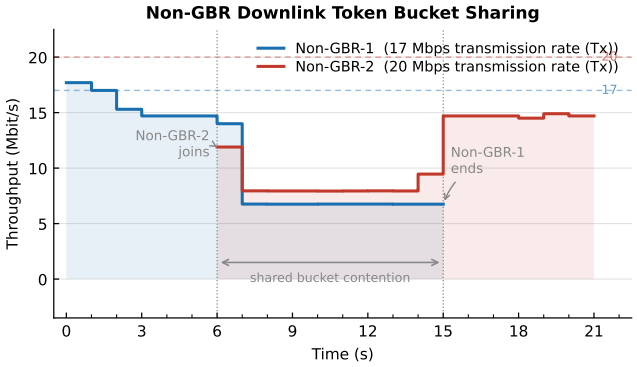


Figure 5 – Downlink AMBR enforcement for non-GBR flows in the same session.

3.4 Implementation details

We extend Open5GS to integrate our NWDAF design and support the closed-loop workflow, including real-time data collection, analysis, and automated policy enforcement.

UPF extensions. We implemented the UPF EES and integrated it directly into the existing Open5GS UPF codebase to enable real-time data collection from the user plane. We also implemented a token-bucket mechanism to enforce bandwidth limits.

NWDAF implementation. In order to receive data from the UPF, we implement the UPF client inside the NWDAF SBI component. The collected data is stored in a MongoDB database. We develop the engines and the MLMPS. This latter integrates MLflow [23], which provides a systematic framework for storing and managing ML models, offering model version control and seamless operations. We use the MLFlow query option to efficiently search for models that match the filters specified in the subscription request.

We implement a custom API for the ML model registry to simplify management operations. While there is an option to use the MLflow deployment terminal to register trained models, our custom API makes it easier to manage the ML model registry. We also leverage MLflow to serve different ML models for inference, receiving inference data from the engines and returning model predictions. We modified other parts of NWDAF in Go to make the engines reachable when generating reports for subscribers.

SMF and PCF extensions. We extend the Open5GS codebase of the SMF and PCF to support subscription to NWDAF analytics and automated policy enforcement.

4. ANOMALY DETECTION AND MITIGATION VIA NWDAF AND SMF

This section presents the first use case instantiation of the NWDAF framework introduced in Section 3. We focus on detecting abnormal UE behavior and on a closed-loop mitigation workflow in which the NWDAF analyzes UPF EES reports, and the SMF enforces appropriate actions. We detail below the detection pipeline, analytics integration, SMF-based mitigation logic, and the experimental setup used to evaluate end-to-end responsiveness.

4.1 Application scenario and objectives

One of the NWDAF use cases defined in the 3GPP specifications [13] is the detection of UEs exhibiting abnormal behavior. The specification categorizes abnormal UE behavior into two types: *mobility-related* anomalies, such as a UE appearing in an unexpected location, and *communication-related* anomalies, such as a UE exhibiting unexpected traffic patterns, for example, during a DDoS attack. In addition, 3GPP identifies possible actions that different NFs may take once unexpected UE behavior is detected. Detecting and mitigating such behavior during network operation is important, as it helps minimize the impact of malicious activity on the network and other users' experiences.

Enabling this use case in 5G networks requires effective data collection, analytics, and action. We chose bot detection as one of our case studies to investigate the potential of NWDAF for enabling zero-touch management, as it allows us to demonstrate multiple parts of our system. Detecting bot behavior requires the NWDAF to collect UE activity data at runtime, which helps us assess whether the UPF EES can provide usable data to the NWDAF in a timely manner. It also requires an NWDAF engine that can detect abnormal behavior and report it, with the corresponding mitigation action triggered as quickly as possible.

In this section, we show how the proposed framework can be instantiated to realize this closed loop, from UPF-based data collection to NWDAF-based anomaly detection and SMF-based mitigation.

4.2 End-to-end workflow for bot detection mitigation

We integrate the framework presented in Section 3 with a bot detection engine and an extended SMF to enable a zero-touch bot detection and mitigation workflow. Fig. 6 shows the sequence diagram, and the workflow can be summarized in four phases.

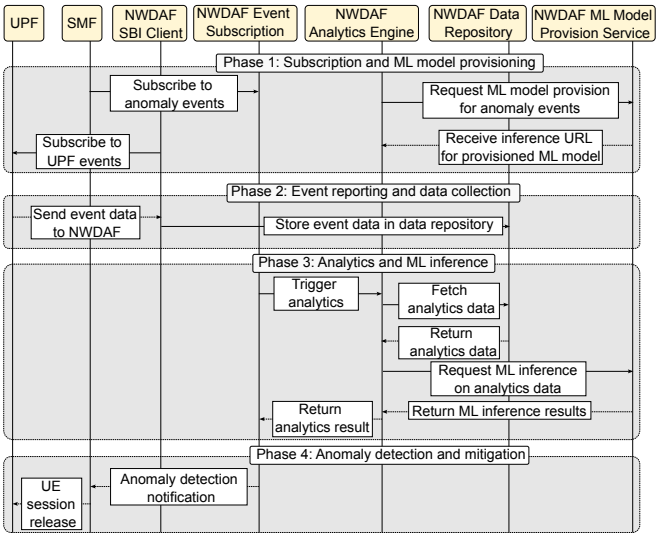


Figure 6 – Sequence diagram for the closed-loop anomaly detection and mitigation workflow

- *Subscription phase*: Upon deployment, the SBI subscribes to the UPF EES, and the engine subscribes to the ML model provision service, asking for a model for abnormal behavior events. Simultaneously, the SMF subscribes to the NWDAF for the abnormal behavior detection event.
- *Data collection phase*: When the UEs start their communication, the UPF EES starts sending periodic reports to the NWDAF SBI.
- *Analysis phase*: Based on the report period requested by the SMF, the NWDAF NBI will trigger the bot detection engine to investigate the network traffic. The bot detection engine then retrieves data from the NWDAF database and preprocesses it to derive graph-based features for each UE communication; it then uses these features, along with the inference model provided by the ML model provision service, to investigate the behavior of each UE.
- *Mitigation phase*: NBI receives the detected results from the bot detection engine and encapsulates them in 3GPP-compliant notifications and sends them to the SMF. If abnormal behavior is reported in the notification, the SMF will release the corresponding PDU sessions to protect the network.

4.3 Anomaly detection pipeline

To enable this workflow, the NWDAF SBI subscribes to UPF EES volume-measurement reports and stores the notifications it receives in the NWDAF database. In parallel, the bot detection engine relies on a pre-trained ML model for runtime inference, together with a data-preparation pipeline that preprocesses raw UPF EES notifications retrieved from the database into inference-ready features. This section presents the dataset and training process used for the bot detection model and

describes how the resulting analytics engine is integrated into the proposed framework.

4.3.1 Data source and collection

We rely on a public dataset that includes diverse malicious activities with reliable labels. In particular, we utilize bot behavior data from scenarios 1, 2, 3, and 5 of the CTU-13 dataset [25], which include Neris, Rbot, and Virut bots, as part of our training set. We extract packet-level information, such as source and destination IP addresses and packet size, to form a flow-based dataset.

For benign traffic, we subscribe to the NWDAF to receive UPF EES reports, run various scripts on UEs, and use the collected data as the source for the benign class. This allows us to train the ML model on data collected in the same environment, under the same runtime conditions, and through the same preprocessing pipeline used during system operation.

Finally, we preprocess both benign and malicious traffic to ensure that each entry uses the same feature set. Because we expect our system to detect and mitigate attacks before they are completed, we also train ML models on data from ongoing flows rather than complete flow logs.

4.3.2 Analytics engine integration

4.3.2.1 Graph-based traffic representation

We use a methodology inspired by BotChase [26, 27] to convert flow-based benign and malicious traffic information into a graph and use graph-based features to train a random forest model. We derive these features by constructing a directed communication graph, in which each IP address is treated as a unique node, and the number of transmitted packets is assigned as the weight of each edge. We then extract node-level features, including the numbers of incoming and outgoing edges, the weighted in and out-degrees, and the weighted betweenness centrality, which indicates how often a node appears on the shortest paths between pairs of nodes in the network. This model can be formulated more precisely as follows:

Let $G = (V, E, w)$ be a directed, weighted communication graph, where:

- V is the set of nodes (IP addresses),
- $E \subseteq V \times V$ is the set of directed edges,
- $w : E \to \mathbb{R}^+$ assigns weights (e.g., packet counts) to edges.

Each node $v \in V$ represents an IP address (e.g., UE or remote server). For each UE $u \in V$, we extract a feature vector:

$$\mathbf{x}_u = \begin{bmatrix} \deg^{\text{in}}(u) \\ \deg^{\text{out}}(u) \\ w\text{-}\deg^{\text{in}}(u) \\ w\text{-}\deg^{\text{out}}(u) \\ \text{BC}(u) \end{bmatrix}$$

where:

- $\deg^{\text{in}}(u)$: in-degree of node u,
- $\deg^{\text{out}}(u)$: out-degree of node u,
- $w\text{-}\deg^{\text{in}}(u)$: weighted in-degree,
- $w\text{-}\deg^{\text{out}}(u)$: weighted out-degree,
- $\text{BC}(u)$: betweenness centrality of node u,

The set of all such vectors for UEs of interest is:

$$X = \{\mathbf{x}_u \mid u \in \mathcal{U} \subseteq V\}$$

We then pass each $\mathbf{x}_u$ to a machine learning model f to produce an output label:

$$\hat{y}_u = f(\mathbf{x}_u) \in \{0, 1\}$$

where:

- $\hat{y}_u = 0$ indicates benign behavior,
- $\hat{y}_u = 1$ indicates detected abnormal behavior

4.3.2.2 Runtime integration with the NWDAF pipeline

A trained ML model must be registered in the MLMPS to be made available for online inference. This service enables different analytics engines to request and use the appropriate model during operation. The bot detection analytics engine subscribes to the ML model provision service to request a model for detecting abnormal UE behavior. The MLMPS provides the model that matches the engine request; in our case, this is the random forest model that we trained offline.

