國立台灣科技大學電子工程系

Taiwan Tech Department of Electronic and Computer Engineering

114 學年度第一學期實務專題

114 Year of First Fall/Spring Semester Special Project

期中進度報告

Mid-term Report

專題題目

OAI 與 NTN 衛星通訊之整合

組　　別：1141B30

組　　員：Team Member

姓名 Name：黃　又　儀 學號 Student ID：B11202220

姓名 Name：陳 黃 承 諭 學號 Student ID：B11202222

姓名 Name：吳　亭　寬 學號 Student ID：B11202116

指導老師：鄭　瑞　光 教 授

中　華　民　國　1　1　4　年　1　2　月　2　0　日

題目 Subject：OAI 與 NTN 衛星通訊之整合

組員姓名及學號 Member name and student ID：

姓名 Name：黃　又　儀 學號 Student ID：B11202220

姓名 Name：陳 黃 承 諭 學號 Student ID：B11202222

姓名 Name：吳　亭　寬 學號 Student ID：B11202116

組別 Team Tier：1141B30

指導老師：鄭　瑞　光 教　授

I.　摘要

隨著第五代行動通訊（5G）的迅速發展，全球對於高速率、低延遲及廣域連線的需求不斷提升。現有的地面網路架構雖能支援多數應用，但在偏遠地區、海洋航線、空中交通及災害現場等環境中，仍難以提供穩定的連線品質。為達成全球無縫覆蓋的通訊目標，非地面網路（Non-Terrestrial Network, NTN）逐漸成為 5G 與未來 6G 網路的重要發展方向。其中，低軌衛星（Low Earth Orbit, LEO）因具備低延遲與高傳輸效率的特性，成為實現廣域連線的關鍵技術。

本研究針對目前學界與業界主流的 Xeoverse、SNS3 (Satellite Network Simulator 3) 以及 OAI (OpenAirInterface) NTN 三種模擬平台，從模擬器安裝測試開始，逐步分析模擬平台的軟體功能架構，並探討支援的物理層 (OAI) 及網路層 (Xeoverse,SNS3)模擬能力，以及程式模組與規格的對應等不同角度，進行分析與研究。

II. 簡介

本專題旨在研究並建構一套多層次的非地面網路（Non-Terrestrial Network, NTN）模擬與驗證環境，透過整合三種不同特性的模擬器——OpenAirInterface (OAI)、SNS3 以及 Xeoverse (XEO)，全面分析 5G 衛星通訊在協議層、鏈路層及大規模網路層的行為與限制。

第一部分(OAI)根據 3GPP 定義[1][2]建立非地面網路（NTN）之端對端（End-to-End, E2E)測試環境，並研究 NTN 的問題及解決方法。非地面網路(NTN)之端對端(E2E)環境，User Equipment(UE)端使用 OAI 的 nr-uesoftmodem 作為軟體 UE，包含 RRC、PDCP、RLC、MAC 與 PHY 等 NR 協定層，負責將上層 IP 封包封裝為 5G NR 無線傳輸的數據。RF-Simulator 置於 UE 與 5G 基站 next-generation NodeB（gNB） 之間，作為軟體通道模擬器。透過調整 RF-Simulator 的參數，可以控制 NTN 通道條件，觀察 OAI gNB/UE 在不同延遲與 Doppler 下的行為。gNB 端使用 OAI 的 nr-softmodem，提供 5G NR 基站功能。在 gNB 後方部署 OAI 5G Core（AMF、SMF、UPF 等），完成 UE 附著與 PDU Session 建立。

第二部分 SNS3（ns-3），目標是 建立一套可重現、可量測、可切換參數的衛星網路模擬環境，用來分析不同流量 型態與網路設定下的傳輸效能，並作為後續專題整體整合與比較的基礎。 在 SNS3 部分，我們以衛星通訊模組提供的範例程式為起點，建立基本拓樸與鏈 路配置，並透過指令列參數控制模擬條件（例如：選定 beam、設定模擬時間、 切換 traffic model、切換測試情境）。模擬過程中，系統會自動輸出吞吐量、 延遲、封包錯誤率、負載等統計資料，讓我們能用一致的量測方式建立 baseline， 並進一步做參數掃描與情境比較。

第三部分以 Xeoverse（XEO）建構用於非地面網路（Non-Terrestrial Network, NTN）研究之網路層模擬與測試環境，聚焦於大規模低軌道（Low Earth Orbit, LEO） 衛星星座之拓樸建模與端對端（End-to-End, E2E）資料傳輸行為驗證。相較於直 接於實體衛星或完整 5G 系統上進行實驗，XEO 可透過軟體方式建立具可重現性 與可擴充性的模擬平台，作為 NTN 研究之前期驗證工具。

截至目前為止，各模組之進度如下：

1. **OAI 部分**：已成功架設 5GCN 並建立地面網路端對端基線，完成未加入 NTN 通道之 RTT 與 iperf3 吞吐量量測 。同時已演示在未更改協定參數情況下，直接加入 NTN 高延遲通道會導致連線失敗之結果 。
2. **SNS3 部分**：已完成模擬環境建置與可執行性驗證，成功執行 **RTN (Return Link)** 系統測試範例 。已建立可重現的執行流程，確保未來可快速進行參數掃描與對照 。
3. **Xeoverse 部分**：成功建置 Mininet 模擬環境並完成基本網路拓樸設定 。在未加入額外延遲或丟包條件下，已完成端對端連線驗證（ping）與 TCP 傳輸吞吐量量測，作為後續加入衛星效應後之對照基準 。

III. 環境建置與部署

A. 系統需求分析

1. OAI 部分:

此部分在一台安裝 VMware Workstation 的 Windows 11 主機上進行，於 VMware 中建立一台虛擬機作為實驗環境，虛擬機作業系統為 Ubuntu 22.04.3 LTS，配置 8vCPU 與 16 GB RAM，作為 OpenAirInterface 及 OAI 5GCN 的執行平台。無線接取網路部分採用 OpenAirInterface 5G New Radio (NR) Radio Access Network (RAN)，版本為 tag 2025.w46（commit 92980ceb72，detached HEAD）。核心網路則採用 OAI 5GCN Docker，AMF/SMF/UPF 版本分別為 AMF：oaisoftwarealliance/oai-amf:develop（image ID 8cd127976ea6）/ SMF：oaisoftwarealliance/oai-smf:develop（image ID 8cdb93e00a1b）/ UPF：oaisoftwarealliance/oai-upf:develop（image ID 4eb612eda6fa）
主要相依軟體如下：
版本控制與除錯工具：git、wireshark。
容器環境：Docker CE 與 Docker Compose（執行 OAI 5GCN）。
開發與編譯工具：build-essential（gcc/g++）、cmake。
測試工具：iperf3（吞吐量量測）、ping（RTT 量測）。
其他依賴套件：依 OAI 官方文件安裝所需開發函式庫與 python3。

2. SNS3 部分:

此部分在 win11 電腦上安裝 VirtualBox 虛擬機環境建置 SNS3（ns-3 satellite） 模擬平台。整體環境包含作業系統、編譯工具鏈、版本控制與資料分析工具，並 以固定版本（commit/tag）確保模擬結果具可重現性。

- 作業系統（OS）：Ubuntu 22.04 LTS
- CPU / RAM 建議：8 vCPUs、16 GB RAM
- ns3 版本：ns-3.43
- 版本控制：git（用於取得 ns-3 與 satellite 模組程式碼並鎖定 commit）
- 編譯工具：g++ / gcc、python3、waf（ns-3 建置系統）
- 除錯與觀察：可搭配 pcap 輸出、log（NS_LOG）與 trace callback

3. Xeoverse 部分:

此部分之模擬環境建置於 Linux 作業系統下，主要用於執行 Xeoverse（XEO） 與 Mininet 虛擬網路模擬。系統需具備基本之 Python 執行環境，

並安裝相關 套件以支援星座建模、拓樸計算與虛擬網路部署，虛擬機作業系統為 Ubuntu 20.04.6 LTS，配置 16 GB RAM，用以執行 Xeoverse 與 Mininet 相關模擬作 業。整體系統以 Linux 作業系統為基礎，作為非地面網路之網路層與傳輸層 模擬平台。 Xeoverse（XEO）主要執行於 Python 環境下，系統需具備 Python 3 與相關相 依套件，以支援星座幾何計算、網路拓樸建立與路由預計算等功能。虛擬網 路執行部分則採用 Mininet，其依賴 Linux Kernel 所提供之 network namespace 與虛擬網路介面機制，以建立虛擬 Host、Switch 與 Link，並支援 IP/TCP 封 包之實際傳輸。 測試與量測工具方面，使用 ping 進行端對端封包往返時間（RTT）量測，並 透過 iperf（v2）進行端對端吞吐量測試，以驗證虛擬網路環境中之基本傳輸 行為。整體環境配置以支援網路層與傳輸層模擬為主要目標，未涉及實體無 線裝置或物理層相關需求。

B. 安裝流程步驟

1. OAI 部分:

以下會以清單形式整理本研究的安裝與部署流程與相對應之關鍵程式碼，包含 5GCN[3]及 OAI RAN（gNB / UE）[4]之設定。詳細指令與完整安裝可參考專案 GitHub 紀錄(5GCN) [5] (OAI RAN) [6]。

a. 5GCN

更新套件庫與基本工具、Docker 相關套件

```
sudo apt update
sudo apt install -y git net-tools putty \ca-certificates curl gnupg \
docker-ce docker-ce-cli containerd.io \
docker-buildx-plugin docker-compose-plugin
```

下載並解壓 OAI CN5G 教學資源，拉取 CN5G Docker 映像檔

```
cd ~
wget -O oai-cn5g.zip <OAI 官方教學壓縮檔網址>
unzip oai-cn5g.zip
cd ~/oai-cn5g
docker compose pull
```

*詳細程式碼在專案 GitHub [5]

b. OAI RAN（gNB / UE）

安裝相依套件與編譯

```
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
git checkout 2025.w46 # tested tag
./build_oai --ninja -I                    # install dependencies
```

./build_oai --ninja --gNB --nrUE -w SIMU -c # compile gNB and nrUE

*詳細程式碼在專案 GitHub [6]

2. SNS3 部分:

以下會以清單形式整理本研究的安裝與部署流程與相對應之關鍵程式碼，包含 ns-3.43[7]之設定。詳細指令與完整安裝可參考專案 GitHub 紀錄 (ns-3.43)[8]。

安裝依賴

sudo apt update && sudo apt install -y git automake cmake qtbase5-dev python3 dev python3-pip libxml2-dev mercurial graphviz libgraphviz-dev

sudo pip install --upgrade pip setuptools cppyy pygraphviz

下載 bake

mkdir -p ~/workspace && cd ~/workspace && git clone https://gitlab.com/nsnam/bake.git

設定 bake 環境變數

export BAKE_HOME=`pwd`

export PATH=$PATH:$BAKE_HOME/build/bin

export PYTHONPATH=$BAKE_HOME/build/lib

export LD_LIBRARY_PATH=$BAKE_HOME/build/lib

d bake

./bake.py configure -e ns-allinone-3.43

./bake.py check

bake 安裝 ns-allinone-3.43

./bake.py download

*詳細程式碼在專案 GitHub [8]

3. Xeoverse 部分:

以下會以清單形式整理本研究的安裝與部署流程與相對應之關鍵程式碼，詳細指令與完整安裝可參考專案 GitHub 紀錄[9]

更新系統套件並安裝基本開發與網路工具

sudo apt update

sudo apt install -y git net-tools curl wget build-essential python3 python3-pip

Python 套件與相依環境設定

pip3 install numpy scipy networkx pyyaml matplotlib

Xeoverse 專案取得與環境準備

cd ~

git clone https://github.com/raycg/xeoverse

cd xeoverse

Mininet 安裝與設定

安裝 Mininet 套件，作為虛擬網路模擬平台

sudo apt install -y mininet

安裝完成後，可透過 Mininet 內建測試指令確認虛擬網路功能正常

sudo mn --test pingall

*詳細程式碼在專案 GitHub [9]

