Design Document

HPE CTY

NSSF(Network Slice Selection Function)

Under the guidance of

MS. GRACE PRISCILLA NAMBI

DR.JAYAKUMAR S

Team Members

Vinita K Vaswani R, 20BCE0060

Stuti, 20BDS0170

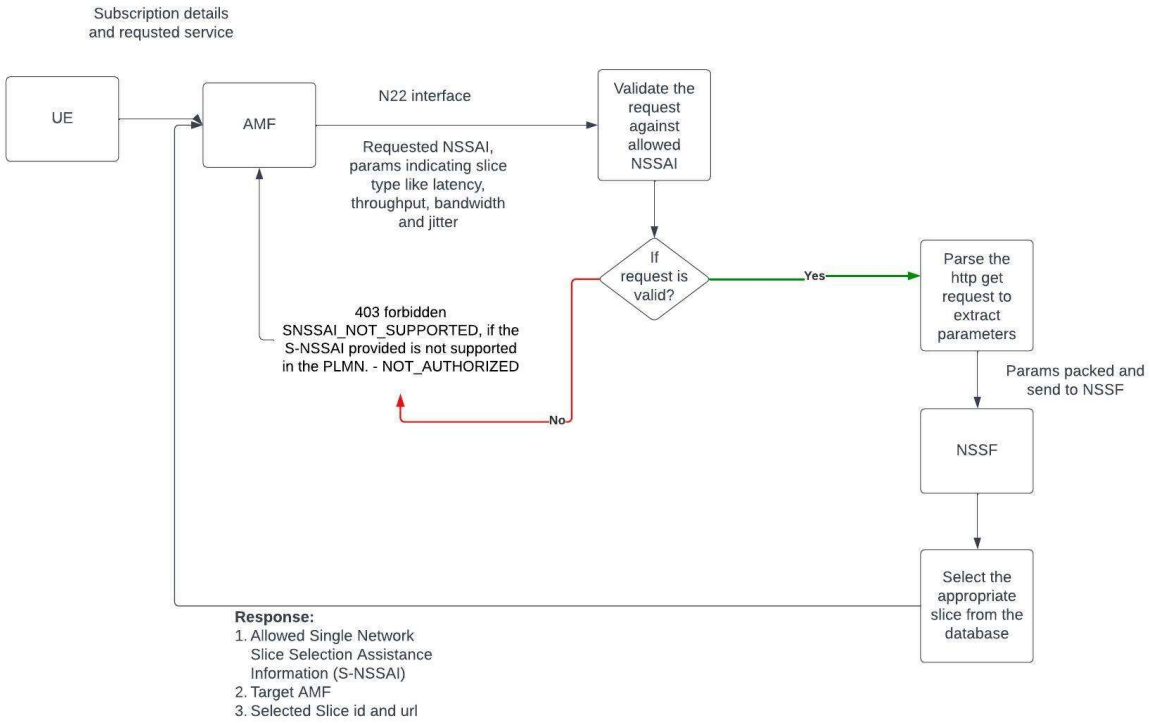
Edil Auxillea, 20BDS0393

Harsh Anand, 20BDS0410

Deepika Pavundoss, 20BEC0285

Table of Contents:

I. ***Architecture / Call Flow:***



1.Firstly, AMF sends an HTTP/2 request to NSSF with the required data, such as the NSSAI and other parameters associated with slice asked for. This request is then parsed to extract the required params like latency, throughput and bandwidth, from the request body.

2.NSSF validates the request against allowed NSSAIs, i.e. if the user asking for a network slice has actually subscribed for it during the registration. Thus, the user is only given access to the subscribed ones, depending upon the SLA agreements, subscriptions, etc.

3. NSSF establishes a connection to the database and executes a query to get the appropriate slice type.

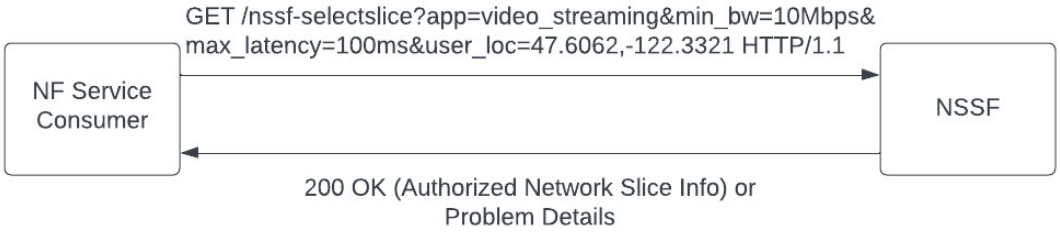
4.It retrieves information such as the network slice instance ID, network slice subnet, and other relevant information.