The bot detectin analytics engine periodically analyzes data collected by the NWDAF at the interval requested by the analytics consumer. However, the NWDAF database accumulates all EES reports collected since the initial subscription to the UPF EES; as a result, it may grow large and contain outdated records. To improve efficiency and ensure that the detection result reflects the current traffic state, the bot detection engine retrieves the EES reports for the last 30 seconds. It then preprocesses these

reports to match the graph-based features described in Section 4.3.2.1. It uses the derived features to run inference with the ML model using the link provided by MLMPS. Finally, it reports the UE IP and session ID of the detected abnormal UEs.

4.4 SMF-based closed-loop mitigation

While there are multiple actions that different network components can take after receiving the requested analytics from the NWDAF, we focus on closing the loop by extending the SMF to act as an NWDAF subscriber.

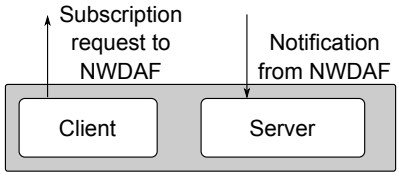


Figure 7 – SMF as NWDAF subscriber

According to the 3GPP specifications, the SMF can subscribe to UE abnormal behavior reports from the NWDAF. Upon receiving such a report, the SMF may initiate the termination of the PDU session of a suspicious UE, thereby preventing it from transmitting any further packets. To enable this automated enforcement, we extend the SMF to subscribe to the NWDAF and receive periodic notifications. We add two modules, *server* and *client*, inside the SMF, where the client module sends a one-time subscription request to the NWDAF, and the server receives periodic notifications from the NWDAF NBI indicating if there is abnormal behavior in the network. Listing 7 shows an example subscription request from SMF. Additionally, Listing 8 shows a sample NWDAF notification after detecting an abnormal UE.

```
{
  "notificationURI": "http://smf-nsmf.../
      notification",
  "eventSubscriptions": [{
    "event": "ABNORMAL_BEHAVIOUR",
    "excepRequs": [{"excepId": "
        SUSPICION_OF_DDOS_ATTACK"}],
    "notificationMethod": "PERIODIC",
    "repetitionPeriod": 1 }]
}
```

Listing 7 – SMF subscription request to NWDAF

```
{ "event": "ABNORMAL_BEHAVIOUR",
  "abnorBehavrs": [{
    "supis": ["10.42.0.2-2742"],
    "excep": {
      "excepId": "SUSPICION_OF_DDOS_ATTACK", }, }]
          }
```

Listing 8 – NWDAF notification to SMF indicating abnormal behavior

Subsequently, the SMF releases the PDU sessions for the suspicious UEs identified in the NWDAF notifications. When an abnormal UE is detected, the SMF terminates the corresponding PDU session by instructing the UPF to release it via the N4 interface. This mechanism enables closed-loop mitigation of misbehaving UEs.

4.5 Evaluation

In this section, we evaluate the end-to-end responsiveness of the proposed anomaly-detection and mitigation loop. In particular, we focus on the time required to observe malicious traffic at the NWDAF, detect abnormal behavior, and enforce mitigation throughthe SMF.

4.5.1 Experimental scenario

We evaluate the proposed closed-loop automation system in a bot detection scenario to assess the end-to-end interaction between data collection, analytics, and mitigation.

We deploy Open5GS as our 5G core using a Kubernetes setup. The Open5GS core includes the modifications we made to the UPF and SMF, as described in sections 3.1 and 4.4. We use UERANSIM [28] to simulate gNB and UE behavior. After deploying the ML model provision service, the trained model is registered through its admin API. We set up 20 servers with distinct IP addresses in our testbed to serve as the bot attack's intended targets. To simulate bot behavior, we used Nmap's [29] –script http-open-proxy to scan for open web proxies. We configured the SMF to receive abnormal behavior reports from the NWDAF every second, and varied the subscription period of NWDAF SBI for data collection from the UPF. For each experiment round, after deploying the system, we started the bot's behavior in the UE.

4.5.2 Closed-loop response time: attack to mitigation

To evaluate the end-to-end responsiveness of the proposed closed-loop system, we measure and report the time from the start of the bot behavior to the point at which the malicious UE is effectively blocked from the network. To do so, we perform a ping from within the UE, setting a 0.5-second timeout for responses. We use ping failures as an indication that the UE has been banned from the network. For each data collection interval (1, 3, and 5 seconds), corresponding to UPF EES data collection period, we repeated the experiment 20 times and report the mean and standard deviation. Fig. 8 shows the result of this experiment.

In this figure, $t1$ shows the time taken after the start of the attack to see the first report of the UE malicious traffic on NWDAF SBI. In all cases, the average time to receive

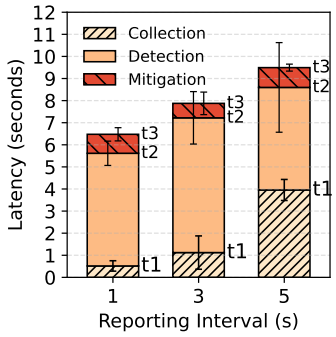


Figure 8 – Response time breakdown for automated attack mitigation with NWDAF

the first report at NWDAF is shorter than the requested subscription periods.

Upon subscription, the NWDAF sends a report of abnormal behavior to the SMF every second; however, it takes until $t2$ for the NWDAF engine to have sufficient data to detect the bot pattern. After detection, the SMF receives a notification from the NWDAF and sends a request to the UPF to release the PDU session via the N4 interface. It takes until $t3$ for the UE to begin experiencing ping failures, indicating that it has been banned from the network.

While the mitigation time is nearly the same across all three cases, we observe that shorter data collection intervals lead to faster mitigation. Additionally, the primary latency bottleneck in this pipeline is the time required for the ML model to detect abnormal traffic patterns in the collected data.

4.6 Discussion

This section demonstrates how the proposed closed-loop system can be applied to a network security scenario. Through this application, we demonstrate that UPF EES reports can be used for anomaly detection, and, through the bot detection use case, we show that these reports can capture traffic patterns more complex than irregular traffic volume alone. We further show that NWDAF can effectively use the runtime data it collects to derive analytics and provide them to the SMF for runtime action.

The evaluation of the end-to-end system responsiveness shows that the time required for the ML model to detect abnormalities in the collected data is the primary latency bottleneck. In addition, the results show that shorter data collection periods lead to faster mitigation. This highlights an inherent trade-off: collecting data over a longer period yields more information and can improve detection accuracy, but it also increases response time. As a result, an important design question is how much detection confidence is required in a given control task before triggering an action. The answer depends on the model characteristics, the acceptable response time, and the required detection precision.

5. DYNAMIC QOS ADAPTATION UNDER CONGESTION VIA NWDAF-DRIVEN ANALYTICS AND PCF

This section illustrates how NWDAF analytics can be integrated into a closed-loop control workflow, enabling fine-grained policy adaptation at runtime via the PCF using standardized interfaces.

5.1 Application scenario and objectives

Future mobile networks are expected to accommodate users and services with heterogeneous QoS requirements, including varying sensitivities to delay and throughput, and varying tolerance to service degradation. At the same time, traffic demand and network conditions evolve over time and are often non-stationary. As a result, policy settings that are suitable under one operating condition may become less effective as traffic patterns change. Since maintaining acceptable QoS under varying network conditions is essential, the network requires mechanisms that can adapt policy decisions at runtime rather than relying exclusively on static provisioning. There are several scenarios in which static policies are not efficient, and dynamic policy adaptation is needed. Table 4 summarizes these scenarios. the NWDAF, as a dedicated analytics function in 5G networks, has access to periodically collected runtime data from various sources. These data reflect the current condition of the network and can also indicate its near-future state. This makes the NWDAF a natural enabler of closed-loop QoS adaptation, as it can continuously analyze the collected data and derive actionable analytics.

The policy control function can naturally act as a consumer of this NWDAF application to enable closed-loop management, since it can adapt policy parameters at varying levels of granularity. By adjusting MBR and GBR, the PCF can enforce adaptation at the level of individual GBR flows, while changing the Session-AMBR affects the policy applied to all Non-GBR flows within a session. Through periodic notifications from the NWDAF, PCF can continuously assess whether policy adaptation is required and apply an appropriate action, thereby enabling a zero-touch management mechanism.

