LAB 8

CELL (Réseaux cellulaires)
Prof. Serge FDIDA
Prof. Mai Trang NGUYEN

Lab Assistants: Dimitris Kefalas, Sokratis Christakis
(Dimitrios.Kefalas@lip6.fr, Sokratis.Christakis@lip6.fr)

Subject: Deploy 5G real-world networks using Docker-Compose and OAI
Date: 25/11/2024

This lab guides the deployment of real-world 5G networks using Docker-Compose and the OpenAirInterface (OAI). It provides hands-on experience in setting up and configuring 5G components within a containerized environment, offering a clear understanding of essential network functions such as the Core Network (CN), Radio Access Network (RAN), and User Equipment (UE) simulation. Docker-Compose simplifies the orchestration of containers, making the deployment and management of OAI's complex 5G architecture more accessible. This practical approach demonstrates the workings of 5G systems and highlights the advantages of containerization, essential to modern network engineering.

OpenAirInterface (OAI) is an open-source software suite designed to simulate and implement 4G and 5G mobile networks, offering a comprehensive platform for research, development, and experimentation with mobile communication systems. Developed and maintained by the OpenAirInterface Software Alliance, OAI includes software implementations of key components in cellular networks, such as the Core Network (CN), Radio Access Network (RAN), and User Equipment (UE). By providing these elements, OAI enables users to deploy, test, and modify real-world mobile network architectures in both simulated and physical environments. Its flexibility and compliance with 3GPP standards make OAI a valuable tool for academic institutions, research centers, and industries exploring the future of wireless communications.

In **Lab 5**, Docker-Compose was introduced as a tool to manage and orchestrate multiple containers in a streamlined manner, essential for complex network applications. Having gained experience with Docker-Compose, you will now use it specifically to deploy the 5G Core Network (CN), while the Radio Access Network (RAN) and User Equipment (UE) will be set up on bare metal. This approach allows for efficient containerized management of core network functions, while providing direct, low-level control over the RAN and UE components, reflecting a realistic 5G deployment scenario.

Exercise 1. Docker-Compose deployment of 5GCN and Bare-Metal deployment of 5G RAN

The provided docker-compose-5g-core.yaml file defines several services for a 5G core network setup, creating multiple network function entities essential for 5G operation. Here's an illustration of the entities created from this configuration:

MySQL Database (mysql):

- Runs a MySQL container as the database for storing network-related data.
- Configured with initialization scripts and health checks.
- Attached to the public_net network.

Unified Data Repository (oai-udr):

- Stores and manages subscriber-related information.
- Depends on MySQL and the Network Repository Function (NRF).
- Exposes TCP ports for service communication.

Unified Data Management (oai-udm):

- Manages user profiles and authentication data.
- Connects to the NRF and depends on it for operational data.
- Exposes ports for internal service communication.

Authentication Server Function (oai-ausf):

- Handles authentication requests for 5G network access.
- Relies on the NRF to manage connectivity and service discovery.
- Exposes ports for communication within the network.

Network Repository Function (oai-nrf):

- Centralized service registry for other network functions.
- Provides discovery services for the UDR, UDM, AUSF, and other functions.
- Configured on the public_net for inter-service communication.

Access and Mobility Management Function (oai-amf):

- Manages UE access, mobility, and session setup.
- Communicates with the NRF and other core network functions.
- Exposes both TCP and SCTP ports for signaling.

Session Management Function (oai-smf):

- Responsible for session management, including setting up and releasing sessions.
- Connects to the NRF and communicates with the UPF for data forwarding.
- Exposes TCP and UDP ports.

User Plane Function (oai-upf):

- Manages user plane data forwarding.
- Communicates with the SMF and connects to external networks.
- Configured with specific permissions (NET_ADMIN, SYS_ADMIN) for network administration.
- Runs with elevated privileges for low-level network operations.

External Data Network (oai-ext-dn):

- Simulates an external data network for user equipment (UE) traffic.
- Configured to communicate with the UPF.
- Uses health checks to ensure connectivity.

Network Configuration (public_net):

- A bridge network, named demo-oai-public-net, for interconnecting all the services.
- Configured with a specific subnet (192.168.70.128/26) to allow controlled IP address assignment.

This setup enables a fully integrated 5G core network using Docker-Compose, providing communication, data management, and user session handling essential for 5G functionality.

In order to deploy all this svcs and containers you have to execute this simple command:

sudo docker-compose -f docker-compose-5g-core.yaml up -d

In the home directory, located at `/home/cell`, there is a folder named `openairinterface5g` that contains the precompiled OpenAirInterface 5G software, which implements both the User Equipment (UE) and the gNodeB (gNB). The path to the gNB executable is `/home/cell/openairinterface5g/cmake_targets/ran_build/build/nr-softmodem`

while for the UE, the executable can be found at

`/home/cell/openairinterface5g/cmake_targets/ran_build/build/nr-uesoftmodem`.

