Test Report: 5G UE Registration, PDU Session Establishment, and De-registration

1 Objective

The primary objective of this test was to evaluate the performance of the 5G network in handling **registration (reg)**, **PDU session establishment (pdu)**, and **de-registration (unreg)** operations for User Equipment (UEs) across multiple **gNBs** running on a low-end server with 16 GB of RAM. The test focused on assessing the success and failure rates of these operations, as well as identifying any system limitations that may arise with increasing traffic.

2 Test Setup

The testing environment consisted of the following components:

- **User Equipment (UEs)**: Simulated using a script to send reg, pdu, and unreg requests.
- **gNBs (gNodeB)**: Deployed on multiple ports ranging from 1200 to 1230, each gNB handling a specific number of UEs.
- **LoxiLB**: Load balancing between AMF instances.
- **Server Configuration**: Host system with 16 GB RAM, gNB ports (1200-1230) for handling simultaneous requests, and UE count varying from 10 to 80 UEs per port.

3 Test Methodology

This test was designed as a **randomness test**, where the ports and UE sequences were changed randomly to simulate real-world traffic patterns and conditions. The aim was to evaluate the performance of the 5G network in handling **registration (reg)**, **PDU session establishment (pdu)**, and **de-registration (unreg)** operations for User Equipment (UEs) across multiple gNBs running on different ports.

For each gNB port, a set of requests for **registration**, **PDU session establishment**, and **de-registration** was sent:

- **Registration & PDU Session Establishment**: UEs sent both reg and pdu requests to the gNBs. The number of requests varied per port, ranging from 10 to 80 UEs. The IMSI values for the UEs were generated dynamically, simulating each UE's unique action.

- **De-registration**: After completing the reg/pdu operations, unreg requests were sent to each port to simulate the removal of UEs from the network.

The time taken to process each set of requests was recorded, with each set of operations for a port taking between 30 seconds and 2 minutes, depending on the number of UEs.

4 Test Results

The test results are summarized below, showing the success and failure rates for **registration**, **PDU session establishment**, and **de-registration** at different gNB ports. The randomness test involved changing both the **ports** and **UE sequences** randomly for each run to simulate real-world network conditions.

4.1 Registration & PDU Session Establishment Results

gNB Port	Requests Sent	Registration Fail Rate	PDU Fail Rate
1200	10	10%	0%
1201	20	5%	10%
1210	40	7.5%	0%
1220	80	0%	81.25%

4.2 De-registration Results

gNB Port	Requests Sent	Unreg Fail Rate
1200	10	10%
1201	20	5%
1210	40	5%
1220	80	0%

5 Performance Metrics

Metric	Value	Calculation
Total Requests Processed	250 requests	Total of all requests sent across all gNB ports
Total Failures	106 failures	Total number of failures across all operations

Failure Rates	Rate	Calculation
Registration Failure Rate	6.25%	14 out of 225
PDU Failure Rate	29.6%	65 out of 220
Unregistration Failure Rate	3.75%	4 out of 106

True Positive Rate (TPR)	Rate	Calculation
TPR for Registration	93.8%	211 out of 225 successful registrations
TPR for PDU Sessions	70.5%	155 out of 220 successful PDU sessions

True Positive Rate (TPR)	Rate	Calculation
TPR for De-registration	96.2%	102 out of 106 successful unreg requests

6 Conclusion

The test results show that the **registration** and **de-registration** operations performed well, with relatively low failure rates. The **PDU session establishment** (pdu) showed considerable failure at higher load, especially at **port 1220**, where **81.25%** of the PDU requests failed.

The **randomness test**—where ports and UE sequences were changed randomly—helped simulate more dynamic and unpredictable network conditions, providing a realistic view of how the system behaves under varying conditions.

Key Conclusions:

- **Registration** and **de-registration** operations were relatively successful, with low failure rates, particularly at higher traffic levels.
- **PDU Session Establishment** needs further investigation, as the failure rate increased significantly at higher gNB loads.
- The **de-registration** process showed high reliability, with only a few failures observed, even at higher UE counts (up to 80).
- The overall system is capable of handling up to 40 UEs per gNB port without significant issues, but performance degrades beyond this load, especially for **PDU session establishment**.

Further testing with larger-scale configurations and performance optimization is recommended, particularly focusing on improving the reliability of **PDU sessions** under load. Resource tuning in Kubernetes and load balancing adjustments may help mitigate some of the observed failures in the PDU session establishment phase.

7 Test Scripts Used

Below are the scripts used during the testing process. Each script served a specific purpose in simulating the UE actions and controlling the network operations.

