Dedicated Monitoring of 5G Service via NEF



Raquel Marques Carneiro
Master in Network and Information Systems Engineering
Department of Computer Science
Faculty of Sciences of the University of Porto
2026

Dedicated Monitoring of 5G Service via NEF

Raquel Marques Carneiro

Dissertation carried out as part of the Master in Network and Information Systems Engineering

Department of Computer Science

2026

Supervisor
Prof. Sérgio Crisóstomo, Assistant Professor, Faculty of Sciences of the University of Porto

Co-supervisor
Prof. Ana Aguiar, Associate Professor, Faculty of Engineering of the University of Porto

Acknowledgements

First of all, I would like to thank my parents for their unconditional love and support throughout my academic journey. Your encouragement gave me the confidence to pursue my goals even during the most stressful times. Thank you for always believing in me.

I also want to thank my grandparents, even though they could not witness the completion of this journey, their love and values continue to inspire and guide me.

To my friends, thank you for your company, encouragement and support. I am so grateful to have shared this chapter of my life with you, I could not have done this without you.

Finally, I would like to express my gratitude to my supervisors, Prof. Sérgio Crisóstomo and Prof. Ana Aguiar, for the opportunity to work on this project. Their guidance, patience and support throughout the development of this thesis were fundamental for its successful completion.

I would also like to thank the Institute of Telecommunications for hosting this work.

Resumo

A transição para as redes móveis de quinta geração (5G) introduz uma Arquitetura Baseada em Serviços (Service-Based Architecture), fundada em princípios cloud-native, onde as funções de rede comunicam através de interfaces normalizadas. O núcleo da rede 5G introduz o Network Exposure Function (NEF), uma função que permite a aplicações externas aceder de forma segura a capacidades e eventos da rede através de Application Programming Interfaces (APIs) padronizadas.

Embora o NEF permita o desenvolvimento de diversas aplicações externas, existem relativamente poucas soluções direcionadas para a investigação de problemas de conectividade que afetam utilizadores em determinadas áreas geográficas. A informação disponibilizada pelo NEF, em particular eventos de perda de conectividade e de localização, possibilita a criação de serviços de monitorização dedicados que apoiam os operadores na identificação e análise de problemas recorrentes de conectividade.

Esta dissertação apresenta o desenho, implementação e avaliação de um serviço de monitorização dedicado para redes 5G que interage com o NEF através da Monitoring Event API. A Network Application proposta utiliza subscrições de Loss of Connectivity e Location Reporting para auxiliar a deteção e investigação de eventos de perda de conectividade. O sistema associa eventos de perda de conectividade à última localização conhecida do equipamento afetado. A informação recolhida é armazenada, processada e apresentada através de uma interface web e de painéis de visualização no Grafana, fornecendo aos operadores contexto adicional sobre onde ocorrem problemas de conectividade e com que frequência estes afetam os utilizadores.

Uma vez que não foi possível aceder a um núcleo de rede 5G durante o desenvolvimento deste trabalho, a solução proposta foi testada com o ATNoG NEF Emulator. Para suportar o fluxo de monitorização proposto, alguns parâmetros adicionais de Location Reporting necessários ao serviço de monitorização foram adicionadas ao emulador. A Network Application foi avaliada num ambiente emulado através de múltiplos cenários de teste. A avaliação permitiu validar o fluxo de monitorização, confirmar a correta comunicação entre a Network Application e o NEF Emulator e demonstrar a capacidade do sistema para processar um elevado volume de eventos de monitorização ao longo dos cenários considerados.

Os resultados obtidos demonstram que o serviço de monitorização é capaz de relacionar

eventos de perda de conectividade com informação de localização e fornecer informação útil para a análise de problemas de conectividade ao nível da célula e do User Equipment (UE). Esta dissertação mostra o potencial do desenvolvimento de serviços de monitorização dedicados que complementem as abordagens tradicionais de gestão e apoiam o diagnóstico e resolução de problemas em redes 5G.

Keywords: 5G; Network Exposure Function (NEF); Serviço de monitorização; Perda de conectividade; Relatório de Localização; Aplicações de Rede; Monitoring Event API.

Abstract

The transition to fifth-generation (5G) mobile networks introduces a cloud-native, Service-Based Architecture (SBA) where network functions communicate through standardized interfaces. The 5G Core (5GC) introduces the Network Exposure Function (NEF), a function that allows third-party applications to securely access network capabilities and events through standardized APIs.

While the NEF enables various external applications, there are relatively few solutions that specifically address the investigation of connectivity issues impacting users in certain geographical areas. The information exposed by the NEF, particularly loss of connectivity and location events, makes it possible to create a dedicated monitoring service to support operators in identifying and analyzing recurring connectivity issues.

This thesis presents the design, implementation and evaluation of a dedicated 5G monitoring service that interacts with the NEF through the Monitoring Event API. The proposed Network Application uses Loss of Connectivity and Location Reporting subscriptions to help detect and investigate connectivity loss events. The system associates connectivity loss events with the last known location of the affected User Equipment (UE). The collected information is stored, processed and presented through a web interface and Grafana dashboards, providing operators with additional context regarding where connectivity issues occur and how frequently they affect users.

As access to a production 5GC was not available at the time of development, the Network Application was tested with the ATNoG NEF Emulator. To support the monitoring workflow, the emulator was specifically extended during this work to support the additional location reporting parameters that were required by the monitoring service. The Network Application was evaluated in the emulated environment through multiple testing scenarios. The evaluation verified the monitoring workflow, the proper communication between the Network Application and the NEF Emulator, and the system's capacity to handle a high volume of monitoring events throughout the given testing scenarios.

The results obtained show that the monitoring service can correlate connectivity loss events with location data and provide useful information for connectivity analysis at the cell and the UE level. This thesis highlights the potential for developing dedicated monitoring services that enhance traditional network management approaches and support troubleshooting operations within 5G networks.

Keywords: 5G; Network Exposure Function (NEF); Monitoring Service; Loss of Connectivity; Location Reporting; Network Application; Network Applications; Monitoring Event API.

Table of Contents

List of Tables viii
List of Figures ix
Listings xi
List of Abbreviations xii

1. Introduction 1
   1.1. Context 1
   1.2. Motivation 2
   1.3. Objectives 2
   1.4. Contributions 3
   1.5. Outline 3
2. Background and State-of-the-Art 5
   2.1. 5G Network Architecture 5
   2.2. Network Exposure Function (NEF) 9
   2.3. NEF for External Service Integration 11
   2.4. Challenges in 5G Monitoring 13
   2.5. Related Work 15
      2.5.1. Service-Based Exposure Research 15
      2.5.2. Edge-Cloud Integration 16
      2.5.3. Vertical-Specific Monitoring Studies 17
      2.5.4. Open-Source 5G Implementations 18
3. System Requirements and Architecture 21
   3.1. Use Case 21
   3.2. System Requirements 23
      3.2.1. Functional Requirements 23
      3.2.2. Non-Functional Requirements 24
   3.3. Architecture 25
      3.3.1. Component Description 26
      3.3.2. Database Schema 27
   3.4. System Interfaces 30
      3.4.1. NEF Interaction 30
      3.4.2. User Interaction 34
   3.5. Security and Privacy Considerations 34
4. System Implementation 36
   4.1. Technology Stack 36

4.2. Database Implementation 37
4.3. Backend Implementation 41
4.3.1. Monitoring Subscription Integration 42
4.3.2. Notification Processing and Event Handling 44
4.4. System Interface 46
4.5. Grafana Dashboard 50
5. NEF Emulator 53
5.1. ATNoG NEF Emulator 53
5.2. Contributions to the NEF Emulator 57
5.3. Interaction with the Network Application 64
6. Testing and Evaluation 68
6.1. Evaluation Setup 68
6.2. Use Case Testing 69
6.2.1. UE Movement Routes 70
6.2.2. Testing Scenarios 73
6.3. Metrics and Results 75
6.3.1. Monitoring Workflow Validation 75
6.3.2. Testing Scenarios Results 77
6.4. Analysis 80
6.4.1. Discussion 82
7. Conclusion and Future Work 84
7.1. Limitations 85
7.2. Future Work 86

Bibliography 87

List of Tables

2.1. Summary of open-source 5G implementations. 20

3.1. Functional Requirements. 24
3.2. Non-Functional Requirements. 24

5.1. Summary of the components altered and extended in the ATNoG NEF Emulator. ... 64

6.1. Time intervals corresponding to the executed testing scenarios. 78
6.2. Summary of monitoring events and subscriptions results for each testing scenario. .. 78
6.3. Summary of the results obtained in the testing scenarios. 83

List of Figures

2.1. Architecture of the 5G System [1, p. 43]. 8
2.2. NEF's Northbound Interface [2, p. 40]. 9
2.3. NEF's Southbound Interface [10, p. 16]. 10
2.4. Event Exposure using Local NEF [9]. 17

3.1. Use Case Diagram. 22
3.2. System Context Diagram. 25
3.3. Container Diagram. 26
3.4. Database Schema. 28
3.5. Message Sequence Diagram between the AF, the NEF and the AMF. 31

4.1. Monitoring Report Workflow. 43
4.2. Notification Processing Workflow. 45
4.3. Manage UEs Interface. 47
4.4. Monitoring Report Creation Interface. 47
4.5. Monitoring Reports Interface. 48
4.6. Monitoring Report Detail. 48
4.7. Events Interface. 49
4.8. Event Detail Interface. 49
4.9. Notifications Interface. 50
4.10 Grafana Dashboard. 51

5.1. High-level overview of the ATNoG NEF Emulator. 54
5.2. PostgreSQL database schema of the ATNoG NEF Emulator. 55
5.3. ATNoG NEF Emulator management interface. 56
5.4. ATNoG NEF Emulator movement control interface. 56
5.5. PostgreSQL database schema highlighting in red the additions to the UE entity. 59
5.6. Initial state of a UE instantiated within the coverage of Cell A. 60
5.7. Location state after UE leaves cell A and loses coverage. 60
5.8. Location state after the UE reconnects to the network through cell B. 61
5.9. Flowchart of the processing workflow that supports multiple *locationTypes* values for Location Reporting subscriptions. 62
5.10 ATNoG NEF Emulator architecture highlighting the components altered during this work. 63
5.11 Updated architecture of the Network Application including the interaction with the ATNoG NEF Emulator API. 65
5.12 Updated use case diagram including the interactions with the NEF Emulator API. 66

6.1. Evaluation environment deployed on Google Cloud, showing the communication interfaces between the Network Application and the ATNoG NEF Emulator. 69
6.2. UE movement routes configure in the ATNoG NEF Emulator. 71
6.3. Cell placement and coverage areas configure in the ATNoG NEF Emulator. 72
6.4. Combined view of the UE movement routes and cell deployment used during the testing. 73
6.5. Report Summary Table generated by Grafana. 79

6.6. Example of a failed notification delivery observed in the NEF Emulator packet captures. 81
6.7. Example of a notification that was successfully delivered observed in the Network Application packet captures. 81
6.8. Comparison between a successful and a failed Location Reporting subscription request. 82

Listings

3.1. Loss of Connectivity subscription creation. 32
3.2. Loss of Connectivity subscription creation response. 32
3.3. Loss of Connectivity notification. 32
3.4. Location Reporting subscription creation. 33
3.5. Location Reporting response. 33
6.1. Network Application log showing the creation of a Loss of Connectivity subscription. 76
6.2. NEF log showing the successful delivery of a Loss of Connectivity notification to the Network Application. 76
6.3. Network Application log displaying the processing of a Loss of Connectivity notification. 76
6.4. Network Application log indicating the creation of a Location Reporting subscription. 77
6.5. Network Application log showing the processing the Location Reporting response. 77
6.6. NEF log showing that the notification was not received by the Network Application. 80

List of Abbreviations

3GPP 3rd Generation Partnership Project.

5GC 5G Core.

5GS 5G System.

AF Application Function.

AKA Authentication and Key Agreement.

AMF Access and Mobility Management Function.

API Application Programming Interface.

ATNoG Aveiro Telecommunications and Networking Group.

AUSF Authentication Server Function.

CAPIF Common API Framework.

CN Core Network.

CP Control Plane.

CUPS Control and User Plane Separation.

DN Data Networks.

eMBB Enhanced Mobile Broadband.

EPC Evolved Packet Core.

gNb gNodeB.

GPSI Generic Public Subscription Identifier.

HTTP Hypertext Transfer Protocol.

IoT Internet of Things.

IP Internet Protocol.

IT Instituto de Telecomunicações.

JSON JavaScript Object Notation.

L-NEF Local Network Exposure Function.

LTE Long-Term Evolution.

MAC Media Access Control.

MEC Multi-Access Edge Computing.

mMTC Massive Machine Type Communications.

MNEF Microservice and Event-Driven Network Exposure Function.

MSISDN Mobile Station International Subscriber Directory Number.

NEF Network Exposure Function.

NF Network Function.

NGW Northbound Gateway.

NRF Network Repository Function.

NSA Non-Standalone.

OAI OpenAirInterface.

ORM Object-Relational Mapper.

PCF Policy Control Function.

QoS Quality of Service.

RAN Radio Access Network.

SA Standalone.

SBA Service-Based Architecture.

SBI Service-Based Interfaces.

SCEF Service Capability Exposure Function.

SGW Southbound Gateway.

SMF Session Management Function.

TCP Transmission Control Protocol.

UDM Unified Data Management.

UDR Unified Data Repository.

UDSF Unstructured Data Storage Function.

UE User Equipment.

UP User Plane.

UPF User Plane Function.

URLLC Ultra-Reliable Low-Latency Communication.

VM Virtual-Machine.

1. Introduction

1.1. Context

Mobile communications networks have experienced significant improvements with the introduction of the fifth-generation (5G) network. In addition to improvements in performance, scalability and flexibility, 5G introduced a cloud-native and service-based architecture in which network functions are implemented as modular services and interact through standardized interfaces [1]. This architectural approach facilitates the development of new services and allows network capabilities to be exposed in a controlled and secure manner.

One of the network functions natively introduced in the 5G Core (5GC) is the Network Exposure Function (NEF). Defined by the 3rd Generation Partnership Project (3GPP), the NEF provides standardized APIs allowing external applications to securely access network information [1]. By serving as an intermediary between external services and the Core, the NEF allows applications to utilize network capabilities without needing direct access to internal network functions [2].

The exposure of network information through the NEF creates new opportunities for application development capable of supporting network operations and service management. Through the APIs provided by the NEF, external services can obtain information regarding network conditions, user connectivity and other operational events occurring within the network [2]. This allows operators to complement the traditional network management approaches with dedicated applications that take advantage of network information to provide additional visibility, support network monitoring and investigation of network issues.

The dissertation developed within the scope of the Route 25: Agenda for Autonomous, Intelligent, Interoperable, and Inclusive Mobility Project, supported by Instituto de Telecomunicações (IT) *. Its primary goal is the development of a specialized monitoring service capable of interacting with the NEF and take advantage of its capabilities to provide troubleshooting and service to users who experience low service quality in a specific geographical area. These services can assist operators in identifying connectivity issues and improving the overall quality of experience delivered to mobile network users.

*https://www.it.pt/Projects/Index/4824

1.2. Motivation

Since the NEF provides a standardized mechanism to expose network events and information to external services [1], the development of applications that leverage these capabilities for dedicated monitoring and troubleshooting remains an area with great potential. Conventional monitoring solutions focus mostly on infrastructure level metrics such as resource utilization, service availability and network performance indicators. These metrics are critical for network operation, but offer limited insights into connectivity problems occurring in specific geographical areas.

As mobile networks grow to accommodate more users and services, operators need tools that allow them to investigate connectivity problems more efficiently. In particular, it can be difficult to identify recurring problems related to specific users, cells or locations when relying only on traditional monitoring approaches. With the ability to access network events and information, the NEF creates an opportunity to complement existing monitoring solutions with services capable of providing additional context regarding user connectivity and network behavior.

The motivation for this dissertation is the possibility to use NEF capabilities to develop a dedicated monitoring service to help operators investigate connectivity problems. This kind of service can combine information and monitoring functionalities to provide additional insights on connectivity events and allow for more efficient troubleshooting processes in 5G networks.

1.3. Objectives

The main objectives of this dissertation are summarized as follows:

- Design, implement and evaluate a monitoring service that exploits the NEF to support the investigation of connectivity problems in 5G networks;
- Develop a Network Application that communicates with the NEF through standardized APIs to monitor UE loss of connectivity events;
- Collect, process and correlate Loss of Connectivity and Location Reporting events to provide additional information on connectivity issues affecting UEs in specific areas;
- Provide visualization tools to assist with troubleshooting and the analysis of recurring events;

- Present a proof-of-concept implementation that shows how NEF capabilities can be used to build dedicated monitoring services that complement other network management approaches.

1.4. Contributions

The work developed throughout this dissertation resulted in the following main contributions:

- Design and implementation of a dedicated 5G monitoring service that uses the NEF Monitoring Event API to detect and investigate loss of connectivity events at the cell and UE level;
- Development of a web monitoring interface and visualization dashboards to provide additional information to users regarding loss of connectivity events and their location information;
- Extension of the ATNoG NEF Emulator to support the Location Reporting attribute *locationType* required by the Network Application to perform the proposed monitoring workflow;
- Evaluation and testing of the Network Application with the NEF Emulator through different scenarios.

1.5. Outline

This thesis is organized into seven chapters. Following this introduction, Chapter 2 presents the background and related work relevant to the development of the proposed monitoring service, including the 5G network architecture, the NEF and its use for external service integration, the challenges of 5G monitoring and similar works.

Subsequently, Chapter 3 describes the requirements and overall design of the proposed monitoring solution. It describes the use case implemented, the system requirements, the architecture of the monitoring service, its main components and their interactions. To finalize, security and privacy considerations are made.

On Chapter 4 it is detailed the implementation of the monitoring service. It describes the technologies used, the implemented functionalities and shows visualizations for both the interface and Grafana.

Chapter 5 presents the ATNoG NEF Emulator. It contains a summarization of the main contributions and implementation choices due to its interaction with the Network Application.

Then Chapter 6 describes the evaluation of the monitoring system, including the evaluation setup, the use case testing, the metrics and results, finishing with the analysis and discussion.

Finally, Chapter 7 concludes the thesis with a presentation of the challenges faced and some suggestions for future work.

2. Background and State-of-the-Art

This chapter provides a theoretical and technical foundation for the development of a dedicated 5G monitoring service based on the NEF. It begins with an overview of the evolution of mobile network architectures, focusing on the transition to the service-based design introduced in 5G. Key components of the 5G Core (5GC) are described with particular emphasis on the NEF and its role in enabling secure, programmable access to network capabilities. The chapter also discusses the challenges associated with performance monitoring in modern mobile networks and reviews existing approaches and solutions in the literature. This analysis highlights the limitations of current monitoring tools and establishes motivation for the solutions proposed in this thesis.

2.1. 5G Network Architecture

The fifth-generation (5G) architecture marks a significant change in the telecommunications infrastructure in comparison to its previous generations, not only for the higher data rates that it can provide, but also for the various services it can accommodate. In comparison to its previous generation, 4G, which contains a more rigid architecture optimized for mobile broadband, 5G implements a Service-Based Architecture (SBA) that provides network slicing, supporting various applications and services, such as Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communication (URLLC), and Massive Machine Type Communications (mMTC) [3].

This architecture, as defined by the 3GPP, is composed of two major domains, those being the Radio Access Network (RAN) and the 5GC Network. The RAN is in charge of handling the wireless communication between the UE and the network. While the core is responsible for managing mobility, sessions, authentication and routing user traffic [1]. Although the RAN plays a crucial role in performance, the focus of this work is on the 5GC, where service exposure and integration with external applications occur.

The 5GC leans on the core principles of the 4G Evolved Packet Core (EPC), as both architectures handle the network's core functions, while still introducing key architectural improvements. The 4G EPC relies on a point-to-point architecture where the network elements are connected via rigid, standardized interfaces, where hardware and software are closely integrated and data is restricted to specific nodes. This design presented some limitations regarding scalability and innovation and to overcome these constraints,

the 5GC was designed to meet the diverse requirements of the 5G era by introducing a cloud-native and service-based design between control-plane functions [4].

A common element between both cores is the Control and User Plane Separation (CUPS) [3]. It was initially introduced in 4G as an enhancement, however it is a necessary and indispensable feature in 5G since it allows operators to scale signaling resources in the Control Plane (CP) and traffic forwarding resources in the User Plane (UP) independently. In order to increase efficiency in real-world scenarios, Control Plane Functions can be based in a data center, while User Plane Functions can be distributed geographically closer to the user [4]. This implementation is crucial for Multi-Access Edge Computing (MEC), facilitating local traffic to minimize latency for mission-critical applications [5].

To enhance resilience and scalability, 5G network functions are designed to be stateless [1]. This is accomplished by decoupling computer resources from data storage, meaning network functions do not keep user context or session state locally between transactions. Instead, they store and retrieve this information from a dedicated storage layer, specifically dynamic state data is stored in the Unstructured Data Storage Function (UDSF) and subscription and policy data are stored in the Unified Data Repository (UDR) [1, 6]. In case a Network Function (NF) instance fails, a new instance can be created by retrieving the context from the common storage layer, preventing service interruption [1].

The functional design of these principles is achieved through the SBA. The monolithic network entities from previous generations are decoupled into modular NF that communicate through service-based interfaces instead of dedicated physical links [1, 6]. The Network Functions expose their capabilities as individual services to the other NFs via standardized API, normally using Hypertext Transfer Protocol (HTTP)/2 over Transmission Control Protocol (TCP) [6].

To manage this complex system, the Network Repository Function (NRF) serves as a centralized directory that allows NF to register and discover each other's services without manual interference [1]. This change improves scalability and supports advanced capabilities such as network slicing. Furthermore, it provides the necessary foundation for Network Capability Exposure, a core principle of 5G that allows the network to share internal information with external entities. This is accomplished with the NEF as the central gateway responsible for bridging the 5GC with external AF, often referred to as Network Applications in vertical industries.

While the 5GC contains a wide range of network functions, this work focuses on the most relevant to session management, mobility and exposure of network information, as described below [1].

- AMF: Responsible for registration, connection, reachability and mobility management. It terminates the N1 (UE) and N2 (RAN) signaling interfaces;
- Session Management Function (SMF): Handles session management. It is responsible for allocating UE Internet Protocol (IP) addresses, as well as controlling the User Plane Function (UPF) for traffic steering and enforcement via the N4 interface;
- User Plane Function (UPF): The anchor point for packet routing and forwarding, serving as the interconnect point between the mobile infrastructure and external Data Networks (DN). It performs packet inspection, traffic usage reporting and Quality of Service (QoS) handling;
- Policy Control Function (PCF): Provides a unified policy framework to govern network behavior. It makes decisions regarding QoS and charging rules and provides these policies to control plane functions for enforcement;
- Unified Data Management (UDM): Manages user identification, subscription data and authentication credentials. It supports the generation of Authentication and Key Agreement (AKA)) credentials and authorizes access based on subscription;
- Authentication Server Function (AUSF): Acts as an authentication server, supporting the unified authentication framework that allows the 5GC to support 3GPP and Non- 3GPP technologies.

Figure 2.1 presents the high-level architecture of the 5G System (5GS), focusing on the main network functions of the 5GC and their interactions through the Service-Based Architecture.

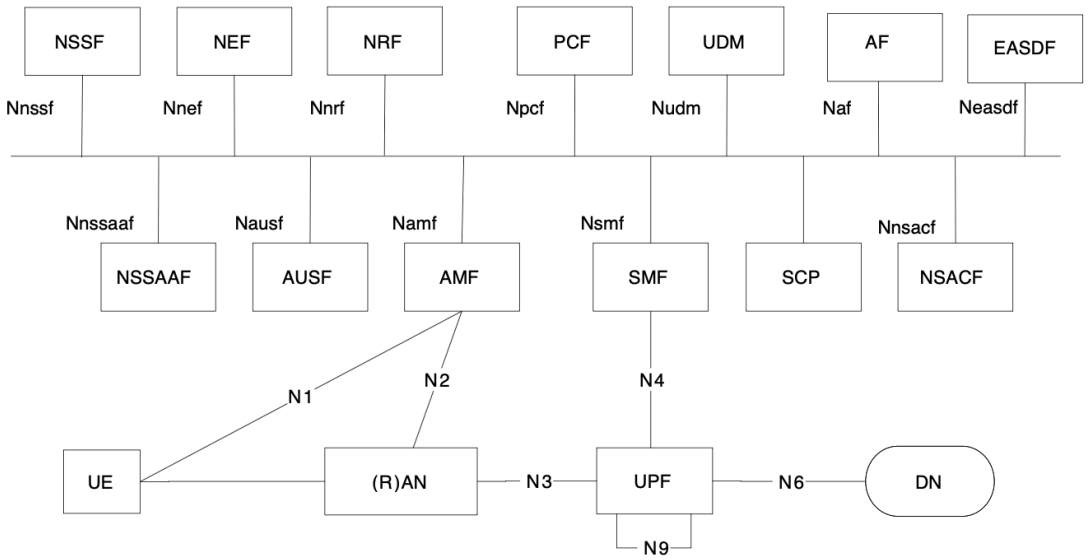


Figure 2.1: Architecture of the 5G System [1, p. 43].

The communication between these network functions is defined as a producer-consumer model. In this framework, a NF acts as a Service Producer when it exposes its capabilities and as Service Consumer to request a capability from another NF. These interactions can usually work as a Request and Response, where a consumer requests a specific action or information and the producer provides a direct response, or a Subscribe and Notify, where a consumer subscribes to specific events and the producer automatically notifies the consumer when those events occur. For monitoring purposes, the Subscribe and Notify model is vital since it allows network functions to post real-time updates to the NEF without constant polling [3].

While the Service-Based Architecture and the producer-consumer model ensure the stability for internal network operations, 5G goes beyond the operator's trust domain to integrate with external vertical industries and third-party applications [3]. However, giving these external Application Functions direct access to internal Service-Based Interfaces (SBI) would pose major security risks and expose sensitive information [7]. To prevent such risks, the 5G architecture includes the NEF to allow external services to monitor and influence network behavior while preserving internal security.

2.2. Network Exposure Function (NEF)

The Network Exposure Function (NEF) is an essential element in the 5G SBA, since it is the primary interface for external applications to interact with the core network. According to the 3GPP standards, the NEF's main purpose is to expose network capabilities, events and analytics to third-party applications. All while ensuring that the external entities can consume complex network insights without compromising its security or exposing its information [1].

Functionally, the NEF evolved from the Service Capability Exposure Function (SCEF) introduced in 4G Long-Term Evolution (LTE). The SCEF was often an add-on for specific Internet of Things (IoT) use cases, while the NEF is a native function of the 5G Core intended to support a broader range of 5G services [8].

