Towards NWDAF-enabled Analytics and Closed-Loop Automation in 5G Networks

Fatemeh Shafiei Ardestani, Niloy Saha, Noura Limam, Raouf Boutaba
Cheriton School of Computer Science, University of Waterloo, Canada
{f2shafie, n6saha, noura.limam, rboutaba}@uwaterloo.ca

***Abstract*—The fifth generation of cellular technology (5G) delivers faster speeds, lower latency, and improved network performance while supporting a large number of users and a diverse range of verticals. This increases the complexity of network management, making closed-loop automation essential. In response, the 3rd Generation Partnership Project (3GPP) introduced the Network Data Analytics Function (NWDAF) to streamline network monitoring by collecting, analyzing, and providing insights from network data. Critical aspects of NWDAF, such as standardized data collection, integrating analytics into closed-loop automation, and end-to-end system evaluation, have received limited attention in prior studies. This work addresses existing gaps by presenting a practical implementation of NWDAF and its integration with a leading open-source 5G core network solution. We develop a 3GPP-compliant User Plane Function (UPF) Event Exposure Service for real-time data collection, and an ML Model Provision Service integrated with MLflow, to support the machine learning lifecycle management for NWDAF. Additionally, we enhance the Session Management Function (SMF) to consume NWDAF analytics and act accordingly. Our evaluation demonstrates our solution's scalability, resource efficiency, and effectiveness in enabling closed-loop security management in 5G.**

***Index Terms*—5G, NWDAF, Analytics, Closed-loop Automation**

I. Introduction

5G and beyond networks are expected to serve diverse use cases, ranging from mission-critical control to industrial automation, each with distinct service-level requirements. To do so, these networks leverage technologies such as software-defined networking and network virtualization, which provide the necessary flexibility but at the cost of increased operational complexity. This calls for network automation. In line with the ETSI Zero-Touch Network and Service Management (ZSM) initiative, which envisions autonomous networks that require minimal human intervention [1], 3GPP introduced the Network Data Analytics Function (NWDAF), a 5G core function that collects data from various network functions (NFs), applies statistical or machine learning techniques, and generates actionable analytics. Other NFs can consume these analytics to form a closed-loop feedback mechanism towards zero-touch network management (Figure 1). While the NWDAF is a promising enabler for autonomous network management, realizing closed-loop automation remains challenging. Existing efforts have explored its use for control-plane monitoring [2], [3], predictive analytics [4], and anomaly detection [5], [6]. However, these systems tackle isolated components in the analytics pipeline rather than the end-to-end loop that connects

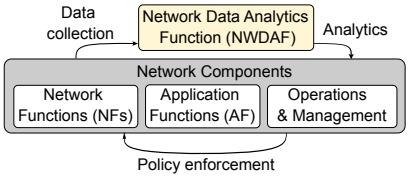


Fig. 1: NWDAF-enabled closed loop network automation

data collection, analytics, and policy enforcement. Such closed-loop control requires addressing three key challenges:

- **Standards-compliant efficient data collection.** The User Plane Function (UPF) must expose fine-grained telemetry while having minimal impact on its primary role of packet forwarding. Capturing per-flow statistics at line rate for multiple User Equipments (UEs) can introduce packet delay and CPU overhead. Prior work [3], [6], [7] has largely relied on passive capture or indirect export of telemetry (e.g., through the SMF) and does not provide implementation of overhead analysis of 3GPP-defined Event Exposure Service (§III-A).
- **Integrating analytics with policy enforcement.** To enable closed-loop automation, NWDAF analytics must lead to timely policy actions. This requires translating analytics into concrete control-plane directives and making sure those directives propagate quickly to enforcement points. However, these goals are often in conflict: larger observation windows increase inference fidelity, while shorter ones enable faster responses. This tension between accuracy and response time requires careful integration and evaluation of both aspects. Existing works [5], [6], [8] fall short of this integration. They focus primarily on addressing the analytics part; they lack demonstration of automated policy actuation or feedback between NWDAF and the control plane (§III-C).
- **Lifecycle management of analytics models.** NWDAF analytics rely on AI/ML models that may need to evolve alongside traffic dynamics and network configuration. To do so, the NWDAF requires mechanisms for model discovery, version control, and live provisioning to enable these models to be updated without disrupting service continuity (§III-B). This aspect has been largely overlooked in prior work.