C. 部署驗證

1. OAI 部分:

E2E 測試流程(地面 baseline 無 NTN)

1. 啟動 5GCN

```
docker compose up -d
```

2. 啟動 gNB

```
./nr-softmodem -O ~/oai-workshops/ran/conf/gnb.sa.band78.106prb.rfsim.conf --rfsim
```

3. 啟動 UE

```
./nr-uesoftmodem -C 3619200000 -r 106 --numerology 1 --ssb 516 -O ~/oai-workshops/ran/conf/ue.conf --rfsim
```

4. 在 UE 端 namespace 裡 ping 5GCN 後面的 server

```
ping -I oaitun_ue1 192.168.70.135
```

```
docker exec -it oai-ext-dn ping <UE IP address>
```

5. iperf3 測試 throughput

在容器內開啟 iperf3 server :

```
docker exec -it oai-ext-dn bash -c "iperf3 -s"
```

在新的終端機跑 iperf3 client :

```
iperf3 -B <UE IP ADDRESS> -c 192.168.70.135 -u -b 50M -R # DL
iperf3 -B <UE IP ADDRESS> -c 192.168.70.135 -u -b 20M    # UL
```

結果展示：圖 1 與圖 2 分別是步驟 4 的執行結果，圖 1 是 UL、圖 2 是 DL，測量的數據結果在表一，圖 3 與圖 4 是步驟 5 的執行結果，圖 3 為 server 端、圖 4 為 client 端，測量的數據結果在表二，表一為 RTT 測量結果表紀錄 UL 與 DL 之 RTT_min/avg/max，表二為吞吐量測試表紀錄 DL 的吞吐量/Jitter/Loss。

```
kuan@kuan-virtual-machine:~/openairinterface5g/cmake_targets/ran_build/build$ ping -I oaitun_ue1 192.168.70.135
PING 192.168.70.135 (192.168.70.135) from 10.0.0.2 oaitun_ue1: 56(84) bytes of data.
64 bytes from 192.168.70.135: icmp_seq=1 ttl=63 time=18.2 ms
64 bytes from 192.168.70.135: icmp_seq=2 ttl=63 time=18.2 ms
64 bytes from 192.168.70.135: icmp_seq=3 ttl=63 time=4.73 ms
64 bytes from 192.168.70.135: icmp_seq=4 ttl=63 time=15.4 ms
64 bytes from 192.168.70.135: icmp_seq=5 ttl=63 time=15.2 ms
64 bytes from 192.168.70.135: icmp_seq=6 ttl=63 time=16.4 ms
64 bytes from 192.168.70.135: icmp_seq=7 ttl=63 time=17.8 ms
64 bytes from 192.168.70.135: icmp_seq=8 ttl=63 time=17.6 ms
64 bytes from 192.168.70.135: icmp_seq=9 ttl=63 time=5.29 ms
64 bytes from 192.168.70.135: icmp_seq=10 ttl=63 time=17.9 ms
64 bytes from 192.168.70.135: icmp_seq=11 ttl=63 time=15.2 ms
64 bytes from 192.168.70.135: icmp_seq=12 ttl=63 time=16.3 ms
64 bytes from 192.168.70.135: icmp_seq=13 ttl=63 time=17.5 ms
64 bytes from 192.168.70.135: icmp_seq=14 ttl=63 time=16.7 ms
^C
--- 192.168.70.135 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13019ms
rtt min/avg/max/mdev = 4.730/15.164/18.208/4.272 ms
```

圖 1. UE → ext-dn server

```
kuan@kuan-virtual-machine:~/openairinterface5g/cmake_targets/ran_build/build$ docker exec -it oai-ext-dn ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=63 time=7.39 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=63 time=17.5 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=63 time=17.0 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=63 time=20.3 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=63 time=15.5 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=63 time=18.3 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=63 time=16.1 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=63 time=16.0 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=63 time=16.3 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=63 time=15.4 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=63 time=23.3 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=63 time=18.5 ms
64 bytes from 10.0.0.2: icmp_seq=13 ttl=63 time=14.8 ms
64 bytes from 10.0.0.2: icmp_seq=14 ttl=63 time=16.2 ms
64 bytes from 10.0.0.2: icmp_seq=15 ttl=63 time=18.5 ms
64 bytes from 10.0.0.2: icmp_seq=16 ttl=63 time=18.0 ms
^C
--- 10.0.0.2 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15019ms
rtt min/avg/max/mdev = 7.389/16.818/23.348/3.189 ms
```

圖 2. ext-dn server → UE

```
kuan@kuan-virtual-machine:~/openairinterface5g/cmake_targets/ran_build/build$ docker exec -it oai-ext-dn bash -c "iperf3 -s"
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

圖 3. server

```
kuan@kuan-virtual-machine:~/openairinterface5g/cmake_targets/ran_build/build$ iperf3 -B 10.0.0.2 -c 192.168.70.135 -u -b 50M
Connecting to host 192.168.70.135, port 5201
[  5] local 10.0.0.2 port 56029 connected to 192.168.70.135 port 5201
[ ID] Interval           Transfer     Bitrate         Total Datagrams
[  5]   0.00-1.00   sec  5.96 MBytes  50.0 Mbits/sec  4313
[  5]   1.00-2.00   sec  5.96 MBytes  50.0 Mbits/sec  4319
[  5]   2.00-3.00   sec  5.96 MBytes  50.0 Mbits/sec  4315
[  5]   3.00-4.00   sec  5.96 MBytes  50.0 Mbits/sec  4316
[  5]   4.00-5.00   sec  5.96 MBytes  50.0 Mbits/sec  4316
[  5]   5.00-6.00   sec  5.96 MBytes  50.0 Mbits/sec  4317
[  5]   6.00-7.00   sec  5.96 MBytes  50.0 Mbits/sec  4317
[  5]   7.00-8.00   sec  5.96 MBytes  50.0 Mbits/sec  4316
[  5]   8.00-9.00   sec  5.96 MBytes  50.0 Mbits/sec  4316
[  5]   9.00-10.00  sec  5.96 MBytes  50.0 Mbits/sec  4315
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Jitter    Lost/Total Datagrams
[  5]   0.00-10.00  sec  59.6 MBytes  50.0 Mbits/sec  0.000 ms  0/43160 (0%)  sender
[  5]   0.00-11.11  sec  59.6 MBytes  45.0 Mbits/sec  0.338 ms  0/43160 (0%)  receiver
```

圖 4. Client

方向	Source Destination	Loss	RTT_min	RTT_avg	RTT_max
UL	UE (10.0.0.2) to DN(192.168.70.135)	0%	4.73	15.16	18.21
DL	DN (oai-ext-dn) to UE (10.0.0.2)	0%	7.39	16.82	23.35

表一 RTT 測量結果表（ping）

方向	設定速率 (Mbit/s)	實測吞吐量 (sender)	實測吞吐量(receiver)	Jitter (ms)	Loss
DL	50	50.0 Mbit/s	45.0 Mbit/s	0.000 / 0.338	0 / 43160 (0%)

表二 吞吐量測試表(iperf3)

2. SNS3 部分:

sns3 測試流程

1. 下載 reference scenario 資料

cd contrib/satellite

git submodule update --init --recursive

2. 測試 sns3

cd ~/workspace/bake/source/ns-3.43

./test.py --no-build

3. 模擬內建程式

./ns3 run "ns3-program"

結果展示：本次程式碼執行 sns3 回傳鏈路(RTN)的模擬。

圖 5 為第一次執行時編譯器的輸出結果，圖 6 為執行結果。

圖 6 為 RTN（Return Link）排程器在分配回傳時槽（TBTP）時的輸出訊息。每一行的數字（115、130、144 Bytes）代表在該傳輸時槽中分配出去的 payload 大小。

這些訊息持續出現，表示排程器正在運作、衛星網路模擬正在「傳送回傳資料流」。此時模擬器正在模擬多個終端的回傳封包，TBTP 正在持續被配置。

```
kevin@kevin-VirtualBox:~/workspace/bake/source/ns-3.43$ ./ns3 run "sat-rtn-system-
test-example"
Consolidate compiler generated dependencies of target stdlib_pch_exec
Consolidate compiler generated dependencies of target stdlib_pch-optimized
Consolidate compiler generated dependencies of target core
Consolidate compiler generated dependencies of target config-store
Consolidate compiler generated dependencies of target antenna
Consolidate compiler generated dependencies of target stats
Consolidate compiler generated dependencies of target network
Consolidate compiler generated dependencies of target bridge
Consolidate compiler generated dependencies of target point-to-point
Consolidate compiler generated dependencies of target csma
Consolidate compiler generated dependencies of target mobility
Consolidate compiler generated dependencies of target traffic-control
Consolidate compiler generated dependencies of target propagation
Consolidate compiler generated dependencies of target internet
Consolidate compiler generated dependencies of target flow-monitor
Consolidate compiler generated dependencies of target applications
Consolidate compiler generated dependencies of target magister-stats
Consolidate compiler generated dependencies of target traffic
Consolidate compiler generated dependencies of target satellite
Consolidate compiler generated dependencies of target sat-rtn-system-test-example
```

圖 5.編譯結果