In order to play its role as a gateway, the NEF's connectivity is divided into a northbound interface to interact with external application functions and a southbound interface to communicate with the internal 5GC network. On the northbound side, the NEF communicates with external application functions over the N33 reference point. Following 3GPP signaling protocols, it exposes network capabilities through standardized RESTful API, which may be managed under the Common API Framework (CAPIF) to ensure its security. As we have mentioned previously, these APIs use HTTP/2 with JavaScript Object Notation (JSON) serialization to ensure compatibility with modern applications [1, 2]. The northbound interface is shown in the figure.



Figure 2.2: NEF's Northbound Interface [2, p. 40].

On the southbound side, as shown in figure 2.3, the NEF interacts with internal Network Functions through the SBA. Within this domain, the NEF works both as a service consumer and a service producer. It acts as a consumer when it receives requests from an AF, translates the standardized API calls and invokes internal services to fulfill specific demands. Alternatively, it serves as a producer when it exposes its own SBI, the Nnef, which offers its services to other internal network entities [9, 10].

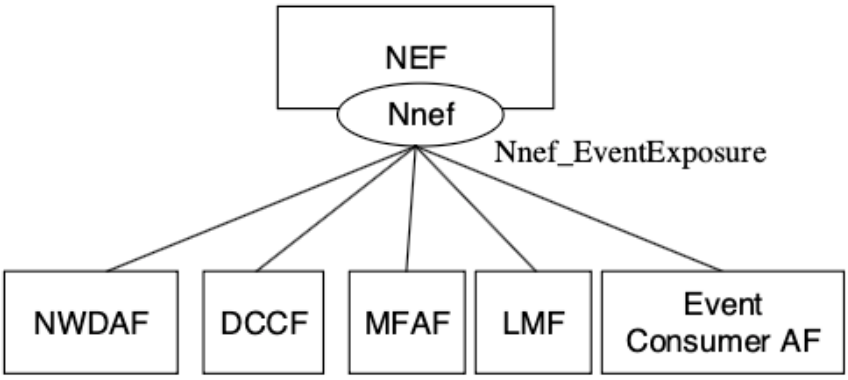


Figure 2.3: NEF's Southbound Interface [10, p. 16].

The NEF exposes various services that enable external applications to interact with network capabilities. However, considering the scope of this thesis, we will focus on the Monitoring Event API. This API allows external applications to subscribe to specific mobility and session-related events, such as location reporting and loss of connectivity events. Although the Monitoring Event API is specified by 3GPP as part of the T8 interface, which was initially introduced for the SCEF, in 5G networks the same functionality is exposed through the NEF [11]. Therefore, external applications communicate with the NEF through the N33 reference point while using the procedures and services defined by the Monitoring Event API [11].

To succeed in providing open access to external services, the NEF is responsible for several critical operations to maintain important internal data secured. Firstly, it manages authentication and authorization by validating the identity of external AFs before processing any requests. Alongside the OAuth 2.0 protocol, the NEF confirms access tokens to assure that the AF has the correct permissions to access precise APIs [1, 2].

A similar and relevant security function that the NEF implements is the concealing of the internal structure of the network. In hiding sensitive details, such as IP addresses and identities of network functions, from external third-parties, it prevents potential attackers from mapping the network's internal structure [4]. Furthermore, it acts as a protocol translator, converting specific 3GPP internal parameters into standardized RESTful APIs using JSON data formatting. This allows for modern web-based applications to interact with the core without the need to implement complex protocols [1].

Another NEF's implementation to protect the core network is to limit processing resources from getting overwhelmed by exaggerated API calls. These policies ensure that incoming

traffic from external software does not impact network balance or consume resources needed for critical network operations [1].

At the same time, the NEF protects the UE's privacy from third-party applications by separating internal and external schemas. The AFs interact with the 5G system using external identifiers, such as Generic Public Subscription Identifier (GPSI), which can either be Mobile Station International Subscriber Directory Number (MSISDN), an External Identifier or an Group External Identifier [1]. This translation is achieved by invoking services from the UDM.

For the purpose of this thesis, the Monitoring Events API is particularly crucial. The NEF serves as a core component for monitoring and allowing AFs to subscribe to specific events regarding UEs [11]. Additionally, it manages subscription state and usually takes advantage of the UDR to store persistent data about active subscriptions.

In summary, the NEF transforms the 5G Core (5GC) from a closed telecommunications system into a programmable platform, allowing secure interaction with third-party services while maintaining the core secure. This architectural capability lays the foundation for the integration of vertical services, as detailed in the following section on NEF for External Service Integration.

2.3. NEF for External Service Integration

The incorporation of external services through the NEF reflects the change from a conceptual standpoint of the network to a more practical application. As we have seen previously, the NEF provides the secure gateway for an open access network, although this heavily depends on how external entities, particularly Network Applications, interact with this exposed layer. By providing a standardized framework for this interaction, 5G allows diverse industries to interact with real-time network information [12].

As established in the 3GPP architecture, the entities consuming exposed APIs are categorized based on their relationship with the operator. Trusted AFs reside within the operator's trust domain. These are allowed to bypass the NEF and interact directly with internal Core Network Functions to request specific services [2]. On the other hand, untrusted AFs are considered third-party or external applications. As such, they do not have direct access to the 5GC and rely on the NEF to communicate with internal functions in order to keep the network integrity [1]. Within the context of vertical industries, these external functions are often referred to as Network Applications. These contain specialized

software components designed to fulfill the rigid demands of particular sectors and are engineered to use 5G data for the optimization of their services.

To simplify how these Network Applications discover and consume these services, the 3GPP introduced the CAPIF [1]. It provides a standardized API management framework that can be used alongside the NEF to simplify the exposure and consumption of network services. This implementation facilitates the integration process for developers, as it regulates important API provider domain functions. Furthermore, CAPIF handles security mechanisms, ensuring proper authentication and authorization, along with logging, throttling and API monetization [7, 11].

This ensures that the integration with a Network Application follows a structured workflow. Initially, the application registers itself in the system and then obtains an authorization token to securely identify itself when it wants to communicate with the network [11]. The application may then use CAPIF discovery mechanisms to identify the network services exposed by the API provider. After all, the Network Application is then able to create subscription requests to the network and receive the respective responses [9].

A critical responsibility of the NEF during this workflow is providing subscriber privacy through Identifier Mapping. The separation between external and internal addresses is key to prevent external services from learning sensitive data regarding UE identities. A Network Application tracks devices through external identifiers, such as GPSI that looks like an email address, or external group identifiers [2]. When an API request is received with an external identifier, the NEF performs the translation procedures that map the identifier used in the 5GC. Such procedures usually rely on subscriber data stored in the UDM. This robust abstraction ensures the user's identity is kept private [1, 2].

The collaboration between Network Applications and the NEF allows the network to function as an active data source. By interacting with the NEF through standardized APIs, external applications can request information or subscribe to specific monitoring events related to a UE [2]. This is made possible through the Monitoring Event API, which enables the creation of subscriptions and the reception of event notifications. With this mechanism, the network can generate event notifications and deliver them to the applications whenever predefined conditions occur. As a result, external stakeholders can obtain real-time information about device connectivity, location or network performance, allowing the development of advanced monitoring services for distinct industries.

One example of location reporting services can be a logistics company that is interested in tracking cargo in order to optimize routing and improve delivery time estimations. A Network Application could implement this by subscribing to location updates regarding that cargo and then would receive information periodically about its position. Another example about location-based alerts and adding geofencing would be an application for drone flight controllers. In this case, the application must be notified when the device enters or leaves a predefined geographic area, enabling operational responses. A further relevant scenario is the detection and investigation of connectivity loss. By collecting information about communication failures and their causes, a Network Application can provide vital insights to help operators diagnose network issues. While the 3GPP defines a vast range of services, this thesis focuses on the Loss of Connectivity and Location Reporting events, as these represent the structure for the presented use case in the following chapters.

In providing a secure and standardized interface for AFs, the NEF allows external entities to interact with the network while maintaining the integrity and privacy of its operations. The capabilities mentioned in this section enable external applications to develop monitoring and analytics services based on network insights. However, despite these advantages, monitoring modern 5G networks introduces several challenges, which will be discussed in the following section.

2.4. Challenges in 5G Monitoring

The NEF provides extensive capabilities for the exposure of network services to external applications. Although, when exposing real-time network events, there are several challenges that must be considered when designing monitoring solutions. The service-based features of the 5GC, combined with the massive number of devices and services, create new constraints relevant to scalability, latency, data consistency, security and resource management.

One of the primary challenges in network monitoring is the signaling overhead generated by monitoring requests. In legacy systems, external applications would consecutively poll the network to obtain device information, which can lead to excessive signaling and can easily saturate the Control Plane's resources. To prevent this, the 5G architecture introduces an event-driven Subscribe-Notify model, where the external application can subscribe to a monitoring event and receive notifications upon its occurrence [2].

In spite of this model, scalability remains an important challenge. Monitoring solutions may need to manage a large number of active subscriptions and process a continuous stream of notifications generated by multiple devices. As the number of monitored UEs increases, the associated signaling and processing requirements can place additional pressure on the control plane [1].

Another challenge arises from the latency introduced by the network exposure. Although 5G promises extremely low latency for user-plane communication, specially for URLLC, monitoring information is delivered through the control plane, which follows a specific workflow [4]. When a monitoring event occurs, it is initially detected by an internal network function, subsequently transmitted to the NEF and then delivered to the AF through a REST-based API call. During this sequence, the NEF might need to interact with other internal functions or translate identifiers [9]. Each individual step includes processing and communication delays, which can affect the delivery of monitoring information that requires real-time responses [1, 2].

Furthermore, the data consistency of these notifications is challenged due to the 5GC's stateless architecture. Although NFs may maintain runtime state information, subscriber and service-related data is typically stored in repositories such as the UDSF and UDR. While this design enhances resilience, it creates difficulties for retrieving real-time monitoring information, as multiple NFs may access and update a UE's state simultaneously. Therefore, there is a possibility for external applications to receive outdated details about an UE status [1, 9].

Beyond performance, one of the many concerns when exposing network information to external applications is the security of the network and the privacy of its users, as monitoring systems require access to sensitive data. As we mentioned in sections 2.2 and 2.3, the NEF handles identifier mapping, access control and topology hiding [1]. While these mechanisms are essential to protect network resources and user privacy, they also limit the visibility available to external applications. As a result, AFs may not have access to all information required to perform detailed troubleshooting of network-related issues [1, 2].

Finally, the dynamic structure of 5G networks introduces additional challenges. As NFs can be scaled, instantiated, or relocated based on network conditions and traffic demands, it is imperative for when these changes occur that subscriptions and events keep being monitored without interruption [1, 9]. As maintaining this continuity requires cooperation

between NFs, systems and data repositories.

These challenges demonstrate that, even though the 5G architecture provides standardized mechanisms for exposing network information and supporting monitoring services, the development of efficient and reliable solutions remains a difficult task. Due to these concerns, several research initiatives and approaches have been proposed to improve monitoring and analytics capabilities in 5G networks. The following section explores the standardization efforts and research contributions pertaining to monitoring and network exposure in 5G systems.

2.5. Related Work

The development of monitoring and exposure mechanisms in 5G networks has been addressed through both standardization and academic research. Their focus being on improving efficiency and integration between the 5GC and external applications. This section reviews the most relevant contributions from Service-Based Exposure Research to open-source projects.

2.5.1 Service-Based Exposure Research

Recent academic research has focused on improving the efficiency and scalability of network exposure mechanisms in the 5G SBA, particularly regarding the performance of NEF's Northbound API and internal service interfaces. For instance, a recent study proposes a Microservice and Event-Driven Network Exposure Function (MNEF) architecture that features native CAPIF support. In order to support AF's demand, the MNEF logically separates NEF functionalities into independent microservices, following a Domain-Driven and API-first design [13].

Similarly to the NEF, this proposed architecture uses a Northbound Gateway (NGW) to handle requests from external AFs and converts them into event-oriented messages. Alongside a Southbound Gateway (SGW) for interfacing with CAPIF and the 5GC. To evaluate this proposal, researchers integrated the MNEF with open-source reference implementations, in particular Open5GS and OpenCAPIF [13].

Performance evaluations for this architecture focused on measuring key indicators under varying request loads. The results indicate that this implementation improves network exposure scalability, allowing the NEF to efficiently handle high traffic volumes and maintain

performance stability. However, the efficiency of the Nnef and associated southbound interfaces revealed that some dependencies within the 5GC can introduce significant constraints. Particularly, in some high request loads, the internal PCF became overloaded and created a system bottleneck, affecting the overall consistency of the exposure chain [13].

Despite the additional overhead from translating and distributing messages among microservices, the study finds that asynchronous communication and detached services are vital for guaranteeing the fault tolerance, scalability and security needed for exposing Northbound APIs to external applications in 5G and future networks.

2.5.2 Edge-Cloud Integration

The centralized 5GC provides a solid foundation for network monitoring, although routing event reports through a centralized control plane introduces latency that can be prejudicial for time-sensitive applications. URLLC, industrial automation and real-time analytics services require almost instant access to both user-plane data and control-plane events. To meet these requirements, the 5G introduces MEC, which extends cloud computing capabilities to the edge of the network [14].

As mentioned in section 2.1, the foundation for the integration of MEC is the CUPS, since it allows operators to decouple resources, enabling the UPF to work independently. To support low-latency services, a local UPF can be deployed at the network's edge, close to the RAN, routing traffic locally to applications. This implementation removes the need to send information to a centralized data center before returning it to the user [5].

While a local UPF optimizes the data plane, a centralized NEF can still introduce additional exposure latency on the control plane and reduce the latency benefits of edge computing. This is solved with the introduction of a Local Network Exposure Function (L-NEF), a NEF instantiated at the edge, closer to the UE and the local UPF [1, 9]. The L-NEF serves as a secure, low-latency gateway that allows local applications to interact with the network without the need to move through the central core. Additionally, the L-NEF can authorize local Network Applications to provision traffic influence rules, ensuring that traffic is always steered to the optimal local UPF based on real-time mobility and monitoring data [9].

A critical optimization in more recent releases is the ability to bypass the centralized core by exposing events directly through the Nupf_EventExposure_Notify service. Instead of

the UPF sending the monitoring events to the SMF, which would then send it to the NEF, the local UPF can send notifications directly to the L-NEF. Once the L-NEF receives the notification from the UPF, then it reports the information directly to the AF, as is displayed on the following diagram [9].

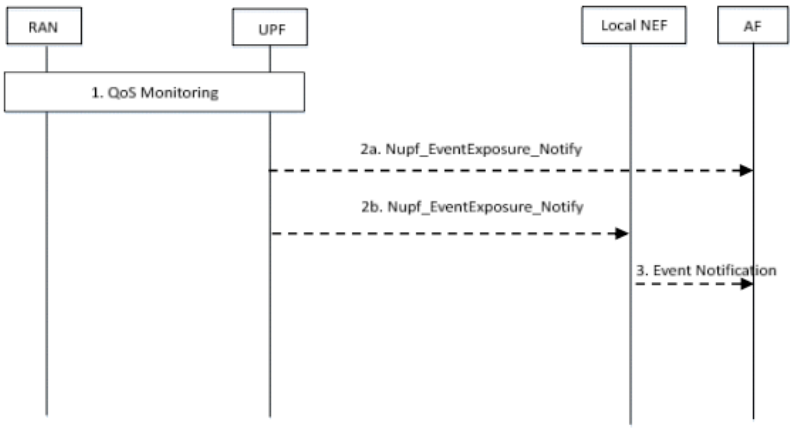


Figure 2.4: Event Exposure using Local NEF [9].

Existing literature explores the challenge of maintaining low-latency exposure since users move across different edge domains. For instance, the Follow Me Edge-Cloud merges MEC with the Follow-Me Cloud principles for 5G automotive and URLLC systems. In this work, automotive edge services and their corresponding exposure contexts are migrated seamlessly between micro-data centers at the edge of the network as the vehicle moves. Research demonstrates that coupling the MEC platform with dynamic service migration algorithms guarantees that the UE is always connected to the geographically optimal edge [14].

The direct interaction between the local UPF and the L-NEF further helps minimize the exposure chain delays, ensuring a faster delivery of monitoring events to applications. Consequently, edge-cloud integration not only optimizes data routing but also enhances the scalability of monitoring services in 5G networks.

2.5.3 Vertical-Specific Monitoring Studies

With the introduction of network exposure capabilities in 5G, it is now possible to develop applications that are tailored to the needs of specific industries and operational environments. These applications go beyond the traditional network management functions and leverage the information exposed by the network to enable use cases in areas such as

smart manufacturing, transportation, healthcare, public safety, and media services. Consequently, monitoring solutions are increasingly evolving from generic infrastructure supervision tools to services capable of providing insights that matter for a given vertical or operational scenario.

One example of a vertical-specific monitoring application is for predictive maintenance in critical infrastructures. In this scenario, a Network Application gathered operational data from industrial machines and processed it to detect abnormal behavior that may indicate future failures. The system can help in maintenance planning and increase the reliability of critical infrastructure [15].

A project presented by 5GASP in the automotive sector is the Virtual On-Board Unit (vOBU) Network Application, which uses MEC to provide a virtual counterpart to the physical On-Board Unit (OBU) installed in vehicles. The application helps protect the OBUs from characteristic disconnections of vehicular networks while allowing them to offload heavy data processing tasks directly to the network edge [16].

The examples illustrate how network information can be combined with application-specific requirements to enable service reliability, mobility management and operational monitoring. Instead of focusing only on infrastructure metrics, these studies show the increasing implementation of monitoring solutions that take into account network context and user information to fit the needs of specific vertical domains.

2.5.4 Open-Source 5G Implementations

Open-source 5G implementations play an important role in enabling experimental validation, performance evaluation and architectural research regarding the 5GC and its service-based interfaces. Platforms such as Open5GS, free5GC and OpenAirInterface (OAI) provide accessible implementations of the core network functions defined by 3GPP. This allows researchers to experiment with control-plane functions, user-plane forwarding and service-based communication mechanisms. These projects have been widely adopted in various research to prototype monitoring solutions, evaluate scalability and analyze exposure-related methods within the 5GC.

Open5GS is an open-source implementation of the 5GC that provides a joint implementation of the 4G EPC and 5GC in both Standalone (SA) and Non-Standalone (NSA) modes. It provides key network functions of the 5GC such as the AMF, SMF, UPF, PCF and UDM,

allowing researchers to emulate registration procedures, session management and policy control interactions. In recent research focusing on network exposure, Open5GS has been successfully deployed alongside the MNEF architecture to validate large-scale API handling. This integration demonstrates that the Open5GS can reliably translate external NEF requests for QoS control and traffic influence into internal core signaling [17].

Another non-proprietary implementation is the free5GC, the first project aiming to comply with the 3GPP Release 15. This platform provides a robust implementation of the 5G SBA. Its design documentation details a wide range of Control Plane network functions and recent specifications officially support a native NEF. This addition allows free5GC to natively expose network capabilities over the N33 interface, making it a complete platform for testing standardized 3GPP northbound APIs and service-based monitoring [18].

OpenAirInterface is another accessible-source project that offers a 3GPP-compliant cellular stack that includes both the RAN and the a Core Network (CN) [19, 20]. Its RAN provides a fully functional software-defined gNodeB (gNb) and UE, while its core implements the key SBA functions. OAI is often used for end-to-end research on commercial off-the-shelf hardware.

Beyond core network implementations, Project CAMARA addresses the critical need for standardized and developer-friendly exposure of telecommunication capabilities. As an open-source project within the Linux Foundation, CAMARA works to define, develop and test APIs that enable seamless access to telco network capabilities. This can be accomplished by abstracting the 5G network complexity and transforming operator networks into serviceable platforms for consumers without specialized telecom expertise to integrate network functions. CAMARA provides free-to-use APIs and reference implementations for specific services such as device location tracking, traffic influence and connectivity insights, while ensuring strict data privacy and regulatory compliance [21].

Additionally, research-focused tools such as the ATNog NEF Emulator have been developed to emulate network exposure interactions without requiring a fully core deployment. This NEF Emulator provides a controlled framework for testing monitoring subscriptions, event notifications and northbound API interactions. Such emulators are particularly useful for evaluating monitoring architectures, validating event-driven workflows and analyzing exposure-related performance metrics [22].

Table 2.1 summarizes the main characteristics of the open-source 5G projects discussed in this subsection.

Platform	Components	Description
Open5GS	5GC	5GC and EPC implementation.
free5GC	5GC	5G Core implementation.
OpenAirInterface	5GC + RAN	Complete open-source 5G implementation.
Project CAMARA	APIs	Standardized telecom APIs over network exposure.
ATNoG NEF Emulator	NEF	NEF emulation for Network Application development and testing.

Table 2.1: Summary of open-source 5G implementations.

Overall, open-source implementations and exposure-oriented projects provide essential platforms for experimentation and validation of 5G monitoring mechanisms. These projects enable practical research into service-based communication, event exposure and network analytics, bridging the gap between theoretical specifications and real-world deployment scenarios.

3. System Requirements and Architecture

Building on the technical foundation introduced in the previous chapter, these sections describe the idea, design and structure of the proposed Network Application. The 3GPP standards offer the groundwork for network exposure via the NEF, but the implementation of a monitoring service requires the architectural definition that translates these capabilities into functional software.

This chapter provides the layout for the Network Application, starting by describing its use case and outlining the context in which the monitoring service is expected to function. The functional and non-functional requirements are identified, specifying all the capabilities the system must provide. These requirements are followed with a description of the system architecture, including an overview of the proposed solution and the interaction between its main components. The component description explains the role and responsibilities of each module, while the database schema clarifies how the data is stored. The system interfaces section examines the application's interaction with the user and the NEF, while following 3GPP standards. Finally, security and privacy considerations are addressed to ensure the system will protect sensitive information.

These elements together provide the design of the monitoring service, which serves as the foundation for the implementation and evaluation presented in the following chapters.

3.1. Use Case

The proposed system is designed as a dedicated monitoring service that leverages the 5G NEF to obtain real-time and event-driven information from the 5GC. The main goal of the service is to detect poor connectivity conditions in a specific geographic area and to generate performance statistics at both the cell and UE levels. The monitoring platform operates as an external AF that interacts with the NEF through the Monitoring Exposure API, subscribing to loss of connectivity and location reporting events and transforming the resulting notifications into key insights [11].

An important objective for this service is the abstraction of network complexity. By serving as an intermediary between the 5GC and the end-user, the system ensures that the user can take advantage of monitoring capabilities without requiring expertise in network

engineering or 3GPP protocols. As a result, the user of the system can be a network operator, a support team or an operations manager.

The following figure presents the use case diagram of the proposed monitoring service. The primary actor is the user that interacts with the Network Application to manage UEs, create monitoring reports and analyze the collected information through the monitoring dashboard. The NEF is represented as an external system, providing the monitoring subscriptions and event notifications required by the application. Internally, the monitoring service processes loss of connectivity and location reporting events, correlates them with UE and cell information, and provides visualization tools.

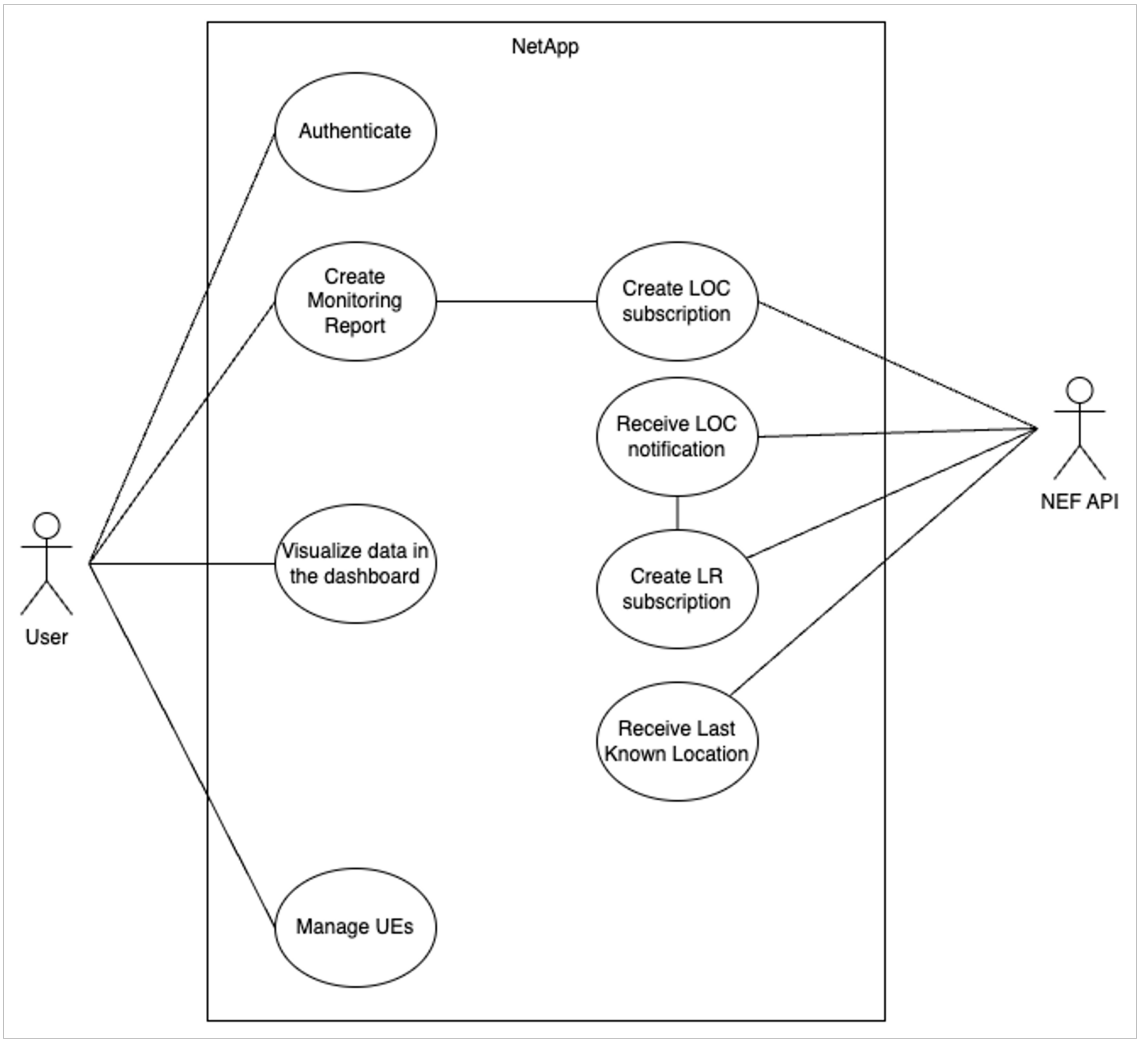


Figure 3.1: Use Case Diagram.