Beyond these architectural gaps, prior work [2], [3], [5]–[8] offers limited quantitative evaluations of the resource costs of data collection, analytics generation, and reporting, as well as NWDAF's effectiveness in closed-loop automation.

Overcoming these challenges requires an end-to-end system that integrates standardized data exposure, real-time analytics,

and automated enforcement within a unified framework. To this end, we develop a standards-compliant NWDAF integrated with Open5GS [9] core. Our implementation demonstrates that fine-grained observability and closed-loop control are feasible in practice: the full pipeline—from UPF telemetry to NWDAF analytics and SMF policy enforcement—runs to completion in a few seconds. Our experiments on a 5G testbed demonstrate lightweight NWDAF modules and low overhead for the UPF Event Exposure Service (EES). We release our implementation as open-source[1]. Our key contributions are as follows:

- We build the first 3GPP-aligned closed-loop NWDAF system, unifying data collection, analytics, and policy enforcement.
- We implement the key control- and data-plane enablers, including a 3GPP-compliant UPF EES, lifecycle management of analytics models, and automated enforcement of SMF-based policies, forming a closed-loop automation framework.
- We provide a comprehensive end-to-end evaluation of our closed-loop NWDAF system on a 5G testbed integrating a popular open-source core (Open5GS [9]). Our experiments cover the UPF EES, NWDAF analytics pipeline, and SMF-based policy enforcement, assessing scalability, resource efficiency, and responsiveness to demonstrate the feasibility of standards-compliant closed-loop automation in 5G networks.

II. Background and Related Work

A. NWDAF and Event Exposure Overview

The 5G core uses a service-based architecture (SBA) in which network functions (NFs) communicate through standardized interfaces [10], [11]. This work focuses on three key NFs: UPF, which forwards user traffic between UEs and data networks; SMF, which controls data sessions; and NWDAF [12]. Using a security use case, we show how these NFs interwork to realize data-driven automated network management (§IV).

The 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 [12].

Data collection. To obtain operational data, NWDAF subscribes to Event Exposure Services 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* [13]. As the standards-based provider of UE traffic telemetry, the UPF EES can enable a wide range of closed-loop analytics and control actions. However, despite being specified in 3GPP, UPF EES remains rarely reported as implemented in existing open-source cores [9], [14], [15] and prior work [2]–[8], [16], [17]. In this work, we present the first standards-compliant realization of the UPF EES, integrated into Open5GS (§III-A).

B. Prior work on Closed-loop Automation using NWDAF

Prior NWDAF-related efforts can be broadly classified into four categories: data collection, analytics generation, model lifecycle management, and closed-loop automation.

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

TABLE I: Comparison of NWDAF implementations

Related Work	Core Stack	UPF EES	ML Lifecycle/ Provisioning	Closed-Loop Automation
[2]	Open5GS	✗	✗	✗
[3]	Open5GS	✗	✗	✗
[8]	Free5GC	✗	✗	✗
[16]	OAI EPC	✗	✗	✓
[5]	Open5GS	✗	✗	✗
[6]	OAI 5GC	✓*	✗	✗
[4]	Free5GC	✗	✗	✗
[17]	OAI 5GC	✗	✓	✗
Ours	**Open5GS**	✓	✓	✓

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

Data collection. The early NWDAF prototype [3] focused on collecting and analyzing 5G core signaling messages using passive packet captures. Work [8] relies on AMF EES for mobility-related metrics, custom packet captures for UE communication, and cAdvisor-based monitoring for NF resource usage. More recent work [4], [6] presented implementations of event exposure services, but these, too, focused on the control plane—i.e., AMF/SMF event exposure—rather than user-plane event exposure. Our work contributes the first open-source realization of the UPF EES [18].

Analytics generation. Several other works focused on ML-based analytics, such as anomaly detection [5], [6]. While these works demonstrate the promise of data-driven analytics at the NWDAF, they rely on custom data-collection mechanisms or SMF-mediated approaches to collect UE communication data, which limits interoperability.

Model lifecycle management. [17] explored integrating an AI-as-a-Service (AIaaS) platform with NWDAF to enable ontology-based model discovery and exchange across NWDAFs. While this addresses interoperability, it does not address runtime model updates or versioning. To address this, our design implements the 3GPP-defined ML Model Provision Service using MLflow [19], enabling continuous model management.