To run the gNB, open another terminal and execute the following command:

sudo ./nr-softmodem -O ~/lab8/ran-conf/gnb.conf --gNBs.[0].min_rxtxtime 6 --rfsim

For the UE, use this command:

```
sudo ./nr-uesoftmodem -r 106 --numerology 1 --band 78 -C 3619200000 --rfsim -O ~/lab8/ran-conf/nr-ue.conf
```

These commands initialize the gNB and UE with specific configurations provided in the `gnb.conf` and `nr-ue.conf` files, located in the `~/lab8/ran-conf` directory.

- Task1: Deploy the 5G core network.
- Task2: See using the docker-ps command which containers have been created.
- Task3: Get the logs of the amf container
- Task4: Deploy the 5G RAN
- Task5: Deploy the NR-UE

Check if the UE has got an ip from the 5GCN by performing ifconfig.

Test deployment using the ping command:
ping google.com -I oaitun_ue1

Answer the following questions:
- What do you observe at the logs of the AMF?
- What is the gNB ID?
- What is the IMSI of the UE?

Exercise 2. Docker-Compose deployment of 5GCN and BM deployment of 5G RAN and FlexRIC

Keeping the same architecture and implementations as in Exercise 1, in this exercise we will deploy an additional component named FlexRIC.

FlexRIC is an open-source software development kit (SDK) designed to facilitate the creation of specialized, service-oriented controllers for 5G Radio Access Networks (RAN). It offers a modular architecture with a minimal footprint, emphasizing flexibility and extensibility to accommodate diverse 5G use cases. FlexRIC supports multi-Radio

Access Technology (RAT) and multi-tenant scenarios, enabling the development of customized RAN Intelligent Controllers (RICs) that can monitor and control RAN elements in real-time. By providing a platform for implementing various service models and integrating with existing RAN infrastructures, FlexRIC serves as a valuable tool for researchers and developers aiming to advance software-defined RAN technologies.

Near-Real-Time RIC (Near-RT RIC)

The Near-Real-Time RIC (Near-RT RIC) is a key component within the O-RAN architecture, designed to manage and optimize 5G Radio Access Network (RAN) operations with a response time in the range of 10 milliseconds to 1 second. Operating at the edge of the RAN, it enables real-time control and rapid decision-making based on data collected from network elements, improving resource allocation, load balancing, and user experience. The Near-RT RIC's main function is to host xApps, which can act on this data to perform targeted optimizations, making the RAN more responsive and adaptable to changing conditions. By supporting open interfaces, the Near-RT RIC allows operators to integrate different vendors' hardware and software, thus fostering interoperability and network flexibility. Its modular design also enables operators to adjust their network dynamically, deploying or updating xApps as needed without altering the underlying RAN infrastructure.

xApps

xApps are modular applications that run on the Near-RT RIC, providing specialized, programmable functionalities for real-time RAN management and optimization. Designed to interact closely with the RAN, xApps enable custom solutions for tasks such as interference mitigation, traffic prediction, load balancing, and KPI monitoring. Each xApp is developed to address specific network challenges and can be independently deployed, managed, and updated. This flexibility allows operators to tailor RAN behavior to network demands and user needs, leveraging real-time data provided by the Near-RT RIC. Many xApps incorporate advanced algorithms, including machine learning, to improve the accuracy of network adjustments, predicting and responding to potential issues proactively. The modular nature of xApps not only enhances network efficiency but also fosters innovation, allowing

developers to create new applications that can seamlessly integrate with existing RAN operations.

To deploy the 5G CN and RAN, simply follow the same procedure as before; however, this time, use a different configuration file for the gNB, located at lab8/flexric-conf/gnb.conf.

When the UE has successfully connected to the 5G network, and you are able to transmit packets, deploy the near RT RIC:

```
cd ~/flexric
./build/examples/ric/nearRT-RIC
```

Once this is deployed and successfully connected with the gNB, execute the xAPP using this command in another terminal:

```
cd ~/flexric
./build/examples/xApp/c/monitor/xapp_kpm_moni
```

Answer the following questions:

- What is the functionality of the xAPP?
- What is the differences between the configuration file located at lab8/gNB-conf/gnb.conf and lab8/flexric-conf/gnb.conf
- Generate traffic and observe how the xAPP reports change.
- Navigate to the source code of the xApp at /home/cell/flexric/examples/xApp/c/monitor/xapp_kpm_moni.c and be prepared to modify it to implement the functionality that will be discussed during the lab session. Compile it using this command and check the corrected functionality:
  sudo ~/flexric/build/examples/xApp/c/monitor/make