The operational context assumes that the user of the system provides a list of UEs to be monitored. The user can then use the service to create a monitoring report for the UEs it wishes to monitor. Upon creating the monitoring report, the system will create individual

subscriptions related to loss of connectivity events. When a UE loses connectivity, the application receives a notification from the NEF and creates another subscription requesting the UE's last known location. The system continuously stores the event notifications and correlates them with cell and UE information to determine where or how often connectivity issues occur.

The main use case for this system is detection and investigation of connectivity loss events. When a monitored UE experiences recurring loss of connectivity, the system compiles these events, along with location information requested, and associates them with its last serving cell. In putting together the frequency and distribution of such events, the monitoring service can provide information on a possible problematic cell, coverage gaps or defective equipment. The combination of this information allows operators to detect recurrent issues in a specific geographic area and focus on troubleshooting.

Additionally, the system will include a dashboard to portray the data collected from the monitoring reports. The panel includes a heat map showing the spatial distribution of connectivity issues across cells, time series graphs illustrating event frequency over time, tables reporting the latest events and a summary of each monitoring report, comparative statistics per UE and per cell, and a pie chart to analyze the different loss of connectivity reasons.

As a result, the proposed monitoring service transforms raw network exposure data into structured performance insights, supporting data-driven decision-making and proactive network management within the 5G system.

3.2. System Requirements

This section defines the criteria used to design and evaluate the Network Application. These requirements ensure that the application can serve as the bridge between the NEF capabilities and the end-user's needs.

3.2.1 Functional Requirements

The functional requirements describe the capabilities that the proposed system must provide. These requirements define the expected behavior of the system in terms of monitoring, processing and reporting connectivity failure for UEs.

ID	Requirement Description
FR1	The system shall authenticate users and manage user sessions.
FR2	The system shall retrieve the access token from the NEF.
FR3	The system shall allow the management of UEs.
FR4	The system shall allow the creation and deletion of monitoring reports for selected UEs.
FR5	The system shall subscribe to monitoring events via the NEF.
FR6	The system shall receive and process monitoring event notifications from the NEF.
FR7	The system shall automatically trigger the subscription of monitoring events based on the received notifications.
FR8	The system shall store monitoring data.
FR9	The system shall aggregate monitoring data per UE and cell.
FR10	The system shall provide a user interface to visualize monitoring data.

Table 3.1: Functional Requirements.

3.2.2 Non-Functional Requirements

The non-functional requirements define the quality attributes and constraints that the monitoring system must satisfy to ensure an effective performance.

ID	Requirement Description
NFR1	The system shall support concurrent processing of monitoring events without degradation of performance.
NFR2	The system shall maintain responsiveness of the user interface while background monitoring operations are executed.
NFR3	The system shall ensure reliable operation, including safe handling of failures during event processing.
NFR4	The system shall have a modular design to allows future extensions and integration of new features.
NFR5	The system shall ensure secure access through user authentication and controlled API interactions.
NFR6	The system shall handle a large number of UEs.
NFR7	The system shall process monitoring events with minimal delay.
NFR8	The system shall ensure data consistency and integrity during storage and processing operations.
NFR9	The system shall keep the user interface updated with the latest monitoring results.

Table 3.2: Non-Functional Requirements.

3.3. Architecture

This section presents the architecture of the proposed monitoring service using the C4 model, with a focus on the System Context Diagram and the Container Diagram. This model enables the architecture to be represented in various levels of abstraction, starting with a general overview of the system and progressively detailing its components and their responsibilities.

At the system context level, the monitoring service functions as an external Network Application that interacts with the user and communicates with the NEF using standardized APIs. The user interacts with the service to select UEs to be monitored, create monitoring reports and consult them. On the other hand, the NEF serves as a gateway to the 5GC, through which the monitoring service creates subscriptions and receives notifications.

As represented in this Figure, the monitoring service acts as a middleman between the user and the NEF. The user defines what should be monitored, while the NEF provides the network events data required to diagnose connectivity issues in devices or even a specific cell.

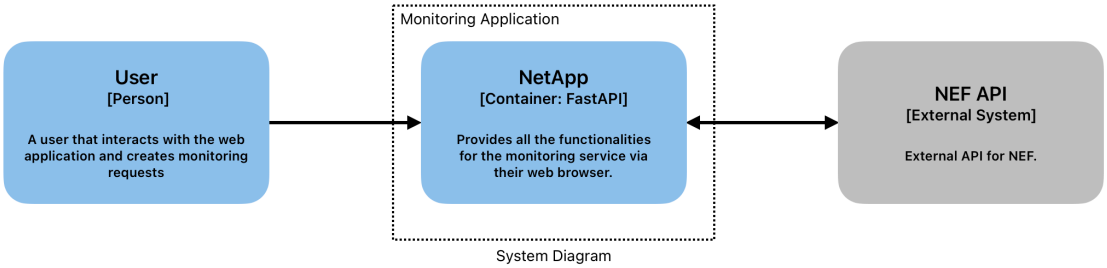


Figure 3.2: System Context Diagram.

At the container level, the monitoring service can be separated into four technical elements, namely, the frontend, the backend, the database and Grafana. The frontend focuses on user interaction and page routing, the backend concentrares on application logic and communication with the NEF, the database handles persistence and Grafana provides data visualization. A representation of the Container Diagram is shown in Diagram 3.3.

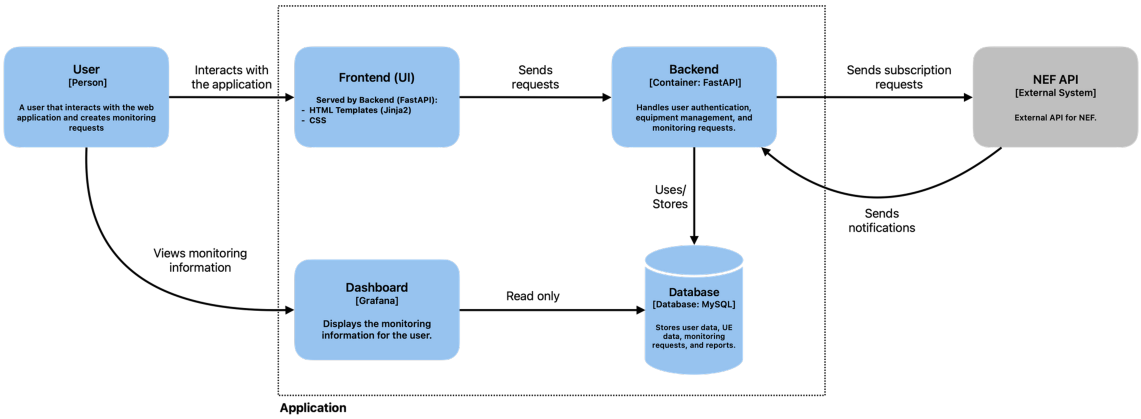


Figure 3.3: Container Diagram.

3.3.1 Component Description

This section describes the key components of the Network Application and their roles as part of the overall architecture. As illustrated in Figure 3.3, the four main components are the frontend, the backend, the database and Grafana.

The frontend is the presentation layer of the system and provides users an interface to interact with the monitoring service. Its key functions include user authentication, management of monitoring reports and UEs, access to monitoring information in their dedicated pages, deletion of monitoring subscriptions and cell retrieval.

The backend is the main component of the system, since it contains the core logic of the monitoring service. It is responsible for handling requests that come from the frontend, managing monitoring reports, communicating with the NEF to create subscriptions and receive event notifications, and processing events.

This component acts as the bridge between the system's interface and the 5G network exposure layer. It combines all business logic into monitoring reports, including event handling, data aggregation and storage. Although the system is developed using the FastAPI framework, the frontend and backend are shown as separate containers due to their distinct responsibilities. This ensures that the user interface remains independent of the network exposure protocols, improving maintainability and scalability.

The database is responsible for storing all the information produced and consumed by the monitoring service. This includes users, monitoring reports, registered UEs, events and cells. By storing all this data, the system is able to aggregate this information and provide the data required for visualization.

Grafana serves as the visualization layer of the system and is used to transform stored monitoring data into graphical representations by performing queries on the database. Through Grafana, users can inspect the latest event trends, connectivity statistics per UE or cell and analyze summarized reports.

3.3.2 Database Schema

The database schema defines how data is organized in the monitoring service, specifying the structure, relationships and constraints of the stored entities. It contains data related to users, token, reports, UEs, subscriptions, events, loss of connectivity notifications, location notifications and cells. The database schema for the monitoring service is shown in Figure 3.4.

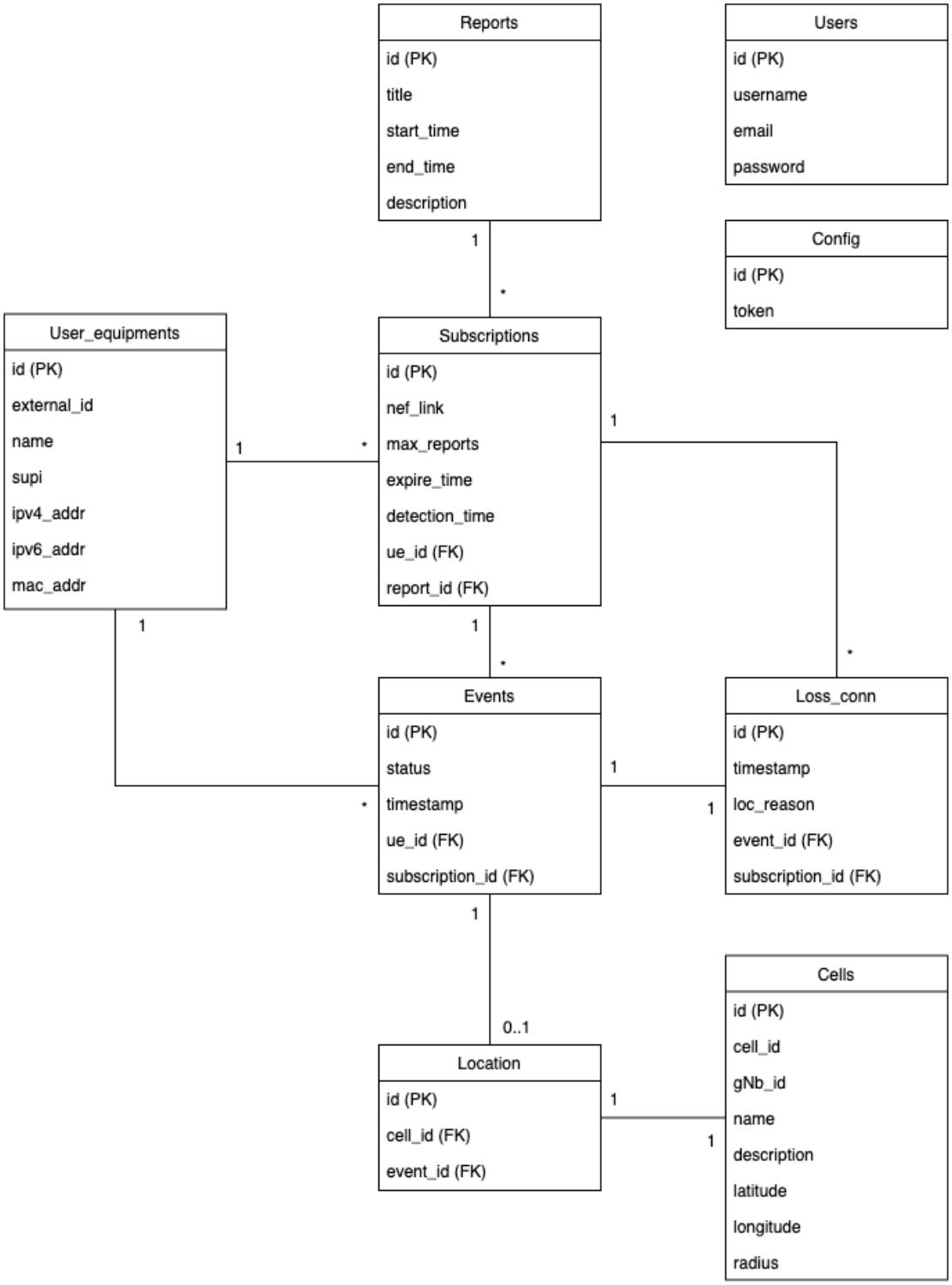


Figure 3.4: Database Schema.

The User entity represents the person responsible for interacting with the monitoring service. Its table is responsible for storing user authentication information. On the other hand, the Token entity is responsible for storing the access token used to interact with the

NEF. This separation ensures that the NEF access token can be updated independently of user accounts.

The User Equipment entity represents the devices that can be selected for monitoring. Its table contains additional attributes, despite only requiring a small set for monitoring purposes, in order to support UE management within the application.

The Report entity represents a monitoring report defined by a user. To monitor a specific set of devices, the user needs to provide the conditions under which monitoring is performed. For each device monitored, a unique subscription is created, therefore a report may involve multiple subscriptions, making this a one-to-many relationship.

The Subscription entity represents the loss of connectivity subscriptions created in the NEF. Each subscription is associated with a unique monitoring report and UE, making them a many-to-one relationship. Additionally, a single subscription can contain multiple events and notifications, representing a one-to-many relationship.

The Event entity represents a combination of loss of connectivity and location notifications in a single entry. For every loss of connectivity notification received, the system creates a new event and requests the device's last known location.

The LossConn entity represents the loss of connectivity notifications received by the NEF. It extends the event model by providing a loss of connectivity reason. Each notification is associated with a subscription and is directly related to a unique event.

The Location entity represents the location information received after the system subscribes to location reporting events in the NEF. It can be linked to an event when location information is available.

The Cells entity represents the radio cells of the network, including their identifiers and geographical information. This allows location information to be associated with specific locations.

Overall, this schema supports the main objectives of the monitoring service by organizing monitoring reports, subscriptions, events and cell information in a structured way. This organization enables historical analysis, event correlation and integration with visualization tools such as Grafana.

3.4. System Interfaces

This section defines the inputs and outputs of the monitoring service, describing how data is exchanged between the system, external users and the 5G network. To ensure interoperability with any 3GPP-compliant network, the Network Application follows the data models established in the standards while simplifying the exposed network data for the user.

The proposed system emphasizes interaction with the NEF, since it serves as the primary source of monitoring data and the main interface for network exposure. In addition, the system also processes user inputs related to monitoring configuration and provides outputs in the form of stored events, reports and visualized data. Together, these inputs and outputs define the monitoring service and support its role as an intermediary between network exposure and the user.

3.4.1 NEF Interaction

The interaction with the NEF is performed through the Monitoring Event API, which allows the Network Application to interact with the Core using standardized monitoring subscriptions and event notifications. Through this interface, the monitoring service can create monitoring subscriptions for specific UEs and receive asynchronous notifications once events occur [11].

As established in Section 3.1, the monitoring requirements of the system focus on Loss of Connectivity and Location Reporting subscription types. The following diagram illustrates the message sequence between the AF and the NEF to support the monitoring service goals. It visualizes its interaction, including subscription creation, corresponding response and asynchronous notifications for requests with more than one report.

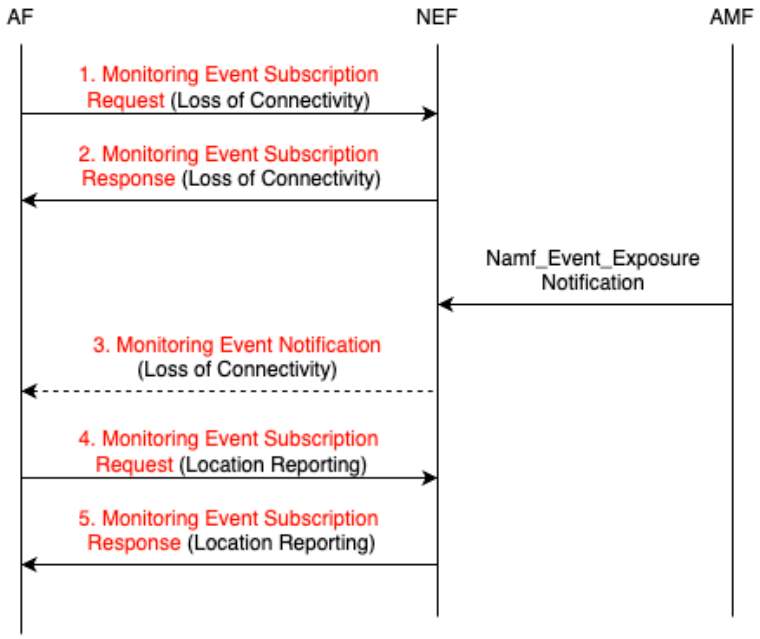


Figure 3.5: Message Sequence Diagram between the AF, the NEF and the AMF.

Monitoring event subscriptions and notifications share a common set of parameters independently of the selected monitoring type. These parameters define the monitoring type, the UE to be monitored, the destination of event notifications and the lifetime of the subscription [11, p. 27].

- *monitoringType* specifies the type of monitoring requested from the NEF, such as Loss of Connectivity and Location Reporting;
- *externalId* identifies the UE associated to the subscription request;
- *notificationDestination* specifies the URL identifying the recipient of the notification;
- *self* or *subscription* is the link to the individual monitoring event subscription.

The subscription lifetime can be configured through either of the following parameters:

- *maximumNumberOfReports* represents the maximum number of reports to be sent by the NEF;
- *monitorExpireTime* specifies the time at which the monitoring event is considered to expire.

According to the 3GPP specification, at least one of these parameters must be present to create a Monitoring Event Subscription [11, p. 27].

The following interactions detail each step of the monitoring process as shown in 3.5.

1. The following request illustrates what is sent by the Network Application to the NEF in order to create a Loss of Connectivity monitoring subscription.

```
{
"externalId":"10001@domain.com",
"notificationDestination":"http://34.14.121.58:8000/notifications",
"monitoringType":"LOSS_OF_CONNECTIVITY",
"maximumNumberOfReports":100,
"monitorExpireTime":"2026-04-27T13:06:42Z",
"maximumDetectionTime":1
}
```

Listing 3.1: Loss of Connectivity subscription creation.

   In addition to the common monitoring parameters previously described, the Loss of Connectivity subscription requires the *maximumDetectionTime* parameter, which defines the maximum period of time at which the UE is considered unreachable by the network.

2. After validating the request, the NEF returns a response containing information related to the subscription.

```
{
"self":"http://34.158.65.20:8888/nef/api/v1/3gpp-monitoring-
    event/v1/localhost/subscriptions/69ef5151aee73889eaf45675",
"externalId":"10001@domain.com",
"notificationDestination":"http://34.14.121.58:8000/notifications",
"monitoringType":"LOSS_OF_CONNECTIVITY",
"maximumNumberOfReports":100,
"monitorExpireTime":"2026-04-27T13:06:42+00:00",
"maximumDetectionTime":1
}
```

Listing 3.2: Loss of Connectivity subscription creation response.

3. When the predefined conditions are met, the NEF sends an asynchronous notification to the Network Application.

```
{
"subscription":"http://34.158.65.20:8888/nef/api/v1/3gpp-monitoring-
    event/v1/localhost/subscriptions/69ef5151aee73889eaf45675",
```

```
"monitoringEventReports":
        [{
        "externalId":"10001@domain.com",
        "lossOfConnectReason":7,
        "monitoringType":"LOSS_OF_CONNECTIVITY"
        }]
}
```

Listing 3.3: Loss of Connectivity notification.

A Loss of Connectivity notification indicates the *lossOfConnectReason* parameter, which identifies the reason why the loss of connectivity event is reported.

4. Upon receiving a loss of connectivity notification, the monitoring system creates a Location Reporting subscription.

```
{
"externalId":"10001@domain.com",
"notificationDestination":"http://34.14.121.58:8000/notifications",
"monitoringType":"LOCATION_REPORTING",
"maximumNumberOfReports":1,
"monitorExpireTime":"2026-04-27T13:07:19.734075Z",
"locationType":"LAST_KNOWN_LOCATION"
}
```

Listing 3.4: Location Reporting subscription creation.

Similarly to a Loss of Connectivity subscription, in addition to the common parameters, the Location Reporting subscription may have the *locationType* parameter, which identifies whether the location is for Current Location, Initial Location or Last Known Location.

5. After completing the request, the NEF returns a response informing the UE's last known location.

```
{
"externalId": "10001@domain.com",
"locationInfo": {
        "cellId": "AAAAA1004",
        },
"monitoringType": "LOCATION_REPORTING"
}
```

Listing 3.5: Location Reporting response.

The Network Application receives an immediate response since, when the *locationType* is set for a Last Known Location, the *maximumNumberOfReports* shall be set

to 1 as a one-time monitoring request. The response contains the *locationInfo* attribute, which provides the UE location, including the last known cell associated with the device.

Once monitoring notifications are received from the NEF, the backend processes the event data and stores relevant information in the database. The data is then used to provide monitoring visualizations through the Grafana dashboard.

3.4.2 User Interaction

Besides the interaction with the NEF, the monitoring service also supports interaction between the user and the system through the frontend interface. Users can provide inputs related to the configuration and management of monitoring operations, while the system returns monitoring information and statistical results. Access to the system is restricted to authenticated users, ensuring that interaction with monitoring operations and access to network information is only approachable by authorized entities.

Initially, the user must provide authentication credentials to establish a valid session within the Network Application. Following authentication, the user will be able access to the monitoring dashboard and the monitoring functionalities. Users, through the interface, can configure monitoring reports, define UEs to be monitored and manage monitoring subscriptions. Regarding monitoring reports, users may select the UEs they wish to monitor and define certain parameters as maximum number of reports, monitor expire time and maximum detection time.

After processing the provided inputs, the system delivers outputs related with monitoring operations and notifications received from the NEF. The information received from monitoring reports is presented through the frontend and Grafana visualizations, allowing users to inspect connectivity events, location-related information and other monitoring data.

3.5. Security and Privacy Considerations

The monitoring service is an intermediary between the end-user and the NEF, therefore security and privacy are inherent parts of the system design. The application handles monitoring information about the connectivity status and location of the UE, as a result, it requires controlled access and secure communication protocols.

The Network Application is only accessible to authenticated users. Only authorized users can perform monitoring tasks and view information. When the user session is validated, access to resources is granted and the risk of unwanted access to monitoring information is minimized.

According to the 3GPP standards, the AF must request an authorization token before being able to communicate with the NEF [11]. This token is stored in *config* separately from user credentials and is only used to communicate with the NEF APIs. This separation improves security by decoupling user authentication from network-exposed credentials and facilitates token management within the application.

The monitoring service follows the exposure principles defined by 3GPP, meaning the system only communicates with the NEF to access the exposed network information and services. As a result, sensitive internal network data and topology information are abstracted from the application, thus preserving the security of the network.

From a privacy standpoint, the system only stores the monitoring data required for the proposed use case. In addition, the information is accessible only to authorized users by means of authenticated access to the frontend interface and visualization tools.

Finally, the proposed architecture decouples its components to reduce the impact of possible failures or unauthorized access between different parts of the system. This separation enhances maintainability as well as the overall resilience of the system.

4. System Implementation

This chapter explains the implementation of the proposed monitoring service, focusing on how the previously defined system design and architectural components were adapted into a functional Network Application. The implementation describes how the different system components are combined to support communication with the NEF.

The proposed system was implemented as a modular web application composed of frontend, backend, database and visualization components. The combination of these elements allows the creation of monitoring reports and subscriptions, the reception of asynchronous notifications, as well as the storage and visualization of monitoring information.

This implementation chapter begins by describing the technology stack, detailing the tools used for the development of the application. The database implementation focuses on the translation from the conceptual schema to a functional relational database. Subsequently, it explores the backend implementation, detailing the development of the Network Application in order to interact with the NEF and describes processing logic associated with monitoring events. Additionally, the system interface and the Grafana dashboard are presented to describe how monitoring data is visualized and exposed to the end-user.

4.1. Technology Stack

The monitoring system was developed by combining technologies that can satisfy the requirements of the system, which include asynchronous communication, data storage and real-time data visualization. The technologies selected were chosen for modularity and scalability and to facilitate integration with the NEF.

The backend is developed using Python, the FastAPI * framework and the Uvicorn † ASGI server. FastAPI was chosen for its native support for asynchronous operations and Uvicorn for providing an efficient execution environment for asynchronous web applications. Together, these technologies allow the system to process concurrent callbacks from the NEF without interrupting the main execution flow. The communication with the NEF is done via RESTful APIs based on HTTP with JSON payloads. Furthermore, the backend is also responsible for handling the business logic, event processing and database interaction. Despite being deployed in the same FastAPI environment, the frontend and

*https://fastapi.tiangolo.com
†https://uvicorn.dev

backend logics remain separate. The frontend is built using HTML and CSS templates, rendered with Jinja2 * through the framework, which allows it to communicate with the backend services.

The system database was implemented with MySQL, a relational database management system. It was chosen because of its reliability in processing concurrent connections, which is a key requirement of the monitoring system, since both the backend and Grafana access the database simultaneously. This ensures that these components can access the database without locking it.

SQLAlchemy † is used as the Object-Relational Mapper (ORM) to serve as the bridge between FastAPI and MySQL. The database schema, instead of being defined by SQL scripts, is defined by Python models. This implementation guarantees that the system is scalable and easier to maintain.

Grafana ‡ was included as the visualization engine. The application leverages Grafana's capabilities to create a dashboard with graphical representations of event information, connectivity statistics and location data.

Finally, the system is containerized and configured using Docker and Docker Compose, allowing the monitoring service to be deployed in an isolated and reproducible environment. Docker simplifies the execution of different components and minimizes the configuration dependencies between environments. Additionally, it facilitates the integration between the backend, the database and Grafana, as each service can run as an independent container and communicate through a shared Docker network.

4.2. Database Implementation

The database implementation converts the conceptual schema, as described in Section 3.3.2, into a relational structure that is capable of supporting the Network Application. The system uses MySQL as the database management system and SQLAlchemy as the ORM layer, allowing the backend to interact with database entities through Python objects instead of raw SQL queries. SQLAlchemy is responsible for defining relational entities, mapping rows to python objects and vice-versa, managing database connections and sessions.

*https://pypi.org/project/Jinja2/
†https://www.sqlalchemy.org
‡https://grafana.com

To maintain a clear distinction between data storage, data validation and database operations, the implementation separates SQLAlchemy models, Pydantic * schemas and CRUD operations. Models specify the database tables, including column types, primary and foreign keys, and relational constraints. Schemas establish data types and validation rules as well as enforce validation at runtime to ensure data integrity. The CRUD operations consist of database functions to create, read, update and delete information within the application.