Closed-loop automation. Only a few works considered closing the loop using NWDAF-provided analytics. In [16], the NWDAF was integrated with an Intent-Based Networking platform to trigger slice scaling or DDoS mitigation based on ML predictions, but it relied on the OAI EPC, which lacks standardized 5G functions and event exposure interfaces. [4] enabled AMF/SMF event generation in Free5GC for handover prediction, but did not take any policy enforcement decisions based on the predictions. In contrast, our system extends the SMF to subscribe to NWDAF analytics via the 3GPP-defined `EventsSubscription` service, enabling automatic release of the user's Protocol Data Unit (PDU) session and achieving standards-compliant closed-loop control.

Earlier works treat data collection, analytics, and control in isolation (Table I). None integrates UPF EES, dynamic ML lifecycle management, and SMF-driven enforcement in a single 3GPP-compliant loop. We present our design in §III.

III. System Architecture and Design

To effectively achieve the NWDAF objectives, we adopt a modular architecture that builds on the OAI NWDAF design and

codebase [6] [14]. We design and implement (i) the NWDAF *ML Model Provision Service* (MLMPS) and (ii) an NWDAF *Analytics Engine*. We extend (iii) the NWDAF *Southbound Interface* (SBI) with an EES client, enabling ingestion of UPF EES notifications. We complete our closed-loop system by developing the (iv) *UPF EES* for data generation and the (v) *NWDAF Subscriber* integrated in SMF, turning the latter into an active consumer of NWDAF reports. We reuse the OAI NWDAF *Northbound Interface* (NBI) to generate 3GPP-compliant reports and notify NWDAF consumers. We provide implementation details in Appendix A. Figure 2 depicts these modules as well as our NWDAF-enabled closed-loop system, which operates in three stages: Data Collection (§III-A), Analytics (§III-B), and Action (§III-C). We address each step in turn in the following sections.

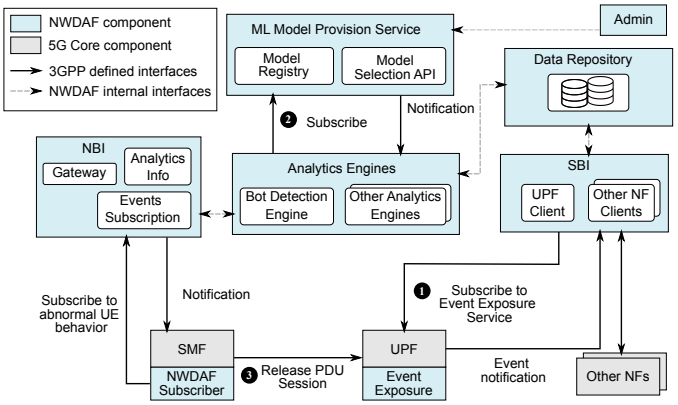


Fig. 2: High-level overview of our NWDAF closed-loop system

A. Data Collection: UPF Event Exposure Service

Data collected at the UPF can provide fine-grained visibility into per-UE flows and usage patterns. This visibility is essential for analytics-driven management across use cases such as QoS assurance and anomaly detection. According to the 3GPP specifications, the UPF EES provides this data. However, no open-source implementation currently exists, leaving unresolved questions about the effectiveness, resource usage, and latency impact of this approach. To address this gap, we implement and evaluate the first UPF EES and NWDAF that leverages it.

Data-collection workflow. The NWDAF SBI is responsible for collecting data from various sources (e.g., NFs). The UPF client inside this module sends a subscription request for data collection to UPF EES. In this request, NWDAF can indicate which data it is interested in collecting by specifying standard fields such as event type (e.g., `usage-measures`), measurement type (e.g., `volume`), or measurement granularity (e.g., `per-flow`). It can also indicate how it wants to receive data in the reporting mode fields, such as the frequency of receiving data. This process is Step ❶ in Figure 2.

After responding to the NWDAF's initial request, the UPF sends periodic reports to the NWDAF, which stores them in its database. Each report includes a timestamp, allowing the NWDAF to learn about the network's historical behavior while monitoring its real-time status. We provide a sample UPF notification in Appendix E.

UPF EES design. We developed the UPF EES by implementing five distinct modules, each providing a specific function. Figure 3 shows these modules in the UPF.