In this section, we study another closed-loop control system enabled by the NWDAF framework presented in Section 3. This allows us to investigate whether UPF EES QoS monitoring reports can be used to derive analytics that reflect the network state, and whether the PCF, as a

Table 4 – Representative runtime scenarios in which dynamic QoS adaptation can be beneficial

Use case	Description	Representative action
Congestion	The network may experience congestion due to temporary resource limitations or due to receiving more traffic than expected. This can lead to QoS degradation. In such cases, QoS settings that were suitable before may no longer be sufficient. Dynamic QoS adaptation can then help protect users while also helping network conditions return to normal.	Adjustment of QoS-related parameters, particularly MBR and AMBR, and, in harsher cases, the GBR of selected users.
Traffic pattern change	The traffic pattern of active users may change over time. For example, users who previously used only part of their allocated bandwidth may start using more of it, making the existing QoS settings less suitable. In such situations, dynamic QoS adaptation can help the network keep a better match between configured policies and the current traffic behavior.	Runtime adjustment of QoS-related policy parameters according to the observed traffic profile.
Malicious condition	In the presence of malicious or abnormal traffic conditions, applying immediate hard mitigation may be unnecessarily disruptive. In such situations, dynamic QoS adaptation can serve as a softer response, first limiting the impact and giving the system time to investigate further. When selective enforcement is possible, such adaptation may also reduce the impact on legitimate traffic while maintaining network protection.	Temporary bit rate reduction or controlled throttling before applying harsher mitigation actions.

consumer of these analytics, can adjust QoS parameters in order to improve network conditions.

5.2 QoS analytics pipeline

We utilize the framework presented in Section 3 to collect and analyze data for actionable QoS analytics. Within this framework, the NWDAF SBI collects data from the UPF EES, and the engine analyzes it to generate reports for PCF. We use the QoS monitoring report as our primary data source because it provides delay and packet-loss indicators that signal QoS degradation. In the following sections, we further discuss the role of the NWDAF in realizing closed-loop QoS control, from data collection to the generation of runtime analytics.

5.2.1 Data sources and collection

We use the QoS monitoring reports from the UPF EES as the primary data source and the basis for runtime analytics for this operational objective. These reports provide parameters that may serve as useful indicators of QoS degradation, including internal UPF uplink and downlink delays at the flow granularity, as well as packet-loss measurements in each direction. The NWDAF SBI subscribes to the QoS monitoring reports exposed by the UPF EES. Per-flow granularity is selected for these reports, as it allows the analytics engine to observe individual flow behavior and helps identify both the flows that consume the most bandwidth and those most affected, i.e., those experiencing higher delay than others. The reporting period is set to one second, which provides sufficient observability of these parameters over time.

These reports are informative because the internal delay measured at the UPF reflects what the UE experiences at the time of reporting. A single delay value, therefore, reflects the current condition of the flow, while the delay trend across several reporting intervals reveals whether the degradation is transient or whether a more persistent condition, such as congestion, is building up. Packet loss may occur for various reasons, and increased packet loss also affects the QoS perceived by the UE. Its trend over time can therefore provide an additional indication of congestion or service disruption.

Instead of relying on a single report, the NWDAF analytics engine retrieves a set of recent reports and uses their timestamps to assess the current network condition and the evolution of packet loss and delay over time. These collected, time-indexed reports are then used by the analytics engine for preprocessing, trend analysis, and report generation.

5.2.2 Selected NWDAF report type and output mapping

3GPP specification [13] defines user data congestion analytics as one of the reports that NWDAF can provide to its subscribers regarding congestion in the control plane, the user plane, or both. According to this specification, the NWDAF may collect the required data from sources such as the AMF, OAM, and UPF.

One of the output parameters of this analytics report is

the Network Status Indication (NSI). The NSI reflects the level of congestion that the network is experiencing; however, its exact definition is not standardized and may therefore vary across implementations. These reports may be either statistical or predictive. For user-plane congestion analytics, the reports may include the top traffic-contributing applications in both uplink and downlink directions. In the predictive form, these correspond to the applications that the NWDAF predicts will contribute most to upcoming traffic. In our implementation, each generated congestion report applies to the recent observation window used by the analytics engine. Since the engine derives each report from the latest 15 seconds of QoS monitoring data, we treat this interval as the applicable time window of the reported congestion state.

In this section, we present NWDAF statistical analytics for user-plane congestion. We selected this report because the network status indication can be configured to represent different levels of QoS degradation in the user plane, thereby enabling the PCF to adjust its actions accordingly. While [13] does not include confidence as part of the statistical analytics output for this report, we include it in our engine output. Since the output of this report is used in a closed-loop orchestration setting, it is important to differentiate actions based on the confidence of the derived NSI.

The specification [13] lists average and peak throughput as parameters that the UPF can provide to the NWDAF for use in this analytics report. While these parameters can be sufficient for identifying the top contributing applications, metrics such as UPF internal delay and packet loss can provide more suitable signals for detecting or reporting congestion. Therefore, although they are not listed in [13] as UPF inputs for this report, we use UPF QoS reports as our primary data source, as explained in Section 5.2.1. In 3GPP terms for this report, if the analytics target is any UE, the subscriber should specify the areas of interest, such as a list of cell IDs. However, in our current setup, which includes only one slice and one gNB, selecting any UE effectively covers all UEs in the network, so no additional area-of-interest filtering is applied.

To summarize, our user-plane congestion report contains three main elements: the network status indication, the confidence, and the list of the top traffic-contributing uplink and downlink flows. In our implementation, these contributors are identified at the flow level because the UPF data available to the NWDAF is flow-based rather than application-based.

5.2.3 Feature construction

We define two time windows for retrieving data from the database: a control window and a state window. The control window corresponds to the reporting period requested by the subscriber and includes the most recent reports used to assess the current network condition. The state window is a longer observation window that includes the control window and captures the network state's recent history. After retrieving the QoS reports for these two windows from the database, we process them to derive the parameters for the user data congestion report. The following sections describe our method for deriving each parameter.

5.2.3.1 Network status indication

We determine the network status by evaluating the severity, persistence, escalation, and scope of the observed delay and packet-loss conditions. We define four categories for delay values: healthy, warning, degraded, and critical. Separating flows into these categories requires threshold values. We selected these thresholds empirically by running the system and observing the reported delays under different network conditions.

- **Delay severity.** For the flows in the control window and the state window, we calculate the mean delay for each flow in the uplink and downlink directions and classify each directional value as healthy, warning, degraded, or critical according to the defined thresholds. We then assign an overall severity label to the flow in each window based on the worst category observed in both directions. This provides an estimate of the delay's severity for that flow.
- **Delay persistence.** Since each flow may have multiple reports within the state window, we examine the delay reported in each sample separately for the uplink and downlink directions. For each direction, we then calculate the fractions of samples whose delays exceed the degraded and critical thresholds. If at least 50% of the samples of a flow exceed the degraded threshold in one direction, we consider the flow persistently degraded in that direction. The flow is then marked as persistently degraded overall if this condition holds in at least one direction.
- **Delay escalation.** We calculate the difference between the latest and the first reported delay for each flow in the uplink and downlink directions within the state window. If this increase exceeds a specified threshold in either direction, we consider that direction escalating and classify the flow as escalating as well.
- **Delay scope.** We consider a flow to be affected in a given direction if its severity in that direction is classified as degraded or critical, or if it is persistently degraded or persistently critical in that direction. A flow is considered overall affected if at least one of

its directions is affected. We then calculate, for each UE, the ratios of affected flows in uplink, downlink, and overall to the total number of flows associated with that UE. A UE is considered broadly affected in a given direction if at least 50% of its flows are affected in that direction. Similarly, a UE is considered broadly affected *overall* if at least 50% of its flows are affected in either direction.

- **Packet loss.** We use the same logic for the experienced packet loss. For each direction, we derive the packet-loss severity from the mean loss observed in the state and control windows. To capture persistence, we examine the packet-loss reports within the state window and calculate the fraction of samples that exceed the degraded and critical thresholds. If at least 50% of the samples exceed the degraded threshold in a given direction, we consider packet loss to be persistently degraded in that direction. Similarly, if at least 30% of the samples exceed the critical threshold, we consider packet loss to be persistently critical in that direction. We consider packet loss to be escalating in a given direction if the mean packet loss over the control window exceeds that over the earlier part of the state window, excluding the control window itself. Considering that packet loss is already reported at the UPF level, we do not derive a separate scope measure for it.

We derive one congestion level from delay and another from packet loss, then combine them to obtain the final network status indication. For the congestion level based on packet delay, we consider only flows present in the control window. However, for these flows, we still use the features calculated over the state window. We use the scoring method described in Table 5 to derive the score of each flow, and we assign a final score to each flow by applying different weights, based on Table 6, to the different feature scores, as shown in Equation *(1)*. Finally, the mean of the weighted scores of all considered flows is divided by the maximum possible score and normalized to obtain the final delay-based congestion level, as shown in Equation *(2)*.

$$W_i = w_s S_i + w_p P_i + w_e E_i + w_{sc} C_i \qquad (1)$$

$$\mathrm{NSI}_{\mathrm{delay}} = 100 \cdot \frac{\frac{1}{N}\sum_{i=1}^{N} W_i}{3w_s + w_p + w_{sc} + w_e} \qquad (2)$$

For packet loss-based congestion, we use the scoring described in Table 8. We use the same severity, persistence, and escalation weights as in Table 6, excluding the scope weight, since scope is not considered in the packet loss-based score. The final weighted score is then normalized according to Equation *(3)* to derive the final packet-loss-based congestion level.

$$\mathrm{NSI}_{\mathrm{packet\_loss}} = 100 \cdot \frac{w_s S_{\mathrm{pl}} + w_p P_{\mathrm{pl}} + w_e E_{\mathrm{pl}}}{3w_s + w_p + w_e} \qquad (3)$$

We derive the final NSI by Equation *(6)*. While we let the worst value between $\mathrm{NSI}_{\mathrm{delay}}$ and $\mathrm{NSI}_{\mathrm{packet\_loss}}$ dominate the reported result, we add 20% of the smaller value to distinguish cases in which both of them are high.