The following lists describe the main database entities implemented in the monitoring service. For each entity, the attributes and relationships is presented.

The users table stores the user authentication information to access the monitoring service:

- id (integer): unique identifier for the user;
- username (string): distinct identifier used for authentication;
- email (string): unique identifier for the user email;
- password (string): securely salted and hashed representation of the user password.

The config table stores the necessary information to interact with the NEF:

- id (integer): unique identifier for the token instance;
- token (string): token retrieved from the NEF.

The reports table represents the monitoring reports configured by users:

- id (integer): unique identifier for the report;
- title (string): name of the report;
- start_time (date time): the time for when the report started;
- end_time (date time): the time for when the report ended;
- description (string): a simple summary of the report.

*https://pydantic.dev/docs/

The subscription table represents the Loss of Connectivity subscription information for each monitored UE:

- id (integer): unique identifier for the subscription;
- nef_link (string): unique URL path for the subscription in the NEF;
- max_reports (integer): maximum number of reports for the subscription;
- expire_time (date time): time at which the subscription expires;
- detection_time (integer): maximum period of time at which the UE is considered unreachable by the network;
- ue_id (integer): links the network monitoring event to the UE being monitored;
- report_id (integer): links the subscription to the its associated report.

The user_equipments table represents the devices that can be selected for monitoring:

- id (integer): unique identifier for the user_equipments;
- external_id (string): the official 3GPP GPSI corresponding to the network device;
- name (string): a name assigned to easily represent the UE;
- supi (string): the UE's SUPI;
- ipv4_addr (string): the UE's IPv4 address;
- ipv6_addr (string): the UE's IPv6 address;
- mac_addr (string): the UE's Media Access Control (MAC) address.

The events table represents the combination of Loss of Connectivity and Location Reporting notifications:

- id (integer): unique identifier to represent the event;
- status (string): the status of the current event, which could be Pending or Completed;
- timestamp (date time): precise moment the event was processed and created in the system;

- ue_id (integer): associates the event to the UE;
- subscription_id (integer): links the event to a specific monitoring event subscription.

The loss_conn table represents the notifications received by the system related to Loss of Connectivity subscriptions:

- id (integer): unique identifier for loss of connectivity notifications;
- timestamp (date time): precise moment the notification is received by the system;
- loc_reason (integer): classifies the distinct reason for the UE's loss of connectivity event;
- subscription_id (integer): associates the notification to its subscription;
- event_id (integer): links the notification to the specific event.

The location table represents the Location Reporting response from the NEF:

- id (integer): unique identifier for location reporting responses;
- cell_id (integer): links the location information to the specific cell it occurred;
- event_id (integer): associates the location information to the specific event.

The cell table represents the retrieved cell information from the NEF:

- id (integer): unique identifier for the cell;
- cell_id (string): unique identifier for the cell in the NEF;
- name (string): a name assigned to easily identify the cell;
- description (text): a description of the cell;
- gNb_id (integer): unique identifier for the gNb;
- latitude (float): latitude coordinate received from the NEF;
- longitude (float): longitude coordinated received from the NEF;
- radius (float): radius of the cell received from the NEF.

The database schema evolution and version control is managed with the use of Alembic *, which is a migration tool for SQLAlchemy. Alembic allows the relational database structure to evolve through development, while maintaining consistency between the models and the physical database schema.

Alembic is responsible for generating migration scripts that transmit changes to the database whenever models suffer changes, such as the creation of new entities, changes to attributes or updates to relationships. This method eliminates the need for manual restructuring and it simplifies the maintenance of the database during the development of the monitoring service.

Additionally, the use of Alembic enhances consistency across development and deployment environments since it allows the database schema to be recreated and modified through version-controlled migrations. This was crucial during the development of the monitoring service, as the database structure evolved alongside the implementation.

Overall, the database implementation provides the foundation needed by the monitoring service, allowing the system to store, validate and manage monitoring information associated with its interactions with both the user and the NEF.

4.3. Backend Implementation

The backend serves as the core component of the monitoring service, acting as the intermediary between the user interface, the database and the NEF. This component is implemented with FastAPI, allowing it to follow a modular architecture organized into routers, services and utility components.

One of the responsibilities of the backend is the management of the monitoring workflow, which includes creating monitoring subscriptions in the NEF through standardized APIs, receiving asynchronous notifications from the NEF, processing monitoring events and displaying the received information for later analysis and visualization. Additionally, the backend is also responsible for handling user authentication, retrieving the authentication token from the NEF, managing UEs for monitoring, importing cell data, and realizing queries to the database.

To support the continuous interaction with the NEF, the application is deployed using Uvicorn with six worker processes, allowing multiple notifications and the subsequent monitoring requests to be handled simultaneously. Communication with the NEF is performed

*https://alembic.sqlalchemy.org/en/latest/

through a dedicated HTTP client based on the httpx * library. The client is configured with connection pooling and persistent connections to support the continuous exchange of requests, responses and notifications between both systems. This configuration improves communication efficiency and enables the backend to handle bursts of monitoring activity while interacting with the NEF.

The following sections describe the integration between the Network Application and the NEF, followed by mechanisms for processing the monitoring events received from the network.

4.3.1 Monitoring Subscription Integration

The communication between the monitoring service and the NEF is performed through the Monitoring Event API, which allows it to create monitoring subscriptions and receive asynchronous notifications associated with the UEs. This interaction follows the Subscribe-Notify model previously explained in Chapter 1.5.

Before communicating with the NEF, the application must first obtain an authentication token from it. This operation is performed during the user authentication process, where the backend validates the user session and before finalizing it, confirms that the current token is valid or retrieves a new one. This token is included in the HTTP requests sent towards the NEF, ensuring that the requests are performed through authenticated and authorized interactions.

Figure 4.1 represents the workflow associated with the creation of a monitoring report and the subsequent Loss of Connectivity subscriptions. The diagram focuses on the interactions between the user, the Network Application components and the NEF during the initial monitoring phase.

*https://www.python-httpx.org

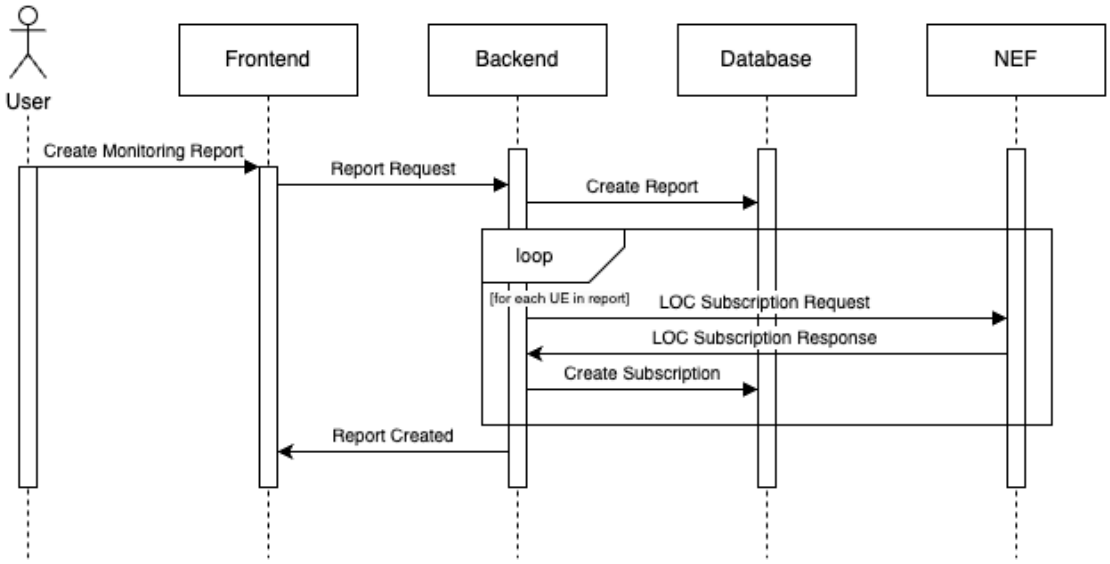


Figure 4.1: Monitoring Report Workflow.

When a user creates a monitoring report through the system interface, they must select the UEs to monitor and define the parameters under which monitoring occurs. The request is forwarded to the backend along with the monitoring configuration. The backend creates and stores the corresponding report in the system's database. This report serves as the logical container for the monitoring session and is associated with the monitoring subscriptions created.

For each selected UE, a monitoring event subscription for Loss of Connectivity is created in the NEF containing the monitoring parameters defined by the user, the monitored UE identifier and the notification destination used by the backend to receive asynchronous monitoring notifications. Each request is realized through HTTP-based requests using JSON payloads that follow the structure defined in the Monitoring Event API.

After creating the subscription request, the NEF returns an HTTP 201 Created status code indicating that the monitoring subscription was successfully created. The response contains the resource link to the individual event subscription, along with the subscription information returned by the NEF, as previously presented in Subsection 3.4.1 [11].

Finally, the backend stores the subscription information in the database and associates it with both the device being monitored as well as the corresponding monitoring report. This allows the system to correlate upcoming notifications with the correct monitoring report and monitored device.

This integration mechanism allows the Network Application to continuously receive monitoring information from NEF while abstracting its complexity from the user. The following subsection describes the notification processing workflow.

4.3.2 Notification Processing and Event Handling

After the monitoring event subscriptions are successfully created, the monitoring workflow becomes event-driven. When a monitored UE loses connectivity, the NEF sends an asynchronous notification to the configured callback endpoint. These notifications contain the monitoring information related to the subscribed UE and represent the starting point of the event processing workflow implemented by the Network Application.

Figure 4.2 illustrates the internal notification processing and event handling workflow implemented by the backend. The diagram presents the sequence of operations performed after receiving a loss of connectivity notification, including payload validation, event correlation and database persistence.

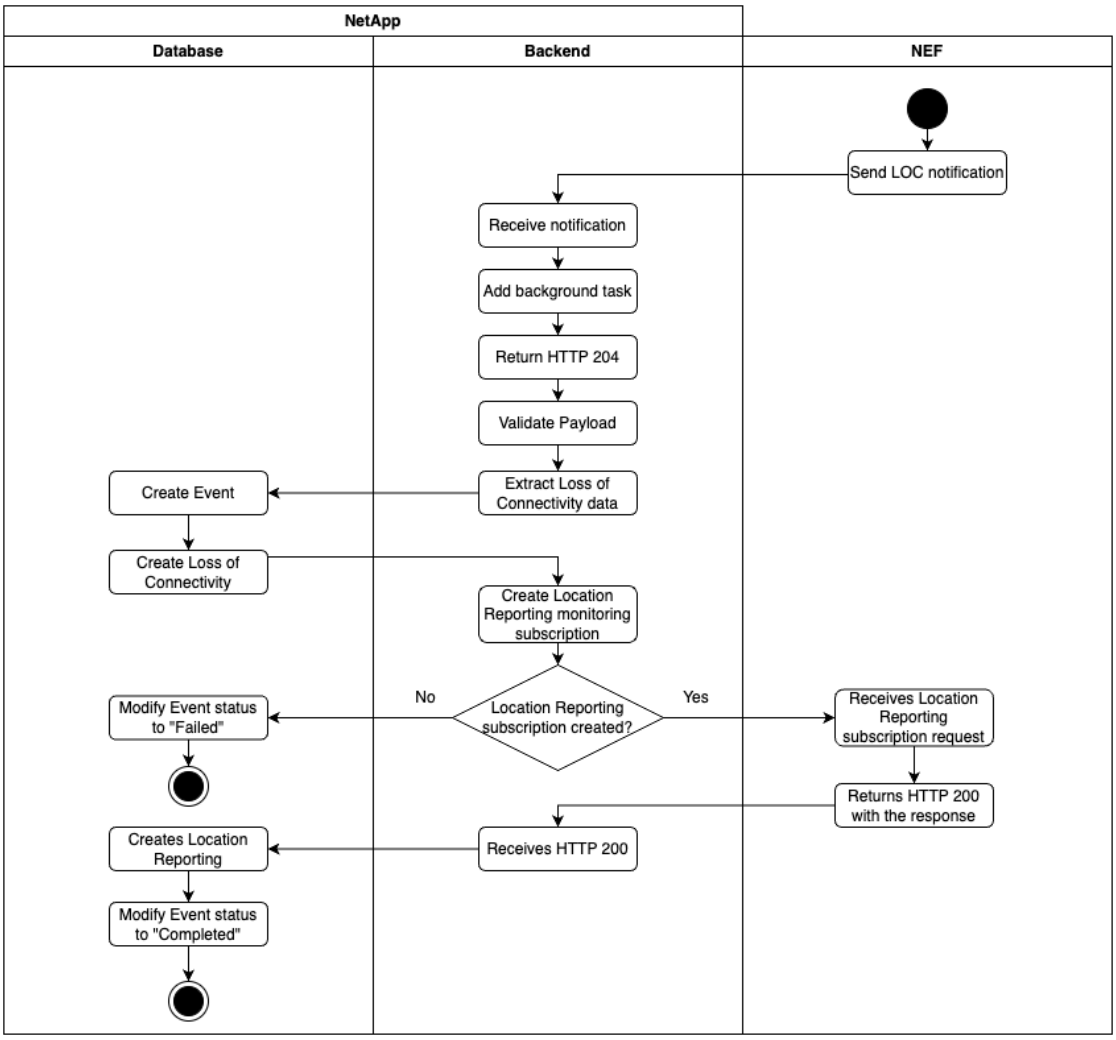


Figure 4.2: Notification Processing Workflow.

When the backend receives a Loss of Connectivity notification from the NEF, the notification is passed to a background task to process it. The backend immediately returns a HTTP 204 No Content response to the NEF, confirming that the notification was received successfully. This approach avoids blocking the reception endpoint and allows the system to continue receiving new notifications while previous events are being processed in the background.

The background task processes the receive payload and extracts the relevant information which includes the subscription reference, associated UE and loss of connectivity reason. The system then creates an event record in the database with its status set as pending and stores the loss of connectivity notification associating it to its originating subscription.

Since the objective of the monitoring service is not only to detect connectivity loss but also to identify where the event occurred, the backend proceeds to create a new monitoring

event subscription regarding Location Reporting. This subscription retrieves the UE's last known location and allows the initial event to be associated to a specific cell or geographic area.

If the subscription is successfully created, the NEF returns an immediate response since Location Reporting subscriptions regarding last known location are limited to one report. The backend stores the location information in the database and updates the event status from pending to completed. In case the Location Reporting subscription fails to be created in the NEF, the event remains registered but its status is set to failed, ensuring that the loss of connectivity event is not discarded when location information is not available.

Through this workflow, the system transforms a raw notification from the NEF into structured data. The resulting data provides the foundation for subsequent analysis and visualization of connectivity issues.

4.4. System Interface

The interface was designed to simplify access to the monitoring capabilities exposed by the NEF while abstracting its complexity. It provides the point of interaction between the user and the monitoring service, allowing users to manage monitored devices, create monitoring reports and inspect events generated by the monitoring process.

The UE management interface, shown in 4.3, provides an overview of the UEs available for monitoring and their associated information. Through this interface, the user can inspect the devices that are registered in the system's database and manage them.

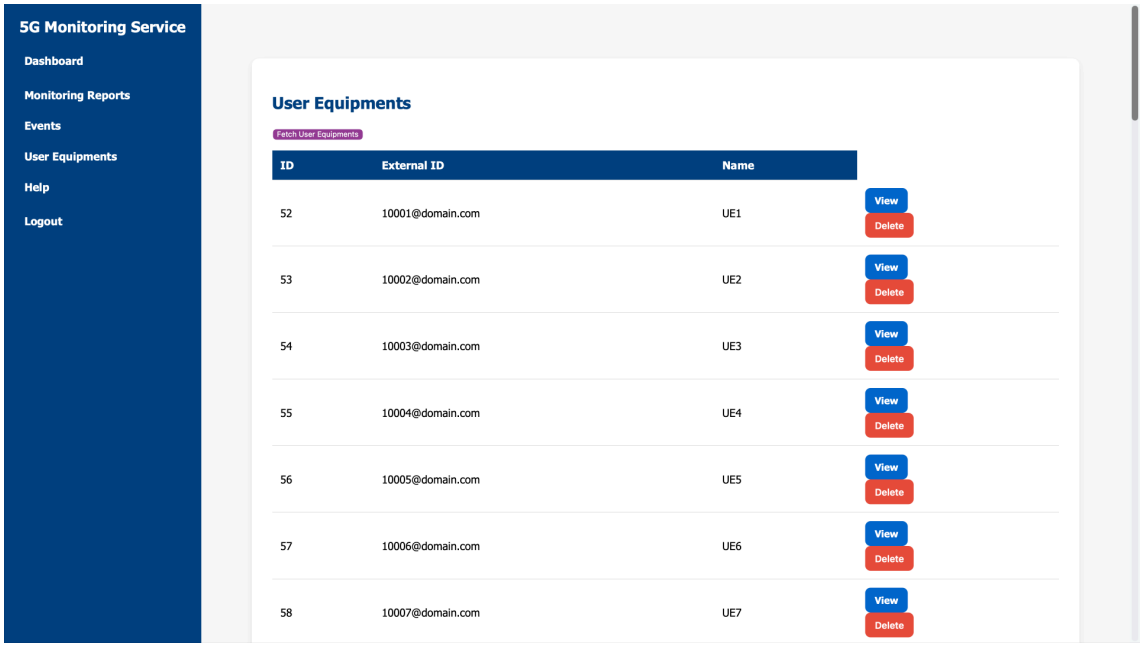


Figure 4.3: Manage UEs Interface.

The monitoring workflow is initiated through the report creation interface presented in Figure 4.4. Users can select one or more UEs to monitor and configure the monitoring parameters associated with the Loss of Connectivity subscription on this page. They can also indicate a name and description for the monitoring report. Once a report is created, the backend automatically creates the corresponding monitoring subscriptions in the NEF and associates them with the newly created monitoring report.

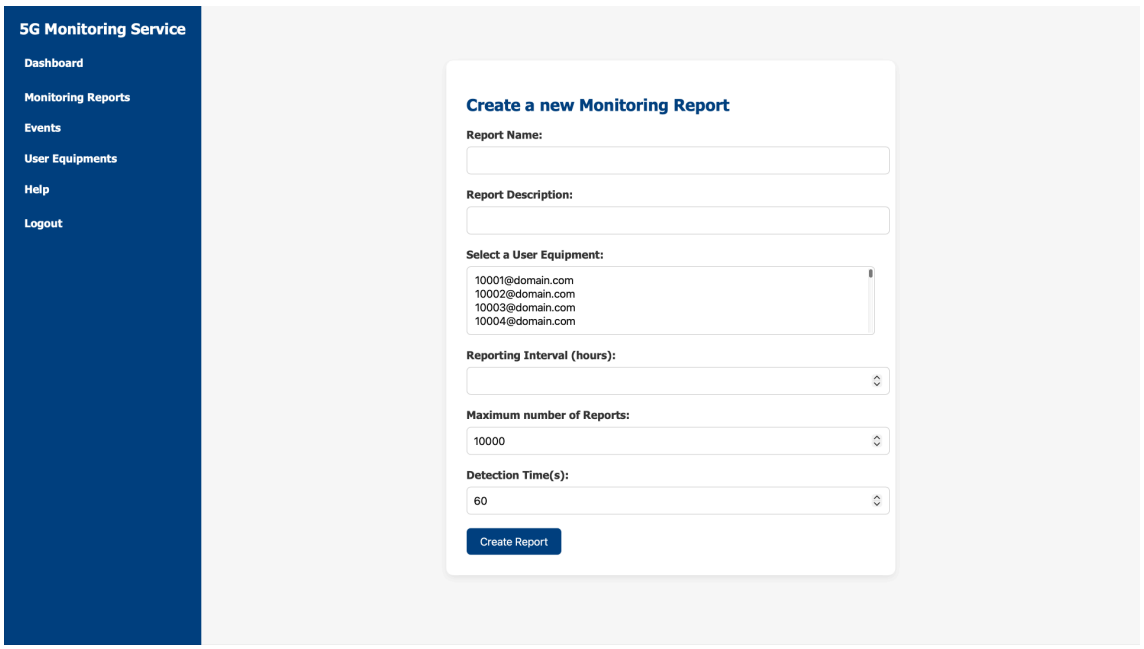


Figure 4.4: Monitoring Report Creation Interface.

Existing monitoring reports can be accessed through the reports interface shown in Figure 4.5. This page offers an overview of the monitoring reports created by the user, enabling the identification and selection of reports for further inspection.

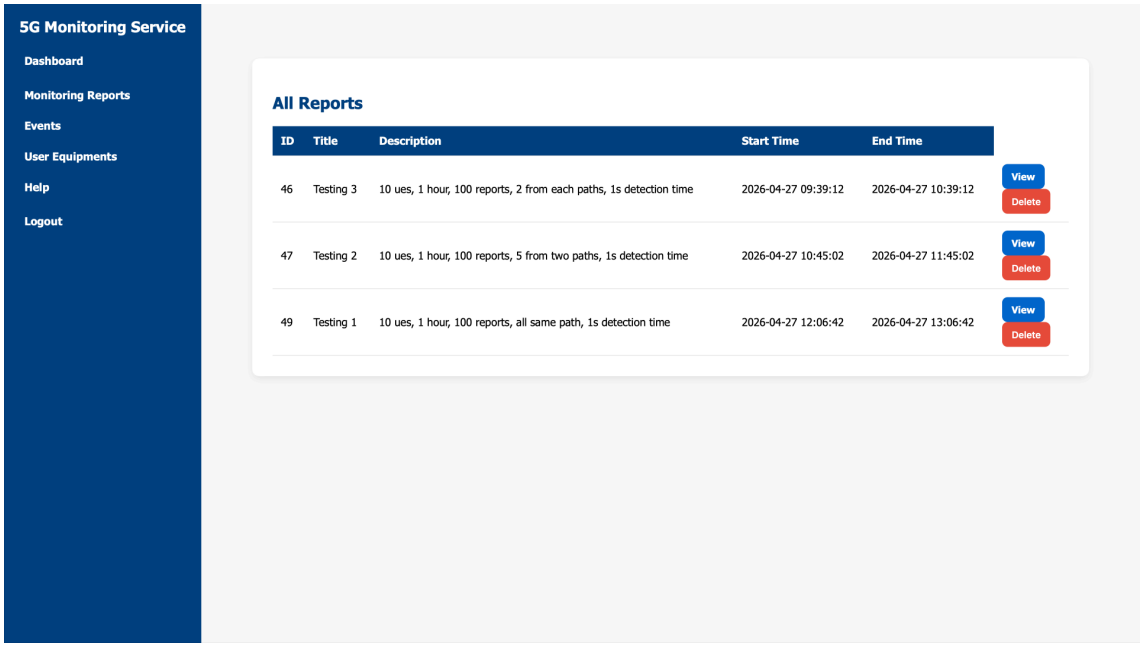


Figure 4.5: Monitoring Reports Interface.

Selecting a report opens the report details view illustrated in Figure 4.6, where information related to the report is presented. This information allows users to review the details and duration of a monitoring session.

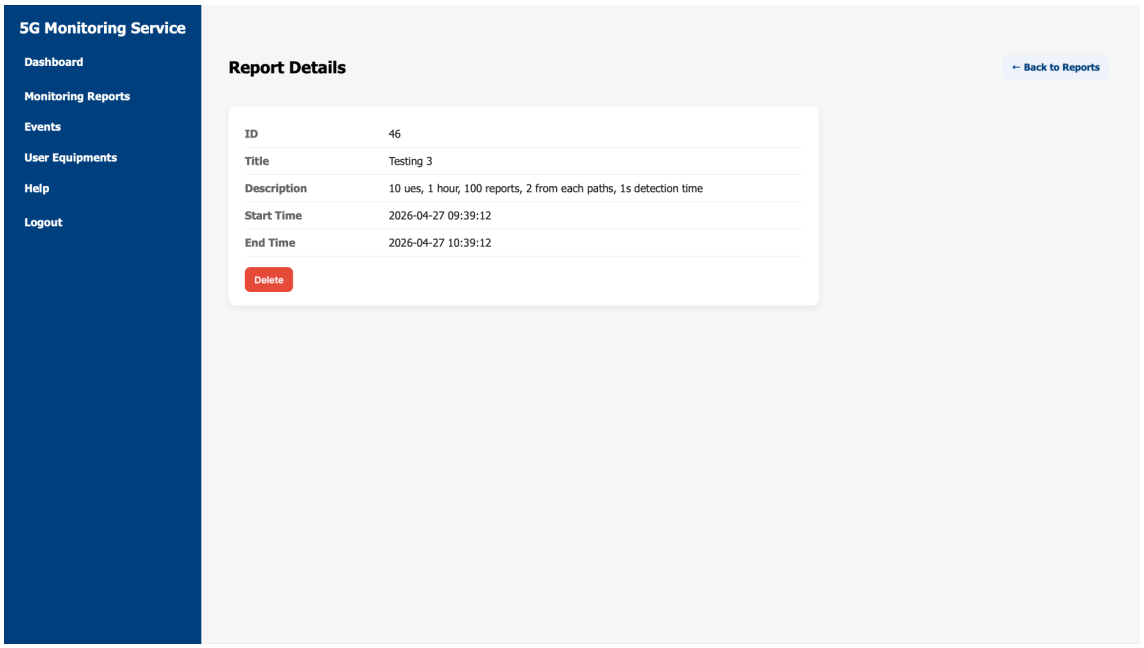


Figure 4.6: Monitoring Report Detail.

The events interface, presented in Figure 4.7, provides access to the monitoring events collected from the NEF. This view allows users to review loss of connectivity occurrences associated with the monitored UEs and obtain an overview of the monitoring activity received by the system.