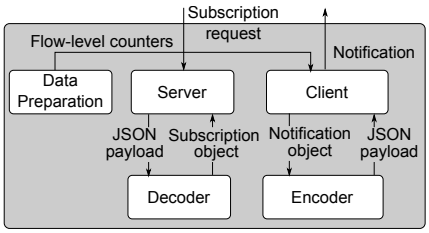


Fig. 3: UPF Event Exposure Service

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 validate subscription requests and send appropriate responses. The server module performs these operations. It stores the subscription details, assigns a unique subscription ID, and replies with the ID and the accepted subscription details, omitting unavailable requested events from the response. It is essential to track each subscriber's requests when generating notifications since subscribers may request different events (types of data). We use a HashMap, mapping each event to its subscriber list, enabling an efficient lookup of subscribers interested in an event when generating notifications. Collecting data without any subscribers is inefficient; therefore, the server will activate the data preparation module once at least one subscriber is present. The subscriber list is shared across modules and 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, as it is executed for every packet traversing the network; hence, its execution frequency increases with the number of packets. Additionally, the high packet rate at the UPF significantly exceeds the notification generation rate. 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 processes the packets passing through the UPF. This module uses a hashing mechanism to categorize the collected data. It uses a combination of the flow identifiers (source IP, destination IP, source port, and destination port), the PDU session ID, and the subscription ID as a key and aggregates critical packet statistics, such as packet volume and direction. This data is shared with the client module for generating notifications.

To ensure that packet processing on the fast path never blocks synchronization, rather than using a lock to ensure the correctness of the shared data, we used a lock-free single-producer/single-consumer (SPSC) ring buffer between the data preparation and the client module. This approach allows us to

gather essential data during packet processing and defers the necessary computation to when we generate the notification. Our evaluation (§V) shows that this approach incurs limited CPU overhead and does not degrade the UPF throughput.

Client Module. The Client module periodically generates and sends notifications to subscribers. Each subscriber defines its expected notification by selecting a combination of the parameters shown in Table II. 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.

TABLE II: Measurements supported by our UPF EES implementation

Parameter	Supported Values
Event Type	`"usage-measures"`, `"usage-trends"`
Measurement Type	`"volume"`, `"throughput"`
Granularity	`"per-flow"`, `"per-session"`

The data preparation module is triggered on each packet arrival, while the server and client modules operate independently on separate threads. This allows concurrency, preventing the loss of a subscription request or incoming packets.

B. Analytics Architecture

Each analytics engine periodically analyzes the data in 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 Figure 2. Below, we detail the newly introduced NWDAF modules and their roles.

Integration with UPF EES. We implement a UPF client in the NWDAF SBI module to leverage UPF EES reports. 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.

Analytics Engine. 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 engine.

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 [19] into this service for model management and versioning. Each Analytics engine can subscribe to MLMPS and request an

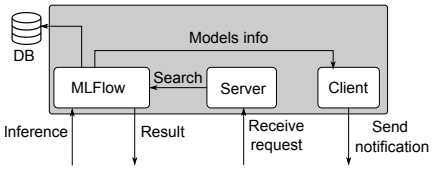


Fig. 4: ML Model Provision Service

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. MLMPS continuously monitors the MLflow model registry and notifies subscribers of the latest versions of the models that match their requests. Beyond the standard MLMPS APIs, we developed complementary MLflow-facing endpoints to log or update models; we refer to these as the admin APIs.

C. Automated Policy Enforcement via SMF

The 3GPP specifications [12] define several possible actions that NWDAF subscribers may take upon detecting abnormal behavior. Among these, SMF may terminate the PDU session of a misbehaving UE to protect network integrity. We extend the SMF to subscribe to and receive NWDAF notifications and automatically enforce session termination (Step ❸ of Figure 2) thereby closing the security management loop.

To realize this functionality, we augment the SMF with two new modules: *NWDAF client* and *NWDAF server*. The client module issues a one-time subscription request to NWDAF, while the server module receives periodic reports from NWDAF's NBI indicating abnormal UE activity. We also change SMF so that, upon notification, it instructs UPF via the N4 interface to release the corresponding PDU session, effectively isolating the misbehaving UE.

This enhancement enables automated policy enforcement within the 5G core, demonstrating a practical realization of 3GPP's vision for analytics-driven closed-loop control.