```
115 Bytes allocated within TBTP
115 Bytes allocated within TBTP
130 Bytes allocated within TBTP
130 Bytes allocated within TBTP
144 Bytes allocated within TBTP
144 Bytes allocated within TBTP
144 Bytes allocated within TBTP
130 Bytes allocated within TBTP
130 Bytes allocated within TBTP
130 Bytes allocated within TBTP
115 Bytes allocated within TBTP
115 Bytes allocated within TBTP
115 Bytes allocated within TBTP
130 Bytes allocated within TBTP
130 Bytes allocated within TBTP
130 Bytes allocated within TBTP
144 Bytes allocated within TBTP
130 Bytes allocated within TBTP
130 Bytes allocated within TBTP
115 Bytes allocated within TBTP
115 Bytes allocated within TBTP
115 Bytes allocated within TBTP
```

圖 6. RTN 執行結果

3. Xeoverse 部分:

1. 啟動 Mininet 虛擬網路環境

   sudo mn --test pingall

   *若所有虛擬節點皆可正常互相連線，則代表 Mininet 虛擬網路環境部署成功。

2. 端對端連線與 RTT 驗證

   量測封包往返時間

   mininet> h1 ping -c 3 h2

   吞吐量測試

   mininet> h2 iperf -s

   mininet> h1 iperf -c h2

結果展示：圖 7 顯示於 Mininet 虛擬網路環境中，透過 ping 指令進行端對端封包往返時間（Round-Trip Time, RTT）量測之結果。實驗中共傳送三個 ICMP 封包，皆成功接收，封包遺失率為 0%，其 RTT 分別為 5.90 ms、1.03 ms 與 0.841 ms，顯示在目前未加入額外延遲或丟包條件下，虛擬網路環境之基本連線與封包傳輸功能可正常運作，表三為 RTT 測量結果表紀錄。圖 8 則為於相同虛擬網路環境下，使用 iperf 工具進行 TCP 吞吐量測試之結果。測試時間區間為 10 秒，期間共成功傳輸 26.8 GBytes 之資料量，平均吞吐量約為 23.0 Gbits/sec，驗證 Mininet 所建立之虛擬網路可支援端對端 TCP 傳輸，並可作為後續效能比較與分析之基礎測試環境，表四為吞吐量測試表紀錄。

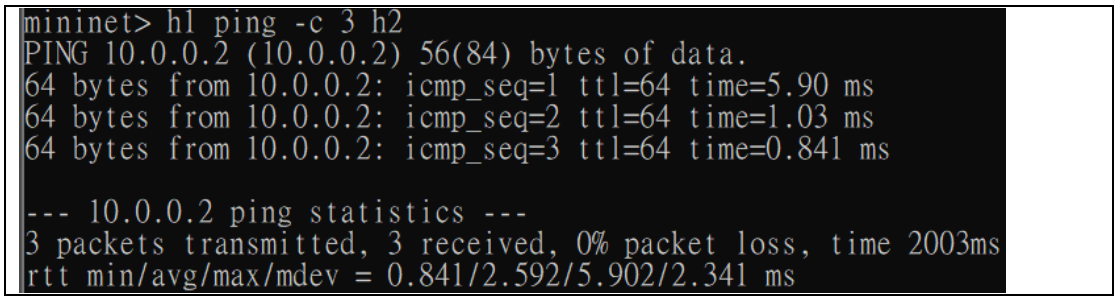

圖 7、使用 ping 測量工具

	RTT
封包一	5.90 ms
封包二	1.03 ms
封包三	0.841 ms

表三、RTT 測量結果表

*結果顯示傳送 3 個封包收到 3 個，no packet loss，測試時間=2003ms，詳細執行結果可參考[10]

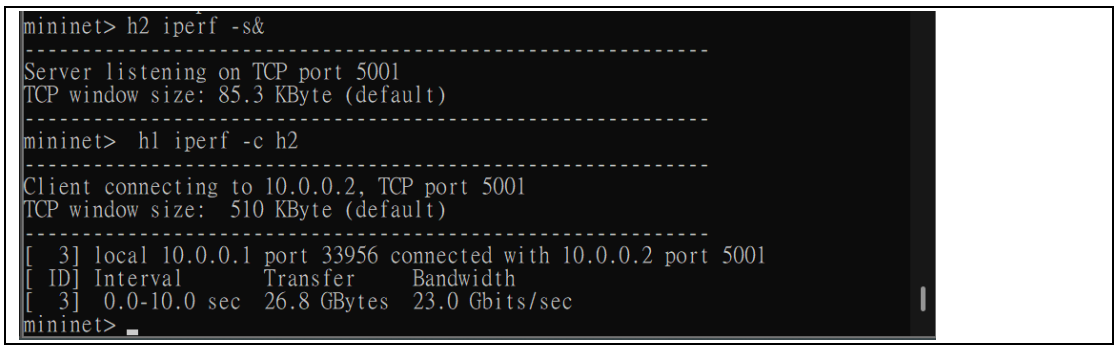

圖 8、使用測量工具 iperf

測試時間區間	總共成功傳送的資料量	平均吞吐量
0.0-10.0 sec	26.8 GBytes	23.0 Gbits/sec

表四、吞吐量測試表(iperf)

*詳細執行結果可參考[11]

IV. 軟體架構與模組設計

A. 整體架構概觀

1. OAI 部分:

此部分計畫旨在建立一套以 OpenAirInterface (OAI) 為核心的 非地面網路 (NTN) 端對端測試環境，有四個主要模組(UE、gNB、RFsimulator、5GCN)；在本研究中,將系統分成 Backstage 與 Main Stage 兩個部分:Backstage 包含 OAI UE、OAI gNB 與 5GCN，負責 5G NR 協定處理與控制訊號；Main Stage 則是 RF-Simulator 所實作的 LEO NTN 通道模型，用來即時產生延遲與 Doppler。

整體架構與流程圖如下圖 9 所示,本研究的端對端模擬架構分為四個部分：UE side、RF-Simulator(channel simulator)、OAI gNB(nr-softmodem)與 5G Core。UE side 由一台 VM 主機與 OAI nr-uesoftmodem 組成。主機上執行 ping 與 iperf3 client 等測試工具產生 user-plane IP 封包，透過 OAI UE 實作 NR 的協定堆疊,包括實體層（PHY）、媒體存取控制層（Medium Access Control, MAC）、無線鏈路控制層（Radio Link Control, RLC）、封包資料匯聚協定（Packet Data Convergence Protocol, PDCP）與 無線資源控制層（Radio Resource Control, RRC），並以 --rfsim 模式將封包轉換為 I/Q 樣本。RF-Simulator 置於 UE 與 gNB 之間,根據 channelmod 設定載入 SAT_LEO_TRANS 或 SAT_LEO_REGEN 通道模型，用來模擬 LEO NTN 環境中的長延遲、多普勒頻移以及路徑損耗。OAI gNB(softmodem)包含 NR gNB 之 RRC、PDCP、RLC、MAC 與 PHY。gNB 會依據 NTN 相關參數（例如 cellSpecificKoffset_r17 與 ta-Common-r17、(Hybrid

Automatic Repeat reQuest)HARQ 與 RLC 計時器）對 RF-Simulator 所產生的長延遲與 Doppler 進行補償，並透過 NG-C / NG-U 介面與 5GCN 溝通。最右側的 5G Core 包含 OAI 5GCN（AMF、SMF、UPF 等元件）以及一個測試伺服器，5GCN 負責完成 UE 的註冊與 PDU Session 建立，並將來自 gNB 的 GTP-U 流量轉發至測試伺服器，使本研究可以量測端到端的 RTT、吞吐量與丟包率。

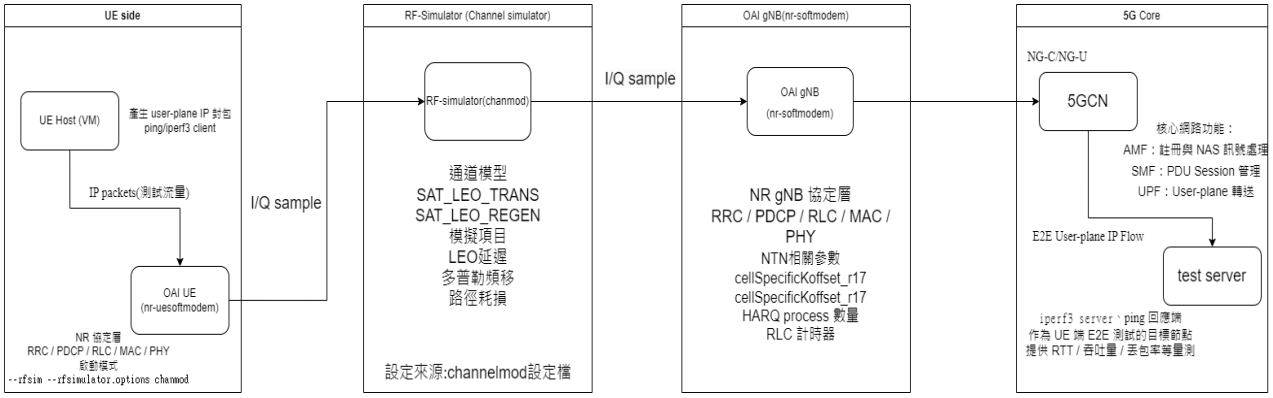

圖 9.本研究之 OAI NTN 端對端模擬架構。由左至右依序為 UE 端、RF-Simulator 通道模擬器、OAI gNB 以及 5G Core Network 與測試伺服器。

2. SNS3 部分:

此部分使用的 SNS3（ns-3 satellite）屬於 NS-3 的衛星通訊擴充模組，其軟體架構採用 NS-3 一貫的模組化設計：在 NS-3 核心（事件驅動模擬引擎、節點 / 網路堆疊）之上，加入衛星專用的 MAC/PHY 協議棧與通道模型，用來描述衛星系統在不同 frame/slot 資源、波束覆蓋、通道衰落與誤碼條件下的傳輸行為。SNS3 的設計特別強調 DVB-S2 / RCS2 相關機制與實體層特性，因此模擬結果不僅呈現封包傳遞，也能反映衛星鏈路的資源配置與傳輸限制。

如圖 10[12]所示，本圖呈現 SNS3（ns-3 satellite）在端到端衛星網路模擬中的 節點分層與資料傳遞路徑。整體而言，封包由左側 End user 的應用層產生後，先經地面接取網路送至 UT（User Terminal），再透過衛星鏈路傳送至 Satellite 與 GW（Gateway），最後抵達右側 End user。在高層次流程上，系統的主要運作可分為以下步驟：(I) 左側 End user 由 Application 產生資料並經 Transport/Network 處理後交由 CSMA 送入地面鏈路；(II) 封包透過 CSMA channel 傳送至 UT；(III) UT 端由 Network/CSMA 接收後，透過 SatNetDevice 將封包導入衛星鏈路；(IV) 封包經 SatChannel（UT↔Satellite、Satellite↔GW）完成衛星段傳輸，並由 Satellite 的 SatGeoNetDevice 進行中繼/轉送；(V) GW 端

由 SatNetDevice 接收後交由 Network，並透過 CSMA channel 傳送至右側 End user，最終交付至目的端的 Network/Transport/Application。

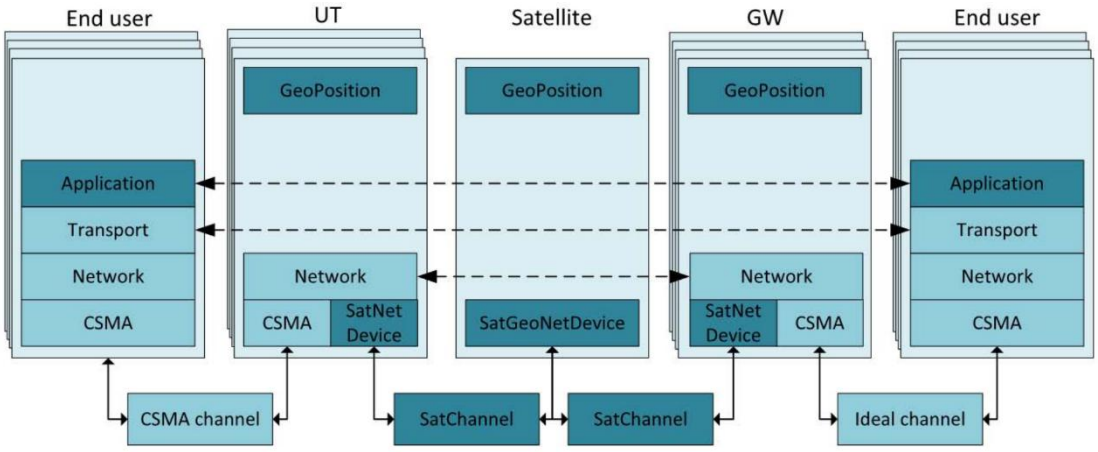

圖 10. Sns3 系統架構圖。由左至右依序為 UE 端、UT 端、衛星節點、GW 端以及 EU 端

1. Scenario / Example Layer（情境與範例層）：

   用於建立與執行模擬情境，將 實驗參數（例如情境設定、波束/節點配置、模擬時間、流量模式、統計輸出） 整合成可重現的模擬流程。通常透過各類 helper 將底層模組快速組裝起來， 並降低情境搭建成本。如圖 11[13]所示，本圖呈現 SNS3 在情境建立 （Scenario/Example Layer）時所使用的 Helper 階層結構；高層次流程可概 括為：(I) 使用者先透過 SatConf / SatFwdCarrierConf / SatSuperFrameSeq 設 定衛星系統的整體參數、前進鏈路載波配置，以及 superframe/frame 的時間 結構;(II) 接著由 SimulationHelper 統一管理模擬流程並呼叫 SatHelper 作 為核心組裝入口；(III) SatHelper 進一步交由 SatBeamHelper 建立與管理波 束相關設定與連結；(IV) 最後由各子 helper 分工完成節點與情境組裝： SatUtHelper 建立 UT 端節點與裝置、SatGwHelper 建立 GW 端節點與裝 置、SatGeoHelper 提供幾何/位置相關配置（例如 UT/GW/衛星位置或計算 所需資料）、SatUserHelper 建立端使用者與流量端點並完成應用掛載，完成 後即可啟動模擬並輸出結果。