$$\mathrm{NSI}_{\max} = \max\left(\mathrm{NSI}_{\mathrm{delay}}, \mathrm{NSI}_{\mathrm{packet\_loss}}\right) \qquad (4)$$

$$\mathrm{NSI}_{\min} = \min\left(\mathrm{NSI}_{\mathrm{delay}}, \mathrm{NSI}_{\mathrm{packet\_loss}}\right) \qquad (5)$$

$$\mathrm{NSI} = \min\left(\mathrm{NSI}_{\max} + 0.2 \cdot \mathrm{NSI}_{\min}, 100\right) \qquad (6)$$

5.2.3.2 Confidence

We include confidence in the user data congestion report to indicate how much data the engine had to rely on to derive its result. Since we use data from both the control and state windows, we account for the availability of data in each window when calculating confidence. We consider one data point per second per window to be the maximum required to support the analytics, and we evaluate confidence against this reference. For each signal, delay, and packet loss, we compute the ratio of actual samples to expected samples in both the state and control windows. The state window receives a weight of 0.6 and the control window 0.4, as the state window provides a longer view of the network conditions, while the control window confirms that the data is still recent. Since the final congestion level combines both packet loss and delay signals, we apply the same logic to the confidence: the dominant signal receives a weight of 0.8, and the secondary a weight of 0.2. Therefore, the overall confidence reflects how well each signal's contribution to the congestion level is supported.

5.2.3.3 Top traffic-contributing flows

For each traffic direction, we report the top five flows that contributed to the traffic, along with their contribution ratios, which range from 0 to 100.

5.2.3.4 Applicable time window

As we use the state window to derive longer-term insights from the collected data, we report its interval as the applicable time window.

5.2.4 Runtime integration in NWDAF

The UPF EES reports UPF's internal delay. Considering the values of these reports under different network conditions, we defined our delay thresholds as shown in

Table 5 – Scoring of the derived delay-based features for congestion-level estimation

Indicator	Symbol	How it is derived
Severity score	S_i	For each flow active in the control window, we map the delay severity category derived for the control window to a numerical severity score according to Table 7.
Persistence score	P_i	For each flow active in the control window, we assign a score of 1 if the flow is marked as persistently degraded or persistently critical based on the state window; otherwise, the score is 0.
Escalation score	E_i	For each flow active in the control window, we assign a score of 1 if the flow is marked as escalating based on the state window; otherwise, the score is 0.
Scope score	C_i	For each flow active in the control window, we assign a score of 1 if the flow belongs to a UE that is broadly affected according to the state window; otherwise, the score is 0.

Table 6 – Weights used for congestion-level estimation

Indicator	Symbol	Weight
Severity	w_s	1.0
Persistence	w_p	0.5
Scope	w_{sc}	0.5
Escalation	w_e	0.5

Table 7 – Severity score mapping

Severity category	Score
Healthy	0.0
Warning	1.0
Degraded	2.0
Critical	3.0

Table 9. Additionally, we used the thresholds defined in Table 10 to map the experienced packet loss to a severity category.

At each reporting period requested by the analytics consumer (in this case, PCF), the engine retrieves the QoS reports for the last 15 seconds from the database. It then uses this data to derive the features and confidence values discussed in Section 5.2.3. Finally, the engine encapsulates the result into the required JSON format and passes it to the NWDAF NBI event module for delivery to the client.

5.3 PCF-based closed-loop mitigation

Given PCF's central role in policy enforcement, it is a natural candidate to consume NWDAF analytics to enable dynamic QoS adaptation. As discussed in Section 2.3, the PCF can enforce policies at different levels of granularity, including the flow level using PCC rules, the session level using the Session-AMBR, and the UE level using the UE-AMBR [14]. This allows policy actions to be tailored to the scope and severity of the detected network condition. Additionally, the PCF can apply graduated responses by adjusting parameters such as AMBR and MBR. This is a less disruptive approach than the PDU session release discussed in Section 4, as it allows the system to throttle a specific flow while the rest continue operating normally. For these reasons, we extend the PCF to subscribe to NWDAF congestion analytics and act as the enforcement point for QoS adaptation in this use case.

5.3.1 PCF subscription to NWDAF

We extend the PCF to operate as an active NWDAF consumer; Fig. 9 illustrates the components added to PCF.

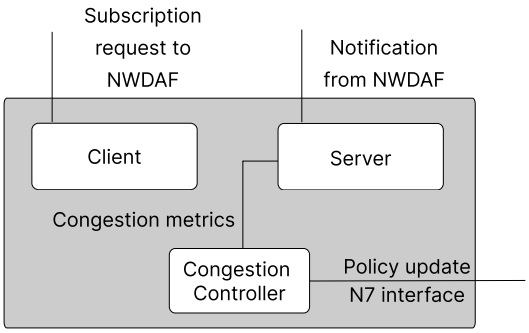


Figure 9 – PCF as NWDAF subscriber

We integrated server and client components into the PCF to enable it to subscribe to the NWDAF and receive notifications. The client component sends a one-time subscription request to the NWDAF for user-data congestion analytics.In this request, it specifies periodic notification as the reporting method, together with the desired reporting period. Since the subscription target is any UE, the PCF provides network area information. However, because our setup consists of a single gNB, this information remains the same across all subscriptions. Listing 9 shows the PCF subscription request sent to the NWDAF.

Table 8 – Scoring of the derived packet loss-based features for congestion-level estimation

Indicator	Symbol	How it is derived
Severity score	S_{pl}	We derive the packet-loss severity separately for the uplink and downlink directions from the packet-loss values observed in the control window. The severity score is then assigned as the worst of the two directional severity scores, using the same severity-to-score mapping as in Table 7.
Persistence score	P_{pl}	We assign a persistence score of 1 if packet loss is marked as persistently degraded or persistently critical in at least one direction based on the state window; otherwise, the score is 0.
Escalation score	E_{pl}	We assign an escalation score of 1 if packet loss is marked as escalating in at least one direction based on the comparison between the control window and the earlier part of the state window; otherwise, the score is 0.

Table 9 – Delay severity thresholds used for flow classification

Severity category	Condition on mean delay
Critical	$\geq 200\,\mu s$
Degraded	$\geq 100\,\mu s$ and $< 200\,\mu s$
Warning	$\geq 80\,\mu s$ and $< 100\,\mu s$
Healthy	$< 80\,\mu s$

Table 10 – Packet-loss severity thresholds used for classification

Severity category	Condition on packet-loss count
Critical	≥ 100 packets
Degraded	≥ 50 packets and < 100 packets
Healthy	< 50 packets

```
{"notificationURI": "http://pcf-npcf.open5gs.svc.
    cluster.local:8081/notification",
    "eventSubscriptions": [{
            "event": "USER_DATA_CONGESTION",
            "notificationMethod": "PERIODIC",
            "networkArea": {
                "gRanNodeIds": [{
                        "plmnId": {"mcc": "001", "
                            mnc": "01"},
                        "gNbId": {"bitLength": 32,"
                            gNBValue": "1"}}]
            },
            "tgtUe": {"anyUe": true},
            "repetitionPeriod": 5}]}
```

Listing 9 – PCF subscription request to NWDAF for user data congestion

The server module receives NWDAF notifications at the callback endpoint exposed by the PCF. It parses the raw notification payload, extracts the congestion metrics relevant to policy adaptation, and forwards them to the congestion controller in an internal format. Listing 10 illustrates a sample NWDAF notification containing the user data congestion report delivered to the PCF.

```
{ "networkArea":{"gRanNodeIds":[...]},
   "congestionInfo":{
      "congType":{"congType":"USER_PLANE"},
      "nsi":{"congLevel":2,
         "avgPacketDelay":48,
         "maxPacketDelay":213},
      "confidence":35,
      "topAppListUl":[
         {"ipTrafficFilter":{
               "flowId":1,
               "flowDescriptions":["src=10.42.0.3,
                   dst=129.97.168.127, sport=47083
                   , dport=27000"]},
               "ratio":20},...],
      "topAppListDl":[
         {"ipTrafficFilter":{
               "flowId":1,
               "flowDescriptions":["src=10.42.0.6,
                   dst=129.97.168.127, sport=6002,
                    dport=6002"]},
               "ratio":33},...]
   },
}
```

Listing 10 – Sample NWDAF notification for user data congestion

The congestion controller implements the logic for adapting policies based on the network status reflected in the NWDAF notifications. Using the heuristic algorithm discussed in Section 5.3.2, it determines the appropriate policy adjustment and sends the corresponding update to SMF for enforcement.

5.3.2 Heuristic policy decision logic

We design a simple heuristic policy decision algorithm to adjust policies based on the reported network status and the confidence in the report. While heuristic approaches are simpler than optimization-based or reinforcement learning-based methods, they provide an interpretable and practical starting point for comparing static and dynamic policies. We therefore use this design to gain an initial understanding of the feasibility of dynamic policy adjustment and its potential to improve network conditions.

As discussed in Section 5.2.3, the congestion level reported in the NWDAF notification reflects the extent to which the network is affected by ongoing traffic, with

higher packet loss and delay translating into a higher congestion level. Policy adjustments should therefore be tied to this congestion level, as more severe conditions require more aggressive actions. In addition, the confidence parameter indicates the reliability of the derived NWDAF result. The policy decision should therefore also take confidence into account, so that uncertain conditions lead to more conservative actions than highly reliable ones.