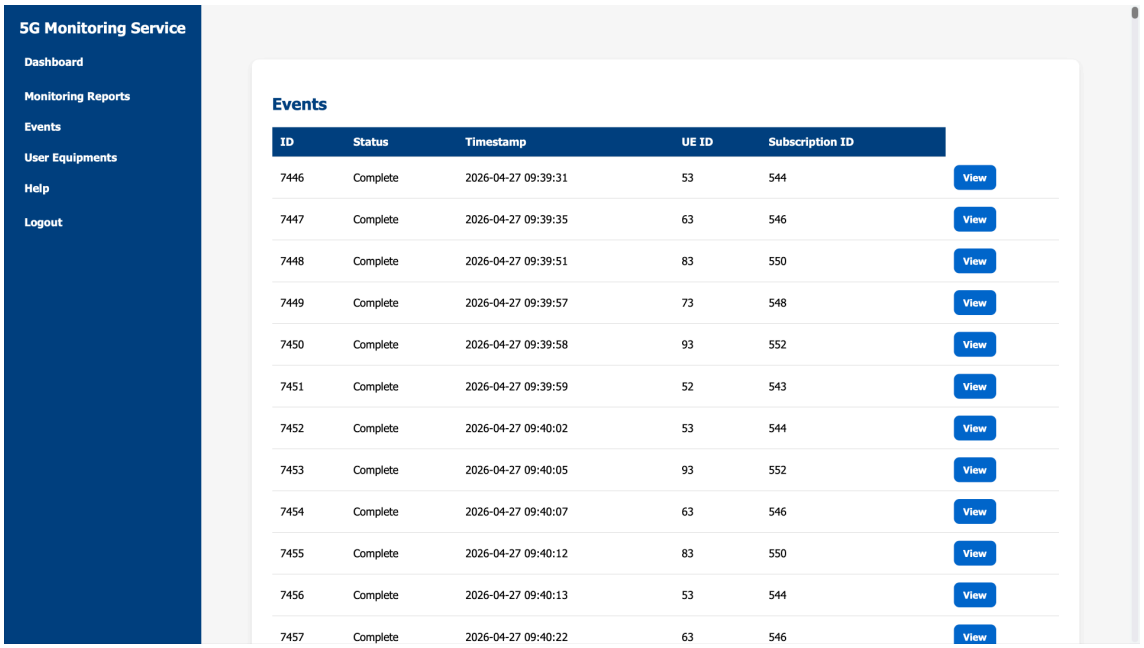


Figure 4.7: Events Interface.

Additional information regarding a specific event can be accessed through the detailed event view shown in Figure 4.8, where information such as event status, timestamps, connectivity details and location-related information are presented.

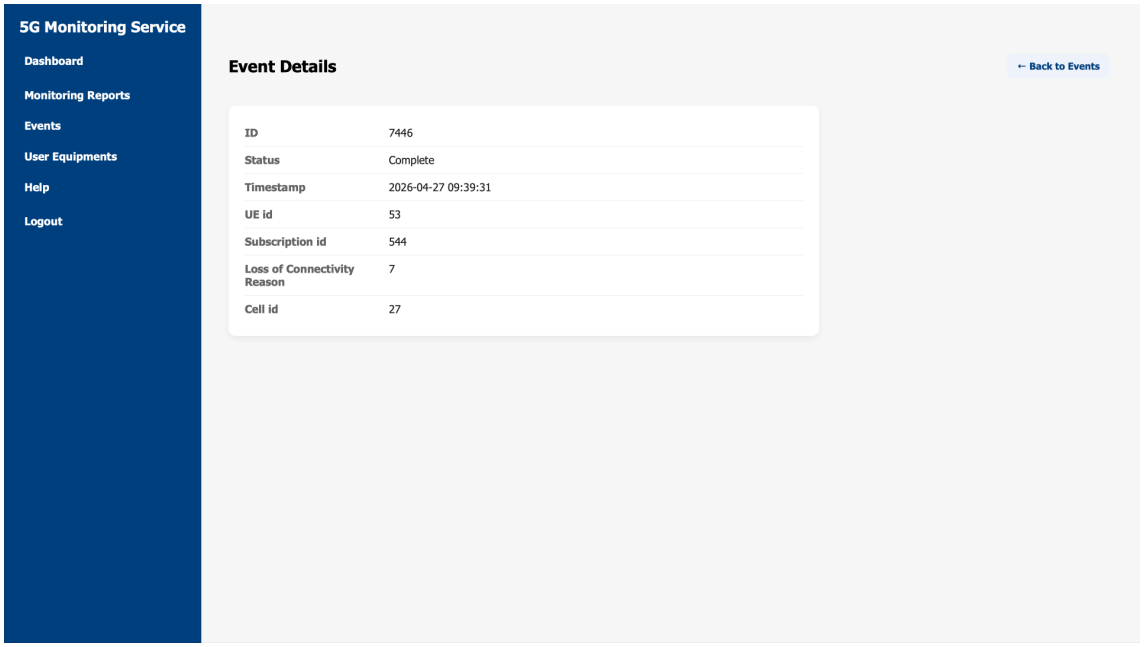


Figure 4.8: Event Detail Interface.

Figure 4.9 displays the notifications interface, which provides a chronological view of the notifications received and processed by the backend. This page allows users to inspect the monitoring activity related to the subscribed UEs and verify the notifications exchanged between the Network Application and the NEF throughout the monitoring workflow.

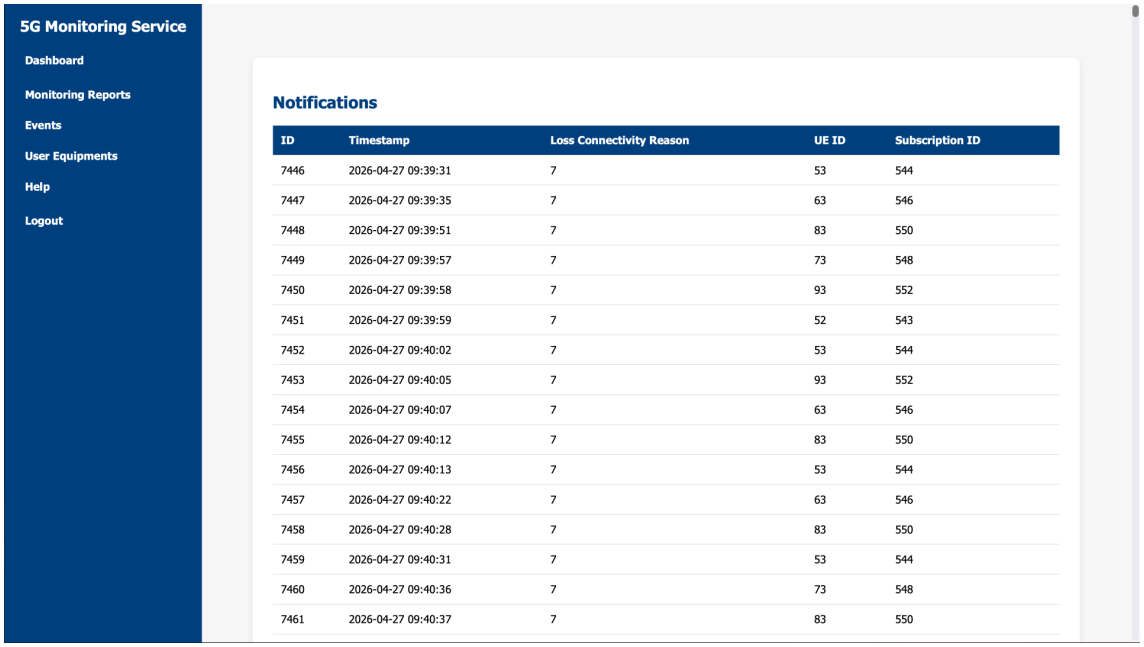


Figure 4.9: Notifications Interface.

Overall, the system interface provides a simple environment for managing monitoring sessions, allowing users to configure monitoring operations and inspect the resulting information through the accessible and user-friendly web application. While the system interface provides the functionalities required to manage the monitoring activities and inspect collected information, the visualization and analysis of monitoring data are performed through dedicated Grafana dashboards, described in the following section.

4.5. Grafana Dashboard

The visualization layer is implemented using Grafana, which serves as the platform for visualization and analysis of the monitoring information received by the Network Application. Grafana is responsible for retrieving monitoring data from the MySQL database and presenting it over interactive panels. Each panel is built from SQL queries over the database. Some panels use direct queries over a single table, while others join multiple tables to correlate events with UEs, reports, loss of connectivity information, location information and cells.

The dashboard was designed to give users an overview of the monitoring data obtained from the loss of connectivity and location reporting subscriptions. By combining statistical, temporal and geographical graphic representations, users can examine event dispersion and identify recurring issues.

Figure 4.10 presents the dashboard for the monitoring service. The dashboard is composed of seven panels, each targeting a different perspective of the collected monitoring data, including a table of the latest events, a timeline of event occurrences, a report summary table, the number of events per UE, a map portraying the distribution of events across cells, the distribution of loss of connectivity reasons and the number of events associated with each cell.

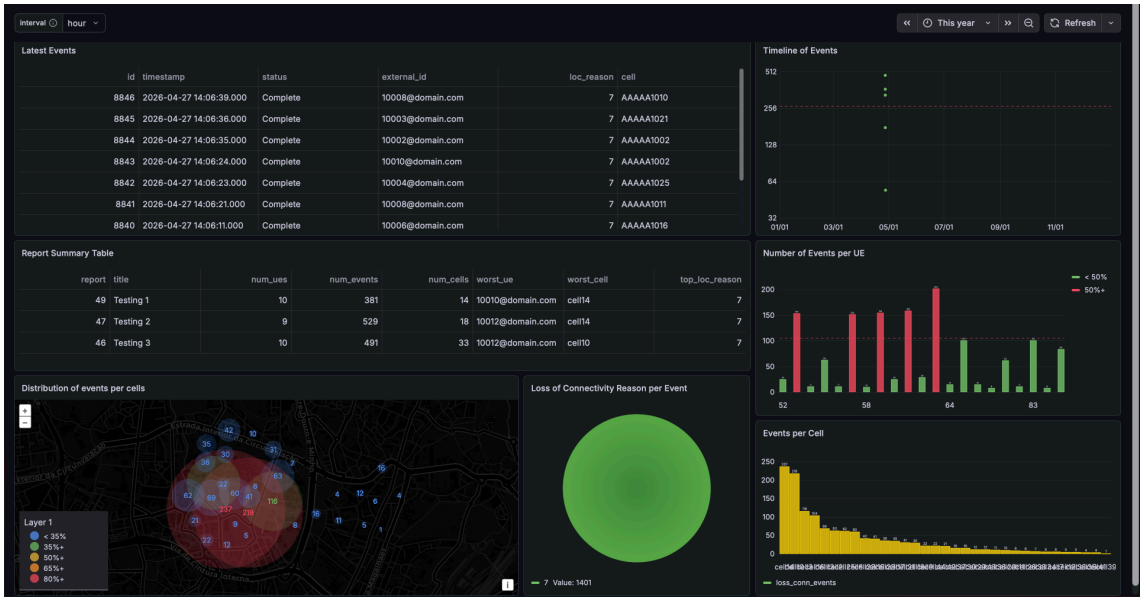


Figure 4.10: Grafana Dashboard.

The latest events table displays the ten most recent monitoring events received by the Network Application. The table combines data from events, user equipments, loss of connectivity data, location information and cells to present the most recent monitoring events. The panel provides a real-time summary of monitoring activity by displaying details about each event, including the timestamp, event status, monitored UE, loss of connectivity reason and associated cell.

The timeline of events panel shows the occurrence of the events in a chronological order by retrieving data from the events table. This allows users to identify recurring behaviors and periods with increased event frequency, facilitating the analysis of temporal trends.

To provide a high-level summary of monitoring activity, the dashboard includes a report summary table. This panel correlates information from various tables to generate statistics for each monitoring report. In addition to the number of monitored UEs, events and affected cells, the table identifies the UE with the highest number of events, the most affected cell, and the most frequent loss of connectivity reason observed during the monitoring report.

The number of events per UE graph uses information from the events table to compare the number of events that occurred regarding a specific device. This visualization helps identify devices that experience repetitive connectivity issues and may require further analysis.

The dashboard also displays the distribution of events over geographical cells, making the analysis of spatial data possible. The graph merges data from the cells, location and events tables to show the geographical dispersion of events. By associating these tables, users are able to identify specific areas with a higher density of connectivity issues.

The pie chart illustrates the distribution of reasons for of loss of connectivity events. This representation offers insight into the most common causes of loss of connectivity events and supports the identification of dominant failure patterns.

The events per cell bar chart complements the geographical distribution of events. This graph allows users to easily identify cells with a higher number of connectivity occurrences. The geographical distribution provides spatial context, while the bar chart facilitates direct comparison between cells and supports the identification of potentially problematic network areas.

Together, these visualizations turn the monitoring data collected from the NEF into valuable insights. The dashboard assists in the investigation of recurring connectivity issues and helps users to identify potential coverage gaps, problematic cells and abnormal network behavior.

5. NEF Emulator

The development and validation of the Network Application required a network environment capable of exposing monitoring capabilities through standardized 3GPP interfaces. In a production environment, these capabilities would be provided by a 5G Core (5GC) network. Since access to a production 5G Core (5GC) was not available at the time of development, an alternative solution was required to support the implementation and evaluation of the monitoring service.

To address this limitation, this thesis utilized the NEF Emulator developed by Aveiro Telecommunications and Networking Group (ATNoG) [22]. The emulator reproduces the behavior of the NEF and exposes the APIs required by the monitoring service. This approach provided a controlled environment that allowed the creation of monitoring subscriptions and the exchange of notifications required to test different monitoring scenarios and validate the complete monitoring workflow of the application without requiring access to a physical telecommunications network.

In addition to serving as the validation environment for the Network Application, the emulator was extended during the course of this work to support additional functionalities required by the monitoring service. The following sections detail the contributions made to the NEF Emulator and explain how the Network Application interacts with the NEF Emulator, focusing on the functionalities added to facilitate it, such as UE management and the retrieval of cell information.

5.1. ATNoG NEF Emulator

The ATNoG NEF Emulator is an open-source project developed by the Aveiro Telecommunications and Networking Group (ATNoG) to emulate the behavior of the 5G NEF [22]. It allows external applications to interact with the NEF via standardized APIs and provides an environment to test and validate Network Applications without requiring a production 5GC.

Figure 5.1 shows a high-level overview of the ATNoG NEF Emulator architecture. The emulator is organized around the backend, which is responsible for implementing the application logic, exposing the available APIs and interacting with the databases, while the web interface provides a way to configure and control the emulated environment.

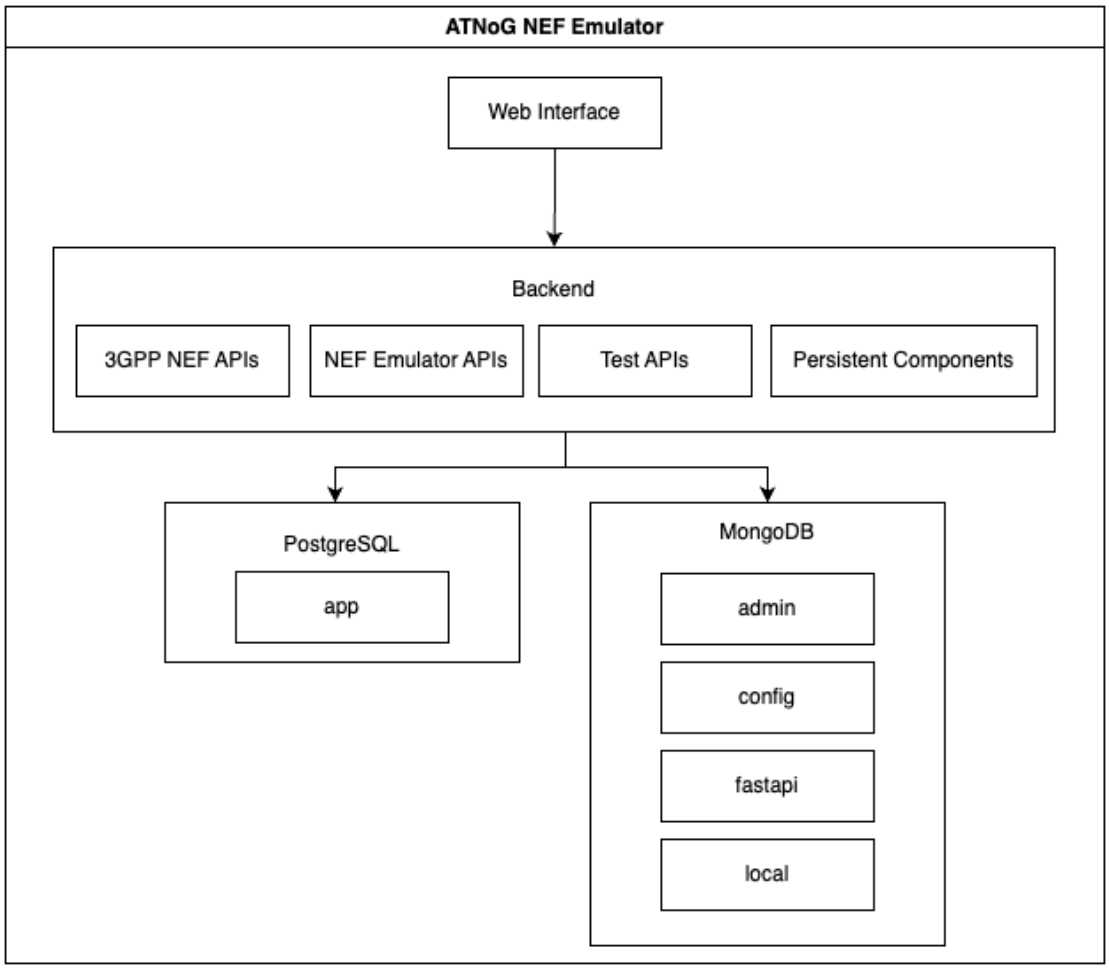


Figure 5.1: High-level overview of the ATNoG NEF Emulator.

The emulator follows a modular architecture that relies on the backend to implement the application logic and expose its interfaces. Each API is responsible for a different aspect of the emulator's functionality:

- NEF APIs: a subset of northbound APIs defined by 3GPP for the NEF, allowing external applications to interact with the emulated network capabilities. For the purpose of this dissertation, only the Monitoring Event API is considered;
- NEF Emulator APIs: provide management and configuration functionalities for the emulator, including user authentication, UE management, cells, gNbs, movement paths and UE movement;
- Test APIs: provide additional information about UE behavior during movement.

The backend interacts with two database management system.s, PostgreSQL and MongoDB, that support the operation of the emulator. In PostgreSQL, the app database stores

monitoring event subscriptions and persistent entities that are required to configure the emulated environment, such as users, cells, UEs and paths. MongoDB contains the standard system databases [23], in particular admin, config and local, and the fastapi database, which stores the events generated during monitoring subscriptions. Figure 5.2 presents a simplified view of the PostgreSQL database schema.

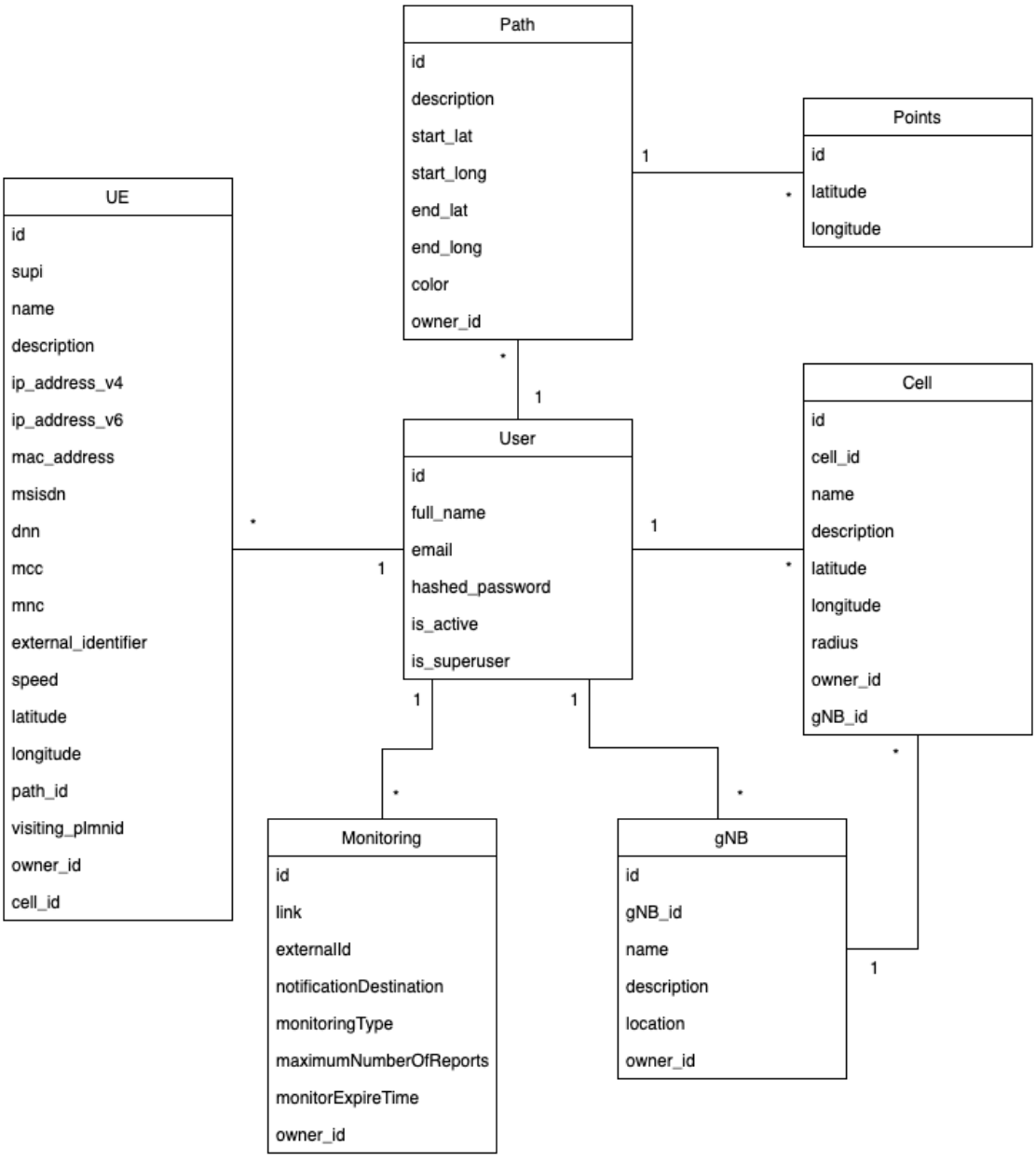


Figure 5.2: PostgreSQL database schema of the ATNoG NEF Emulator.

In addition, the emulator provides a web interface that simplifies the configuration and execution of testing scenarios. Through this interface, users can create and manage

network elements such as gNbs, cells, UEs and movement paths, as well as initiate UE movement. Figures 5.3 and 5.4 present the main interface of the ATNoG NEF Emulator.

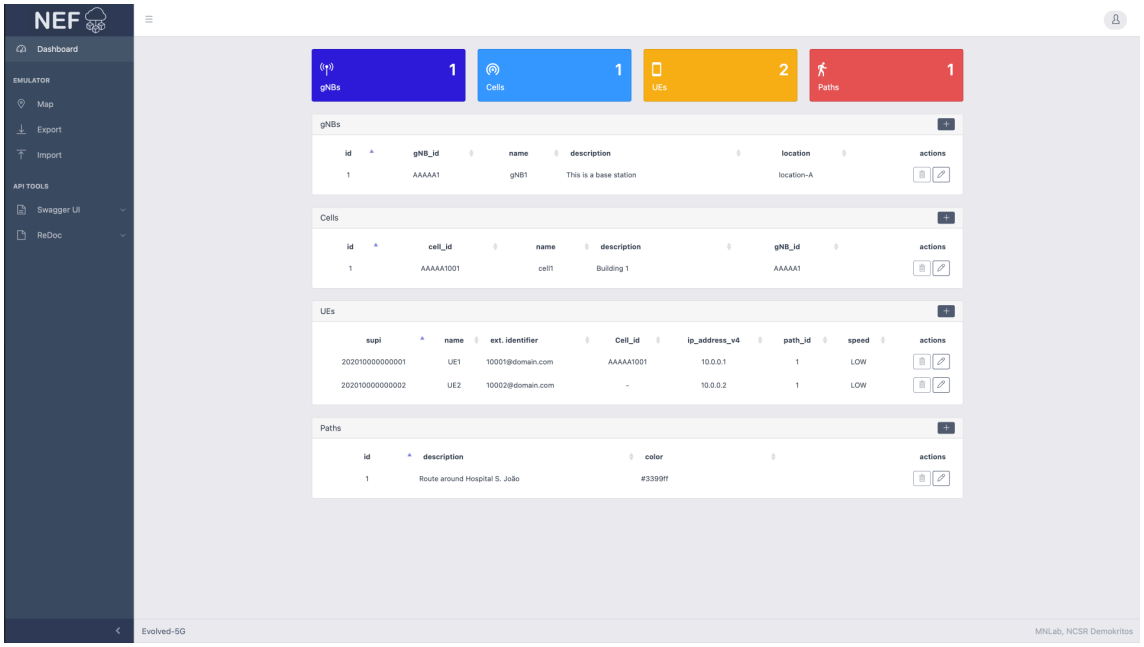


Figure 5.3: ATNoG NEF Emulator management interface.

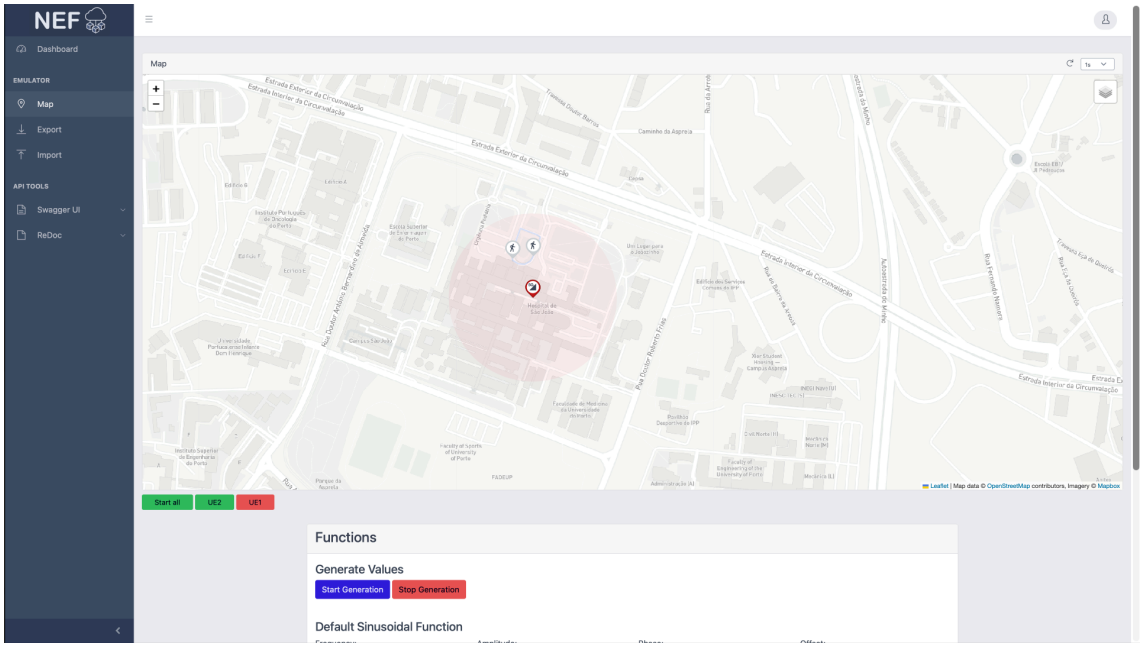


Figure 5.4: ATNoG NEF Emulator movement control interface.

