비면허 다중링크 분석 툴

1. 실행 환경 및 설치

- 운영체제: Ubuntu Linux 22.04 를 권장한다.

- 필요 사항
  - Ubuntu 패키지
    - git, curl, cmake, cmake-format, libgsl-dev, libxml2-dev, python3-dev, libsqlite3-dev, libboost-dev, libgtk-3-dev, gir1.2-goocanvas-2.0, libdpdk-dev, graphviz-dev, libcrypt-dev, python3-pip
  - Python 라이브러리
    - pygraphviz
  - C++ 라이브러리
    - jsoncpp
  - ns3 3.36
    - https://www.nsnam.org/releases/ns-allinone-3.36.1.tar.bz2

- 수동 설치
  - 필요 패키지의 설치

```
$ sudo apt install git curl cmake cmake-format libgsl-dev libxml2-dev python3-dev libsqlite3-dev libboost-dev libgtk-3-dev gir1.2-goocanvas-2.0 libdpdk-dev graphviz-dev libcrypt-dev python3-pip
...
$ pip3 install pygraphviz
...
```

  - ns3 소스코드 다운로드

```
$ wget https://www.nsnam.org/releases/ns-allinone-3.36.1.tar.bz2
```

- ns3 압축 해제 및 폴더 이동

```
$ tar xjf ns-allinone-3.36.1.tar.bz2
...
$ cd ns-allinone-3.36.1/ns-3.36.1/
ns-allinone-3.36.1/ns-3.36.1$
```

- jsoncpp 설치

```
ns-allinone-3.36.1/ns-3.36.1$ git clone
https://github.com/Microsoft/vcpkg.git
...
ns-allinone-3.36.1/ns-3.36.1$ cd vcpkg
ns-allinone-3.36.1/ns-3.36.1/vcpkg$ ./bootstrap-vcpkg.sh
...
ns-allinone-3.36.1/ns-3.36.1/vcpkg$ ./vcpkg integrate install
...
ns-allinone-3.36.1/ns-3.36.1/vcpkg$ ./vcpkg install jsoncpp
...
ns-allinone-3.36.1/ns-3.36.1/vcpkg$ cd ..
```

- 다중링크 분석 툴 소스 설치

```
ns-allinone-3.36.1/ns-3.36.1$ unzip [mlms.zip] -d ./examples/
```

- 빌드 설정

```
ns-allinone-3.36.1/ns-3.36.1$
.. $ temp=$(pwd)/vcpkg/scripts/buildsystems/vcpkg.cmake
.. $ echo "include(₩"$temp₩")" > ./examples/mlms/CMakeLists.txt
.. $ cat ./examples/mlms/base_CMakeLists.txt
>> ./examples/mlms/CMakeLists.txt
.. $ ./ns3 configure --enable-examples --enable-tests --enable-
python-bindings
```

- 스크립트 사용 설치
    - 수동 설치의 자동화 스크립트 install_mlms.sh 사용하여 설치 가능하다.

```
$ sudo ./install_mlms.sh
....
```

- 가상 머신의 사용
    - 위 설정이 모두 적용되어 있는 VMWare 를 위한 가상 머신 이미지 를 Import 하여 사용한다.

2. 실행

- ns3 소스코드 디렉토리에서 ns3 명령을 이용하여 원하는 시나리오의 실험 실행

ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run "wifi-mlms –config=[설정파일]"
예) ./ns3 run "wifi-mlms –config=./examples/mlms/scenario-0.json"

- 실험 환경은 구조화된 데이터를 표현하기 위한 문자기반의 표준 포맷인 JSON 형식으로 이루어진 설정 파일을 이용하여 구성한다.