5.NSSF generates a response message containing the network slice information. If the AMF sending the request can service the selected slice, then the slice information is returned, otherwise a target AMF Set is returned which contains

the list of AMFs that can service the selected slice. It sends this response back to the AMF.

6.AMF parses the response message to extract the network slice information and returns a response to the UE based on the received network slice information and its own policy.

II. GET Request from AMF To NSSF:



Here, after the user gives the type of slice/application required , the amf sends the parameters like latency, bandwidth and through-put to the NSSF. NSSF then selects the slice and returns a response after conducting several checks including the network availability and resources and returns a target AMF set. This includes the AMFs that can be used to service the user requirements.

III. AMF Reallocation:

It occurs when the slice selected by the NSSF is not served by the AMF requesting it. Then, NSSF sends request for AMF-reallocation to ensure that UE is served by the correct AMF.

IV. Database Design:

- **SliceRepo:** A table containing all the information about the available slices. It includes their nssai which will be assigned during to the UE during initial registration, the slice identifiers and slice differentiators, supported application, slice type, QoS parameters and the availability.

- **Subscriptions:** This contains information about the users like the user id, assigned plmn(mcc and ncc), location and the slices which they have subscribed to.
- **AMFTable:** This table, contains information regarding the mapping between the slices and the AMFs.i.e which slice IDs are associated with which AMFs.
- **SliceResource:** This table, contains information about available slices, their Slice IDs and their respective QoS parameters like jitter, latency, bandwidth on the basis of which the most appropriate slice will be selected by the NSSF.

V. Types of Slices:

- **Massively IoT (Internet of Things):** IoT device installations on a large scale are catered for by MIoT slices.They are designed for low-data-rate, low-power applications that involve lots of connected devices.Long battery life, broad coverage, and effective resource utilisation are prioritised by MIoT slices.

- **(eMBB) Enhanced Mobile Broadband:**High-speed data transport capacities are improved and user experiences are better by eMBB slices. For uses including high-definition video streaming, virtual reality, and augmented reality, these slices have been specially designed.
  To accommodate bandwidth-demanding applications, eMBB slices provide greater data rates, lower latency, and increased network capacity.

- **V2X: Vehicle-to-Everything:** Slices of V2X technology are designed to support communication between vehicles and diverse environmental elements. These slices allow communication between vehicles, between infrastructure, and between pedestrians (V2P).
  In order to support sophisticated driver assistance systems, cooperative driving, and intelligent transportation systems, V2X slices prioritise low-latency, high-reliability connections.

- **Low-Latency Ultra-Reliable Communication (URLLC):** It enable extremely dependable, low-latency communication. For essential applications like industrial automation, remote surgery, and public safety, they are created to adhere to high reliability and latency criteria.

To assure mission-critical communications, URLLC slices provide incredibly low latency, great reliability, and robustness against packet loss.

- **mMTC, or massive machine-type communication:**
  In IoT applications, mMTC slices often provide massive-scale connectivity for a large number of devices.
  These slices are designed for occasional, low-data-rate applications including asset tracking, environmental monitoring, and smart city deployments.In order to support a high number of devices, mMTC slices prioritise the effective use of network resources, energy efficiency, and scalability.

VI. Response and Error Codes:

HTTP Response Code	Condition
400 (Bad Request)	All semantic, syntax errors lead to this response code. It indicates that request is not valid according to protocol such as invalid json
500(Internal Error)	Database error. Operation not supported / not valid.
403(Forbidden)	User is not authorized to access the requested type of slice. (not subscribed).
404	Requested service not available at the particular moment
200 (OK)	Success Code

VII. Description of Modules:

1. Request from UE to AMF and AMF to NSSF:

To initiate the process, the user communicates their specific service requirements to the Access and Mobility Management Function (AMF) along with their unique user ID. This service type serves as a crucial parameter in guiding the subsequent steps. The AMF acts as a gateway, facilitating the user's interaction with the 5G network.

Upon receiving the user's request, the AMF springs into action, leveraging the provided service type information to extract pertinent details from the 'slicerepo' table. This repository contains a wealth of information necessary for efficient network slice selection. Key details obtained include the Network Slice

Selection Assistance Information (NSSAI), which encapsulates various slice attributes and capabilities. Additionally, the AMF retrieves the Single Network Slice Selection Assistance Information (SNSSAI), which further refines the specifications of the desired slice. Other essential parameters include the Service Slice Type (SST), which categorizes the type of service required, as well as the Slice Service Differentiator (SSD) that distinguishes between multiple slices of the same service type.

Then the AMF, armed with the extracted information, now proceeds to validate whether the user has subscribed to the requested service type.

2. Authentication:

Requested NSSAI: A user equipment (UE) includes the requested NSSAI as part of the first registration process when it establishes a connection to the 5G network. The precise network slice instance or service that the UE wants to use is specified in the requested NSSAI.