Although the emulator had already implemented the Monitoring Event API, it did not provide all the functionalities required by the monitoring service. Therefore, the emulator was extended during this work, as described in the following section.

5.2. Contributions to the NEF Emulator

To support the validation of the Network Application, it was necessary to contribute directly to the source code of the open-source ATNoG NEF Emulator. Although, the NEF Emulator lacked the *locationType* attribute in Location Reporting monitoring subscriptions, which limited the monitoring service since it required the UE's last known location. The primary objective of this contribution was to implement all four standardized parameters for the *locationType* attribute.

Before implementing these features, an issue with the emulator's deployment was fixed. The build of the project consistently crashed because the Poetry package manager failed to install the dependencies correctly. The Docker configuration was modified to ensure that the dependency was correctly installed, allowing the emulator to compile properly.

The implementation was developed by extending mechanisms already available within the emulator. However, since the emulator does not maintain tracking area information and does not implement the accuracy parameter commonly used in standardized location reporting subscriptions, the location types were implemented exclusively at the cell level. As a result, location retrieval is based on the serving cell associated with a UE rather than on more precise geographical or tracking-area information.

The four supported location types are described as follows:

- CURRENT_LOCATION: Returns the current serving cell associated with the UE [11].
- LAST_KNOWN_LOCATION: Returns the last known serving cell associated with the UE [11].
- CURRENT_OR_LAST_KNOWN_LOCATION: Attempts to return the current serving cell associated with the UE, if it is not available, then it returns its last known serving cell [11].
- INITIAL_LOCATION: The 3GPP specifications define this location type as a request for the initial location of the UE [11]. However, the standard does not detail how it should be maintained internally by the network. Since the emulator tracks UE mobility at the cell level, the initial location was interpreted as the first serving cell associated with the UE when it is instantiated in the emulator. When a UE is created and assigned a movement path, it is placed at a random position along that

path, therefore the corresponding serving cell is stored as the UE's initial location. In case the UE is placed in a specific location not covered by a cell, the UE initial location is the first cell it connects itself to.

To support the implementation of the new location types, slight modifications were made to the PostgreSQL database schema, the UE data model and the corresponding Pydantic schema. Since the emulator previously only stored information regarding the UE's current serving cell as *cell_id*, two additional attributes, *initial_cell_id* and *last_known_cell_id*, were added to both the UE model and schema. These parameters allow the emulator to store additional cell information required by the initial and last known location types, ensuring that the appropriate location can be retrieved when requested in a Location Reporting subscription. Figure 5.5 shows the modifications introduced to the UE entity highlighted in red.

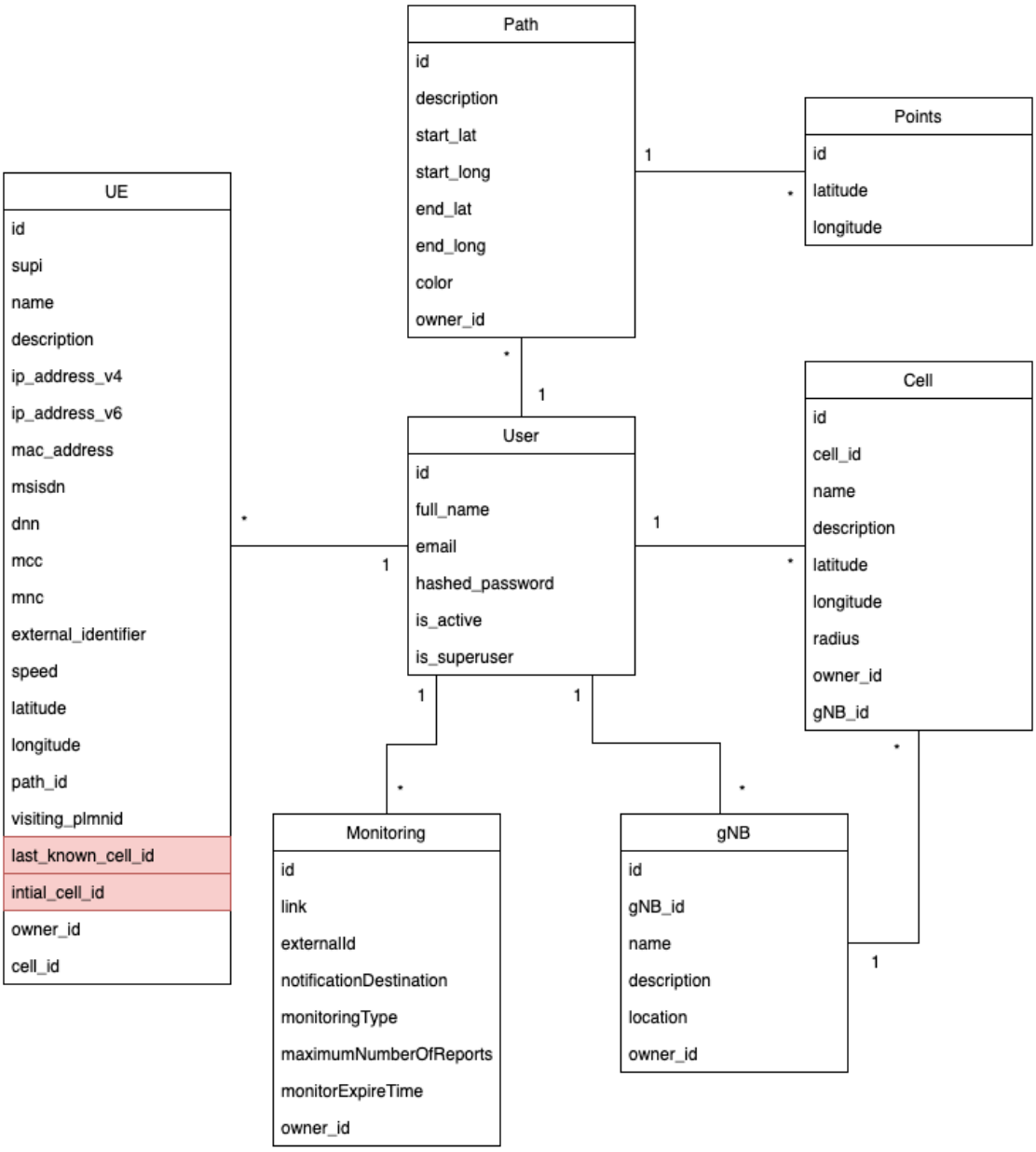


Figure 5.5: PostgreSQL database schema highlighting in red the additions to the UE entity.

Additional modifications were required in the UE creation to support the approach for the INITIAL_LOCATION. When a UE is instantiated and assigned a new path, the emulator places it at a random position along that path. If the selected position is covered by a serving cell, the identifier of that cell is stored in the *initial_cell_id* attribute and remains unchanged. However, it is possible for a UE to be instantiated in an area without cell coverage. The identifier of the first serving cell established during this initial attachment is then stored as the UE's initial location. This approach ensures that all UEs maintain a valid initial location while remaining consistent with the mobility model implemented by

the emulator.

The behavior of the implemented location types is directly influenced by UE mobility. As a UE moves through the network and transitions between serving cells, the values stored in the *cell_id*, *last_known_cell_id* and *initial_cell_id* attributes are updated accordingly. The following figures illustrate how these attributes change during different mobility scenarios.

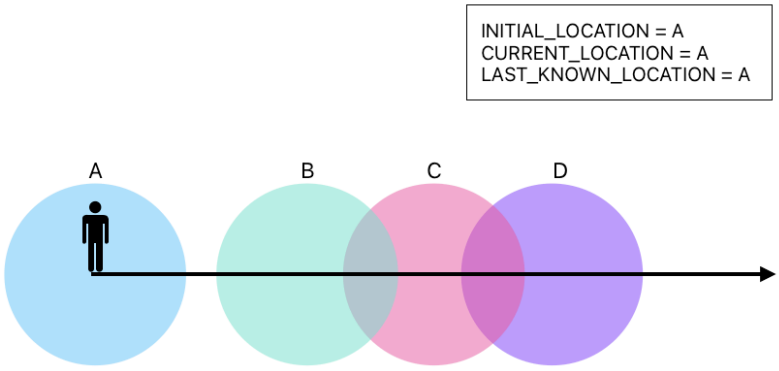


Figure 5.6: Initial state of a UE instantiated within the coverage of Cell A.

Figure 5.6 illustrates the initial state of a UE instantiated within the coverage of cell A. Since this is the first serving cell associated with the UE, the values of the current location, last known location and initial location are all set to cell A.

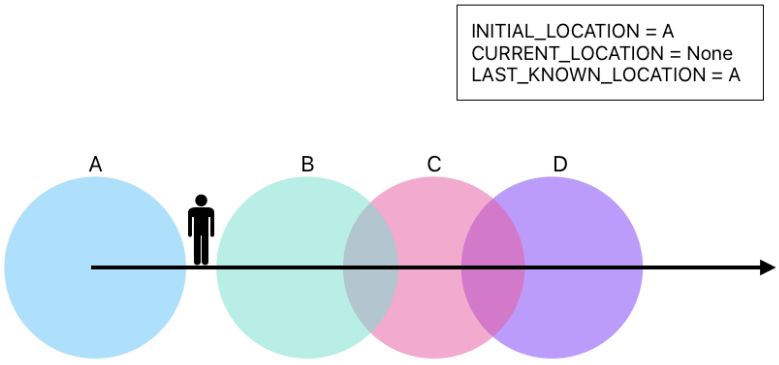


Figure 5.7: Location state after UE leaves cell A and loses coverage.

Figure 5.7 illustrates the UE leaving the coverage area of cell A. As the UE is no longer connected to a serving cell, the current location is set to none. However, cell A remains stored as both the last known location and the initial location.

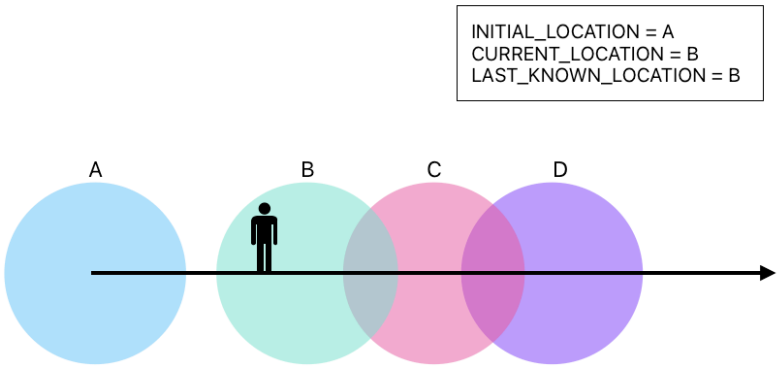


Figure 5.8: Location state after the UE reconnects to the network through cell B.

Figure 5.8 illustrates the UE after connecting to cell B. Both the current location and last known location are updated to cell B. The initial location remains unchanged.

Figures 5.6 to 5.8 depict the behavior implemented for the UE location attributes. The initial location is assigned when the UE first connects to a serving cell and remains constant for its lifetime. The current and last known locations are updated based on UE mobility, allowing the emulator to support the different *locationType* values specified for Location Reporting subscriptions.

The implementation also needed modifications regarding the processing of Location Reporting subscriptions. When creating a subscription, an application may explicitly indicate a value for the *locationType* attribute. The emulator then retrieves and reports the location information appropriate to the requested type, following the logic described previously. In the absence of the *locationType* attribute in the subscription request, the emulator returns the UE's current location. This approach is consistent with the current implementation of the Location Reporting service and guarantees compatibility with subscriptions that do not specify a location type. Figure 5.9 illustrates the processing logic implemented to support different *locationTypes*.

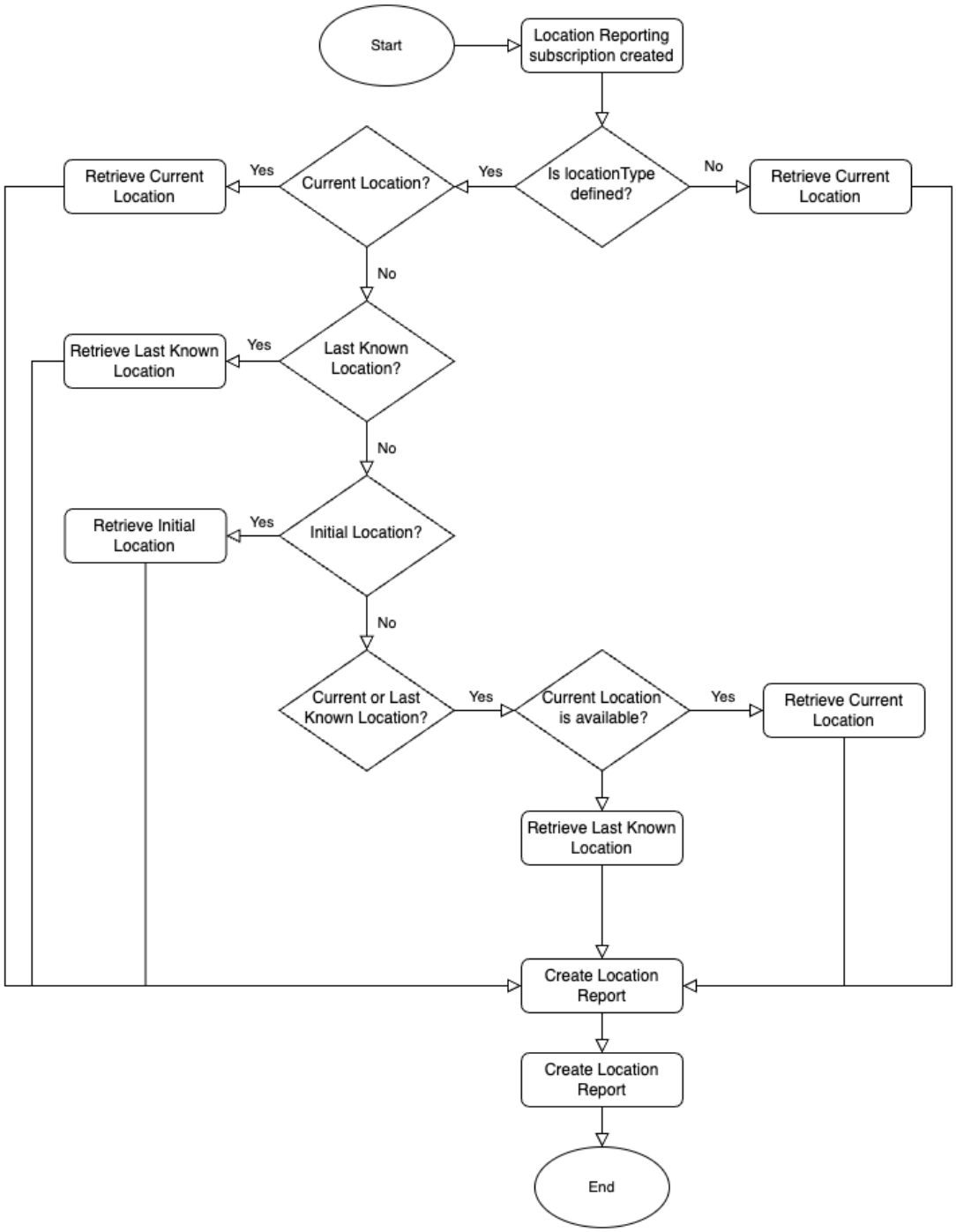


Figure 5.9: Flowchart of the processing workflow that supports multiple *locationTypes* values for Location Reporting subscriptions.

The implementation follows the constraints established by the 3GPP specifications regarding Location Reporting subscriptions with a *locationType* set to LAST_KNOWN_LOCATION. According to the standard, last known location data is exclusively valid to one-time location requests and cannot be used in continuous location

reporting subscriptions [2]. Therefore, the emulator validates the subscription parameters and ensures that the *maximumNumberOfReports* parameter is set to one when the *locationType* is set to LAST_KNOWN_LOCATION, otherwise the subscription is rejected.

The modifications described in this section are summarized in Figure 5.10 and Table 5.1. Figure 5.10 identifies the modified components within the emulator architecture and Table 5.1 summarizes the changes to each component.

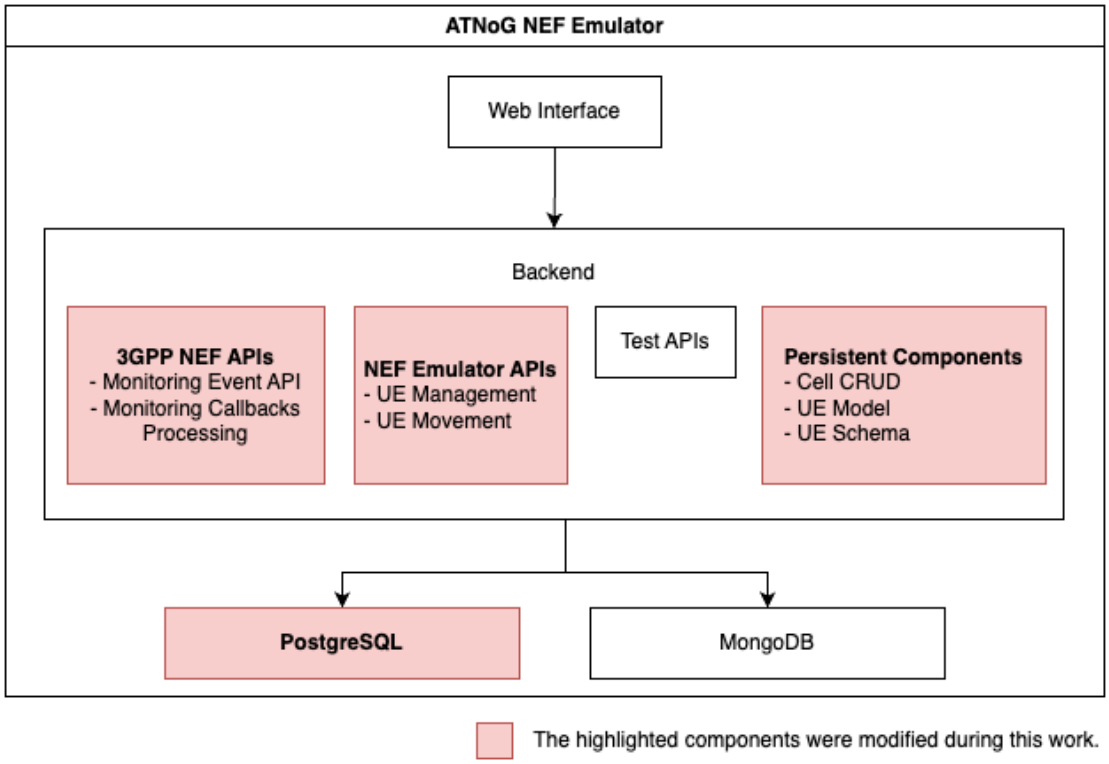


Figure 5.10: ATNoG NEF Emulator architecture highlighting the components altered during this work.

Component	Description
Monitoring Event API	Extended the Monitoring Event API to accept different location types. Limited the maximum number of reports to one for Location Reporting subscriptions regarding the LAST_KNOWN_LOCATION *locationType*.
UE Management API	Added a clause to ensure that the UE's initial location is set once the UE is created and assigned a path.
UE Movement	Altered the UE movement logic to support the initial and last known location.
Cell CRUD	Created a function to retrieve the cell_id by the cell's id.
UE Model	Added initial_cell_id and last_known_cell_id to the UE model.
UE Schema	Updated the Pydantic schema to support the changes made to the UE model.
Monitoring Callbacks	Implemented the processing logic for the different location types implemented.
Dockerfile Backend Configuration	Updated the backend configuration to fix a deployment error caused by Poetry.

Table 5.1: Summary of the components altered and extended in the ATNoG NEF Emulator.

In short, the ATNoG NEF Emulator was extended to support the *locationType* attribute in Location Reporting monitoring subscriptions. Although the implementation is limited due to the emulator's lack of tracking area, it provides an approximation of location reporting behavior. As a result, the enhanced emulator is capable of supporting the location requirements of the developed Network Application, enabling realistic validation of the implemented monitoring service workflows.

5.3. Interaction with the Network Application

Some additional functionalities were incorporated into the Network Application to facilitate integration with the ATNoG NEF Emulator and to help validate the defined use case. These features simplify the management of the emulated environment and enable the Network Application to interact with the emulator more efficiently.

Figure 5.11 presents the updated architecture of the proposed solution. In addition to interacting with the NEF API, in particular the Monitoring Event API, the Network Application also interacts with the NEF Emulator API to retrieve information regarding UEs and cells configured in the emulator.

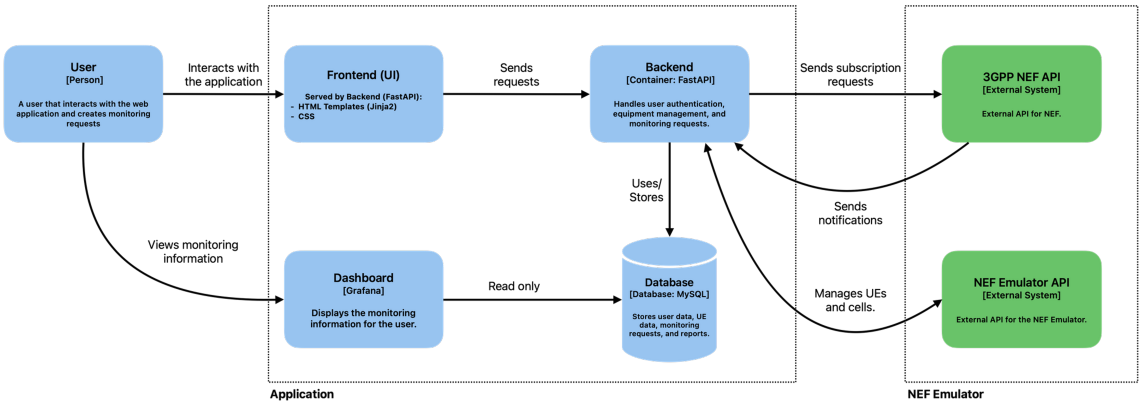


Figure 5.11: Updated architecture of the Network Application including the interaction with the ATNoG NEF Emulator API.

It should be noted that these functionalities were implemented to support the emulated environment and would not be required in a production deployment. In a real mobile network, UEs correspond to physical devices managed by the operator's infrastructure and therefore do not need to be created or removed by an application through the NEF. Likewise, information regarding network cells is generally obtained through network management systems rather than directly through an API. However, within the context of the ATNoG NEF Emulator, exposing UE and cell management capabilities through the emulator API simplifies the configuration of the test environment and provides the Network Application with the information required to validate the monitoring service. Therefore, the use case previously presented in Section 3.1 was updated to include these additional interactions, as shown in Figure 5.12.

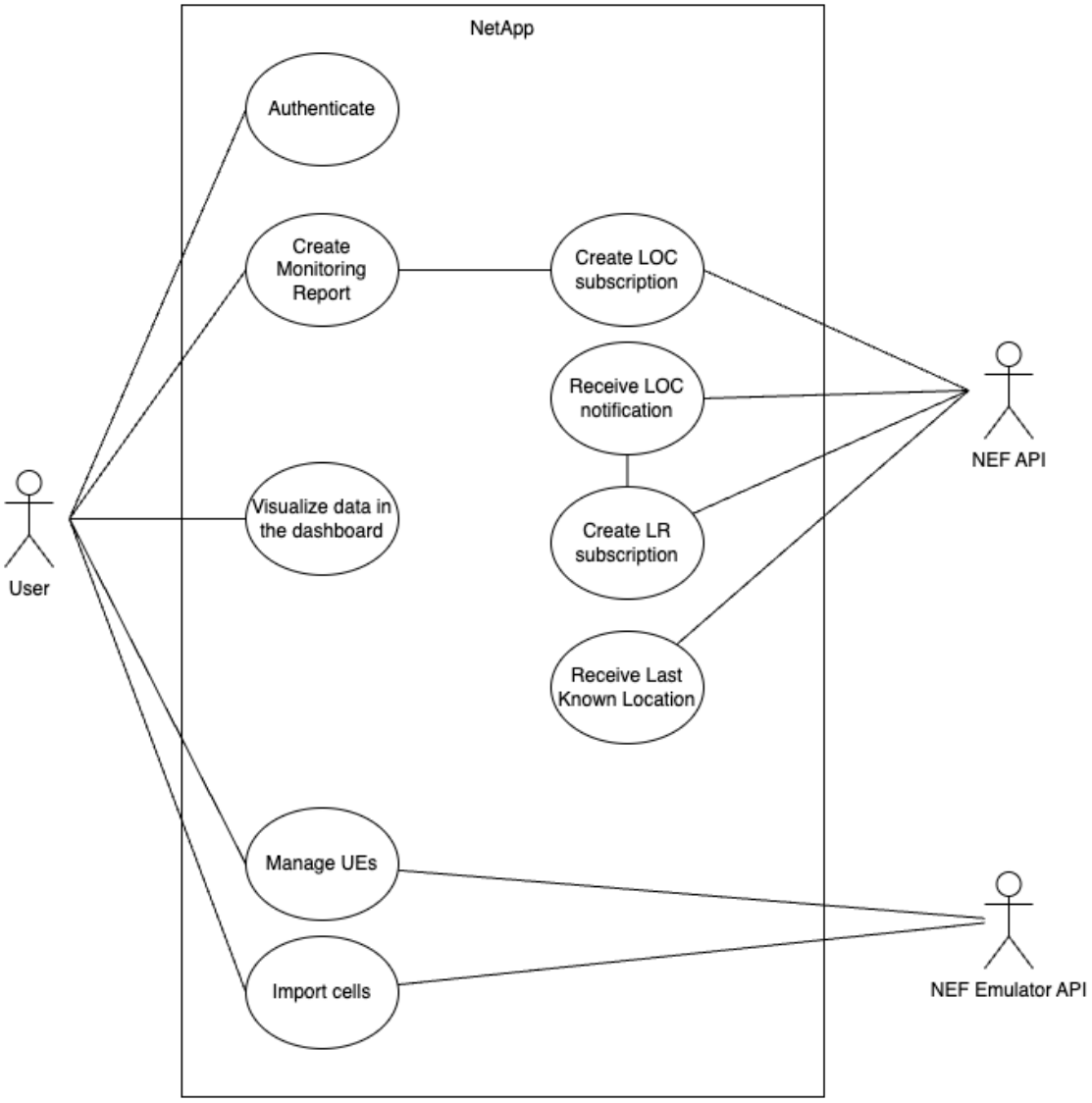


Figure 5.12: Updated use case diagram including the interactions with the NEF Emulator API.