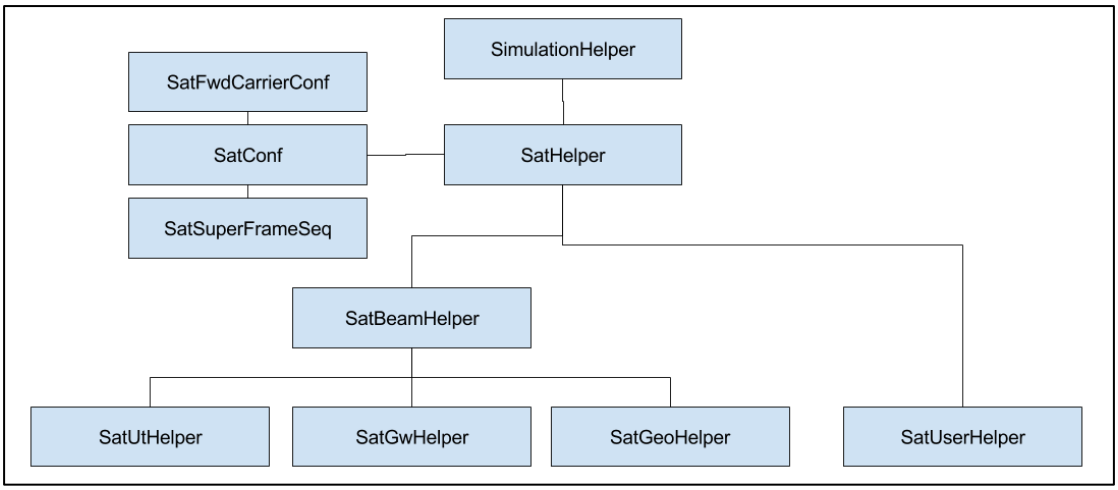

圖 11. Scenario / Example Layer（情境與範例層）

2. DVB-S2 / DVB-RCS2

SNS3 在設計上以 DVB-S2。（前進鏈路） 與 DVB-RCS2。（回傳鏈路） 的系 統概念作為建模參考，因此其時間結構與資源分配可用「前進以 frame 為主、回傳以 slot/burst 為主」來理解： 前進鏈路（Forward Link，對應 DVB-S2）：前進鏈路通常由 GW/衛星向 UT 傳送，重點在於以 Frame 組織下行傳輸；在 SNS3 中，這會反映為「以 frame 邊界驅動的傳輸組織與控制流程」， 並搭配通道模型（延遲、損耗、誤碼）決定接收端可達吞吐與錯誤率。換言 之，DVB-S2 的角色在於提供「下行以 frame 結構化傳輸」的建模視角，使 模擬能呈現下行鏈路在不同通道條件下的效能限制。 回傳鏈路（Return Link，對應 DVB-RCS2）： 回傳鏈路由 UT 上行至衛星/GW。，核心特是是「資源先分配、再傳輸」：UT 必須取得可用的 Slot（或等價的上行資源）才能在該 slot 內送出 burst/封 包。SNS3 因此特別強調 frame/slot 的時間結構，並在 slot 邊界觸發上行傳 輸事件，使模擬結果能反映回傳鏈路在排程/配置限制下的吞吐與延遲表現。

3. Frame/Slot

如圖 12 所示，本圖說明 SNS3 在衛星鏈路資源建模時所採用的階層式時間結構：Superframe → Frame → Slot。；高層次流程可概括為：(I) 模擬時間以 Superframe 為週期性重複的最大單位，用於組織整體衛星鏈路的時間骨架；(II) 每個 Superframe 由多個 Frame。（ frame 0…frame n） 依序構成，Frame 邊界是觸發排程/配置事件的主要時間點；(III) 每個 Frame 內部再細分為多個 Slot（slot 0…slot n），Slot 是回傳鏈路（Return Link）資源分配與上行 傳輸的基本單位；圖中「slot 1 copy 1 / slot 2 copy 2 / slot n copy 3」表 示同一類型的時槽可依情境設定在 Frame 中以多個 copy 形式重複出現， 用於表達

特定時槽配置在時間上可被多次安排（例如控制/資料或不同配置 段落的重複出現）。整體而言，此階層式結構使 SNS3 能以離散事件方式在 frame/slot 邊界觸發控制流程與資料傳輸，進而反映衛星系統「資源何時可 用、誰可用、可用多少」的行為。

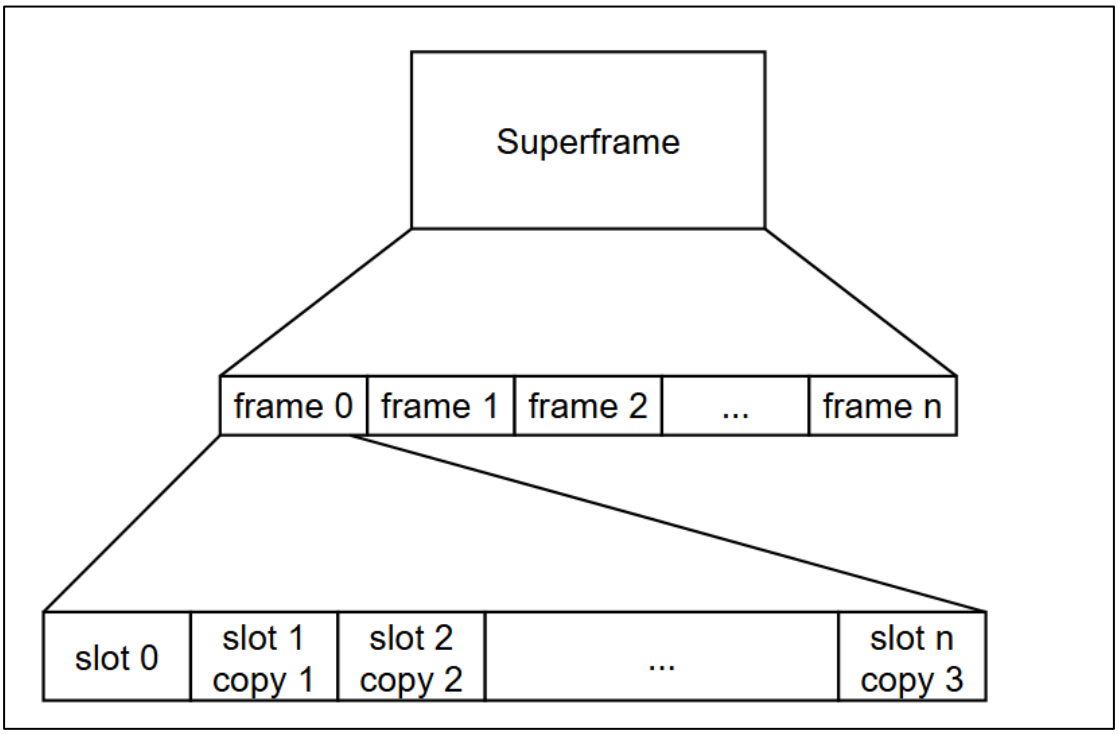

圖 12.frame/slot

3. Xeoverse 部分:

Xeoverse（XEO）整體系統架構可分為 Backstage 與 Mainstage 兩個階段，分別負責模擬資料的預計算與虛擬網路環境的實際執行。此種設計方式可將衛星網路之幾何與拓樸計算，與實際封包傳輸行為加以分離，提升系統架構的清晰度與可維護性。

1. Mainstage

主台階段負責將後台所產生之拓樸與路由資訊，導入 Mininet 虛擬網路模擬環境中，建立可實際執行之 IP 網路架構。透過 Mininet 提供之 Host、Switch 與 Link 機制，系統可在 Linux 環境中模擬端對端封包傳輸行為。

在此階段中，網路節點會被對應為 Mininet 中的虛擬 Host 與 Switch，並透過 IP/TCP 連線進行資料傳輸。使用者可於主台環境中執行 ping 與 iperf 等工具，量測封包往返時間（RTT）與吞吐量，以驗證網路層與傳輸層之行為。

2. Backstage

後台階段主要負責進行與衛星星座相關之預計算作業(pre-processing)，包含節點定義、幾何關係推導以及網路拓樸建立等流程。此階段不涉及實際封包傳輸，而是產生描述網路結構與連線關係之中介資料。
在後台中，系統會根據輸入之設定檔與星座參數，建立模擬所需的節點集合，並依據既定規則判斷節點之間是否存在連線關係，形成對應之網路拓樸，同時後台也會完成路徑計算相關的預處理，產生後續虛擬網路部署所需之拓樸與路由資訊。

圖 13 是 Xeoverse(XEO)模擬流程與量測架構圖，此架構可分為輸入(Input)、XEO Core 建立、Mininet 以及量測（Measurement）四個模組。其中，輸入模組（Input）負責讀入設定檔（config.yaml）與 TLE 資料，以定義衛星星座參數與整體模擬場景；XEO Core 建立模組則依據上述輸入進行網路層建模，包含星座節點建立（Constellation）、節點間連線關係生成（Topology）以及路徑計算（Routing），以完成網路拓樸與路由資訊之預先計算。接著，Mininet 模組負責將 XEO Core 所產生之網路拓樸與路由結果部署為可實際執行之虛擬網路環境，建立對應之虛擬主機（Hosts）、交換節點（Switches）與節點間連線（Links），並透過 Linux 核心的 IP/TCP 機制進行封包轉送。最後，量測模組（Measurement）於既定虛擬網路結構下，使用 ping 與 iperf 等工具進行端對端傳輸測試，取得封包往返時間（RTT）與吞吐量等效能指標。

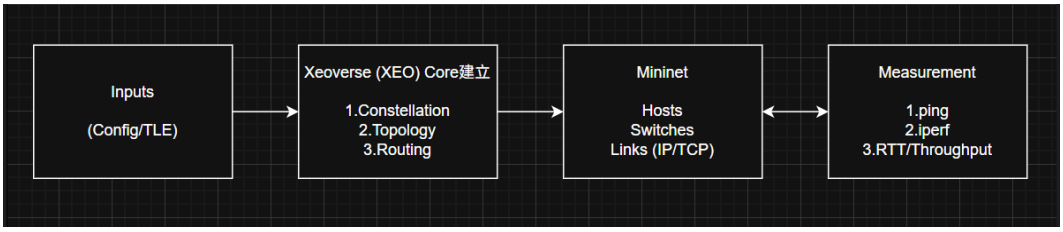

圖 13、系統架構圖

B. 關鍵模組說明

1. OAI 部分:

此部分計畫旨在建立一套以 OpenAirInterface (OAI) 為核心的非地面網路 (NTN) 端對端測試環境，有四個主要模組(UE、gNB、RFsimulator、5GCN) 如圖 14 顯示，以下會對這四個關鍵模組各自的功能與在本研究中的角色進行說明。

1. UE 模組模擬一支 5G NTN UE，包含 PHY、MAC、RLC、PDCP、RRC、NAS 等協定堆疊。實作上使用 cmake_targets/ran_build/build/nr-uesoftmodem

執行，搭配 ci-scripts/conf_files/nrue.uicc.ntn-leo.conf 作為 UE 端設定檔（包含 IMSI、金鑰與 DNN 等參數）。啟動時透過 --rfsim 選項，將 UE 端產生的 I/Q 樣本送往 RF-Simulator，並經由 RF 模擬通道再連到 gNB 與 5GCN。

2. RF-Simulator 衛星模擬通道取代真實的無線硬體，在 UE 與 gNB 之間接收並轉發 I/Q samples，同時加入通道模型。其主要程式位於 OAI 專案的 radio/rfsimulator/，通道參數則由設定檔中的 modellist_rfsimu_1 區段指定，例如 type="SAT_LEO_TRANS" 表示採用 LEO 透明衛星通道模型，並可調整 ploss_dB（路徑損耗）、noise_power_dB（雜訊功率）、offset（通道偏移）、ds_tdl（TDL delay spread）等參數，用來模擬不同 NTN 通道條件。
3. gNB 模組模擬一個 NTN gNB，實作 NR PHY/MAC/RLC/PDCP/RRC 協定，並對接 5GCN。實作上使用 cmake_targets/ran_build/build/nr-softmodem 執行，配合 ci-scripts/conf_files/gnb.sa.band254.u0.25prb.rfsim.ntn-leo.conf 作為設定檔。gNB 端的上行 / 下行 I/Q 都經由 RF-Simulator 傳遞，並透過 NGAP（SCTP）與 GTP-U（UDP）與 5GCN 交換控制面與使用者面訊息，負責處理 UE 的 PRACH/RAR、RRC 連線與資料傳輸。
4. 5GCN 模組使用 OAI 官方的 CN5G Docker 組件（oai-amf、oai-smf、oai-upf、mysql 等）實現 5G Core Network。該模組與 gNB 透過 NGAP 建立控制面連線，進行 UE 註冊與 PDU Session 建立；並透過 GTP-U 建立使用者面的隧道，將 UE 的 IP 封包轉送到資料網路中的伺服器，完成端對端的 IP 封包轉發。

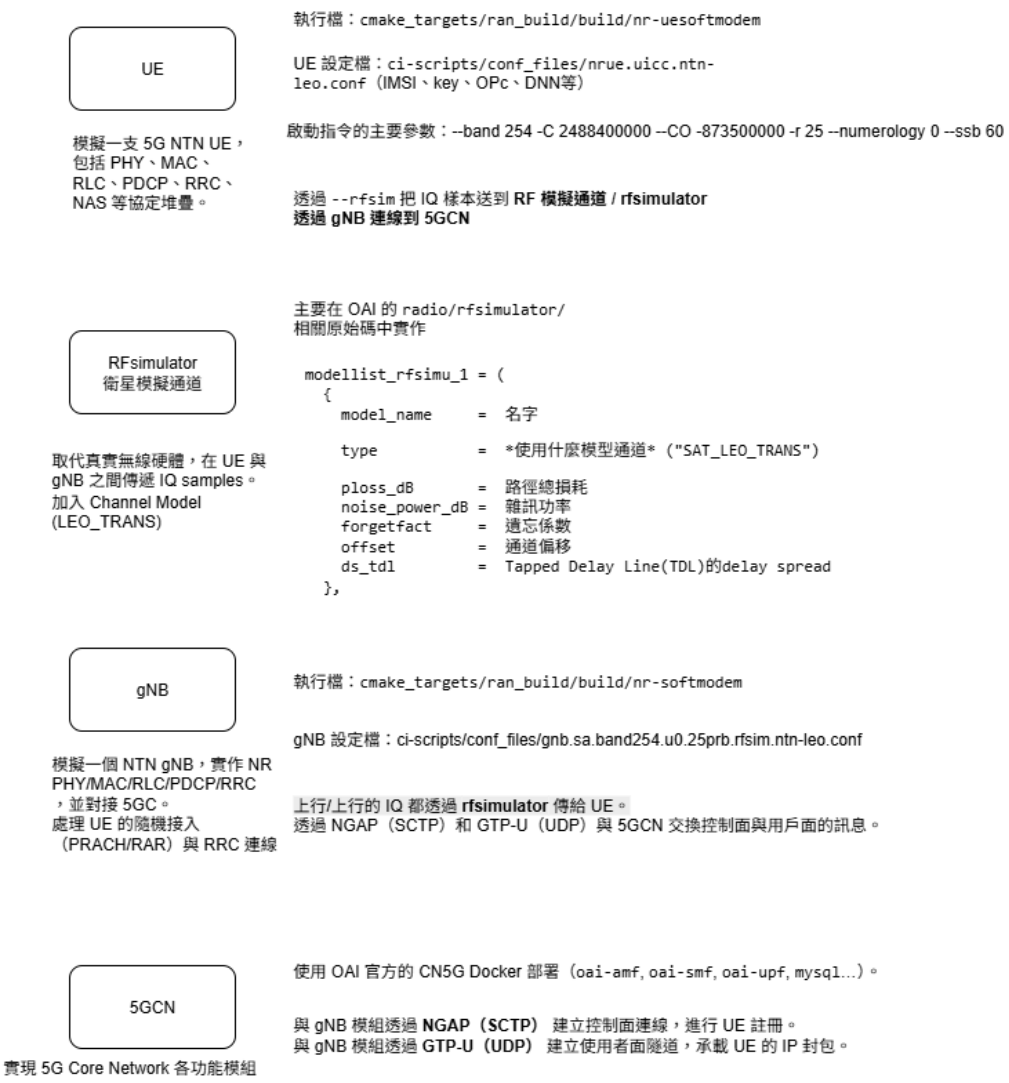

圖 14.OAI 模組圖（UE、RF-Simulator、gNB 與 5GCN 之模組劃分）

NTN 通道模型（LEO_TRANS / LEO_REGEN）與 RF-Simulator 參數

在此部分中，UE 與 gNB 之間並未使用實體 RF 卡，而是透過 OAI 的 RF-Simulator 來模擬無線通道；以下會介紹兩個本研究會用到通道模型（LEO_TRANS / LEO_REGEN）及 RF-Simulator 的相關參數。在 RF-Simulator 的通道設定中：

1. type 用來選擇通道模型的種類。例如：AWGN：（Additive White Gaussian Noise）只加入加性白雜訊、TDL：使用多抽頭延遲線（Tapped Delay Line, TDL）以及本研究會用到的兩個通道模型，SAT_LEO_TRANS： NTN 透明衛星（Transparent）模式、SAT_LEO_REGEN： NTN 再生式衛星（Regenerative）模式。

2. delay 代表 UE 與 gNB 之間單向的通道傳播延遲，在 LEO NTN 中主要由衛星軌道高度與相對幾何距離決定，可近似為 $delay \approx d(t)/c$ (其中 d(t)為某一時刻 UE 與衛星／gNB 的幾何距離、c 為光速 (~3×10^8 m/s))
3. Doppler shift f_D 代表由於衛星與 UE 之間的相對運動所造成的頻率偏移。在 LEO NTN 中，相對速度可達數 km/s。若 gNB / UE 的頻率同步與補償不足，將導致解調錯誤率上升，進而影響 RACH 成功率與 HARQ 的 Block Error Rate (BLER)