Algorithm 1 summarizes the controller logic used to translate the reported congestion level and confidence into a policy adjustment. Taking these two parameters into account, for each congestion level and confidence value reported in a notification, the congestion controller first uses the mappings shown in tables 11 and 12 to derive the corresponding factors. The mitigation factor serves as a multiplicative coefficient that adjusts the current user rate based on the reported congestion level. The mapping in Table 11 is designed so that low congestion levels, i.e., levels below 30, do not trigger a rate change, while increasingly severe congestion leads to progressively stronger rate reduction.

Table 11 – Mapping from reported congestion level to mitigation factor

Congestion level range	Mitigation factor
0–29	1.00
30–39	0.90
40–49	0.80
50–59	0.70
60–69	0.55
≥ 70	0.40

Table 12 – Mapping from report confidence to confidence factor

Confidence range	Confidence factor
0–14	0.00
15–39	0.40
40–74	0.75
75–100	1.00

The congestion controller uses the confidence factor derived from Table 12 to further scale the user-rate adjustment. Under this mapping, reports with insufficient confidence (i.e., confidence values below 15) do not trigger any rate adjustment, regardless of the reported congestion level. For reports with higher confidence, the resulting adjustment is scaled by the corresponding confidence factor, such that lower-confidence reports lead to more conservative actions. Therefore, the controller computes the final user-rate adjustment factor as

$$\text{f\_adj} = 1 - \Big((1 - \text{f\_mit}) \cdot \text{f\_conf}\Big) \qquad (7)$$

where f_adj, f_mit, and f_conf denote the adjustment, mitigation, and confidence factors, respectively. Following this equation, lower confidence reduces the strength of the congestion-driven adjustment.

The adjustment factor derived by this approach considers only the metrics extracted from the most recent NWDAF notification. However, if the reported congestion level has not improved since the previous action, or if the improvement is not significant, the next adjustment should be stronger than the previous one. To achieve this, we compare the current congestion level reported by the NWDAF with the previous one. When the improvement is insufficient, we further compare the newly derived adjustment factor with the factor applied in the previous step. If the new factor is equal to, or less aggressive than, the previous one, the controller instead decreases the previous factor by 0.1 to derive the new adjustment factor.

In contrast, the controller also requires a mechanism to relax the applied action when the congestion level has sufficiently improved. In our implementation, instead of immediately applying a relaxation based on the most recent NWDAF notification, the controller waits until the improvement is observed over two consecutive notifications, each reported with at least 50% confidence. Once this condition is satisfied, the recovery mechanism is triggered. When the improvement is confirmed, the controller compares the adjustment factor derived from Equation *(7)* with the previously applied factor. If the derived factor exceeds the previous factor by more than 0.1, the controller increases the applied factor by only 0.1. Conversely, if the derived factor is smaller than the previous factor, the controller keeps the previous factor unchanged.

5.3.3 AMBR/MBR enforcement workflow

Once the controller derives the adjustment factor, it applies it to the non-guaranteed bit rate parameters that can be safely adapted at runtime, at both per-flow and per-session granularity. At the flow level, the MBR is a non-guaranteed rate parameter and is therefore suitable for runtime adjustment. The controller thus scales the MBR of each PCC rule by the derived adjustment factor, provided that the resulting value does not fall below the corresponding GBR. At the session level, the Session-AMBR defines the aggregate bit rate limit for Non-GBR flows and can likewise be adjusted under congestion. The controller, therefore, applies the same adjustment factor to the Session-AMBR as part of the mitigation process. The resulting policy update is then sent by the PCF to the SMF for enforcement.

For each active session, after applying the adjustment factor to the MBR of each PCC rule and the Session-AMBR, the controller compares the resulting policy with

Algorithm 1 Heuristic update of the adjustment factor

Require: current congestion level c, previous congestion level c_{prev}, confidence value γ, previous adjustment factor r_{prev}, improvement counter n_{imp}
Ensure: Updated adjustment factor r

Derive mitigation factor r_{mit} from the congestion-level mapping
Derive confidence factor r_{conf} from the confidence mapping
$r \leftarrow 1 - \big((1 - r_{\text{mit}}) \cdot r_{\text{conf}}\big)$
if $c_{\text{prev}} < 0$ **or** $r_{\text{prev}} \leq 0$ **then**
    return r
end if
if $c \geq c_{\text{prev}}$ **or** $(c_{\text{prev}} - c) < 10$ **then**
    if $r \geq r_{\text{prev}}$ **then**
        $r \leftarrow r_{\text{prev}} - 0.1$
        $r \leftarrow \max(r, 0.05)$
    end if
else if $(c_{\text{prev}} - c) \geq 10$ **then**
    if $n_{\text{imp}} \geq 2$ **then**
        if $r \geq r_{\text{prev}} + 0.1$ **then**
            $r \leftarrow r_{\text{prev}} + 0.1$
        end if
        if $r \leq r_{\text{prev}}$ **then**
            $r \leftarrow r_{\text{prev}}$
        end if
    else
        $r \leftarrow r_{\text{prev}}$
    end if
end if
$r \leftarrow \min(1, \max(0, r))$
return r

the previously applied congestion-adjusted policy stored in the PCF. If a difference is detected, the PCF invokes the SM Policy Control Update Notify service to notify the SMF of the updated policy. Otherwise, the update is skipped, thereby avoiding unnecessary signaling when the computed policy remains unchanged. Fig. 10 shows the closed-loop workflow from data collection to policy adjustment.

5.4 Evaluation

We evaluate the proposed system to demonstrate its effectiveness. In this section, we first describe the experimental setup and then present each experiment and its results.

5.4.1 Experimental setup

We use the same testbed as in Section 4.5, with the addition of the features implemented to realize dynamic QoS adaptation under congestion using the NWDAF. To evaluate the effectiveness of our system in adapting policies, we constrain the UPF resources to make it the bottleneck in the forwarding path, while UERANSIM is provisioned with sufficient resources so that it does not become the limiting factor. Table 13 summarizes the resources allocated to the UPF. This makes QoS degradation observable at lower traffic rates, thereby making the experiments easier to conduct. We use another server in our lab as the traffic destination for the UE-generated packets, thereby keeping the end-to-end packet path short. This allows us to better isolate the effect of UPF forwarding on the observed results.

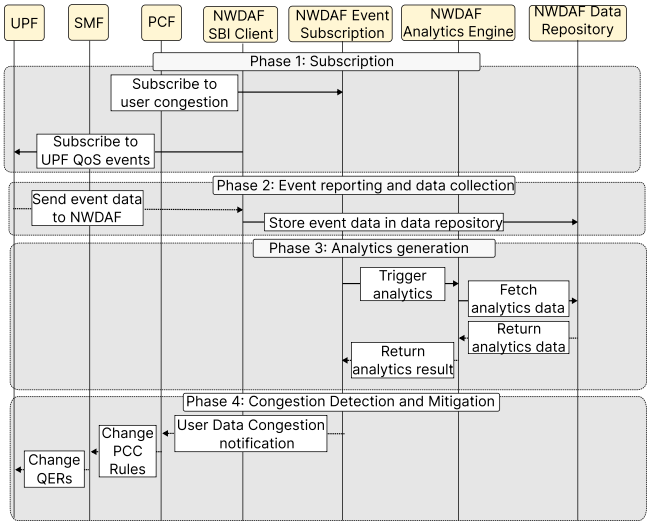


Figure 10 – Sequence diagram for the closed-loop policy adjustment workflow

To observe the effect of PCF rate adjustment on GBR flows, we defined two PCC rules and registered them in the UE profile during system initialization. These rules define synthetic GBR traffic profiles intended to represent different bit rate requirements across traffic classes. Table 14 summarizes the two configured PCC rules. We also set the AMBR of all UEs to 15 Mbps in the downlink and 100 Mbps in the uplink. Although our implementation supports rate adjustment in both directions, we could not evaluate uplink GBR enforcement in our testbed because the lack of RAN support causes uplink traffic to match the default uplink rule rather than the configured GBR-specific rules. We therefore keep the uplink AMBR sufficiently high to avoid introducing uplink bottlenecks that could affect the results.

Table 13 – Resource allocation for the UPF pod

Resource	Request	Limit
CPU	200m	400m
Memory	512Mi	1024Mi

Table 14 – Configured GBR PCC rules used in the evaluation.

PCC Rule	Proto.	Port Range	GBR UL/DL	MBR UL/DL	5QI
VoIP media	UDP	16384–20000	64/64 Kbps	128/128 Kbps	1
Real-time gaming	UDP	27000–27999	5/10 Mbps	10/20 Mbps	3

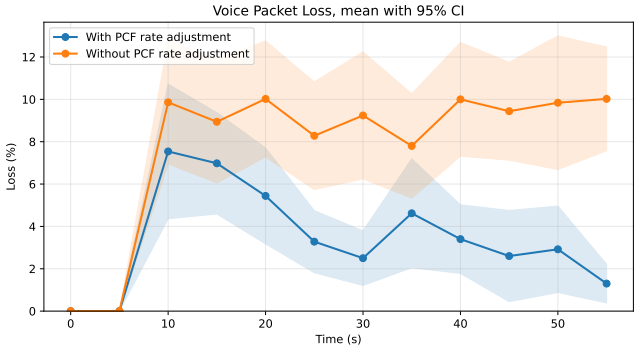


Figure 11 – Effect of PCF rate adjustment on light voice packet loss.