The first addition concerns UE creation, deletion and retrieval through the Emulator API. These functionalities were implemented to manage the set of UEs participating in the application tests. To create a UE, the Network Application must provide some parameters required by the emulator, such as a name, an external ID, an IPv4 address, an IPv6 address, a MAC address and a SUPI. Similarly, to remove a UE it is necessary to provide its SUPI, which uniquely identifies the UE within the emulator. The UE retrieval functionality allows the Network Application to obtain information regarding UEs that have already been instantiated in the emulator. These functionalities simplify the preparation and management of the test environment by allowing UEs to be managed directly from the Network Application. However, the control of UE mobility, including the start and stop

of UE movement, remains managed via the NEF Emulator.

The Network Application was also extended with the ability to retrieve cell information from the NEF Emulator. Through the corresponding API endpoint, the application can obtain details regarding cells that have been configured in the emulated environment. Access to this information allows the Network Application to correlate monitoring notifications with specific cells and provide additional context when processing Location Reporting responses.

Together, these additions improve the interaction between the Network Application and the NEF Emulator, providing the necessary mechanisms to manage emulated entities and access network information required for the validation of the monitoring service.

6. Testing and Evaluation

This chapter presents the testing and evaluation of the monitoring service developed in this thesis. The goal is to validate the use case defined in Chapter 3 and assess the interaction between the Network Application and the NEF Emulator during the execution of monitoring workflows. In particular, the evaluation focuses on verifying the complete sequence of operations associated with the use case, including the creation of Loss of Connectivity subscriptions, the reception of event notifications, the creation of Location Reporting subscriptions requesting the UE's last known location, the storage and correlation of the collected information within the Network Application.

To accomplish this objective, a set of testing scenarios was executed in an emulated environment. During the tests, logs and network captures were collected from both the Network Application and the NEF Emulator to analyze the behavior of the implemented service and identify any problems affecting the monitoring workflow.

The chapter begins by describing the evaluation setup and the configuration of the testing environment. Next, the testing scenarios used to validated the use case are presented. The collected metrics and results are then analyzed, followed by a discussion of the observed behavior and the issues identified during the execution of the tests.

6.1. Evaluation Setup

To test and evaluate the monitoring service presented in this thesis, an evaluation environment was configured with two Virtual-Machine (VM) hosted on Google Cloud. The environment was designed to resemble a realistic deployment where both components, the Network Application and the ATNoG NEF Emulator, run as separate network entities.

The Network Application and the NEF Emulator are running on different VMs, which are configured as Google Cloud e2-medium instances with two virtual CPUs, 4GB of RAM and 15GB of balanced persistent disks. For the evaluation, both VMs were assigned static external IP addresses, allowing the Network Application and the emulator to consistently access each other's APIs and notification endpoints.

Firewall rules were created to allow incoming traffic on the ports required by each service. The VM with the NEF Emulator was configured to listen on port 8888 and the VM with

the Network Application was configured for connections on port 8000. These rules allow the two components to communicate in both directions while performing tests.

The following diagram illustrates the architecture of the evaluation environment and the communication routes between the two VMs. As shown in the figure, the Network Application interacts with the NEF Emulator through its REST API exposed on port 8888 to perform operations such as subscription management, UE management and retrieval of cell information. On the other hand, the NEF Emulator communicates with the Network Application through the notification callback endpoint on port 8000, which is used to send monitoring event notifications generated during the execution of the tests.

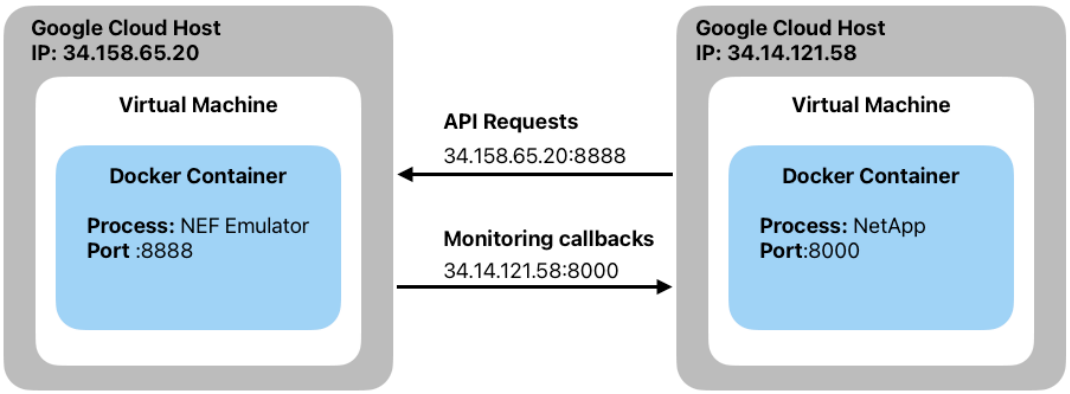


Figure 6.1: Evaluation environment deployed on Google Cloud, showing the communication interfaces between the Network Application and the ATNoG NEF Emulator.

The use of separate VMs allows the interaction between both components to be evaluated in conditions that closely resemble a real deployment environment, where the Network Application and the NEF Emulator operate independently. This architecture further supports the identification of communication failures and integration issues that may happen when exchanging monitoring requests and notifications.

This deployment provided the necessary foundation for the monitoring application to test the monitoring workflow, from setting up a subscription to receiving and processing notifications.

6.2. Use Case Testing

The purpose of the testing phase was to validate the use case for the monitoring service defined in this thesis. The goal was to ensure that the Network Application could successfully create Loss of Connectivity subscriptions for a specific set of UEs, receive

the corresponding notifications from the NEF, request the last known location of the affected UEs through Location Reporting subscriptions and store the collected information for further analysis.

Beyond validating the monitoring workflow, the testing phase also aimed to evaluate the interaction between the Network Application and the NEF Emulator in circumstances that produce a significant number of monitoring events. Since each Loss of Connectivity notification triggers additional processing and the creation of a Location Reporting subscription, the chosen testing scenarios were designed to repeatedly exercise the complete workflow during the testing period.

For this purpose, the emulated environment was configured to allow monitored UEs to move through the network and trigger Loss of Connectivity events. The next subsections describe the design of this environment, including the UE movement routes and cell deployment, and the testing scenarios.

6.2.1 UE Movement Routes

A set of UE movement routes and the corresponding cell deployment were designed and configured in the ATNoG NEF Emulator to support the evaluation of the monitoring service. The goal was to create an environment capable of generating Loss of Connectivity events in a controlled and repeatable manner and allow the interaction between the Network Application and the NEF Emulator to be validated.

In the testing, we utilized five UE movement routes as shown in Figure 6.2. These routes are defined within the emulator as a sequence of geographical points that are travelled by the UEs. All the routes were designed as closed circuits to better simulate realistic mobility patterns, allowing UEs to follow the same path continuously during a test and preventing the unrealistic behavior where a UE reaches the end of a route and is forced to return to its initial position.

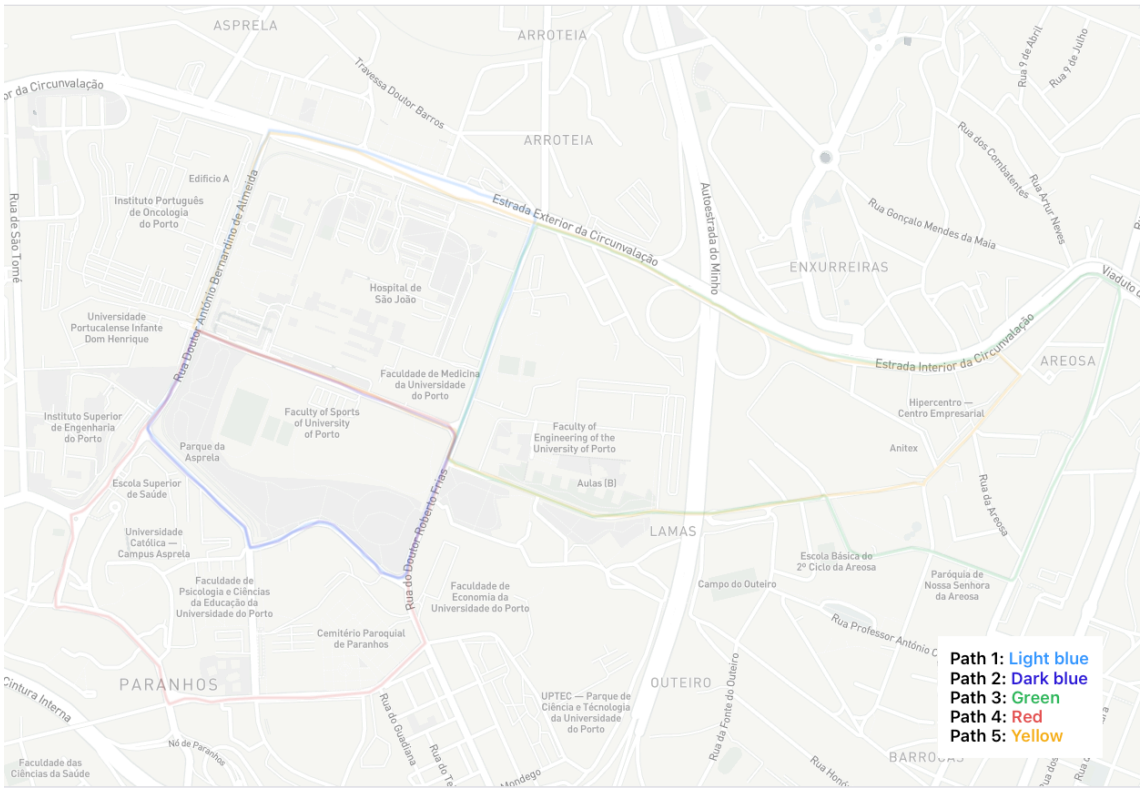


Figure 6.2: UE movement routes configure in the ATNoG NEF Emulator.

The position of the cells in the emulated environment is displayed in Figure 6.3. Since the testing was performed in an emulated environment, connectivity loss events could not be produced in real radio propagation conditions. Therefore, areas without cell coverage were intentionally introduced along the routes, causing UEs to temporarily lose connectivity and trigger a notification to the Network Application.

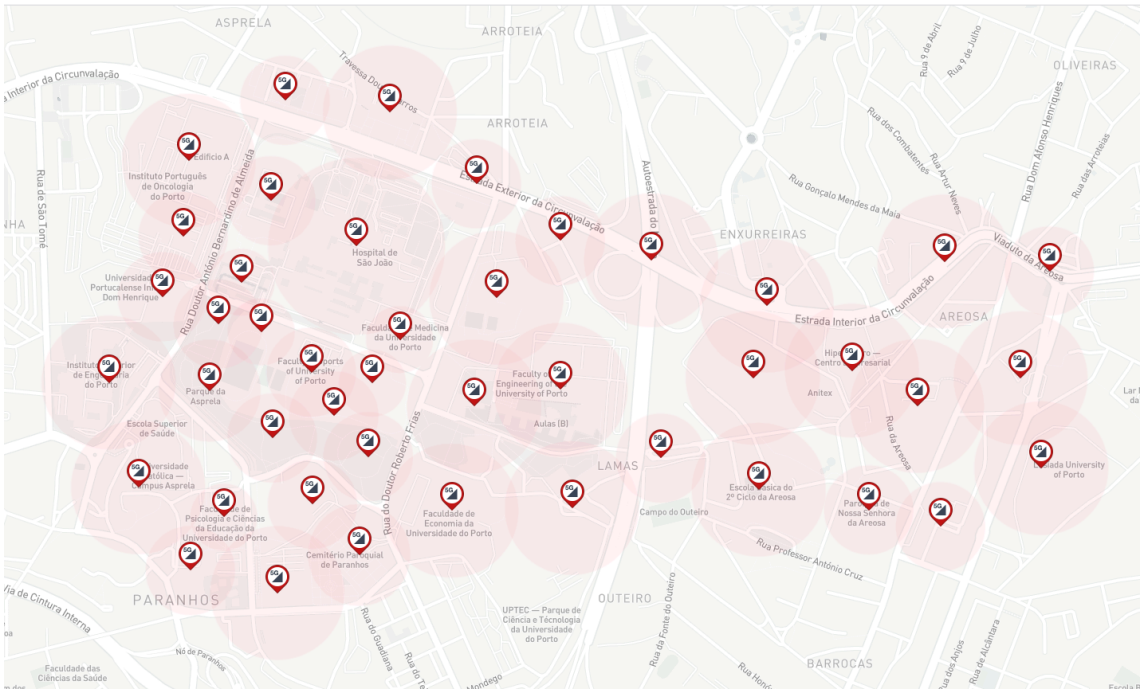


Figure 6.3: Cell placement and coverage areas configure in the ATNoG NEF Emulator.

Figure 6.4 presents a combined view of the UE movement routes and the cell deployment. The Figure illustrates how the routes cross the covered and uncovered areas of the emulated environment. The occurrence of an event depends on the monitoring parameters configured by the user, specifically the maximum detection time. For instance, if the maximum detection time is set to one second, a Loss of Connectivity notification is only sent if a UE remains without connectivity for at least the configured period.

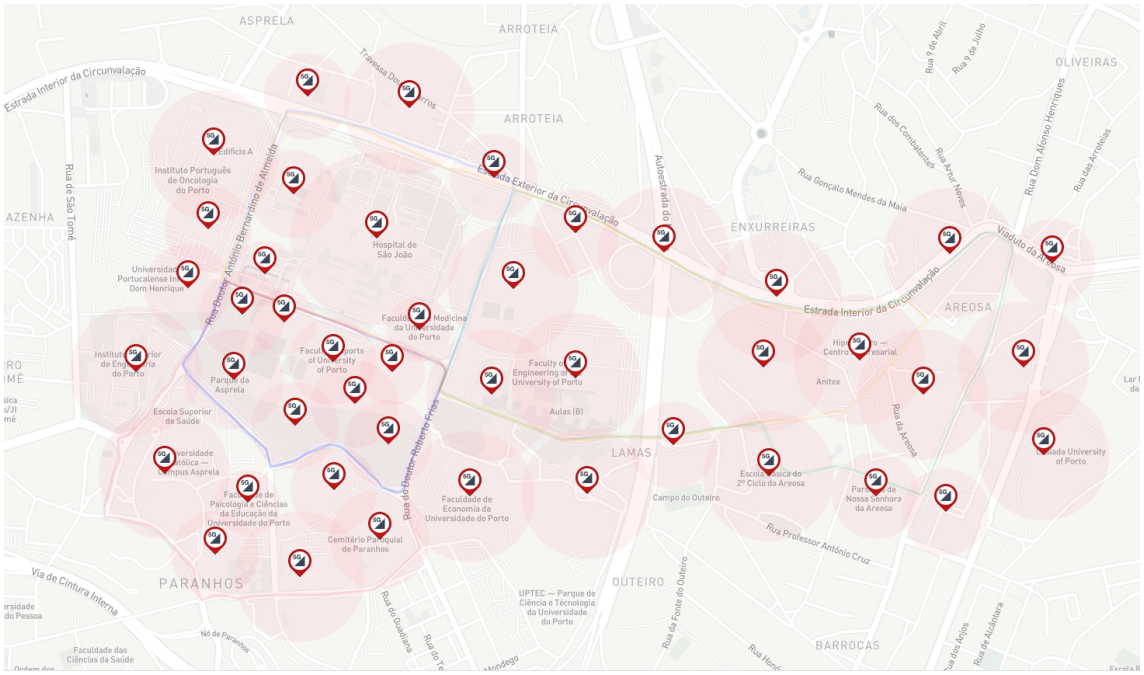


Figure 6.4: Combined view of the UE movement routes and cell deployment used during the testing.

The combination of multiple movement routes, strategically placed coverage gaps and configurable monitoring parameters provides a controlled environment for producing Loss of Connectivity events within the emulated network. This environment serves as the foundation for the testing scenarios described in the following subsection.

6.2.2 Testing Scenarios

The testing scenarios were designed to test the monitoring service with different UE distributions while maintaining the same monitoring configuration. In all scenarios, the monitoring report was configured with a maximum of 100 reports, a detection time of 1 second and the duration of 1 hour. The corresponding Loss of Connectivity subscriptions subsequently created remain active until the one hour monitoring period expires or the maximum number of 100 reports is reached.

Each scenario was executed using 10 UEs. As the ATNoG NEF Emulator supports different UE movement speeds, UEs can move at either a slow or fast pace. Therefore, to maintain consistency and introduce movement diversity within the emulated environment, all scenarios were configured with five slow UEs and five fast UEs. Additionally, all UEs movement must be started manually in the NEF Emulator.

The selected configuration serves as a moderate stress test for the monitoring service and allows to test the interaction of the Network Application and the NEF Emulator in a consistent flow of monitoring events. The UE routes, as described above, were designed to include areas without cell coverage. When a UE enters one of those areas and remains disconnected for at least one second, a Loss of Connectivity notification is sent to the Network Application. Therefore, the continuous movement of several UEs in the emulated environment causes a large number of notifications, exercising several times the communication and subscription procedures implemented by the monitoring service during the test period.

Three testing scenarios were considered:

Scenario 1: A Single Route

In the first scenario, all UEs were assigned to path 1. This configuration concentrates all UE activity in the same area. As a result, multiple UEs lose connectivity in similar places along the route.

Scenario 2: Two Routes

In the second scenario, the UEs were split equally across two paths, specifically path 1 and 2, resulting in five UEs assigned to each path. The distribution expanded the geographical coverage of generated events in the emulated environment without changing the number of monitored UEs.

Scenario 3: Five Routes

In the third scenario, the UEs were distributed over the five paths, with two UEs per route, one moving slow and the other moving fast. This configuration provides the maximum distribution of UEs in the emulated environment and triggers Loss of Connectivity events from multiple different locations.

The scenarios allow the evaluation of the monitoring service in different scenarios of movement and event triggering patterns, increasing the distribution of UEs over the available routes in a progressive way, while maintaining the same number of monitored devices. The following section analyzes the results originating from these testing scenarios.

6.3. Metrics and Results

This section presents the results obtained during the execution of the testing scenarios described previously. The goal is to test that the monitoring service is working properly and examine the interaction between the Network Application and the NEF Emulator during the tests.

This evaluation was supported by data collected from both the Network Application and the NEF Emulator. The data retrieved include application logs generated by both components, as well as packet captures from the communication interfaces of the corresponding VMs. Together, this information provides visibility into the entire monitoring workflow, including the creation of monitoring subscriptions, the delivery of notifications and the exchange of Location Reporting requests and responses.

The Network Application logs provide detailed information regarding the processing of Loss of Connectivity notifications, and the creation and processing of Location Reporting subscriptions. Furthermore, the logs contain the execution time of the different processing stages, allowing evaluation of the duration of the monitoring operations. On the other hand, the NEF Emulator logs show the response that the Network Application provides once a notification is delivered.

Packet captures were collected from both VMs and filtered to include only the traffic exchanged between the Network Application and the NEF Emulator. These captures provide an independent record of the TCP and HTTP packets exchanged between the two components and allow for verification at the network level of the communication observed in the application logs.

The evaluation, based on the collected data, is focused on metrics such as the number of Loss of Connectivity notifications and the number of Location Reporting subscriptions created after these notifications. These metrics provide an overview of the monitoring service behavior for the different test scenarios and are the basis for the analysis presented later in this chapter.

6.3.1 Monitoring Workflow Validation

The collected logs and packet captures were used to verify the correct operation of the monitoring service workflow. Since the interaction between the Network Application and the NEF has already described in subsection 3.4.1, the goal is to demonstrate that the

implemented workflow was successfully executed during the testing scenarios. This can be accomplished by correlating the information available in the Network Application logs, the NEF Emulator logs and the capture packets retrieved on both VMs.

The workflow begins when the user creates a monitoring report in the Network Application. As described previously, based on the list of UEs selected by the user, the application creates individual Loss of Connectivity subscriptions for each equipment in the NEF Emulator. The Network Application logs confirm the successful creation of these subscriptions, as shown in Listing 6.1.

```
[INFO] httpx: HTTP Request: POST
http://34.158.65.20:8888/nef/api/v1/3gpp-monitoring-event/v1/localhost/subscriptions
"HTTP/1.1 201 Created"
```

Listing 6.1: Network Application log showing the creation of a Loss of Connectivity subscription.

Once a monitored UE remains without connectivity for at least one second, the NEF Emulator triggers a Loss of Connectivity notification and delivers it to the Network Application. Listing 6.2 presents an example extracted from the NEF Emulator logs showing that the Network Application successfully received the notification.

```
[INFO] httpx: HTTP Request: POST http://34.14.121.58:8000/notifications
"HTTP/1.1 204 No Content"
```

Listing 6.2: NEF log showing the successful delivery of a Loss of Connectivity notification to the Network Application.

The Network Application logs display the reception and processing of the notification, as shown in Listing 6.3.

```
[INFO] root: LOC notification processing started
[INFO] root: LOC start external_id=10002@domain.com
[INFO] root: get_subscription_by_nef_link took 0.003s
[INFO] root: get_equipment_by_external_id took 0.002s
[INFO] root: create_events took 0.013s
[INFO] root: create_loss_conn took 0.015s
[INFO] root: LOC end external_id=10002@domain.com total=0.034s
[INFO] root: LOC notification processing completed in 0.03s
```

Listing 6.3: Network Application log displaying the processing of a Loss of Connectivity notification.

After processing the Loss of Connectivity notification, the Network Application automatically creates a Location Reporting subscription requesting the last known location of the

UE. Listing 6.4 presents an example of the log entries associated with the creation of a Location Reporting subscription.

```
[INFO] root: LR subscription creation started for external_id=10002@domain.com
event_id=7446
[INFO] root: Creating LR subscription for external_id=10002@domain.com, event_id=7446
[INFO] httpx: HTTP Request: POST
http://34.158.65.20:8888/nef/api/v1/3gpp-monitoring-event/v1/localhost/subscriptions
"HTTP/1.1 200 OK"
```

Listing 6.4: Network Application log indicating the creation of a Location Reporting subscription.

Since the Location Reporting subscription requesting the UE last known location is a one time report, the NEF Emulator immediately returns the location of the UE. Listing 6.5 presents an example of the corresponding log entries recorded by the Network Application.

```
[INFO] root: Handling LR notification for external_id=10002@domain.com, event_id=7446
[INFO] root: Received LR notification for UE with external_id=10002@domain.com
[INFO] root: get_cell_by_cell_id took 0.008s
[INFO] root: get_equipment_by_external_id took 0.002s
[INFO] root: create_location took 0.013s
```

Listing 6.5: Network Application log showing the processing the Location Reporting response.

The entire monitoring cycle can be recreated by combining the data available in the Network Application logs, the NEF Emulator logs and the packet captures. The information from the logs shows that the monitoring service successfully creates Loss of Connectivity subscriptions, receives related notifications, requests the last known location of the affected UEs and stores the resulting information. In addition, the packet captures retrieved from both VMs also support the interactions observed in the logs.

Therefore, the results obtained during the testing scenarios confirm the correct operation of the monitoring workflow implemented in this work. Having validated the successful execution of the use case, the results obtained for each testing scenario are presented in the following subsection.

6.3.2 Testing Scenarios Results

Following the validation of the monitoring workflow, the results obtained during the execution of the three testing scenarios were analyzed. The goal of this subsection is to quantify

the interactions between the Network Application and the NEF Emulator and to evaluate the behavior of the monitoring service under the previously defined UE distributions.

The results presented were extracted from the Network Application and the NEF Emulator logs generated during the execution of the testing scenarios. It should be noted that all Loss of Connectivity subscriptions associated with the monitored UEs were successfully created. Therefore, the collected logs were analyzed to determine the number of Loss of Connectivity notifications successfully delivered to the Network Application, the number of notifications that failed to be delivered, the number of Location Reporting subscriptions created effectively by the Network Application and the number of subscriptions that failed to be created.

To facilitate the verification of the presented results in the logs, Table 6.1 identifies the start and end timestamps corresponding to each testing scenario. These timestamps indicate the intervals in the Network Application and the NEF Emulator logs from which the metrics were extracted.

Scenario	Start Time	End Time
Scenario 1	2026-04-27 12:06:42	2026-04-27 13:06:42
Scenario 2	2026-04-27 10:45:02	2026-04-27 11:45:02
Scenario 3	2026-04-27 09:39:12	2026-04-27 10:39:12

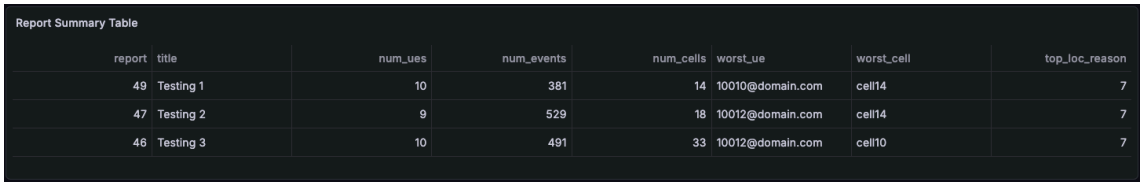
Table 6.1: Time intervals corresponding to the executed testing scenarios.

The metrics obtained in each testing scenario are summarized in Table 6.2.

Metrics	Scenario 1	Scenario 2	Scenario 3
Loss of Connectivity Notifications Received	381	529	491
Loss of Connectivity Notifications Failed to Deliver	51	5	8
Location Reporting Subscriptions Created	347	522	472
Location Reporting Subscription Failed to Create	34	7	19

Table 6.2: Summary of monitoring events and subscriptions results for each testing scenario.

In addition to the log analysis, the monitoring reports generated by the Network Application were also inspected through the Grafana dashboard described in Section 4.5. Figure 6.5 presents the report summary table containing details of each testing scenario.



Report Summary Table

report	title	num_ues	num_events	num_cells	worst_ue	worst_cell	top_loc_reason
49	Testing 1	10	381	14	10010@domain.com	cell14	7
47	Testing 2	9	529	18	10012@domain.com	cell14	7
46	Testing 3	10	491	33	10012@domain.com	cell10	7

Figure 6.5: Report Summary Table generated by Grafana.