3. 설정 파일 문법

- 하나의 JSON 객체로 구성하기 위해 **{** 로 시작하고, **}**로 마친다.
- 객체의 하위 데이터는 "simulationTime", "networks", "stations"가 존재한다.
  따라서 아래와 같은 기본 구조를 가진다.

```
{
    "simulationTime": .....,
    "networks": ....,
    "stations": .....
}
```

- "simulationTime"은 시뮬레이션 시간 (seconds)를 의미하고 실수로 표현한다.

```
{
    "simulationTime": 10.0,
    "networks": ....,
    "stations": .....
}
```

- "networks"은 다중링크를 제공하는 Access Point 가 구성한 각각의 네트워크에 대한 객체의 배열이다.

```
{
    "simulationTime": 10.0,
    "networks": [ { ... },  ... ] ,
    "stations": .....
}
```

- "stations"는 네트워크에 연결된 station 에 대한 객체의 배열이다.

```
{
    "simulationTime": 10.0,
    "networks": [ { ... },  ... ] ,
    "networks": [ { ... },  ... ]
}
```

○ 하나의 네트워크 객체는 아래와 같은 하위 데이터를 가진다

프로퍼티 (데이터)	내용
"ssid"	SSID (문자열)
"standard"	WiFi 표준 (문자열)
"wifiManager"	WiFi Rate 제어 ("Ideal" 외 다수)
"AP"	Access Point 정보 (객체)
"AP_applications"	AP 에서 작동하는 application 정보 (배열)

```
{
    "simulationTime": 10.0,
    "networks": [ {
                        "ssid": "network-1",
                        "standard": "802.11ax",
                         "wifiManager": "Ideal",
                         "AP": { ... },
                         "AP_applications": [ { ... }, ... ]
                    },
                     ....
                ] ,
    "stations": .....
}
```

- "ssid"는 알파벳, 숫자, 특수기호를 조합한 문자열로 지정할 수 있다.
- "standard"는 현재 버전 기준으로 "802.11n", "802.11ac", "802.11ax" 3 가지 중 하나로 설정할 수 있다.
- "wifiManager"는 MAC rate control algorithm 을 "Ideal", "Aarf", "Cara" 등으로 설정할 수 있다.
- "AP"는 네트워크를 제공하는 Access Point 의 정보를 나타내는 객체이다 (아래에서 설명).
- "AP_applications"는 AP 에서 트래픽을 생성하는 애플리케이션의 정보를 나타내는 객체의 배열이다. 트래픽을 생성하는 애플리케이션이 없을 경우 빈 배열로 정의할 수 있다.

○ “AP” 객체는 아래와 같은 하위 데이터를 가진다.

프로퍼티 (데이터)	내용
“location”	X, Y, Z 좌표 (실수 배열, 좌표계 단위는 m)
“links”	AP 가 지원하는 링크 객체의 배열

```
{
    “simulationTime”: 10.0,
    “networks”: [ {
                        “ssid”: “network-1”,
                        “standard”: “802.11ax”,
                        “wifiManager”: “Ideal”,
                        “AP”: {
                                “location”: [ 0.0, 0.0, 0.0 ],
                                “links”: [ { … }, … ]
                            },
                        “AP_applications”: [ ]
                    },
                    ....
                ] ,
    “stations”: .....
}
```

- “location”은 meter 를 단위로 하는 3 차원 좌표계에서 X, Y, Z 좌표를 나타내는 실수 배열이다.
- “links”는 AP 가 지원하는 링크에 대한 정보를 나타내는 객체의 배열이다.

○ 하나의 link 객체는 아래와 같은 하위 데이터를 가진다.

프로퍼티 (데이터)	내용
"band"	사용할 밴드 ("2.4Ghz", "5Ghz", "6Ghz")
"channel"	사용할 channel 번호. 0 은 default channel 사용 의미
"Ipv4Address"	해당 링크의 IPv4 주소 (문자열)
"Ipv4Mask"	해당 링크의 IPv4 넷 마스크 (문자열)

```
{
    "simulationTime": 10.0,
    "networks": [ {
                    "ssid": "network-1",
                     "standard": "802.11ax",
                     "wifiManager": "Ideal",
                     "AP": {
                                "location": [ 0.0, 0.0, 0.0 ],
                                "links": [ {
                                                "band": "2.4Ghz",
                                                "channel": 0,
                                                "Ipv4Address": "192.168.1.0",
                                                "Ipv4Mask": "255.255.255.0"
                                            },
                                            ... ]
                                },
                     "AP_applications": [ { ... }, ... ]
                  },
                  ....
                ] ,
    "stations": .....
}
```

- 하나의 AP_application 객체는 아래와 같은 하위 데이터를 가진다.