IV. Proof-of-Concept Case Study

In this section, we demonstrate our closed-loop automation framework using a security management scenario that aims to (i) detect bots by monitoring and analyzing UE traffic in real time and (ii) autonomously mitigate detected threats. Our testbed uses the open-source, 3GPP Release 16-compliant Open5GS core [9]. We deploy all NFs and NWDAF modules on Kubernetes and simulate gNB/UE behavior with UERANSIM [20].

Bot Detection Model. We use the bot behavior data from Scenarios 1, 2, 3, and 5 of the CTU-13 dataset [21] as the source of malicious behavior. Benign samples were generated in our testbed by emulating legitimate UE traffic, with data collected from the NWDAF. We use a methodology inspired by BotChase [22], [23] to build a communication graph among several UEs (mapped as nodes) and extract features to train a Random Forest model. For each UE, the features include the number of incoming and outgoing edges, the weighted indegree and outdegree, and the weighted betweenness centrality, which measures how often the UE appears on the shortest path

between pairs of nodes in the graph. We register our trained bot-detection model using the MLMPS APIs, as described in §III-B.

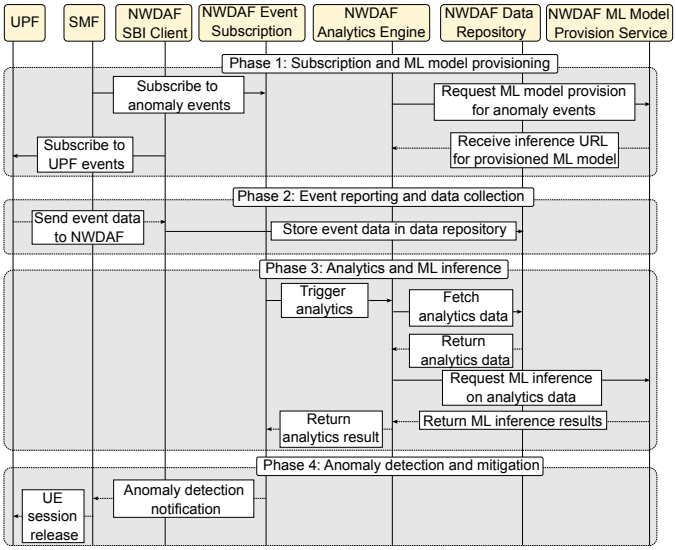


Fig. 5: Sequence diagram for closed-loop workflow

Closed-loop Workflow. Figure 5 depicts the closed-loop workflow, which consists of:

- *Subscription Phase*: Upon deployment, SBI subscribes to the UPF EES, and the Bot Detection engine subscribes to the MLMPS, requesting a model for abnormal behavior events. Simultaneously, the SMF subscribes to NWDAF for abnormal behavior detection. Appendix C provides an example SMF subscription request.
- *Data Collection Phase*: Once UEs send traffic, the UPF EES streams periodic reports to the NWDAF SBI.
- *Analysis Phase*: Based on the report period requested by SMF, NWDAF NBI will trigger the bot detection engine to investigate the network traffic. The bot detection engine retrieves data from the NWDAF database, preprocesses it to derive graph-based features for each UE communication, and uses these features, along with the inference model provided by MLMPS, to investigate each UE's behavior.
- *Mitigation Phase*: The NBI receives detection results from the bot detection engine and encapsulates them in notifications; a sample notification is provided in Appendix C. Upon receiving notifications indicating abnormal behavior, the SMF releases the PDU session of the detected UEs to protect the network from further malicious activity.

V. Performance Evaluation

We evaluate the performance of our NWDAF-based closed-loop system, focusing on three critical aspects: (i) the overhead and scalability of our UPF EES (§V-A), (ii) NWDAF resource consumption under varying workloads (§V-B), and (iii) the end-to-end latency of the closed-loop detection and mitigation workflow (§V-C). Our evaluation demonstrates that the system introduces negligible performance overhead while enabling real-time analytics and automated threat response.