The information shown on Grafana is consistent with the values obtained from the application logs. In particular, the number of events reported by the dashboard matches the number of Loss of Connectivity notifications successfully received by the Network Application in each scenario. These events are the correlation between a Loss of Connectivity notification and the corresponding Location Reporting subscriptions' response. If the Location Reporting subscription cannot be completed successfully, the event remains stored, but its status is set to Failed and does not contain the location information that would normally be associated with the connectivity loss event.

The results indicate that all three scenarios produced a significant number of monitoring events during the execution period. This is reflected in the high number of Loss of Connectivity notifications received by the Network Application and the matching Location Reporting subscriptions created during the tests. Scenario 2 had the highest number of Loss of Connectivity notifications, with 529 notifications received by the NEF, followed by Scenario 3 with 491 notifications and Scenario 1 with 381 notifications. The number of successfully created Location Reporting subscriptions is similar with 522 in Scenario 2, compared to 472 in Scenario 3 and 347 in Scenario 1.

Most of the interactions were completed successfully, however, some failures in notification delivery and Location Reporting subscriptions creation were observed in all scenarios. Scenario 1 had the highest number of failures, with 51 Loss of Connectivity notifications failing to be delivered and 34 Location Reporting subscriptions failing to be created. Meanwhile Scenarios 2 and 3 had significantly fewer unsuccessful interactions. The results show that the monitoring service was capable of handling a high number of monitoring events during the evaluation and also presented a small number of failures requiring further investigation.

The next section analyzes the corresponding packet captures in detail to explain the cause of the observed failures.

6.4. Analysis

The results presented in the previous section demonstrate that the monitoring service successfully executed the use case and processed a significant number of monitoring events in all the testing scenarios. However, a small number of failures were also observed during the evaluation. In particular, some Loss of Connectivity notifications generated by the NEF Emulator were not successfully delivered to the Network Application and some Location Reporting subscriptions were not created in the NEF.

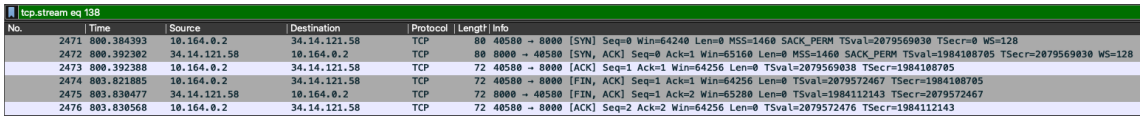
Before analyzing the observed failures, it is important to address an inconsistency in the results of Scenario 2. Although all Loss of Connectivity subscriptions were successfully created, the corresponding monitoring report, as shown in 6.5, indicates that only 9 UEs participated in the test. This behavior is unrelated to the notification delivery and subscription creation discussed in this section. Instead, it results from the operation at the NEF Emulator, where the movement of each UE must be started manually. During the execution of Scenario 2, the movement of one UE was not correctly initiated, preventing it from generating monitoring events and consequently excluding it from the resulting report.

The first failure corresponds to cases where Loss of Connectivity notifications generated by the NEF Emulator were not successfully delivered to the Network Application. These are reflected in the results as Loss of Connectivity notifications failed to deliver and Listing 6.6 shows the NEF Emulator log.

```
[ERROR] asyncio: Task exception was never retrieved
```

Listing 6.6: NEF log showing that the notification was not received by the Network Application.

The packet captures indicate that these failures do not originate from a network connectivity problem between the two VMs. Figure 6.6 shows that the NEF Emulator successfully establishes a TCP connection with the Network Application, completing the three-way handshake. However, after the connection is established, the NEF Emulator, after 3 seconds, closes the connection by sending a TCP FIN, ACK segment without transmitting the expected HTTP POST request containing the notification.



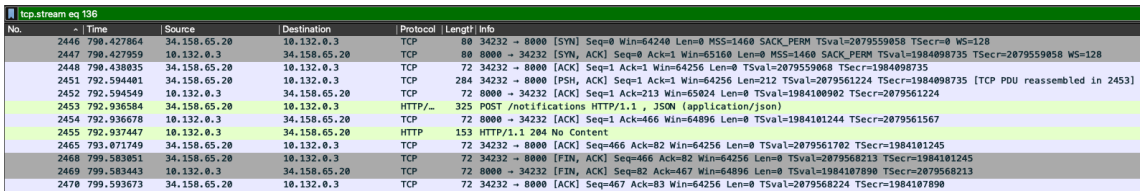
tcp.stream eq 138

No.	Time	Source	Destination	Protocol	Length	Info
2471	800.384393	10.164.0.2	34.14.121.58	TCP	80	40580 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2079569030 TSecr=0 WS=128
2472	800.392302	34.14.121.58	10.164.0.2	TCP	80	8000 → 40580 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1984108705 TSecr=2079569030 WS=128
2473	800.392388	10.164.0.2	34.14.121.58	TCP	72	40580 → 8000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2079569038 TSecr=1984108705
2474	803.821885	10.164.0.2	34.14.121.58	TCP	72	40580 → 8000 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2079572467 TSecr=1984108705
2475	803.830477	34.14.121.58	10.164.0.2	TCP	72	8000 → 40580 [FIN, ACK] Seq=1 Ack=2 Win=65280 Len=0 TSval=1984112143 TSecr=2079572467
2476	803.830568	10.164.0.2	34.14.121.58	TCP	72	40580 → 8000 [ACK] Seq=2 Ack=2 Win=64256 Len=0 TSval=2079572476 TSecr=1984112143

Figure 6.6: Example of a failed notification delivery observed in the NEF Emulator packet captures.

This demonstrates that the Network Application endpoint is reachable and that communication between both VMs is functioning correctly. Even though, the Network Application is unable to receive the Loss of Connectivity notification because no notification is transmitted.

A successful transmission of the notification, as shown in Figure 6.7, would consist of the three-way handshake, followed by a TCP PSH, ACK, holding the header of the HTTP request, and the HTTP POST containing the payload of the notification. Afterward, the Network Application responds with a HTTP 204 No Content and then the NEF Emulator closes the connection.



tcp.stream eq 136

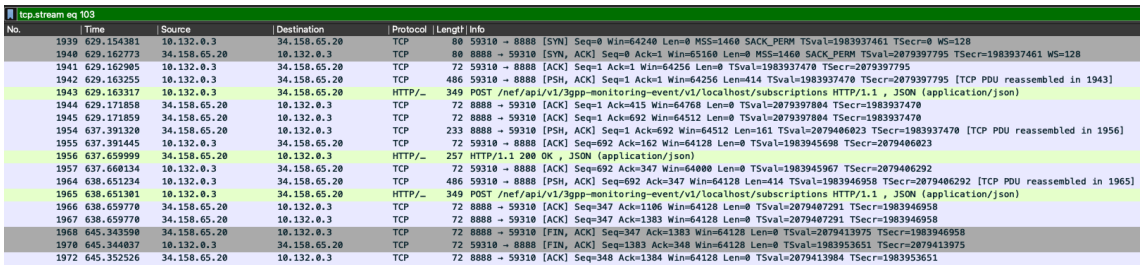
No.	Time	Source	Destination	Protocol	Length	Info
2446	790.427864	34.158.65.20	10.132.0.3	TCP	80	34232 → 8000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2079559058 TSecr=0 WS=128
2447	790.427959	10.132.0.3	34.158.65.20	TCP	80	8000 → 34232 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1984098735 TSecr=2079559058 WS=128
2448	790.438035	34.158.65.20	10.132.0.3	TCP	72	34232 → 8000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2079559068 TSecr=1984098735
2451	792.594401	34.158.65.20	10.132.0.3	TCP	284	34232 → 8000 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=212 TSval=2079561224 TSecr=1984098735 [TCP PDU reassembled in 2453]
2452	792.594549	10.132.0.3	34.158.65.20	TCP	72	8000 → 34232 [ACK] Seq=1 Ack=213 Win=65024 Len=0 TSval=1984100902 TSecr=2079561224
2453	792.936584	34.158.65.20	10.132.0.3	HTTP/…	325	POST /notifications HTTP/1.1 , JSON (application/json)
2454	792.936678	10.132.0.3	34.158.65.20	TCP	72	8000 → 34232 [ACK] Seq=1 Ack=466 Win=64896 Len=0 TSval=1984101244 TSecr=2079561567
2455	792.937447	10.132.0.3	34.158.65.20	HTTP	153	HTTP/1.1 204 No Content
2465	793.071749	34.158.65.20	10.132.0.3	TCP	72	34232 → 8000 [ACK] Seq=466 Ack=82 Win=64256 Len=0 TSval=2079561702 TSecr=1984101245
2468	799.583051	34.158.65.20	10.132.0.3	TCP	72	34232 → 8000 [FIN, ACK] Seq=466 Ack=82 Win=64256 Len=0 TSval=2079568213 TSecr=1984101245
2469	799.583443	10.132.0.3	34.158.65.20	TCP	72	8000 → 34232 [FIN, ACK] Seq=82 Ack=467 Win=64896 Len=0 TSval=1984107890 TSecr=2079568213
2470	799.593673	34.158.65.20	10.132.0.3	TCP	72	34232 → 8000 [ACK] Seq=467 Ack=83 Win=64256 Len=0 TSval=2079568224 TSecr=1984107890

Figure 6.7: Example of a notification that was successfully delivered observed in the Network Application packet captures.

The second failure detected corresponds to cases where the Network Application successfully receives a Loss of Connectivity notification and attempts to create a Location Reporting subscription, but does not receive the expected response from the NEF Emulator.

The packet captures show that these failures occur after the subscription request has already been transmitted successfully. Figure 6.8 presents an example containing two consecutive Location Reporting subscription requests. The Network Application sends an HTTP POST containing the details of the subscription, transmitted in frame 1943, and the NEF Emulator returns an HTTP 200 OK response in frame 1956, indicating that the subscription was successfully processed. On the other hand, a second subscription is transmitted in frame 1965 and is acknowledged at the TCP level by the emulator. However, no corresponding HTTP response is sent by the NEF Emulator. After a period

of inactivity, the NEF Emulator closes the connection with a TCP FIN, ACK observed in frame 1968.



tcp.stream eq 103

No.	Time	Source	Destination	Protocol	Length	Info
1939	629.154381	10.132.0.3	34.158.65.20	TCP	80	59310 → 8888 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1983937461 TSecr=0 WS=128
1940	629.162773	34.158.65.20	10.132.0.3	TCP	80	8888 → 59310 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=2079397795 TSecr=1983937461 WS=128
1941	629.162905	10.132.0.3	34.158.65.20	TCP	72	59310 → 8888 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1983937470 TSecr=2079397795
1942	629.163255	10.132.0.3	34.158.65.20	TCP	486	59310 → 8888 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=414 TSval=1983937470 TSecr=2079397795 [TCP PDU reassembled in 1943]
1943	629.163317	10.132.0.3	34.158.65.20	HTTP/…	349	POST /nef/api/v1/3gpp-monitoring-event/v1/localhost/subscriptions HTTP/1.1 , JSON (application/json)
1944	629.171858	34.158.65.20	10.132.0.3	TCP	72	8888 → 59310 [ACK] Seq=1 Ack=415 Win=64768 Len=0 TSval=2079397804 TSecr=1983937470
1945	629.171859	34.158.65.20	10.132.0.3	TCP	72	8888 → 59310 [ACK] Seq=1 Ack=692 Win=64512 Len=0 TSval=2079397804 TSecr=1983937470
1954	637.391320	34.158.65.20	10.132.0.3	TCP	233	8888 → 59310 [PSH, ACK] Seq=1 Ack=692 Win=64512 Len=161 TSval=2079406023 TSecr=1983937470 [TCP PDU reassembled in 1956]
1955	637.391445	10.132.0.3	34.158.65.20	TCP	72	59310 → 8888 [ACK] Seq=692 Ack=162 Win=64128 Len=0 TSval=1983945698 TSecr=2079406023
1956	637.659999	34.158.65.20	10.132.0.3	HTTP/…	257	HTTP/1.1 200 OK , JSON (application/json)
1957	637.660134	10.132.0.3	34.158.65.20	TCP	72	59310 → 8888 [ACK] Seq=692 Ack=347 Win=64000 Len=0 TSval=1983945967 TSecr=2079406292
1964	638.651234	10.132.0.3	34.158.65.20	TCP	486	59310 → 8888 [PSH, ACK] Seq=692 Ack=347 Win=64128 Len=414 TSval=1983946958 TSecr=2079406292 [TCP PDU reassembled in 1965]
1965	638.651301	10.132.0.3	34.158.65.20	HTTP/…	349	POST /nef/api/v1/3gpp-monitoring-event/v1/localhost/subscriptions HTTP/1.1 , JSON (application/json)
1966	638.659770	34.158.65.20	10.132.0.3	TCP	72	8888 → 59310 [ACK] Seq=347 Ack=1106 Win=64128 Len=0 TSval=2079407291 TSecr=1983946958
1967	638.659770	34.158.65.20	10.132.0.3	TCP	72	8888 → 59310 [ACK] Seq=347 Ack=1383 Win=64128 Len=0 TSval=2079407291 TSecr=1983946958
1968	645.343590	34.158.65.20	10.132.0.3	TCP	72	8888 → 59310 [FIN, ACK] Seq=347 Ack=1383 Win=64128 Len=0 TSval=2079413975 TSecr=1983946958
1970	645.344037	10.132.0.3	34.158.65.20	TCP	72	59310 → 8888 [FIN, ACK] Seq=1383 Ack=348 Win=64128 Len=0 TSval=1983953651 TSecr=2079413975
1972	645.352526	34.158.65.20	10.132.0.3	TCP	72	8888 → 59310 [ACK] Seq=348 Ack=1384 Win=64128 Len=0 TSval=2079413984 TSecr=1983953651

Figure 6.8: Comparison between a successful and a failed Location Reporting subscription request.

This shows that the failure does not originate from the communication between the Network Application and the NEF Emulator, since the request reaches its destination. When this situation occurs, the Loss of Connectivity notification remains stored in the application's database. However, since no location information is provided, the event is set to a Failed status.

6.4.1 Discussion

The evaluation results show that the monitoring service can support the use case provided. In all testing scenarios, the Network Application was able to receive and process hundreds of Loss of Connectivity notifications, create the corresponding Location Reporting subscriptions and store the resulting information. Furthermore, the consistency between the application logs, packet captures and Grafana reports shows that the monitoring workflow was working correctly during the tests.

The results indicate that the monitoring service remained operational throughout the testing, despite the small number of failures that occurred. Across the three testing scenarios, the Network Application received 1401 of the 1465 Loss of Connectivity notifications, resulting in a notification delivery success of about 95.6%. At the same time, 1341 out of 1401 Location Reporting subscriptions requests were successfully created, indicating a subscription success rate of approximately 95.7%. These results show that a large percentage of monitoring events were properly processed.

Metrics	Successful	Total	Percentage of success
Loss of Connectivity Notifications	1401	1465	95.6%
Location Reporting Subscriptions	1341	1401	95.7%

Table 6.3: Summary of the results obtained in the testing scenarios.

The packet captures provide valuable information into the cause of the identified failures. In both cases, communication between the Network Application and the NEF Emulator was established correctly. The NEF Emulator initially established a TCP connection with the Network Application but closed it before transmitting the expected HTTP notification, resulting in a failed notification delivery. Similarly, when a Location Reporting subscription failed to be created, the emulator received the subscription request, but the expected HTTP response was never provided. These observations prove that the failure is not caused by connectivity problems between the two VMs.

Although the exact cause cannot be determined based on the data provided, the packet captures suggest that the failures occur when monitoring events are being processed by the NEF Emulator. The timestamps visible in the captures reveal that after communication has already been established, some requests take longer to receive a response. The most probable explanation is that the emulator has difficulties processing monitoring events during higher activity periods, which results in notifications or responses not being delivered correctly.

Since the testing of the monitoring service was performed using an emulated environment, there are some limitations that may be specific the emulator rather than to the monitoring service. The ATNoG NEF Emulator serves as a research and testing platform and does not necessarily offer the same level of service expected from a production NEF.

To sum up, the evaluation shows that the proposed monitoring service supports the intended use case and is capable of correlate connectivity loss events to UE location information. The next chapter concludes this work by summarizing this work, discussing its limitations and outlining ideas for future work.

7. Conclusion and Future Work

The objective of this thesis was to design, implement and evaluate a monitoring service capable of detecting and investigating connectivity loss events through the use of NEF capabilities. The proposed Network Application allows users to monitor a set of UEs through Loss of Connectivity subscriptions and, upon receiving a connectivity loss notification, automatically requests the UE's last known location via a Location Reporting subscription. The service provides additional information which can help in the identification and analysis of connectivity issues in a mobile network by combining loss of connectivity events with location information.

Apart from the implementation of the Network Application, this work also contributed to the ATNoG NEF Emulator by extending the support for Location Reporting subscriptions. The implementation of the standardized location types has allowed the retrieval of the UE location information required by the monitoring service and facilitated the testing of the proposed use case in the emulated environment.

The developed application was tested using a series of testing scenarios involving multiple UEs moving through predefined paths that included areas without cell coverage. The results demonstrated that the monitoring service could successfully receive and process Loss of Connectivity notifications, create the subsequent Location Reporting subscriptions and integrate the resulting information into a dashboard for users to analyze. In all scenarios, the monitoring service demonstrated a high success rate in executing the monitoring workflow. Although some failures were observed, the analysis of logs and packet captures revealed that they were related to limitations on the emulated environment rather than the monitoring service itself.

The objective and requirements defined in this work have been achieved. The implemented application demonstrates how NEF capabilities can be used to provide additional context about connectivity issues in mobile networks, such as the correlation of connectivity loss events with UE location information and supporting the detection of problematic cells or devices.

The next sections discuss the main limitations found during the development and evaluation of the monitoring service, and present some functionalities that could be implemented in the application in the future.

7.1. Limitations

The development and evaluation of the proposed monitoring service revealed some challenges, despite its success in meeting the defined objectives. These limitations are related to the characteristics of the testing environment, but also to the complexity of implementing standardized NEF procedures and the tools available to obtain monitoring information. It is important to understand the limitations of the obtained results for correct interpretation and identification of opportunities for future improvements.

One of the challenges in this work was the complexity associated with interpreting and implementing some 3GPP specifications. The standards provide detailed definitions of NEF services and procedures, but some aspects require interpretation when it comes to translating them into a concrete implementation. An example is the implementation of the INITIAL_LOCATION location type in the ATNoG NEF Emulator [11]. The specifications define the existence of this location type, but do not specify how the network should keep the initial location internally. Therefore, when it came to implementing it in the emulator, the initial location was considered as the first serving cell associated with a UE. This example illustrates that the implementation of standardized functionalities may need some additional decisions on top of the specification.

A further observation is the tools available to retrieve the UE's last known location. The application is an event-driven service, as Loss of Connectivity notifications are triggered once the predisposed conditions are fulfilled. However, once a connectivity loss notification is received, the application must create a Location Reporting subscription to obtain the last known location of the UE. In the use case implemented, the requested location information information is returned immediately by the NEF Emulator and no continuous monitoring is needed. Nevertheless, the 3GPP specifications define location retrieval through the creation of a monitoring subscription, requiring the use of an HTTP POST request. So, the application is creating a new subscription for every notification received, despite only requiring a single response [11]. A simpler mechanism would be a query-based model, where the application could get the location information right away without waiting for a response. However, the 3GPP specifications define location retrieval by creating monitoring subscriptions and therefore the monitoring service must follow the standardized approach.

Another limitation is associated with the use of the ATNoG NEF Emulator as the platform to validate the monitoring service. While the emulator provides a practical environment

for testing applications that interact with the NEF, it does not possess all the network functionalities that would be expected of a NEF in a production environment. Therefore, the service's scalability cannot be fully assessed in the emulated environment.

Despite these challenges, the developed solution provided a functional proof of concept that takes advantage of the NEF to support the investigation of connectivity loss in the network. The following section presents possible implementations for future work.

7.2. Future Work

The current implementation correctly validates the intended use case, but some improvements can be considered to extend its functionality and improve its applicability in more demanding environments. The experience gained throughout the development of this work highlighted areas where the solution could be improved in terms of deployment, scalability and user experience.

A potential improvement is related to the processing of notifications received from the NEF. During the evaluation, all notifications were directly processed by the Network Application in background tasks. This approach was effective for the proposed use case, but using a dedicated message queue could help enhance the scalability and reliability of the monitoring service. This would allow the application to process a higher volume of notifications more efficiently.

Another important addition would be to enhancement of the monitoring service's visualization capabilities. In the current implementation, the Grafana interface is decoupled from the web application's interface. A possible improvement would be to integrate these dashboards into the Network Application so users could access statistics and charts in a single interface. The goal of this enhancement would be to improve usability and provide a smoother user experience.

The most important goal for future work is to validate the monitoring service in a production 5G environment. The testing presented in this thesis with the ATNoG NEF Emulator, which provides a practical platform for development and testing, does not fully reproduce the behavior of a real deployment. Therefore, testing the application with a production 5G Core (5GC) would provide valuable information regarding scalability, performance and practical applicability of the monitoring service in an operational network.

Finally, the monitoring service could be extended to support additional NEF capabilities and monitoring events beyond Loss of Connectivity and Location Reporting. This would

give the application a broader view of the network conditions and further assist in the analysis and troubleshooting of network issues.

References

[1] 3GPP, “5G; System architecture for the 5G System (5GS),” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 12 2024, version 18.8.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144 [Cited on pages ix, 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 14, and 16.]

[2] ——, “5G; 5G System; Network Exposure Function Northbound APIs; Stage 3,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.522, 9 2025, version 18.11.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3437 [Cited on pages ix, 1, 9, 10, 11, 12, 13, 14, and 63.]

[3] G. Brown, P. Analyst, and H. Reading, “Service-based architecture for 5g core networks,” 11 2017, accessed: 2025-06-26. [Cited on pages 5, 6, and 8.]

[4] H. Holma, A. Toskala, and T. Nakamura, *5G Technology: 3GPP Evolution to 5G-Advanced*, 2nd ed. John Wiley Sons Ltd., 2024, accessed: 2025-06-26. [Online]. Available: https://books.google.pt/books?id=g5jxEAAAQBAJ&printsec=copyright&redir_esc=y#v=onepage&q&f=false [Cited on pages 6, 10, and 14.]

[5] A. Franklin and S. Tambe, “Multi-access edge computing in cellular networks,” *CSI Transactions on ICT*, vol. 8, 05 2020, accessed: 2025-06-26. [Cited on pages 6 and 16.]

[6] 3GPP, “5G; 5G System; Technical Realization of Service Based Architecture; Stage 3,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.500, 6 2025, version 18.9.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3338 [Cited on page 6.]

[7] ——, “Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Architecture enhancements to facilitate communications with packet data networks and applications,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.682, 3 2024, version 18.0.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=862 [Cited on pages 8 and 12.]

[8] C. Chiasserini, “5g core network architecture: Network exposure function,” Ph.D. dissertation, Politecnico de Torino, 2020, accessed: 2025-06-26. [Online]. Available: https://webthesis.biblio.polito.it/12557/?template=default [Cited on page 9.]

[9] 3GPP, “5G; Procedures for the 5G System (5GS),” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.502, 6 2024, version 18.6.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3145 [Cited on pages ix, 9, 12, 14, 16, and 17.]

[10] ——, “5G; 5G System; Network Exposure Function Southbound Services; Stage 3,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.591, 3 2024, version 18.5.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3681 [Cited on pages ix, 9, and 10.]

[11] ——, “Universal Mobile Telecommunications System (UMTS); LTE; 5G; T8 reference point for Northbound APIs,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.122, 9 2024, version 18.7.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3239 [Cited on pages 10, 11, 12, 21, 30, 31, 32, 35, 43, 57, and 85.]

[12] 5GASP, “Publications - conferences and papers,” 5gasp.eu, accessed: 2026-04-15. [Online]. Available: https://www.5gasp.eu/publications/conferences.html [Cited on page 11.]

[13] J. Fernandes, L. Rosa, J. Gameiro, P. Valente, E. Ramos, J. Oliveira, D. Raposo, A. T. Ferrer, J. M. Portela, P. Rito, L. Cordeiro, and S. Sargento, “Pushing the boundaries of scalable 5g core networks: Cloud-native nef and capif interplay,” in *2024 3rd International Conference on 6G Networking (6GNet)*, Oct 2024, pp. 201–205, accessed: 2026-03-05. [Cited on pages 15 and 16.]

[14] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, “On enabling 5g automotive systems using follow me edge-cloud concept,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5302–5316, June 2018, accessed: 2026-04-14. [Cited on pages 16 and 17.]

[15] S. Giannakidou, P. Radoglou Grammatikis, S. Koussouris, M. Pertselakis, N. Kanakaris, A. Lekidis, K. Kaltakis, M. Koidou, C. Metallidou, K. Psannis, S. Goudos, and P. Sarigiannidis, “5g-enabled netapp for predictive maintenance in critical infrastructures,” 10 2022, accessed: 2026-05-10. [Cited on page 18.]

[16] J. Gallego-Madrid, A. Hermosilla, and A. Skarmeta, “Dynamic deployment and testing of virtual on-board units in 5g,” Oct. 2022, accessed: 2026-05-10. [Online]. Available: https://doi.org/10.5281/zenodo.7221666 [Cited on page 18.]

[17] S. Lee, “Documentation,” Open5GS, 2025, accessed: 2026-04-15. [Online]. Available: https://open5gs.org/open5gs/docs/ [Cited on page 19.]

[18] L. Foundation, “Documents - free5gc,” Free5gc.org, 2024, accessed: 2026-04-15. [Online]. Available: https://free5gc.org/doc/ [Cited on page 19.]

[19] OpenAirInterface, “Oai ran,” Openairinterface.org, accessed: 2026-04-15. [Online]. Available: https://openairinterface.org/ran/ [Cited on page 19.]

[20] ——, “Core network,” Openairinterface.org, accessed: 2026-04-15. [Online]. Available: https://openairinterface.org/core-network/ [Cited on page 19.]

[21] L. Foundation, “Camara project – linux foundation project,” Camara Project, 2023, accessed: 2025-04-10. [Online]. Available: https://camaraproject.org [Cited on page 19.]

[22] ATNoG, “Github atnog/nef-emulator: Configurable, emulated environment for providing network exposure function (nef) apis,” GitHub, accessed: 2025-05-16. [Online]. Available: https://github.com/ATNoG/NEF_emulator [Cited on pages 19 and 53.]

[23] GeeksforGeeks, “Mongodb database, collection, and document.” GeeksforGeeks, accessed: 2026-05-12. [Online]. Available: https://www.geeksforgeeks.org/mongodb/mongodb-database-collection-and-document/ [Cited on page 55.]