7.1 SKEL.txt

This script contains the template for UE configuration. It includes essential information such as the IMSI, encryption keys, and network parameters, which are dynamically replaced by the **gen-file.sh** script to create individual UE configuration files.

```
{
  "ipversion": "0",
  "use_ngap": "0",
  "ue_enb_id": "0",
  "ue_mme_id": "0",
  "e_rab_id": "0",
```

```
  "msip": "",
  "gwip": "",
  "cause_choice": "-1",
  "cause_number": "-1",
  "attach_complete": "0",
  "eps_id": "1",
  "usim": {
        "mcc": "999",
        "mnc": "99",
        "routing_indicator": "1",
        "uplinknascount": "1",
        "downlinknascount": "1",
        "SQN": "7136",
        "msin": "SKEL",
        "K": "000102030405060708090A0B0C0D0E0F",
        "CK": "",
        "IK": "",
        "oletusarvoinen_O_pPpPP_c": "BC2BCE2A23BE2FAE32E4F1B4546004F7",
        "OPc": "BC2BCE2A23BE2FAE32E4F1B4546004F7",
        "Oo Pee is":  "bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb"
  }
}
```

7.2 gen-file.sh

This script generates multiple UE configuration files by replacing the **SKEL** placeholder in the SKEL.txt template with unique IMSI values. It accepts input for the number of UEs and their starting IMSI number.

```
#!/bin/bash
# Prompt for starting IMSI with a default value
read -p "Enter starting IMSI [default: 1000000001]: " START_IMSI
START_IMSI=${START_IMSI:-1000000001}

# Prompt for UE count with a default value
read -p "Enter number of UEs to generate [default: 10]: " UE_COUNT
UE_COUNT=${UE_COUNT:-10}

SKEL_TEMPLATE="SKEL.txt"

for ((i=0; i<UE_COUNT; i++)); do
    IMSI=$((START_IMSI + i))

    # Create the output file name in the format 99999IMSI.txt
    OUTPUT_FILE="./99999${IMSI}.txt"

    # Replace "SKEL" in the template file with IMSI and create the new file
    sed "s/SKEL/$IMSI/g" "$SKEL_TEMPLATE" > "$OUTPUT_FILE"
```

```
    echo "Generated file: $OUTPUT_FILE"
done
```

7.3 gnb.sh

This script is responsible for starting multiple gNB instances on different ports (1200-1230) and logging their outputs. It also cleans up any old logs before starting the gNB instances.

```
#!/bin/bash
GNB_IP="10.10.3.219"
LOXILB_IP="10.10.3.215"
LOXILB_PORT="38412"
GNB_START_PORT=1200
GNB_COUNT=$1
LOG_DIR="./logs"

if ! [[ "$GNB_COUNT" =~ ^[0-9]+$ ]]; then
    echo "Usage: $0 <gnb_count>"
    exit 1
fi

echo "Cleaning old logs and UE state..."
rm -rf ./gnb/*
mkdir -p "$LOG_DIR"
rm -f "$LOG_DIR"/gnb-*.log

echo "Killing any existing gNB processes..."
pkill -f "./run/bin/cmd" 2>/dev/null

for ((i=1; i<=GNB_COUNT; i++)); do
    PORT=$((GNB_START_PORT + i - 1))
    LOG_FILE="$LOG_DIR/gnb-${i}.log"

    echo "Starting gNB-$i on port $PORT... Logging to $LOG_FILE"
    ./run/bin/cmd ./run/config_gnb.json -v -t -u "$PORT" -e "$GNB_IP" -g -n "$LOXILB_

    if [ $? -eq 0 ]; then
        echo "gNB-$i started successfully."
    else
        echo "Failed to start gNB-$i."
    fi
done

echo "All gNBs started in background. Use 'ps' or check '$LOG_DIR/' for logs."
```

7.4 all.sh

This script is used to simulate the registration, PDU session establishment, and de-registration processes for a specified number of UEs. It can be configured to run these operations in parallel on the specified gNB port.

```
#!/bin/bash
# Configuration
GNB_IP="10.10.3.219"
UE_SRC_IP="10.10.3.218"

# Prompt for required inputs
echo "Select operation:"
echo "  1) reg     - Registration"
echo "  2) pdu     - PDU Session Establishment"
echo "  3) unreg   - De-registration"
read -p "Enter option (reg/pdu/unreg): " ACTION

read -p "Enter UE count (range): " UE_COUNT
read -p "Enter starting IMSI (default: 1000000001): " START_IMSI
START_IMSI=${START_IMSI:-1000000001}

read -p "Enter GNB port (default: 1200): " GNB_PORT
GNB_PORT=${GNB_PORT:-1200}

# Validate inputs
if ! [[ "$UE_COUNT" =~ ^[0-9]+$ ]]; then
    echo "Error: UE count must be a number."
    exit 1
fi

if ! [[ "$GNB_PORT" =~ ^[0-9]+$ ]]; then
    echo "Error: GNB port must be a number."
    exit 1
fi

# Map action to command value
case "$ACTION" in
    reg)
        CMD=1
        ;;
    pdu)
        CMD=2
        ;;
    unreg)
        CMD=3
        ;;
    *)
        echo "Invalid action. Use reg, pdu, or unreg."
```

```
        exit 1
        ;;
esac

# Background function for execution
run_ue_action() {
    echo "Starting $ACTION for $UE_COUNT UEs from IMSI $START_IMSI using GNB port $GN

    for ((i=0; i<UE_COUNT; i++)); do
        IMSI=$((START_IMSI + i))
        echo "Processing IMSI: $IMSI"

        ./bin/ue -m "$IMSI" -g -y 999 -u 99 -c "$CMD" -v -v -t -s "$UE_SRC_IP" -n "$G

    done

    echo "Completed $ACTION for $UE_COUNT UEs."
}

# Run in background
run_ue_action &

PID=$!
echo "$ACTION job started in background with PID $PID."
```