5.4.2 *Impact of PCF-based rate adaptation on GBR traffic*

To demonstrate the effect of PCF action on the performance of guaranteed bit rate traffic, we compare the packet loss and delay experienced by GBR flows in two cases: when PCF subscribes to NWDAF and can adapt network policies accordingly, and when it does not subscribe and therefore performs no policy adaptation. We compare these two cases in scenarios where the load on the UPF is caused either solely by high-volume non-GBR traffic or by a mixture of GBR and non-GBR traffic. In these experiments, we disable recovery of the adjusted rate so that we can observe the effect of the PCF action without PCF later relaxing the rate.

5.4.2.1 GBR traffic under heavy non-GBR load

In this experiment, we generate background traffic to create a load on UPF. Specifically, a set of UEs receives non-GBR downlink traffic from the server, generated

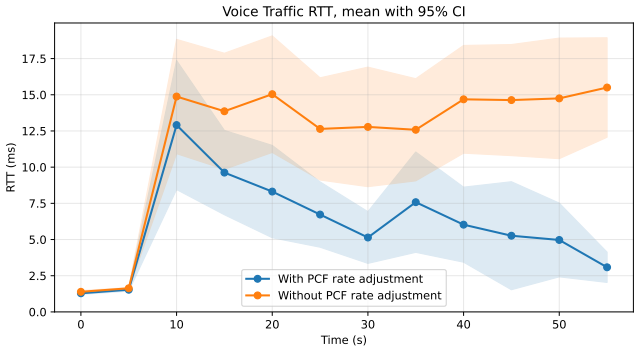


Figure 12 – Effect of PCF rate adjustment on light voice traffic RTT.

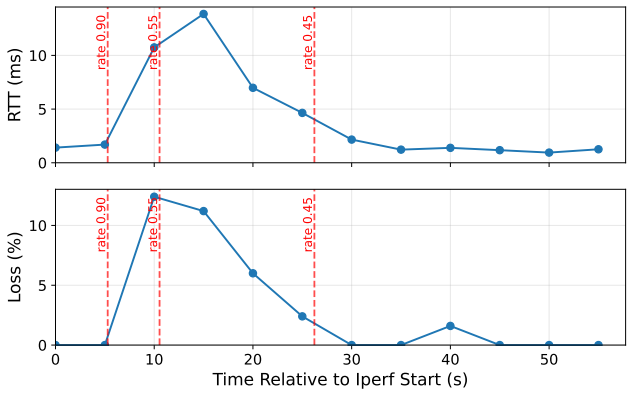


Figure 13 – Voice traffic RTT and packet loss under PCF-enforced rate limiting, showing resilience of GBR flows.

using iperf. For the GBR PCC rules, we use lightweight UDP traffic. Consequently, the packet-loss results do not reflect a direct effect of rate adaptation on the GBR flows themselves; rather, they show how throttling the non-GBR flows affects the conditions experienced by the GBR flows. To generate and measure this lightweight GBR traffic, we used a custom script that ran on the UE. The script periodically sends UDP probe packets for 60 seconds, while a parallel receiver thread collects echoed packets and measures the Round-Trip Time (RTT) for each probe. Each packet carries a sequence number and a timestamp, allowing the script to compute per-packet RTT and identify packet losses through timeouts.

We repeated this experiment for 20 rounds, each lasting 60 seconds. In each round, three UEs generated background traffic, while the packet loss and average RTT of the GBR traffic were logged every second. We then averaged the results across the 20 rounds. Figures 11, and 12 present the results of this experiment, with each point showing the average over a 5-second window.

As shown in figures 11 and 12, enabling PCF-driven rate adjustment improves the conditions experienced by the GBR flows under background load. This improvement is not immediate. Instead, packet loss and RTT begin to decrease after an initial delay, reflecting the time required for the closed-loop process to observe the network condition, generate analytics, and apply the updated policy. Since rate adaptation is applied to non-GBR traffic, the gain observed for GBR flows is indirect. By throttling competing non-GBR traffic, the PCF reduces congestion at the UPF, which in turn improves the service conditions for GBR flows. This trend is observed in both voice and gaming traffic, as both operate below their GBR, indicating that the proposed mechanism can help protect GBR traffic under load.

Without rate adjustment, the packet loss of the guaranteed flow remains between approximately 8% and 10%

throughout the experiment. In contrast, when PCF-based rate adjustment is enabled, the packet loss decreases over time, from about 8% at the beginning of the experiment to around 1% by the end (Fig. 11). A similar trend can be observed for RTT. Without PCF action, the RTT remains between approximately 12.5 and 15 ms throughout the experiment, whereas with PCF-based rate adjustment, it decreases from about 12.5 ms at the beginning of the experiment to around 3 ms by the end (Fig. 12).

As discussed in Section 5.3.2, the PCF adjusts the rate according to the confidence and congestion level reported by the NWDAF. Across different experiment runs, the PCF may take no action, even when the control logic is triggered, if the NWDAF does not report an actionable congestion level. In other cases, it may apply different levels of rate adjustment over time. Fig. 13 presents one experimental run, showing when the PCF applied rate changes during the 60-second interval and how packet loss and RTT evolved over the same period. As the figure shows, the first applied rate adjustment is 0.90. After some time, PCF applies a stronger adjustment of 0.55, and the final applied rate is 0.45. After this point, PCF does not apply any further rate updates. While packet loss begins to decrease after the 0.55 rate adjustment, the RTT starts to decrease about 5 seconds later. Following the final action, the RTT remains stable, while packet loss decreases and stays below 2.5%.

5.4.2.2 GBR traffic under combined GBR and non-GBR load

In this experiment, we combine non-GBR traffic with gaming and voice GBR traffic to put pressure on the UPF. The gaming flow receives downlink traffic at 15 Mbps, which lies between its GBR and MBR, while the voice traffic receives data at 60 Kbps, slightly below its GBR. In this setting, we expect the rate adjustment to reduce the congestion experienced by the voice flow, since it operates below its GBR. In contrast, we expect the gaming flow to experience packet loss when the PCF reduces its rate below the original MBR.

Figures 14, 15, and 16 compare the packet loss experienced by each traffic type in the cases with and without PCF-based rate adjustment. Fig. 14 shows that the loss experienced by non-GBR traffic increases when the PCF adjusts the rate based on reports received from the NWDAF. When the UPF experiences overload, and this condition is reflected in the NWDAF reports, the PCF reduces the AMBR to protect the GBR flows. The effect of this throttling is evident in the higher loss experienced by non-GBR traffic in this experiment.

Fig. 15 shows the results for the gaming traffic. Since gaming traffic transmits above its GBR, when the PCF throttles its MBR, it limits the excess portion of the flow

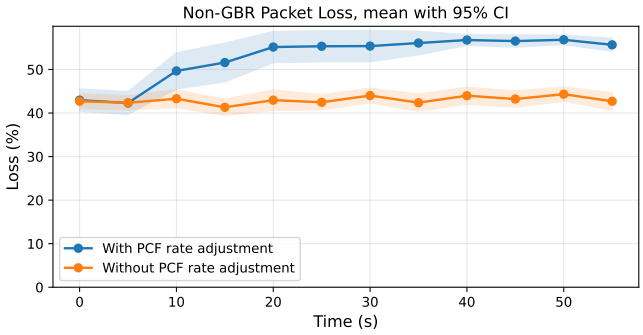


Figure 14 – Effect of PCF rate adjustment on non-GBR packet loss.

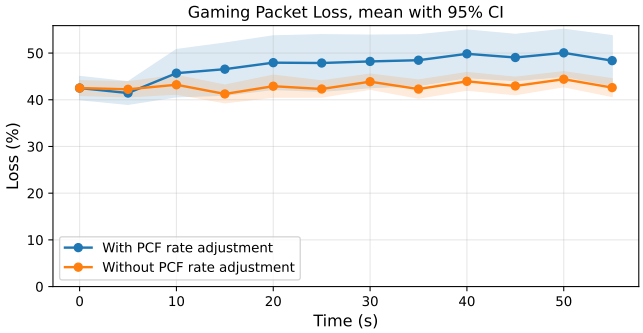


Figure 15 – Effect of PCF rate adjustment on gaming packet loss.

under network pressure. However, because the number of non-GBR flows remains higher in this experiment, the gaming traffic is already experiencing packet drops at the UPF. Converting part of this drop into intentional throttling increases gaming packet loss by only about 10%, while helping protect the other GBR flows. As Fig. 16 shows, the loss of the voice traffic decreases from 40% to 20% when PCF-based rate adjustment is enabled, indicating that this flow is protected.

6. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our NWDAF-based closed-loop system under the bot de-

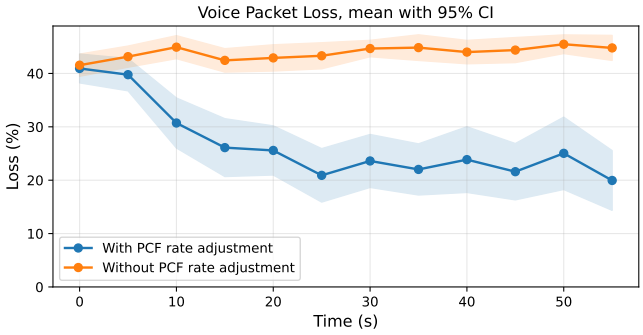


Figure 16 – Effect of PCF rate adjustment on voice packet loss.

tection use case, focusing on two critical aspects: (i) the overhead and scalability of our UPF EES (Section 6.1), (ii) NWDAF resource consumption under varying workloads (Section 6.2). Our evaluation demonstrates that the system introduces negligible overhead while enabling real-time analytics and automated threat response.