Subscribed NSSAI: The subscribed NSSAI is kept in the 5G core network's Home Subscriber Server (HSS) or Unified Data Management (UDM) and is linked to the subscriber's service subscription. The network slice instances or services that the subscriber is authorised to are represented by this.

UE authentication is carried out via the AMF, the requested and subscribed NSSAI. Using the uid,(in the ‘subscription’ table), we have made a list of type of slices that a user has subscribed for. If the user is requesting for a slice among them, then we return a success code 200 and forward the request to the NSSF with the payload in order to select the most appropriate slice.

The subscription NSSAI connected with the subscriber's profile serves as the basis for the central entity, the AMF, to authenticate and authorise the UE's access to the requested network slice. In case a user is requesting for a slice, for which it has not subscribed then, error code(403 Forbidden) is returned. This process makes sure that the UE is authenticated and permitted to use the desired network slice instance or service. Once the authorization is successful, the necessary parameters are send to the NSSF for slice selection.

3. Slice Selection by NSSF and AMF Reallocation:

• As requested is parsed to NSSF, the very first step is to check for Bad Request which means whether the Request contains all the Parameters required by the NSSF for slice selection.

• Secondly, the NSSAI will be used to find the SNSSAI which is available and also the QoS will be matched to check if the slice is providing the required service or not, if not then the Slice Not Found will be given as an output.

• After the SNSSAI is found out the respective SliceID will be considered as the User Slice.

• The respective SliceID will be used to associate its AMF from AMFRepo table and if the AMF is Initial AMF which we considered as AMF-1 then the Response will be sent to UE and if not then the Target AMF set will be sent which will help in AMF Reallocation.

Inputs and Responses of each operation in NSSF:

1.Network Slice Selection:

- **Inputs:**

  UE capabilities: Information about the capabilities of the User Equipment (UE), such as supported radio access technologies, frequency bands, and QoS requirements.

  Service requirements: type of application, required latency, throughput, reliability, and other service-specific requirements.

  Network conditions: details about the current network conditions, such as available network resources, congestion levels, and other network-related parameters.

- **Responses:**

  Selected network slice: The NSSF provides information about the selected network slice, which may include the slice ID, slice type, QoS parameters, and other slice-specific information.

Resource URI: {apiRoot}/nnssf-nssaiavailability/{apiVersion}/nssai-availability/subscriptions

Slice-specific information: additional information about the selected network slice, such as available resources, slice instance ID, and other slice-related information.

2.Network Slice Availability and Status:

- **Inputs:**

  Network slice status: Information about the current status of network slices in the network, such as their availability, health, and resource usage.

- **Responses:**

  Slice availability and status: The NSSF provides information about the availability and status of network slices in the network, which may include the current status, resource usage, and health of network slices.

3.Authentication and Authorization:

- **Inputs:**

  authentication tokens, authentication keys, or other credentials.

- **Responses:**

  Authentication and authorization result: whether the UE or service is authenticated and authorized to access the selected network slice.

4. Docker and Grafana:

There are 4 docker images:

a. nssf-db
b. nssf-host
c. nssf-request
d. nssf-automate

a) nssf-db:

Dockerising the SQL database involves running the database server inside a Docker container. All the tables required by the Network Slice Selector are present in the "nssf" database of this docker container.

It is implemented using 2 files:

- Dockerfile: the docker file used to containerize the database by setting up the details of mysql connection such as:
    - Base image for the container being: MySQL image's latest version from Docker Hub
    - Database name: nssf
    - Root Password: password
    - Root user: root
    - Port: 3307 which is the default port used by MySQL, to allow connections from outside the container
- init.sql: consists of all SQL queries starting right from creation of the NSSF database for the system to creating all the tables involved to store and obtain values for the slice selector.

Key points:

- Containerizing the database prevents any need to the creation of the database in host system and occupying the memory.
- This isolation of the database by containerizing it ensures that the database and its dependencies are kept separate from the host system and other containers helping in minimizing any potential conflicts created if a similar database is present in the host system or any MySQL credential issues.
- Other Docker containers can access the SQL database container by connecting to the network of the database container, "mynetwork".
- When accessing the SQL database from other Docker containers, the connection details (hostname, port, username, password) of the database container are used to establish a connection. This allows other services or applications running in different containers to interact with the database.

b) **nssf-host:**

This container helps in hosting the Flask Application by running the image, connecting to the "mynetwork" network, on port: 5000

It consists of the following files to containerize:

- app.py: consists of the Flask framework and the code to obtain parameters and allocate the slice by accessing the database through the container – nssf-db.
  It accesses the database image by specifying the following connection details.
    1. Host: mysql-container (which is the image name of the nssf-db container)
    2. User: root
    3. Password: password
    4. Database: nssf
    5. Port: 3306
- requirements.txt: Consists of a list of all the dependencies along with their versions to be installed to run all the libraries in app.py python file.
- Dockerfile:
    - It uses base image as Python 3.9 running on a slim version of Debian Buster from Docker Hub.
    - Pip is upgraded to the latest version and all the necessary dependencies for MySQL client are first installed.
    - All the dependencies mentioned in requirements.txt are installed.
    - Exposes port 5000 within the container indicating that the container will listen on this port when it runs.
    - It runs the Flask application using the flask run command, specifying --host=0.0.0.0 to allow external connections.

c) <u>nssf-request:</u>

It containerizes the files which obtain the user input and display the slice allocated.

It consists of the following files:

- requestApp.py: It consists of accepting user input, accessing database and the hosted Flash application to allocate the slice.
- testing.py: It consists of code to test the working of the database and the application.
- Dockerfile:
    - It uses base image as Python 3.9 running on a slim version of Debian Buster from Docker Hub.

- Pip is upgraded to the latest version and all the necessary dependencies for MySQL client are first installed.
- All the dependencies mentioned in requirements.txt are installed.
- The image execution starts with requestApp.py which is a python file.

- requirements.txt: Consists of a list of all the dependencies along with their versions to be installed to run all the libraries in app.py python file.

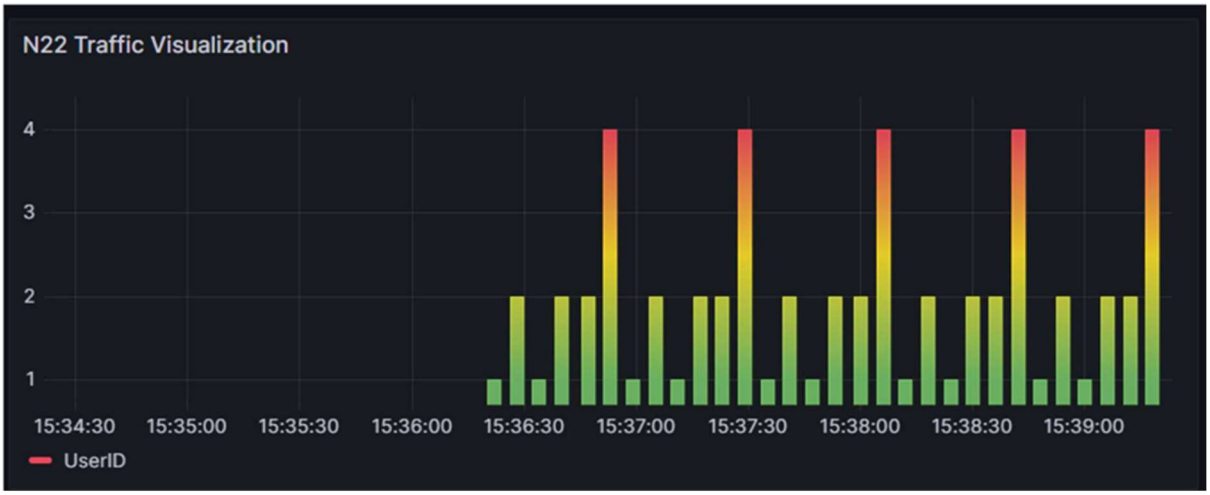
d) <u>nssf-automate:</u>

It consists of a set of pre-defined user inputs which are run continuously unless and until the image is run. On running this Image, the testing details of the application is displayed along with the slices allocated for each of the pre-defined user inputs. It consists of the files same as nssf-request with 1 additional file – input.py which consists of all the pre-defined user inputs.

Grafana Visualization:

This part helps visualize the traffic on N22 interface on Grafana.

A live visualization of the network traffic is displayed on the Grafana Dashboard and it is as follows:



Each bar represents a communication via the N22 interface. The X-axis represents the timestamp of each such traffic and the Y-axis represents the UserID of an authorized User, who is requesting for the slice allocation.

First a Grafana data source has to be created with following details:

Host: localhost:3307

Database: nssf

User: root

Password: password

A dashboard using the same data source should be created and the following query has to be executed:

select datetime as time, UserID as val from nssf.Network

5. Unit Testing:

- The unit testing verifies the input parameters that's taken from users and the response generated from each , if it met the expected output . Its a mechanism to test for the accuracy of the NSSF function based on the response generated for particular users, and measure the deviations for the same.
- We have defined slices as objects of a class named "Network slice" with attributes like bandwidth range, latency range and jitter.
- We have also defined inputs with the expected output against the same as a list.
- We'll run for each test case, checking if the expected output is meeting the true output based in the actual output obtained in the database. This will be done only after establishing a connection to the 'NSSF' module in the database.