圖 15 為本研究採用之 NTN LEO_TRANS 透明衛星架構示意。UE 端執行 OAI 的 nr-uesoftmodem，經由 RF-Simulator 所建立的 NTN LEO_TRANS 通道 與地面 gNB（nr-softmodem）連線；圖左側的 LEO 衛星僅代表實際物理路徑經過透明轉發衛星，本研究並未實作實體衛星，而是由 RF-Simulator 以 SAT_LEO_TRANS 模型注入長延遲與多普勒等通道效應。gNB 透過 N2 介面將控制面訊息送往 5GC，透過 N3 介面傳送使用者面資料，5GC 再經由 N6 介面 與 DN（Data Network） 連接，完成 UE→gNB→5GC→DN 的端對端資料傳輸。圖 16 為本研究考量之 NTN LEO_REGEN 再生式衛星架構示意。UE 同樣執行 nr-uesoftmodem，與搭載 gNB 功能的 LEO 衛星（NTN-gNB） 之間的 Ground-to-satellite link 由 RF-Simulator 以 SAT_LEO_REGEN 通道模型模擬，產生 LEO 通道的延遲與多普勒效應。NTN-gNB 再透過 N2 介面(控制面)與 N3 介面(使用者面)連接地面 5GC，此一 backhaul link 在本研究中簡化為理想 IP 連線，不額外加入 RF-Simulator。5GC 之後仍透過 N6 介面 連至 DN（Data Network），完成 UE→NTN-gNB→5GC→DN 的端對端傳輸路徑。

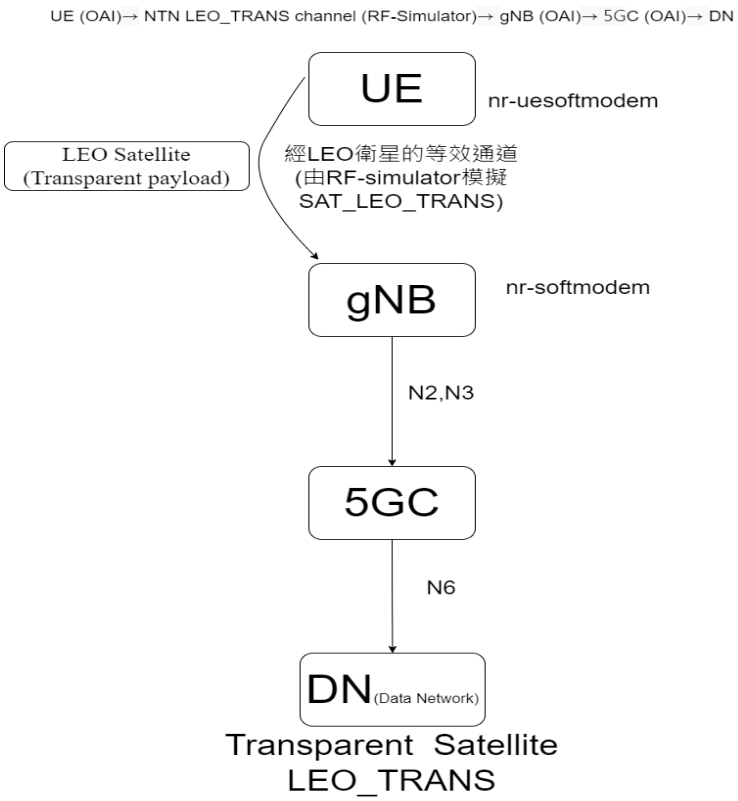

圖 15. NTN LEO_TRANS 透明衛星架構示意圖

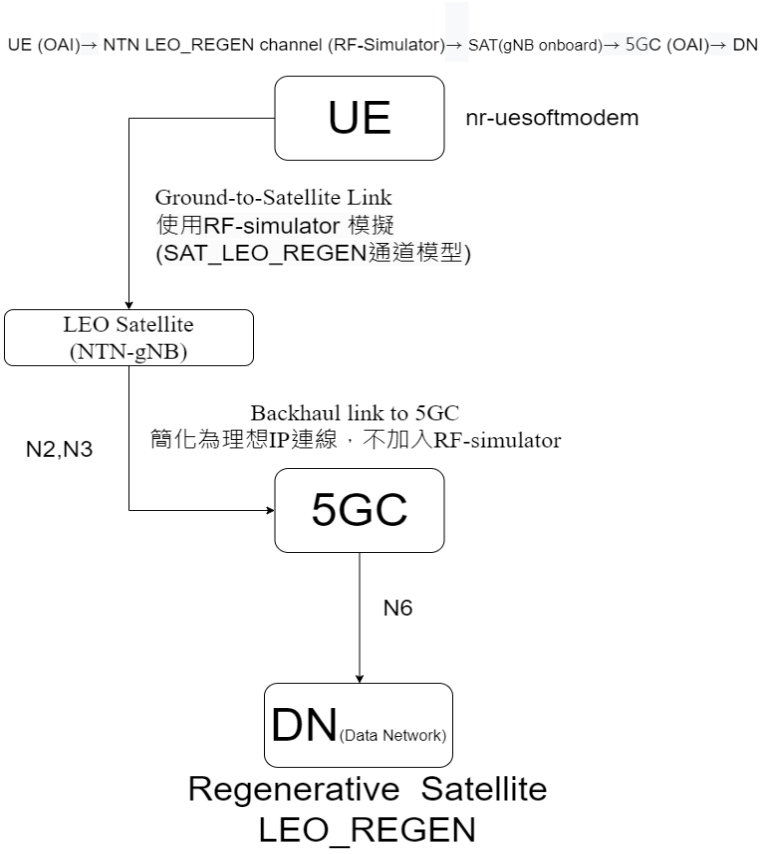

圖 16. NTN LEO_REGEN 再生式衛星架構示意圖

2. SNS3 部分:

本節整理 SNS3（ns-3 satellite）在本專題中最核心的模組與其分工，並以程式碼 目錄結構對應說明各模組在模擬流程中的角色。整體而言，SNS3 建立於 NS-3 的離散事件模擬架構之上，衛星相關行為主要集中在 contrib/satellite/，而情境配 置、流量產生與統計輸出則由 helper、traffic 與 stats 模組共同完成。

1. Satellite Model（衛星核心模型模組）

   SNS3 的核心功能位於 contrib/satellite/model/，此區塊負責衛星通訊系統的 主要建模內容，包含衛星節點與網路裝置、通道與波束相關元件，以及衛星 MAC/PHY 協議棧等。由於此模組決定衛星鏈路在模擬中的運作方式，因此最終 輸出的吞吐量、延遲、封包錯誤率與負載等指標，皆會受到此區塊模型設計與參 數設定的直接影響。

2. 通道與傳播模組（Channel / Propagation）

   通道與傳播相關功能屬於 contrib/satellite/model/ 的核心範圍，主要用於描 述封包在衛星鏈路中傳輸時的物理層效應，例如鏈路損耗、雜訊與衰落造成的錯 誤率變化。此模組提供可控的鏈路條件，使後續可以在相同情境下切換不同通道 參數，觀察 throughput、delay、error 等指標的變動趨勢。

3. 波束管理模組（Beam Management）

   波束管理用於描述多波束衛星系統中的波束覆蓋與資源作用範圍，並支援依 beam 進行統計與比較（例如 per-beam throughput、beam load）。實作集中於 contrib/satellite/model/，後續若要擴充不同 beam 配置或比較不同波束負載情況， 此模組是主要的調整與追蹤位置。

4. 衛星 MAC/PHY 協議棧模組（MAC/PHY Stack）

   SNS3 的衛星 MAC/PHY 行為（包含 frame/slot 相關機制、資源配置、排程 與傳輸流程）在 NS-3 的事件時間軸下以離散事件方式被建模，並與 DVB S2/RCS2 相關概念相呼應。此模組為影響吞吐量與延遲的關鍵因素之一，因為封 包如何被排入 frame/slot 資源、何時可傳送、以及傳送成功或失敗，都由此區塊 的機制所主導。

5. Helper 模組（Scenario Construction Helpers）

   SNS3 透過 contrib/satellite/helper/ 提供高階 helper 介面，用於快速建立模 擬情境並簡化配置流程。此模組主要負責：建立衛星情境與

(UT/GW/beam)、套 用參數設定、串接流量產生以及掛載統計輸出。helper 模組可視為「情境層」的 主要入口。

6. Traffic 流量模組（Traffic Module）

   流量產生模組位於 contrib/traffic/，提供常用 traffic pattern（例如 CBR、OnOff）以建立不同型態的資料流量，作為對照實驗的輸入條件。此模組雖不屬 於衛星核心協議棧，但會明顯影響系統被「刺激」的方式，因此在後續比較實驗 中，traffic model 是常用的自變數，用於分析穩定流量與突發流量對吞吐與延遲 的影響。

7. 統計與輸出模組（Statistics / Output）

   統計與輸出模組位於 contrib/magister-stats/，主要用於將模擬過程中的行為 彙整成可量化指標並輸出成檔案，以支援後續表格與圖表整理。模擬輸出結果通 常會集中在 data/sims/ 目錄中，作為後續分析吞吐量（throughput）、 延遲（delay）、封包錯誤率（packet error）與負載（load）等指標的資料來源。

3. Xeoverse 部分:

1. XEO Core 模組

圖 17 為 Xeoverse（XEO）之 Core 模組架構圖，此架構分成 Constellation、Topology、Routing 與輸出至 Mininet 等幾個部分。其中，Constellation 模組根據輸入之設定檔建立模擬中所需的節點集合(Node definition)，定義衛星與相關節點的基本結構；接著，Topology 模組依據節點之間的關係產生對應的網路拓樸，決定哪些節點之間存在連線(Link generation)，以描述整體網路結構。完成拓樸建立後，Routing 模組以該網路拓樸作為輸入進行路徑計算與路由決策(Path calculation)，產生封包在網路中轉送時所依循之路由資訊。最後，XEO Core 會將所產生之網路拓樸與路由相關結果(Topology/Routes)輸出至 Mininet。

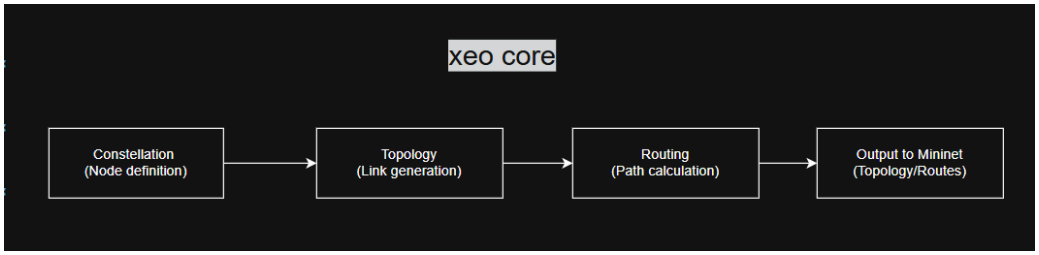

圖 17.xeocore 模組架構

2. Mininet 模組

圖 18 為 Mininet 虛擬網路模組架構圖，此架構可分為 Host、Switch 與 Links 等幾個部分。

其中，Host（h1、h2、h3）負責模擬網路中的端點節點，代表使用者或應用程式所在位置，可於其上執行 ping 與 iperf 等量測指令，以進行端對端封包傳輸與效能測試；Switch（switch1）則負責封包轉送，模擬網路中交換節點之行為，依據既定之網路拓樸結構將封包轉送至對應之目的端；Links（IP/TCP）則代表節點之間的虛擬連線，負責在 Linux 核心網路堆疊下進行 IP/TCP 層之封包傳輸。

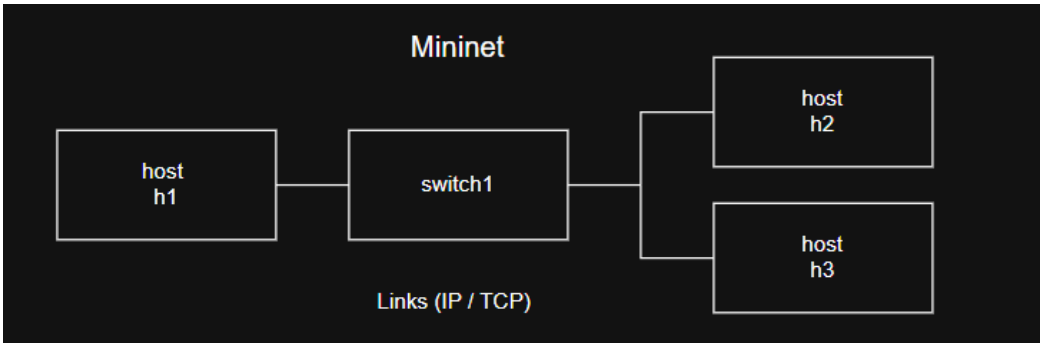

圖 18.Mininet 模組架構

V. 功能實現與源碼追蹤

A. OAI 部分:

1. RF-Simulator：虛擬 RF 板卡與 I/Q 橋接

功能描述：

使用 RF-Simulator 模擬實體 RF 卡，在 UE 與 gNB 間透過 TCP socket 傳送 I/Q 樣本。

關鍵檔案：

radio/rfsimulator/README.md [14]

radio/rfsimulator/rfsimulator.c [15]

核心函數：

device_init()

rfsimulator_write()

rfsimulator_read()

實作概念：

OAI gNB/UE 的 PHY 層原本會呼叫 RF driver 讀寫 I/Q buffer；當加上 --rfsim 選項時，改由 rfsimulator.c 建立一個 TCP 連線，把 I/Q 樣本送到對端的 RF-Simulator 實例。

2. NTN LEO 通道模型：SAT_LEO_TRANS / SAT_LEO_REGEN

功能描述：

依照設定檔中 channelmod 區段的參數設定載入 LEO 通道模型，在 RF-Simulator 內模擬 LEO 衛星的時間變動延遲與 Doppler shift 目前 RF-Simulator 支援 SAT_LEO_TRANS 與 SAT_LEO_REGEN。

關鍵檔案：

radio/rfsimulator/apply_channelmod.c [16]

openair1/SIMULATION/TOOLS/ channel_sim.c [17]

核心函數：

rxAddInput()

do_DL_sig()

do_UL_sig()

實作概念：

通道模擬程式會先解析設定檔中的 channelmod 區段（例如 type = "SAT_LEO_TRANS"），再根據當前的時間索引計算對應的 LEO 衛星位置，進一步得到傳播延遲與 Doppler 頻移。這些時間與頻率偏移會轉成多路徑通道 tap，最後在 apply_channelmod() 中對 I/Q buffer 做時間位移與相位旋轉，達到模擬 LEO 通道的效果。

3. gNB MAC：隨機接入流程與 Timing Advance (TA) 命令

功能描述：

gNB MAC 負責處理 UE 的隨機接入（Random Access, RA）流程，接收 PRACH preamble 後估計 UE 上行延遲，並在 Msg2 (RAR) 中下發 Timing Advance (TA) 命令。對 NTN 來說，延遲變很大，所以會出現問題。

關鍵檔案：

openair2/LAYER2/NR_MAC_gNB/gNB_scheduler_RA.c [18]

openair2/LAYER2/NR_MAC_gNB/nr_mac_gNB.h [19]

核心函數：

schedule_nr_prach()

nr_initiate_ra_proc()

nr_schedule_RA()

實作概念：

當 gNB PHY 偵測到 UE 的 PRACH preamble 時，會把測得的 delay 回報給 MAC。MAC 在 RA scheduler 中建立一個 RA context，計算對應的 TA command，並在 nr_fill_rar() 產生的 RAR MAC PDU 內填入 ta_command 欄位。之後 UE 收到 Msg2 後根據 TA 調整 uplink 時序。