6.1 UPF overhead and scalability

UPF is a critical component in the 5G user plane, and any additional processing overhead directly impacts network throughput and latency. We examine how well our EES implementation scales and how it affects UPF packet forwarding. For this, we compare three UPF variants:

- Unmodified Open5GS UPF (Baseline).
- UPF with the EES enabled but no active subscriptions, accounting for control overhead only (EES: 0 Subscriber).
- UPF with one active subscriber receiving per-flow volume reports every 3s (EES: 1 Subscriber).

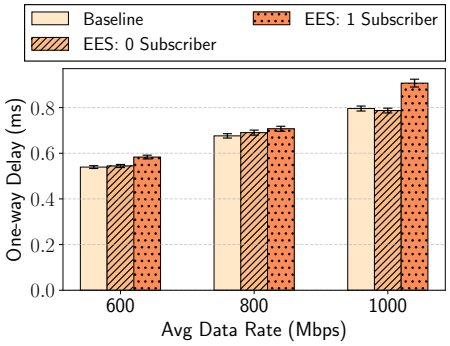


Figure 17 – Impact of data rate on UPF performance across latency

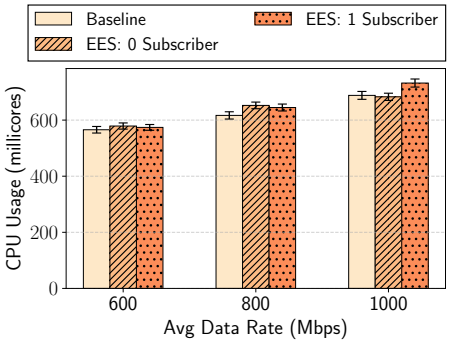


Figure 18 – Impact of data rate on UPF performance across CPU usage

Since the UPF EES performs real-time data collection and reporting, it may introduce additional CPU overhead or delay packet forwarding. We evaluate this overhead along two critical stress dimensions: (i) increasing data rate, which stresses per-packet processing load, and (ii) increasing number of UEs, which stresses the number of flows, as well as reporting messages UPF must manage. Finally, we discuss its signaling overhead.

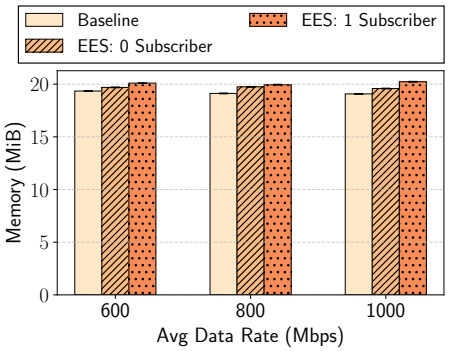


Figure 19 – Impact of data rate on UPF performance across memory usage

6.1.1 Impact of data rate

We compare the average one-way delay, as well as the CPU and memory usage, of UPF across 3 UPF variants at varying data rates. For each data rate, one UE generates UDP traffic to a server in our testbed using iperf3, with the bandwidth parameter set to the target rate. Simultaneously, another UE sends pings to the server. Figures 17, 18, 19 show the outcome of this experiment with 95% confidence intervals.

While at lower rates the latency incurred by the EES with one subscriber is imperceptible, at 1000 Mbps, it increases the one-way delay by only 0.11 ms compared to the baseline. This demonstrates that the proposed implementation imposes a negligible performance degradation on the UPF, whose primary function is high-speed packet forwarding. CPU utilization increases proportionally with throughput across all UPF variants. In our extended UPF with an active subscriber, a small, throughput-independent overhead is observed due to data collection and notification generation. This additional CPU usage remains below 6.5% across all tested data rates, confirming minimal impact on processing.

Memory consumption remains stable and throughput independent across all UPF versions. The EES introduces only a small, constant overhead due to subscription state management and temporary data buffering. No upward drift in memory usage was observed, indicating no memory leaks.

Overall, as throughput increases, the system efficiently scales with the volume of packets processed for data collection, demonstrating that the proposed implementation maintains high scalability and resource efficiency

under load.

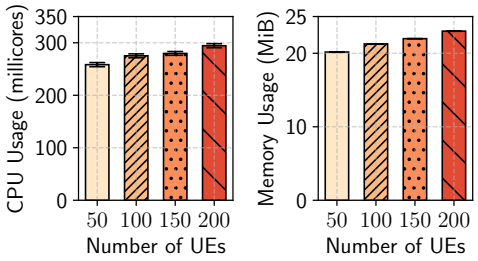


Figure 20 – Effect of active UE count on UPF resource usage

6.1.2 Impact of number of UEs

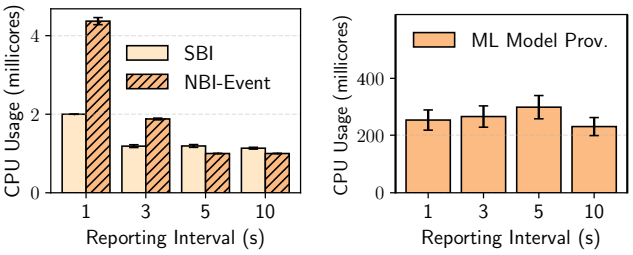
In this experiment, the total data rate is fixed at 200 Mbps while the number of active UEs is incrementally increased. Fig. 20 presents the corresponding results. As the number of UEs increases from 50 to 200 (while aggregate throughput remains constant), CPU utilization increases by only about 36 millicores, and memory usage increases only marginally by 2.84 MiB. This minor rise is attributed to the EES tracking more concurrent flows, which proportionally increases the total number of per-flow reports generated.

6.1.3 Signaling overhead

At the end of each reporting period, the UPF EES sends one notification per UE, containing information about all of that UE's flows. Therefore, the number of notifications per reporting period grows linearly with the number of UEs. In contrast, increasing the number of flows per UE keeps the number of notifications fixed but increases the size of each notification. As a result, bandwidth consumption increases linearly with both the number of UEs and the number of flows. For example, for the volume measurement report and a 3-second notification period, the average signaling overhead is about 14 Kbps for 2 UEs with 2 flows each, and 52 Kbps for 4 UEs with 4 flows each.

6.2 NWDAF resource usage

The NWDAF must support efficient real-time decision-making without incurring excessive overhead. To evaluate its scalability, we measure the CPU usage of its components at different reporting frequencies (1s to 5s). For each interval, we configured the SMF to request periodic reports on abnormal behavior, which trigger the engine's periodic inference to the MLMPS and the event subscription module (within the NBI) to generate periodic notifications based on the analytics engine's results. We also configured the NWDAF to collect data from the UPF with the same reporting period. Consequently, both SBI and NBI event subscription modules (referred to as NBI-Event) use a common interval. Fig. 21 shows the CPU usage of different NWDAF modules. Fig. 21a shows that longer reporting intervals lead to lower resource consumption for both the SBI and NBI, while Fig. 21b shows that the CPU usage of the MLMPS remains unchanged across intervals. To confirm this, we conducted another experiment, deployed the MLMPS, and logged and served the ML model so that it was ready for inference; however, we did not deploy the engine, and as a result, there is no periodic inference to this service; however, the CPU utilization was 281 millicores (95% CI: 240–323 millicores), which is consistent with our earlier observation. These experiments indicate that the MLMPS consumes substantially more resources than the other NWDAF modules. This is expected since it performs compute-intensive tasks such as ML model querying, filtering, and inference.



(a) NBI-Event and SBI **(b)** MLMPS

Figure 21 – Resource usage of different NWDAF modules

This highlights that data collection and mitigation do not bottleneck the NWDAF and that MLMPS should be scaled independently.

Additionally, in the bot detection use case, with a 1-second notification period, the signaling overhead generated by NWDAF is about 4 Kbps, and can be reduced to 1 Kbps if the minimum required fields are used for reporting an abnormal UE. While only one notification is sent to the SMF in each reporting period, the report's size increases by about 17 Bytes for each detected abnormal UE.

7. CONCLUSION AND FUTURE WORK

This paper investigated whether the 3GPP-defined NWDAF can serve as a practical foundation for closed-loop network orchestration. While the NWDAF enables data-driven network operations, prior work has largely focused on isolated components rather than realizing a complete pipeline spanning data collection, analytics, and action. In particular, the integration of standardized user-plane data collection with runtime policy enforcement has remained limited.

To address this gap, we designed and implemented a standards-compliant NWDAF framework that integrates UPF-based data collection, analytics generation, and automated action by network functions. We demonstrated the framework's ability to support different network control objectives through two proof-of-concept use cases: (i) SMF-based anomaly mitigation via PDU session release and (ii) PCF-based QoS adaptation via runtime policy updates. For anomaly detection, we showed that shorter data collection intervals reduce detection and mitigation latency. For QoS adaptation, we demonstrated that dynamically adjusting QoS bit-rate parameters can protect GBR traffic during congestion.

Our evaluation shows that fine-grained telemetry can be collected efficiently, incurring at most 6.5% CPU overhead and an additional 0.11 ms delay compared to the baseline. We further analyzed the resource usage of NWDAF components, showing that most are lightweight, except for the MLMPS, which can be scaled independently to meet the load. Beyond scaling an individual NWDAF instance, large-scale networks may benefit from multiple NWDAF instances working together. Depending on the network's requirements, an operator may deploy multiple NWDAF instances. These instances can be specialized along different dimensions: the analytics types they support, the geographic area they serve, or their location within the network. This distribution can be driven by where the input data resides. For example, when an analytics task requires data from NFs at both the edge and the core, deploying a separate NWDAF instance at each location avoids transporting large volumes of raw NF data back and forth across the network, which would otherwise consume substantial bandwidth and add latency.