프로퍼티 (데이터)	내용
"band"	애플리케이션이 사용하는 링크의 밴드
"protocol"	트래픽의 프로토콜 (문자열)
"interval"	패킷 전송 간의 시간 간격 (seconds)
"randomInterval"	전송 간의 시간 간격이 위의 값을 평균으로 한 지수 분포를 따르는 랜덤 값을 가짐
"packetSize"	패킷의 크기
"randomPacketSize"	패킷의 크기가 위의 값을 평균으로 한 지수 분포를 따르는 랜덤 값을 가짐
"destType"	목적지 종류 ("broadcast", "sta")
"destIndex"	"sta"일 경우 네트워크에서 몇 번째 sta 인지 기입 (정수)
"destPort"	트래픽의 목적 포트번호 (정수)
"AC"	EDCA 의 AC ("BK", "BE", "VI", "VO")

```
{
    "simulationTime": 20,
    "networks": [ {
                        ……,
                         "AP_applications": [ {
                                                    "band": "2.4Ghz",
                                                    "protocol":"UDP",
                                                    "interval":1,
                                                    "randomInterval":false,
                                                    "packetSize":1472,
                                                    "randomPacketSize":false,
                                                    "destType":"broadcast",
                                                    "AC":"BE",
                                                },
                                                …
                                                ]
                    },
                    ….
                ] ,
    "stations": …..
}
```

○ 하나의 station 객체는 아래와 같은 하위 데이터를 가진다

프로퍼티 (데이터)	내용
"ssid"	속하는 네트워크의 ssid (문자열)
"band"	속한 네트워크에서 사용할 링크의 밴드
"location"	X, Y, Z 좌표 (실수 배열, 좌표계 단위는 m)
"application"	Station 에서 작동하는 application 정보 (객체)

```
{
    ...
    "stations": [ {
                        "ssid": "network-1",
                        "band": "5Ghz",
                        "location": [ 0.0, 10.0, 0.0 ],
                        "application": { ... }
                  },
                  ...
                ]
}
```

○ application 객체는 아래와 같은 하위 데이터를 가진다.

프로퍼티 (데이터)	내용
"protocol"	트래픽의 프로토콜 (문자열)
"interval"	패킷 전송 간의 시간 간격 (seconds)
"randomInterval"	전송 간의 시간 간격이 위의 값을 평균으로 한 지수 분포를 따르는 랜덤 값을 가짐
"packetSize"	패킷의 크기
"randomPacketSize"	패킷의 크기가 위의 값을 평균으로 한 지수 분포를 따르는 랜덤 값을 가짐
"destType"	목적지 종류 ("broadcast", "ap", "sta")
"destIndex"	"sta"일 경우 네트워크에서 몇 번째 sta 인지 기입 (정수)
"destPort"	트래픽의 목적 포트번호 (정수)
"AC"	EDCA 의 AC ("BK", "BE", "VI", "VO")

```
{
    …
    "stations": [ {
                    "ssid": "network-1",
                    "band": "5Ghz",
                    "location": [ 0.0, 10.0, 0.0 ],
                    "application": {
                                    "protocol":"UDP",
                                    "interval":0.0001,
                                    "randomInterval":false,
                                    "packetSize":1472,
                                    "randomPacketSize":false,
                                    "destType":"AP",
                                    "destPort":10,
                                    "AC":"BE",
                                }
                  },
                  …
                ]
}
```

- 예제: 다음은 (0, 10, 0)에 놓인 AP 가 2.4Ghz, 5Ghz, 6Ghz 세 개의 링크로 network-1 이 ssid 인 네트워크를 구성할 때, 5Ghz 링크에 연결되고 (0, 20, 0)에 놓인 station 이 고정간격, 고정패킷크기로 UDP 트래픽을 AP 에 보내는 경우의 전체 설정 파일과 실행 스크린샷이다.