在 NTN 模式下，因為真實 RTT 比地面情境大很多，原本為地面設計的 RA timer 與 TA range 可能不足，導致在實驗中觀察到「RA 卡住、連線失敗」的情形。

4. gNB / UE NTN 參數的引入與 Command Line Interface(CLI) 設定

功能描述：

處理 NTN 相關的 RRC / MAC 參數（例如 ta_CommonDrift_r17、NTN

timing offset 等），以及 command line / config 檔中的 --ntn-*、--rfsim、rfsimulator.options="chanmod" 等參數，把它們寫進 gNB / UE 的全域設定中。

關鍵檔案：

doc/RUNMODEM.md [20]

executables/nr-softmodem.c [21]/ executables/nr-uesoftmodem.c[22]

核心函數：

nr-softmodem::main()

nr-softmodem::create_gNB_tasks()

nr-softmodem::wait_RUs() / wait_gNBs()

nr-softmodem::start_L1L2() / stop_L1()

nr-uesoftmodem::main()

nr-uesoftmodem::create_tasks_nrue()

nr-uesoftmodem::get_options() / set_options()

nr-uesoftmodem::init_openair0()

實作概念：

在 shell 裡下的 --rfsim、--ntn-* 或 config 檔中 ntn-leo.conf 的設定，都會在執行時被 nr-softmodem / nr-uesoftmodem 解析，寫進 config。後續 RF-Simulator、MAC 層的 TA / HARQ / timer 會讀這些欄位來決定如何在 NTN 環境下運作。未來若修改 TA 或 HARQ 行為，也會從這裡再加新的 CLI / config 欄位進去。

B. SNS3 部分:

1. 回傳鏈路 Logon 機制

功能描述：

UT(使用者終端)必須先向 NCC 完成 logon,回傳鏈路(return channel)的流 量才會開始送出，用來描述回傳通道啟用的控制流程。

關鍵檔案：

contrib/satellite/examples/sat-logon-example.cc [23]

核心函數：

logon, NCC, return channel, UT

實作概念：

透過 logon 功能讓 UT 先完成登入流程；在登入完成前，回傳通道的 traffic 不 會送出，以符合衛星系統的接入/啟用邏輯。

2. NCR 同步機制

功能描述：

回傳鏈路的 frame 排程下，UT 需定期與 NCC 同步時鐘（NCR），避免因 UT 時鐘漂移導致時序不準、影響 slot/ frame 的排程與傳輸。

關鍵檔案：

contrib/satellite/examples/sat-ncr-example.cc [24]

核心函數：

NCR, synchronization, clock, NCC

實作概念：

UT 一般採用較便宜的時鐘，需由 NCC 週期性重新同步，才能維持回傳鏈路 排程正確性（特別是在 return channel 上行 frame 發送時序）

3. 流量產生（CBR / OnOff 基礎流量）

功能描述：

以固定速率（CBR）或突發型（OnOff）流量觸發衛星鏈路負載，觀察吞吐、延 遲與錯誤率等指標變化。

關鍵檔案：

contrib/satellite/examples/sat-cbr-example.cc[25] contrib/traffic/（traffic 模組）

核心函數：

CBR, OnOff, Application, Socket

實作概念：

sat-cbr-example.cc 提供最基本的固定速率流量情境，可作為後續 RTN/回傳鏈路 測試的 baseline。

4. 通道與波束模型（Channel / Beam Management）

功能描述：

本模組提供「beam ↔ channel pair」的管理機制：針對每個 (satId, beamId)，儲存並 回傳一組 Forward channel 與 Return channel，使模擬在建立多波束架構時，可以快速 定位該 beam 對應的衛星通道。

關鍵檔案：

satellite-beam-channel-pair.cc[26]

核心函數：

GetChannelPair(satId, beamId)：依衛星與波束取得(ForwardChannel, ReturnChannel)。StoreChannelPair(satId, beamId, fwdFrequencyId, fwdChannel, rtnFrequencyId, rtnChannel)： 建立並儲存該 beam 的 forward/return channel 配對。 GetForwardChannel(satId, frequencyId) / GetReturnChannel(satId, frequencyId) ：用 frequencyId 查對應通道。

HasFwdChannel(...) / HasRtnChannel(...)：檢查是否已存在該 frequency 的通道。

實作概念：

建立情境時呼叫 StoreChannelPair() 將 (satId, beamId) 綁定 (fwdFrequencyId, rtnFrequencyId) 與對應的 SatChannel → 執行期間需要通道時呼叫 GetChannelPair() → 由內部 map 查到 frequencyId，再分別透過 GetForwardChannel() 與 GetReturnChannel() 回傳通道。

輸入與輸出參數:

輸入參數（Inputs）：

satId (uint32_t)：衛星 ID（索引維度之一）。 beamId (uint32_t)：波束 ID（索引維度之一）。 fwdFrequencyId / rtnFrequencyId (uint32_t)：forward/return 頻率（或顏色/頻點）索引。 fwdChannel / rtnChannel (Ptr)：要存入的 forward/return 通道物件。

輸出參數（Outputs）：

GetChannelPair(...) -> ChannelPair_t ： 回 傳 一 組 (Ptr forward, Ptr return)。 GetForwardChannel(...) / GetReturnChannel(...) -> Ptr：回傳指定頻點的通道物件。

5. 統計收集與結果輸出（Stats / Output）

功能描述：

將模擬過程中的指標（吞吐、延遲、錯誤率、負載等）輸出成檔案，供後續畫圖與表格比較。

關鍵檔案：

contrib/magister-stats/（統計框架） data/sims//（輸出資料夾）

核心函數：

stats, output, cdf, scalar

實作概念：

SNS3 的範例情境在執行後會自動生成統計結果，並集中輸出到 data/sims/，作為報告的數據來源。

C. Xeoverse 部分:

1. XEO 星座與節點定義

功能描述:

根據輸入之設定檔與星座參數，建立模擬所需之衛星與地面節點，作為後續網路拓樸建立之基礎

關鍵檔案:

xeoverse/constellation/

核心函數:

build_constellation()

實作概念:

系統讀取設定檔（如 config 與星座參數），初始化模擬中所需的節點物件，包含衛星與地面節點，並將節點資訊傳遞至後續拓樸建立模組使用

2. XEO 網路拓樸建立

功能描述:

根據已定義之節點集合，產生節點之間的連線關係，建立整體網路拓樸結構**關鍵檔案**

關鍵檔案

xeoverse/topology/

核心函數:

generate_topology()

實作概念:

系統依據節點資訊與拓樸規則，判斷節點之間是否建立連線，並產生對應的網路結構資料，用以描述整體網路的連線關係

3. XEO 路徑計算與路由產生邏輯

功能描述:

根據已建立之網路拓樸，計算節點之間的傳輸路徑，並產生封包轉送所需之路由資訊

關鍵檔案:

xeoverse/routing/

核心函數:

compute_routes()

實作概念

系統以網路拓樸作為輸入，透過既有的路由邏輯計算節點間的傳輸路徑，並將結果整理為路由資訊，供後續虛擬網路部署使用

4. XEO 拓樸與路由輸出至 Mininet

功能描述:

將 XEO 所計算完成之網路拓樸與路由結果輸出，供 Mininet 建立虛擬網路環境使用

關鍵檔案:

xeoverse/mininet_interface/（或相關輸出模組）

核心函數:

export_to_mininet()（或對應之拓樸輸出函數）

實作概念:

系統將拓樸與路由資料轉換為 Mininet 可讀取之格式，並由 Mininet 根據該資料建立對應之 Host、Switch 與 Link，完成虛擬網路部署

5. Mininet 端對端測試流程

功能描述:

於 Mininet 虛擬網路環境中執行端對端傳輸測試，以驗證網路連線與傳輸行為

關鍵檔案:

Mininet 腳本或互動式指令

核心函數:

ping、 iperf（於 Host 上執行）

實作概念:

在建立完成之虛擬網路中，於不同 Host 上執行 ping 與 iperf 指令，量測端對端封包往返時間（RTT）與吞吐量，作為網路層模擬之驗證結果

VI. 模擬實現概念深度探討

A. OAI 部分:

1. SIB19 (System Information Block 19)：NTN 專用系統資訊區塊

為因應 NTN 中長傳播延遲、衛星移動與可能缺乏 GNSS 等特性，3GPP 在 Release 17 於 TS 38.331 [16]中新增 SystemInformationBlockType19（SIB19），作為 NTN 專用的系統資訊區塊。SIB19 由 gNB 透過 PDSCH 以 SI-RNTI 廣播，所有欲接入衛星 NTN 之 UE 均須先接收並解碼此區塊才能取得必要的 NTN 輔助資訊。SIB19 的核心內容可分為三類：