Several directions remain open for future work. The QoS monitoring delay reported by the UPF EES is currently limited to UPF-internal delay; incorporating the RAN-side component to capture the end-to-end delay would bring an interesting new perspective. The heuristic algorithm used to adjust bit rates could be replaced with learned policies to better handle non-stationary traffic and to compare the effectiveness and cost of learning-based decision-making at runtime. More broadly, the proposed framework provides a foundation for evaluating new analytics methods within a runtime closed-loop orchestration and can be extended to additional use cases, operational objectives, and enforcement strategies as open-source 5G cores continue to mature.

ACKNOWLEDGEMENT

We thank Niloy Saha for valuable assistance and feedback on this work. This work was supported in part by funding from the Innovation for Defence Excellence and Security (IDEaS) program from the Department of National Defence (DND).

REFERENCES

[1] ETSI. *Zero-touch network and Service Management (ZSM); Closed-Loop Automation; Part 3: Advanced topics*. Group Report (GR) ZSM 009-3. Version 1.1.1. ETSI, Aug. 2023.

[2] Niloy Saha, Nashid Shahriar, Muhammad Sulaiman, Noura Limam, Raouf Boutaba, and Aladdin Saleh. "Monarch: Monitoring Architecture for 5G and Beyond Network Slices". In: *IEEE Transactions on Network and Service Management* 22.1 (2025), pp. 777–790. DOI: 10.1109/TNSM.2024.3479246.

[3] Dimitrios Michael Manias, Ali Chouman, and Abdallah Shami. "An NWDAF approach to 5G core network signaling traffic: Analysis and characterization". In: *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE. 2022, pp. 6001–6006.

[4] Abebu Ademe Bayleyegn, Zaloa Fernández, and Fabrizio Granelli. "Real-Time Monitoring of 5G Networks: An NWDAF and ML Based KPI Prediction". In: *2024 IEEE 10th International Conference on Network Softwarization (NetSoft)*. 2024, pp. 31–36. DOI: 10.1109/NetSoft60951.2024.10588931.

[5] Sebastian Peters, Fikret Sivrikaya, Satyatma Winarga Rochadi, and Amina Ayadi-Miessen. "Cobra-5G – an AI-Driven Solution for Resilient Industrial Applications in Private 5G Environments". In: *2024 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS)*. 2024, pp. 92–99. DOI: 10.1109/ICCNS62192.2024.10776531.

[6] Ali Chouman, Dimitrios Michael Manias, and Abdallah Shami. "Towards supporting intelligence in 5G/6G core networks: NWDAF implementation and initial analysis". In: *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE. 2022, pp. 324–329.

[7] Abdelkader Mekrache, Karim Boutiba, and Adlen Ksentini. "Combining Network Data Analytics Function and Machine Learning for Abnormal Traffic Detection in Beyond 5G". In: *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE. 2023, pp. 1204–1209.

[8] Open5GS Contributors. *Open5GS: Open Source 5G and EPC*. Accessed: 2025-04-06. 2025. URL: https://github.com/open5gs/open5gs.

[9] Fatemeh Shafiei Ardestani, Niloy Saha, Noura Limam, and Raouf Boutaba. "Towards nwdaf-enabled analytics and closed-loop automation in 5g networks". In: *arXiv preprint arXiv:2505.06789* (2025).

[10] Fatemeh Shafiei Ardestani, Niloy Saha, Noura Limam, and Raouf Boutaba. "Towards NWDAF-enabled Analytics and Closed-Loop Automation in 5G Networks". In: *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. Rome, Italy, May 2026.

[11] Massimo Condoluci and Toktam Mahmoodi. "Softwarization and virtualization in 5G mobile networks: Benefits, trends and challenges". In: *Computer Networks* 146 (2018), pp. 65–84. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.09.005. URL: https://www.sciencedirect.com/science/article/pii/S1389128618302500.

[12] Faqir Zarrar Yousaf, Michael Bredel, Sibylle Schaller, and Fabian Schneider. "NFV and SDN—Key Technology Enablers for 5G Networks". In: *IEEE Journal on Selected Areas in Communications* 35.11 (2017), pp. 2468–2478. DOI: 10.1109/JSAC.2017.2760418.

[13] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Services and System Aspects; Architecture enhancements for 5G System (5GS) to support network data analytics services; (Release 19)*. Tech. rep. TS 23.288. 3GPP, 2024.

[14] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 19)*. Tech. rep. TS 23.501. 3GPP, 2024.

[15] Lionel Morand. *OpenAPIs for the Service-Based Architecture*. https://www.3gpp.org/technologies/openapis-for-the-service-based-architecture. Accessed: 2026-05-01.

[16] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Services and System Aspects; Policy and charging control framework for the 5G System (5GS); Stage 2, (Release 19)*. Tech. rep. TS 23.503. 3GPP, 2025.

[17] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Core Network and Terminals; 5G System; Session Management Policy Control Service; Stage 3, (Release 19)*. Tech. rep. TS 29.512. 3GPP, 2025.

[18] 3rd Generation Partnership Project (3GPP). *3rd Generation Partnership Project;Technical Specification Group Core Network and Terminals; Interface between the Control Plane and the User Plane Nodes; Stage 3, (Release 19)*. Tech. rep. TS 29.244. 3GPP, 2025.

[19] 3rd Generation Partnership Project (3GPP). *Technical Specification Group Core Network and Terminals; 5G System; User Plane Function Services; Stage 3 (Release 18)*. Tech. rep. TS 29.564. 3GPP, 2024.

[20] Khizar Abbas, Talha Ahmed Khan, Muhammad Afaq, Javier Jose Diaz Rivera, and Wang-Cheol Song. "Network Data Analytics Function for IBN-based Network Slice Lifecycle Management". In: *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2021, pp. 148–153. DOI: 10.23919/APNOMS52696.2021.9562662.

[21] Henok Daniel, Omar Alhussein, Jie Liang, Cheng Li, and Ernesto Damiani. "Enhanced Open-Source NWDAF for Event-Driven Analytics in 5G Networks". In: *IFIP Networking*. IEEE. 2025.

[22] Ali Nadar and Jérôme Härri. "Enhancing Network Data Analytics Functions: Integrating AIaaS with ML Model Provisioning". In: *2024 22nd Mediterranean Communication and Computer Networking Conference (MedComNet)*. 2024, pp. 1–4. DOI: 10.1109/MedComNet62012.2024.10578162.

[23] MLflow Contributors. *MLflow:A Tool for Managing the Machine Learning Lifecycle*. Accessed: 2025-04-06. 2025. URL: https://mlflow.org/#core-concepts.

[24] OpenAirInterface Software Alliance. *OAI 5G Core Network (CN5G)*. Accessed: 2025-04-06. 2025. URL: https://gitlab.eurecom.fr/oai/cn5g.

[25] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. "An empirical comparison of botnet detection methods". In: *computers & security* 45 (2014), pp. 100–123.

[26] Abbas Abou Daya, Mohammad A. Salahuddin, Noura Limam, and Raouf Boutaba. "A Graph-Based Machine Learning Approach for Bot Detection". In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2019, pp. 144–152.

[27] Abbas Abou Daya, Mohammad A Salahuddin, Noura Limam, and Raouf Boutaba. "BotChase: Graph-based bot detection using machine learning". In: *IEEE Transactions on Network and Service Management* 17.1 (2020), pp. 15–29.

[28] Ali Güngör. *UERANSIM: Open source 5G UE and RAN (gNodeB) implementation*. Accessed: 2025-04-06. 2025. URL: https://github.com/aligungr/UERANSIM.

[29] *Nmap Network Scanner*. https://nmap.org. 2009.

AUTHORS

Fatemeh Shafiei Ardestani received her bachelor's degree in computer engineering from the University of Isfahan, Iran. She received her Master of Mathematics in computer science from the David R. Cheriton School of Computer Science at the University of Waterloo, Canada, in 2026. Her research focuses on enabling closed-loop orchestration in 5G networks, particularly through analytics from 5G core network functions. More broadly, she is interested in the use of AI for the management and orchestration of complex systems.

Noura Limam received the MSc and PhD degrees in computer science from Sorbonne University in 2002 and 2007, respectively. She is a research professor of Computer Science at the University of Waterloo, Canada. She is an active researcher and a contributor in the area of network and service management. Her current interests revolve around network automation and cognitive network management. She is the recipient of the 2026 IEEE Communications Society Salah Aidarous award, an associate editor of IEEE Transactions on Network and Service Management, and a guest editor of IEEE Communications Magazine Series on Network Softwarization and Management. She previously served as the Chair of the IEEE Communications Society Technical Committee on Network Operations and Management (2021-2025).

Raouf Boutaba received M.Sc. and Ph.D. degrees in computer science from Sorbonne University in 1990 and 1994, respectively. He is currently a university chair professor and the Director of the David R. Cheriton School of Computer Science, University of Waterloo, Canada. His research interests fall in the areas of computer networking and distributed systems. He is a fellow of the Engineering Institute of Canada, the Canadian Academy of Engineering, and the Royal Society of Canada. He served as the founding Editor-in-Chief for IEEE Transactions on Network and Service Management (2007–2010) and the Editor-in-Chief for the IEEE Journal on Selected Areas In Communications (2018–2021).