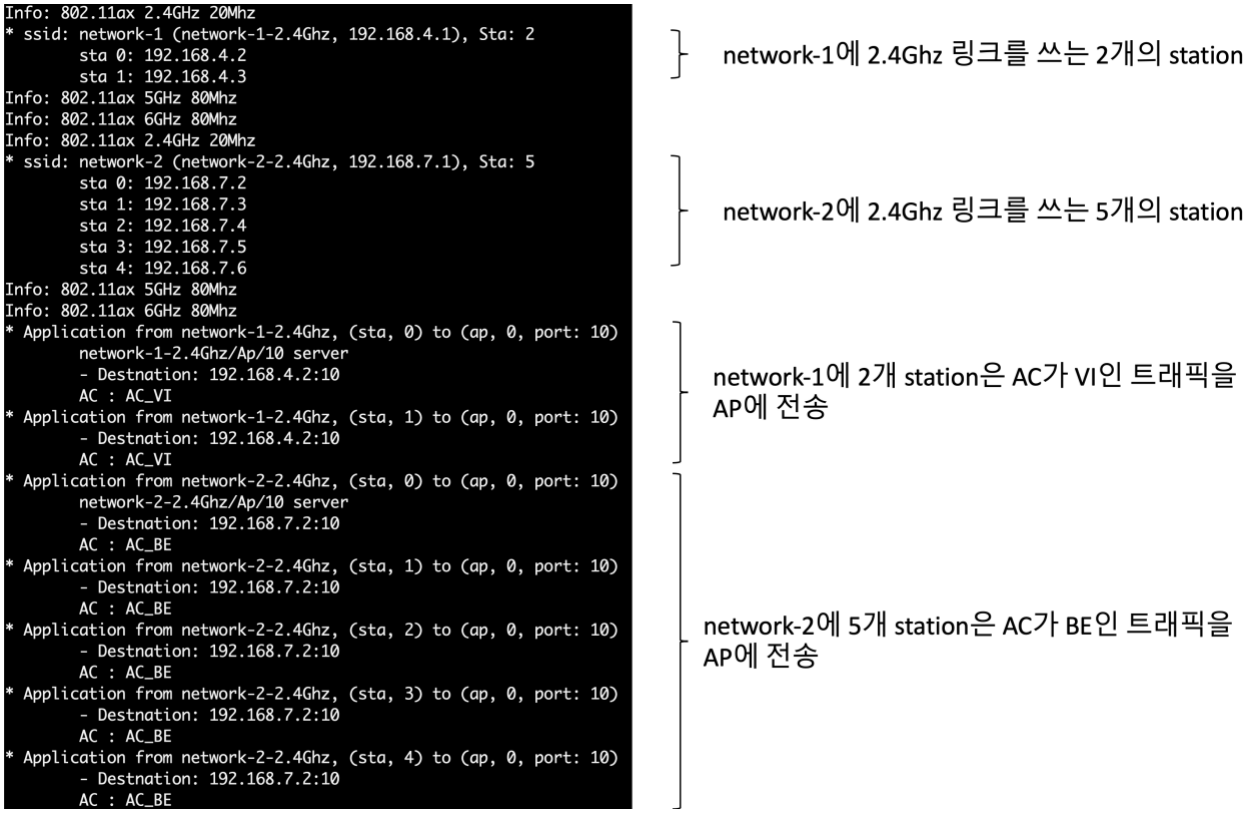
```
{
    "simulationTime": 10.0,
    "networks": [
        {
          "ssid": "network-1",
          "standard": "802.11ax",
          "wifiManager": "Ideal",
          "AP": {
            "location": [0.0, 10.0, 0.0],
            "links": [
                {
                  "band": "2.4Ghz",
                  "channel": 0,
                  "Ipv4Address": "192.168.4.0",
                  "Ipv4Mask": "255.255.255.0"
                },
                {
                  "band": "5Ghz",
                  "channel": 0,
                  "Ipv4Address": "192.168.7.0",
                  "Ipv4Mask": "255.255.255.0"
                },
                {
                  "band": "6Ghz",
                  "channel": 0,
                  "Ipv4Address": "192.168.6.0",
                  "Ipv4Mask": "255.255.255.0"
                }
            ]
          },
          "AP_applications": [ ]
```

```
        }
      ],
  "stations": [
    {
      "ssid": "network-1",
      "band": "5Ghz",
      "location": [0.0, 20.0, 0.0],
      "application": {
        "protocol": "UDP",
        "interval": 0.00001,
        "randomInterval": false,
        "packetSize": 1472,
        "randomPacketSize": false,
        "destType": "Ap",
        "destPort": 10,
        "AC": "BE"
      }
    }
  ]
}
```

```
(ns) ylim@ylim-serv1:~/hdd/ns-allinone-3.36.1/ns-3.36.1$ ./ns3 run "wifi-mlms --config=./examples/mlms/scenario-0.json"
Info: 802.11ax 2.4GHz 20Mhz
Info: 802.11ax 5GHz 80Mhz
* ssid: network-1 (network-1-5Ghz, 192.168.7.1), Sta: 1
        sta 0: 192.168.7.2
Info: 802.11ax 6GHz 80Mhz
* Application from network-1-5Ghz, (sta, 0) to (ap, 0, port: 10)
        network-1-5Ghz/Ap/10 server
        - Destnation: 192.168.7.2:10
        AC : AC_BE
(ns) ylim@ylim-serv1:~/hdd/ns-allinone-3.36.1/ns-3.36.1$
```