1. 軌道與時間輔助資訊：ephemerisInfo 提供 serving/neighbor 衛星的軌道或位置資訊，搭配 epochTime 作為參考時刻，使 UE 能估算特定時刻的傳播延遲與都卜勒偏移；ntn-UlSyncValidityDuration-r17 則定義自 epoch 起，該組上行同步與 TA 模型可使用的有效期間。
2. 上行同步與排程參數：commonTimingAdvance 與 koffset 為 NTN 特有的 TA／排程補償參數。前者提供共用的 TA 初始值，讓所有 UE 在進行 RACH 及初始 UL 傳輸前，先對 LEO 的大延遲做預補償；後者則為 cell-specific offset，用於 UL HARQ 以在長 RTT 環境中維持 gNB 期望的收訊時間視窗。
3. 覆蓋範圍與移動性輔助資訊：t-Service 用來告知 UE 此 NTN cell 何時會停止服務當前覆蓋區域；referenceLocation 與 distanceThresh 用來定義「離 cell 參考位置多遠時要啟動位置觸發量測」；ntn-NeighCellConfigList 則提供可切換之 NTN 鄰居 cell 清單，搭配前述參數輔助 UE 在 beam

接近 coverage 邊界前完成 reselection 或 handover。

在本研究探討的 RACH TA 補償與 HARQ timer 調整中，許多參數（如 ta-Common-r17、cellSpecificKoffset_r17、上行同步有效時間等）皆可視為 SIB19 所廣播資訊在協定實作層面的呈現。

2. 問題分析

在地面 5G NR 系統中，隨機接入 (RACH) 與 HARQ 機制是建立連線與提供可靠傳輸的關鍵。在 NTN LEO 場景下，單向傳播延遲大概會達到數十毫秒，RTT 會超過 100 ms，且 Doppler shift 隨時間快速變化，使得原本假設「幾毫秒級 RTT」的地面 5GNR 參數設定不再適用。

在 RACH 程序方面，gNB 需要在有限的 RAR window 內回應 UE 的前導 (preamble)，並在後續的 Msg3 / contention resolution 計時器到期前完成隨機接入。當 RTT 因 LEO 通道大幅增加時，UE 容易在收到 RAR 之前就觸發 timeout，或是 contention resolution 未在規定時間內完成，導致 RACH 失敗。

HARQ 部分則面臨類似問題：原本設計給地面系統的 HARQ RTT timer 與 process 數量，假設回饋 (ACK/NACK) 能在幾個毫秒至一個 Transmission Time Interval (TTI) 內完成；在 NTN_LEO 下，每一個 HARQ 回合都可能被拉長到數十毫秒甚至更久，若不調整 timer 或 process 數量，容易出現 process 被過早釋放、重傳與 Automatic Repeat request(ARQ) 疊加造成效能下降等問題。提高 HARQ process 數量到 32。

3. RACH 改編

為了在 NTN 中維持 RACH 程序的可靠性，3GPP Rel-17 引入了多項針對長延遲的增強機制，其中包含 Common Timing Advance (Common TA)、K_{offset} 與 k_{mac}，在 OAI 的 NTN 實作中，分別對應到 RRC / MAC 中幾個參數：
cellSpecificKoffset_r17：對應 3GPP 的 K_{offset}
ta-Common-r17 與 ta-CommonDrift-r17：作為 UE 初始 TA 與 TA 漂移的起始值，幫助 UE 在初始接入時就大致對齊 NTN 的長延遲與緩慢變化。
本研究規劃的 RACH 協定改編步驟如下：

1. **建立基準情境**：在 RF-Simulator 中啟用 LEO_TRANS 通道模型，設定對應軌道高度與相對速度，在「未啟用 cellSpecificKoffset_r17 / ta-Common-r17」的情況下觀察 RACH 是否成功。預期會出現 RAR 收不到或 contention resolution timeout 的情況。
2. **導入 Common TA / Koffset**：根據 LEO 單向延遲估計值，設定適當的 ta-Common-r17 與 cellSpecificKoffset_r17，使 UE 與 gNB 在 PRACH 與隨後 UL 格位排程時，能預先補償大部分 propagation delay。

4. HARQ 改編

在長 RTT 的 NTN 環境下，HARQ 機制面臨兩項主要挑戰：一個是每一次 HARQ 回合的時間大幅變長，導致原本為地面系統設計的 HARQ RTT timer 過短，容易出現 process 尚未收到 ACK/NACK 就被釋放或重傳被誤判；另一個則是在 RTT 過長時，HARQ 的效益下降，重傳資訊到達時，對上層 RLC/PDCP 來說可能已經超過可接受的延遲範圍。3GPP 對此提出的方向包括：增加 HARQ process 數量或直接關閉 HARQ 回饋，改由上層 ARQ 承擔可靠度。
相關對應的參數：
num_dlharq / num_ulharq 可配置為 32。
disable_harq 參數可關閉 HARQ 回饋。
本研究規劃的 HARQ 協定改編實驗方向如下：

1. **預設設定下的行為觀察**：在 LEO 通道模型下，使用預設 HARQ process 數量與 timer，量測 RTT end-to-end 吞吐量；預期會觀察到 HARQ RTT timer 不匹配、重傳效率低下等問題。
2. **增加 HARQ process 數量與調整 timer**：gNB / UE 的 HARQ process 數量提升至 32，並適度放寬 HARQ 相關計時器；比較在相同 RF-Simulator 通道條件下，吞吐量是否有所改善。
3. **關閉 HARQ，讓 RLC-AM 接手**：啟用 disable_harq，使實驗系統在 LEO 通道下完全依賴 RLC-AM 進行 ARQ 重傳；比較「開啟 HARQ + RLC」與「僅 RLC」兩種策略下的效能與暫時性緩衝需求，分析在不同 RTT 條件下，哪一種配置更適合作為 NTN 的設計參考。

B. SNS3 部分:

1. 情境資料與線上模擬的整合方式

sns3 主要特色是把衛星系統行為直接整合在 NS-3 的線上離散事件模擬流程中：情境建立（UT/GW/衛星/beam）由範例與 helper 組裝完成；傳輸效果（通道、波束、MAC/PHY、排程）則由 contrib/satellite/model/ 內部模型在模擬時間軸上逐步驅動。
在資料使用上，SNS3 仍可能依賴「參考情境資料」（例如衛星相關 scenario data），但這類資料主要用於情境參數化/配置，真正的封包傳輸、排程與錯誤行為仍由模擬事件推進來決定。此設計使得使用者可以用一致的 ns-3 執行方式（run / args）重現同一套情境與輸出結果，並利於後續做參數掃描與對照實驗。

2. 離散事件模擬（DES）機制的落實

SNS3 的概念核心是「離散事件驅動（Discrete-Event Simulation, DES）」：系統時間不是連續流動，而是由一系列事件（event）推進，例如「某個 frame 開始」、「某個 slot 分配完成」、「某個封包開始傳輸」、「某個封包接收成功/失敗」等。這種做法特別適合衛星系統，因為衛星通訊常以 frame/slot 為單位進行資源配置與排程，事件點清楚、可控且易於統計。

SNS3 在 DES 狀態呈現以下特徵：

1. Frame/Slot 時序事件化：衛星 MAC/PHY 的資源分配與封包傳輸會對齊 frame/slot 的時間結構；每個時間邊界點會觸發排程、分配或傳輸事件。
2. 通道/波束以事件影響傳輸結果：通道損耗、干擾、衰落等模型不必以連續微分方式計算，而是在「傳輸事件發生時」決定其成功率/錯誤率，並回饋到統計輸出（throughput、delay、error）。

3. 現行模型限制與擴充性探討

雖然 SNS3 能以 DES 精確描述衛星系統在 frame/slot 下的協定行為，但在實作與研究應用上仍存在幾個常見限制與取捨

1. 規模擴張的計算成本
   多波束、多使用者、多流量情境會使事件數量快速增加，模擬時間與資源消耗上升；因此在大規模星座或長時間模擬時，可能需要取捨（例如降低細節、縮短模擬時間、分批掃參數）。
2. 模型細節與真實性取捨
   SNS3 的強項是衛星 MAC/PHY 與通道行為的可控建模，但任何模型仍是抽象化；例如某些硬體層細節、實際設備實作差異，可能需要額外參數校正或透過實測資料對齊。
3. 實驗設計依賴參數設定品質
   DES 模擬的輸出高度依賴情境與參數（beam 配置、通道參數、流量模型、排程策略）。因此在後續工作中，需建立 baseline（固定版本與固定參數）再逐一變更單一因素，才能讓比較結果具說服力。

VII. 結論與規劃

目前為止已完成之工作項目如下：

OAI 部分:

1. 將 5GCN 架設好並鎖定版本，建置、編譯好 gNB (nr-softmodem) 與 nrUE (nr-uesoftmodem)，建立地面網路端對端(E2E)環境，在未加入 ntn 通道的情況下跑通 UE↔gNB↔5GCN 基線，並且量測與收檔並鎖定版本，量測並紀錄 最大 RTT 與 iperf3 吞吐/丟包。
2. 在未更改參數的情況下，使用 RF-Simulator 模擬 NTN 低軌衛星(LEO)的 LEO_TRANS 模式下的延遲、丟包、時漂、頻偏， 模擬失敗的結果，演

示在未更改參數的情況下加入 NTN 通道是無法完成連線。

SNS3 部分:

完成 SNS3（ns-3 satellite）模擬環境之建置與基本驗證，成功整合 SNS3 satellite 模組所需的相依套件與參考情境資料（submodule）。在環境完成後，已可正常執行衛星模組範例，完成 RTN 範例情境之成功執行，確認模擬流程能夠順利從情境建立、模擬推進到產生輸出結果，代表後續實驗具備可重現的基礎。

後續規劃：

1. 導入 traffic 模組進行流量情境設計