A. UPF Overhead and Scalability

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

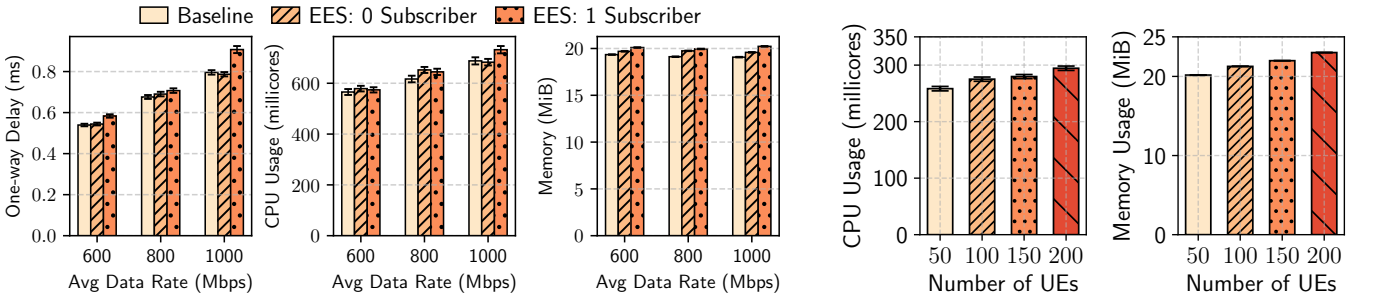
Since the UPF EES performs real-time data collection and reporting, it may introduce additional CPU overhead or delay packet forwarding. We evaluate this overhead along two critical stress dimensions: (i) increasing data rate, which stresses per-packet processing load, and (ii) increasing number of UEs, which stresses the number of flows as well as reporting messages the UPF must manage. Figure 6 presents the results with 95% confidence intervals. Further details of these experiments are provided in Appendix B.

Impact of Data Rate. We compare the average one-way delay (RTT/2), as well as the CPU and memory usage of the UPF for the 3 UPF variants with varying data rates. For each data rate, one UE generates UDP traffic using iperf3 to a server in our testbed, with the bandwidth parameter set to the target rate. Simultaneously, another UE sends pings to the server. Figure 6a shows the outcome of this experiment.

- *the latency* incurred by EES with one subscriber is imperceptible at lower rates; however, at 1000 Mbps, it increases the one-way delay by only 0.11 ms compared to 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 processing impact.
- *Memory consumption* remains stable and throughput-independent across all UPF versions. 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 the absence of 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.

Impact of Active UE Count. In this experiment, the total data rate is fixed at 200 Mbps while the number of active UEs is incrementally increased (Figure 6b). 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



(a) Impact of Data Rate on UPF performance

(b) Impact of Active UE Count on UPF performance

Fig. 6: Impact of Data Rate and Active UE Count on UPF performance

more concurrent flows, which proportionally increases the total number of per-flow reports generated.

B. 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 different components with varying reporting frequencies (1s to 5s). For each interval, the SMF requests abnormal behavior reports. As a result, the engine performs inference to the MLMPS, and the Event subscription module (inside the NBI) generates notifications periodically, on the same interval. 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. Result shows that longer reporting intervals lead to lower resource consumption for both SBI and NBI, while the CPU usage of the MLMPS remains unchanged across intervals. To confirm this, we repeated the experiment with the model ready for inference but with no engines subscribing to or invoking the service, thereby measuring its idle-state resource consumption; 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.

This highlights that data collection and mitigation do not bottleneck the NWDAF and that MLMPS should be independently scaled. Appendix D provides further details about this experiment.

C. Closed-Loop Workflow: Attack to Mitigation

To evaluate the responsiveness of our proposed closed-loop automation system, we examine the time taken for a bot attack to be detected and malicious UEs to be banned.

We set up 20 servers with distinct IP addresses in our testbed to serve as the intended target of the bot attack. To simulate bot-like scanning activity, we used Nmap's [24] http-open-proxy to scan for open web proxies. We configured SMF to receive abnormal behaviour reports from NWDAF every second and varied the subscription period for NWDAF SBI to collect data from UPF. For each experiment round, after deploying the system, we initiated the bot's behavior within the UE. We also performed a ping inside the UE, setting a timeout of 0.5 seconds for responses. We used ping to illustrate the time required for the network to block the UE after detection. For each data collection interval $(1, 3, 5)$, we repeated this experiment 20 times and reported the mean and standard deviation. While the mitigation time is almost identical 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. This highlights an inherent trade-off: collecting data over a more extended period provides more information and can improve detection accuracy, but it also increases response time. Optimizing this trade-off depends on the model's characteristics and the required detection precision. Appendix D presents additional experimental details.