4. 실행결과

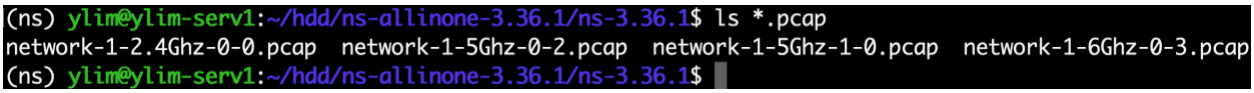
- 설정 파일에 따라 시뮬레이션을 수행하며 화면에 아래와 같은 기초 정보를 출력한다.
    - 네트워크 구성 정보 (ssid, 링크 밴드, 해당 구성의 IP 주소)
    - 네트워크에 속한 station 에 할당된 IP 주소
    - 802.11 표준과 밴드에 따른 링크 대역폭
    - 트래픽을 생성하는 애플리케이션 정보 (송신자, 수신자 정보 등)



- 각각의 flow 에 대한 실험결과는 실행 폴더의 results.xml 에 저장된다.

```
<?xml version="1.0" ?>
<FlowMonitor>
  <FlowStats>
    <Flow flowId="1" timeFirstTxPacket="+1e+09ns" timeFirstRxPacket="+1
="+0ns" txBytes="1500000000" rxBytes="1500000000" txPackets="1000000" r
      <delayHistogram nBins="1" >
        <bin index="0" start="0" width="0.001" count="1000000" />
      </delayHistogram>
      <jitterHistogram nBins="1" >
        <bin index="0" start="0" width="0.001" count="999999" />
      </jitterHistogram>
```

- 시뮬레이션의 각 노드에서 발생한 packet 을 캡쳐한 결과 역시 pcap 파일로 저장되어 wireshark 등 외부 도구를 이용하여 추가적인 분석이 가능하다.



- 리포트 도구의 이용