規劃導入 CBR / OnOff 等流量型態，建立「穩定流量 vs 突發流量」的對照實驗，以觀察在相同衛星系統設定下吞吐量、延遲與錯誤率的變化趨勢。

2. 導入 stats（magister-stats）建立統一輸出格式

導入 magister-stats 建立固定輸出格式與指標集（如 throughput、delay、packet error、beam load 等），並統一輸出位置與檔名規則，以利後續繪圖與比較。

3. 完成實驗結果彙整

以 baseline 為起點，規劃多組參數對照（每次只改一個變因），最後將輸出整理成表格與圖表，以便後續其他組員測量與對照。

4. 建立可供 OAI 量測的實驗數據輸出介面

Xeovers 部分:

已完成 **Xeoverse（XEO）**基本模擬環境之建置，並成功與 **Mininet** 整合，建立可實際執行之虛擬網路環境。在此架構下，已可透過 Mininet 進行端對端傳輸測試，並使用 **iperf** 與 **ping** 工具量測吞吐量與封包往返時間（RTT），以驗證基礎資料傳輸行為之正確性。

待完成的項目:

OAI 部分:

1. 更改 gNB 的參數，使 NTN 能成功連線
2. 測試 LEO_REGEN 模式下的模擬結果
3. 量測記錄加入 NTN 模擬通道的最大 RTT 與 iperf3 吞吐/丟包。比較
4. 整理數據比較 無 NTN、LEO_TRANS、LEO_REGEN 三者的數據

SNS3 部分:

規劃輸出為 CSV / JSON 其中一種，並統一欄位命名、單位與時間戳。

並以 OAI 可量測的指標為對齊目標，固定每次實驗必輸出的欄位，例如：

吞吐量（Throughput, bps/ Mbps）

延遲（One-way delay / RTT，ms）

封包遺失率（Packet loss rate, %）

抖動（Jitter, ms）（若後續需要）

建立輸出管線：整合 SNS3 的統計輸出（預計導入 magister-stats 或自訂 trace callback），在模擬結束後自動生成。

Xeovers 部分:

後續將在目前已完成之 XEO 與 Mininet 架構基礎上，進一步評估引入其他輔助模組之可行性,像是整合內建的天氣 API,將天氣資訊轉換為網路層參數(如連線延遲或封包遺失率），以觀察其對端對端 RTT 與吞吐量之影響。

同時，也將進一步研究 XEO 中的路徑計算機制，了解其目前採用之路由演算法與路由邏輯，並分析不同路徑選擇對端對端傳輸效能的影響，作為網路層行為分析的延伸方向。

VIII. 時間進度表

OAI

月次 Month / order / 工作項目 Work item	第1月	第2月	第3月	第4月	第5月	第6月	第7月	第8月	第9月	備註 Remark
收集與整理資料										
5GCN架設										
了解RAN程式碼架構並完成5GRAN架設										
測試Core Network 與 RAN 的整體功能										
量測記錄(無NTN效應的數據)										
研究RF simulator的參數並加入模擬										
使用測試TRANS 與 REGEN 兩種模式										
量測記錄(有NTN效應的數據)										
整理數據與彙整實驗結果										
完成專題期末報告										

預定進度累計百分比 Scheduled progress cumulative percentage	5%	10%	25%	50%	60%	75%	85%	95%	100%	

*藍色為預期結果，黃色為實際進度

SNS3

月 次 Month order 工作項目 Work item	第 1 月	第 2 月	第 3 月	第 4 月	第 5 月	第 6 月	第 7 月	第 8 月	第 9 月	備註 Remark
收集與整理資料										
SNS3 環境建立與理解結構										
衛星網路模型與通道設計										
讓 SNS3 與外部真實網路互通										
完成專題期中報告										
與 OAI 整合										
整理數據與彙整實驗結果										
完成專題期末報告										
預定進度累計百分比 Scheduled progress cumulative percentage	5%	10 %	16 %	25 %	40 %	55 %	70 %	85 %	100 %	

*藍色為預期成果，黃色為實際成果

XEO

月 次 Month order 工作項目 Work item	第 1 月	第 2 月	第 3 月	第 4 月	第 5 月	第 6 月	第 7 月	第 8 月	第 9 月	備註 Remark
收集&整理資料										
XEO 基本環境建置與理解										
XEO 與 Mininet 整合										
端對端傳輸測試與基礎量測										
完成專題期中報告										
進階網路層情境測試										
路徑計算與路由機制分析										
彙整實驗結果										
完成專題期末報告										
預定進度累計百分比 Scheduled progress cumulative percentage	5 %	10 %	16 %	25 %	40 %	55 %	70 %	85 %	100 %	

*藍色為預期成果，黃色為實際成果

IX. 參考資料

[1]3GPP TR 38.821

[2]3GPP TS 38.211

[3] doc/NR_SA_Tutorial_OAI_CN5G.md · develop · oai / openairinterface5G · GitLab

[4] doc/NR_SA_Tutorial_OAI_nrUE.md · develop · oai / openairinterface5G · GitLab

[5]https://github.com/Kuan-K/2025_kuan_project/blob/main/OAI/CN5G%20built%20environment_.md

[6]https://github.com/Kuan-K/2025_kuan_project/blob/main/OAI/E2E%20hands-on.md#installation-of-dependencies-and-compilation

[7] https://github.com/sns3/sns3-satellite

[8]https://github.com/kevin940822-beep/OpenAirInterface-NTN-Integration/blob/main/sns3/SNS3_installation.md

[9] https://github.com/raycg/xeoverse

[10]https://www.notion.so/xeoverse-2b40f1e3ee8980dabde6e5206b65d03f?source=copy_link#2ca0f1e3ee8980c58c2cf4e097686d5b

[11]https://www.notion.so/xeoverse-2b40f1e3ee8980dabde6e5206b65d03f?source=copy_link#2b80f1e3ee898082af51feb19ce696a4

[12] https://www.sns3.org/doc/_images/satellite-general-architecture.png

[13] https://www.sns3.org/doc/_images/satellite-helper-structure.png

[14]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/radio/rfsimulator/README.md

[15]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/radio/rfsimulator/simulator.c

[16]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/radio/rfsimulator/apply_channelmod.c

[17]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/openair1/SIMULATION/TOOLS/channel_sim.c

[18]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/openair2/LAYER2/NR_MAC_gNB/gNB_scheduler_RA.c

[19]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/openair2/LAYER2/NR_MAC_gNB/nr_mac_gNB.h

[20]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/doc/RUNMODEM.md?ref_type=heads

[21]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/executables/nr-softmodem.c?ref_type=heads

[22]https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/executables/nr-uesoftmodem.c?ref_type=heads
[23] https://www.sns3.org/api/sat-logon-example_8cc.html
[24] https://www.sns3.org/api/sat-ncr-example_8cc.html
[25] https://www.sns3.org/api/sat-cbr-example_8cc.html
[26]https://github.com/sns3/sns3-satellite/blob/master/model/satellite-beam-channel-pair.cc
[27] 3GPP TS 38.331