VI. Conclusion

We present a 3GPP-compliant, NWDAF-based closed-loop automation framework for 5G and Beyond networks. Our system realizes the UPF Event Exposure Service, an NWDAF ML Model Provision Service integrated with MLflow, and extends the SMF to enforce actions based on NWDAF analytics. We showed that the UPF EES incurs minimal latency and scales with data rates and UE counts, while most NWDAF components are lightweight, with MLMPS dominating computational cost. The Closed-loop response time analysis confirms that NWDAF can be effectively embedded into the 5G control loop, enabling practical, standards-aligned automation. The architecture is modular and extensible: any ML model can be integrated with minimal changes, and SMF actions can be generalized to a broad set of anomaly-detection and policy-enforcement scenarios. Future work includes supporting online training and adaptive ML models that evolve with network dynamics, as well as broadening the range of policy enforcement mechanisms for richer, context-aware closed-loop control.

Acknowledgment

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, ETSI, 08 2023. Version 1.1.1.

[2] A. Chouman, D. M. Manias, and A. Shami, "Towards supporting intelligence in 5g/6g core networks: Nwdaf implementation and initial analysis," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 324–329, IEEE, 2022.

[3] D. M. Manias, A. Chouman, and A. Shami, "An nwdaf approach to 5g core network signaling traffic: Analysis and characterization," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pp. 6001–6006, IEEE, 2022.

[4] H. Daniel, O. Alhussein, J. Liang, C. Li, and E. Damiani, "Enhanced open-source nwdaf for event-driven analytics in 5g networks," in *IFIP Networking*, IEEE, 2025.

[5] S. Peters, F. Sivrikaya, S. W. Rochadi, and A. Ayadi-Miessen, "Cobra-5g – an ai-driven solution for resilient industrial applications in private 5g environments," in *International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS)*, pp. 92–99, 2024.

[6] A. Mekrache, K. Boutiba, and A. Ksentini, "Combining network data analytics function and machine learning for abnormal traffic detection in beyond 5g," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*, pp. 1204–1209, IEEE, 2023.

[7] N. Saha, N. Shahriar, M. Sulaiman, N. Limam, R. Boutaba, and A. Saleh, "Monarch: Monitoring architecture for 5g and beyond network slices," *IEEE Transactions on Network and Service Management*, vol. 22, no. 1, pp. 777–790, 2025.

[8] A. A. Bayleyegn, Z. Fernández, and F. Granelli, "Real-time monitoring of 5g networks: An nwdaf and ml based kpi prediction," in *IEEE 10th International Conference on Network Softwarization (NetSoft)*, pp. 31–36, IEEE, 2024.

[9] O. Contributors, "Open5gs: Open source 5g and epc," 2025. Accessed: 2025-04-06.

[10] 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.

[11] 3rd Generation Partnership Project (3GPP), "Technical specification group services and system aspects; procedures for the 5g system (5gs); stage 2 (release 19)," Tech. Rep. TS 23.502, 3GPP, 2024.

[12] 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.

[13] 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.

[14] O. S. Alliance, "Oai 5g core network (cn5g)," 2025. Accessed: 2025-04-06.

[15] free5GC Contributors, "free5gc: Open source 5g core network," 2026. Accessed: 2026-02-08.

[16] K. Abbas, T. A. Khan, M. Afaq, J. J. Diaz Rivera, and W.-C. Song, "Network data analytics function for ibn-based network slice lifecycle management," in *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 148–153, 2021.

[17] A. Nadar and J. Härri, "Enhancing network data analytics functions: Integrating aiaas with ml model provisioning," in *22nd Mediterranean Communication and Computer Networking Conference (MedComNet)*, pp. 1–4, 2024.

[18] 3GPP, "5G; 5G System; User Plane Function Services; Stage 3 ," Tech. Rep. 3GPP TS 29.564 version 18.4.0, 3rd Generation Partnership Project (3GPP), May 2024.

[19] M. Contributors, "Mlflow:a tool for managing the machine learning lifecycle," 2025. Accessed: 2025-04-06.

[20] A. Güngör, "UERANSIM: Open source 5g UE and RAN (gnodeb) implementation," 2025. Accessed: 2025-04-06.

[21] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[22] A. Abou Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "A graph-based machine learning approach for bot detection," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 144–152, 2019.

[23] A. Abou Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "Botchase: Graph-based bot detection using machine learning," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 15–29, 2020.

[24] "Nmap network scanner." https://nmap.org, 2009.

Appendix

A. System Implementation

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 in C and integrated it directly into the existing UPF codebase to enable real-time data collection from the user plane.

NWDAF Implementation. In order to receive data from UPF, we implement the UPF client in Go inside the NWDAF SBI component. The collected data is stored in a MongoDB database. We develop the engines and the MLMPS using Python. This latter integrates MLflow [19], 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 Extensions. We extend the SMF in C for Open5GS to support subscription to NWDAF analytics and automated policy enforcement.

B. Experiment Setup

To facilitate testing of the UPF EES, we implemented and deployed a lightweight Python subscriber to receive notifications from UPF. Table III shows the resources assigned to the UPF and gNB for the experiment on the impact of data rate (V-A). Although the results indicate that the UPF used significantly less than the allocated resources, we set higher limits to ensure that neither the UPF nor the gNB became a bottleneck, allowing us to obtain clear and unbiased results. However, for the experiment on the impact of the number of UEs, since the augmented data rate is fixed at 200 Mbit/s, we allocate less CPU and memory to both the UPF and gNB.

TABLE III: Resource limits allocated to each pod in the 5G testbed

Component	CPU Limit (milicores)	Memory Limit (MiB)
UPF	2500	2048
gNB	3500	1024

Since per-packet processing cost is central to our evaluation, we keep the packet count constant in every run across all UPF variants, thereby keeping per-packet processing overhead

constant. To eliminate the run-to-run variation in packet counts, we set the UDP datagram size to 1200 bytes. We then compute the total payload, given the bit rate (M Mbit/s) and the experiment duration (T). Sending at a target rate of M Mbit/s for T seconds would transmit exactly B Bytes.

Traffic was generated with iperf3 (UDP mode) using the target bitrate (-b M), datagram length (-l L), and total byte budget (-n B). We enabled –repeating-payload to keep packet contents deterministic and avoid per-packet randomization costs. For the Impact of the number of UEs experiment, we divide the rate and the total byte budget by the number of UEs.

C. Sample Notifications

Listing 1 SMF subscription request to NWDAF

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

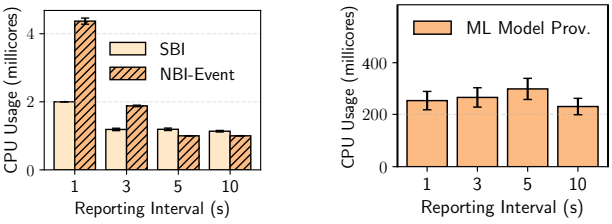
Listing 2 NWDAF notification to SMF indicating abnormal behavior

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

D. Experiment Result

NWDAF Resource Usage.

Figure 7 shows the CPU usage of the NWDAF modules. This confirms that NBI and SBI modules are lightweight and their CPU usage decreases as the reporting interval increases (Figure 7a). However, the resource usage of MLMPS is independent of the reporting interval (Figure 7b).



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

Fig. 7: Resource usage of different NWDAF modules

Closed-Loop Workflow: Attack to Mitigation. Figure 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 the NWDAF SBI. The average time to receive the first report at the NWDAF across all cases is shorter than the requested subscription periods.

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

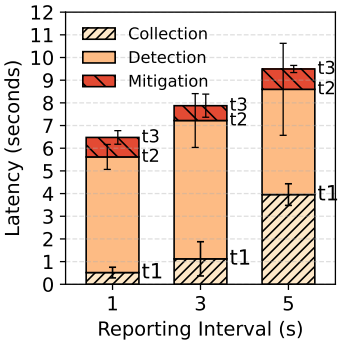


Fig. 8: Timing breakdown for automated attack mitigation

E. Sample UPF Notification

Listing 1 shows a sample per-flow notification. When the subscriber selects *User Data Usage Measures* as the event type, it chooses either *Volume* or *Throughput* as the measurement type and receives the corresponding data. If it selects *User Data Usage Trends* as the event type and *Throughput* as the measurement type, then the throughput statistics measurement is populated. Listings 2, 3, and 4 show samples for measurement fields.

Listing 1: Sample user data usage trends notification per flow

```
{"eventType": "USER_DATA_USAGE_MEASURES  |
     USER_DATA_USAGE_TRENDS",
  "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": null | {..},
    "throughputMeasurement": null | {..},
    "throughputStatisticsMeasurement": null | { .. }}]}
```

Listing 2: Sample Volume Measurement notification

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

Listing 3: Sample Throughput Measurement notification

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

Listing 4: Sample Throughput Statistics Measurement notification

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