Deployment and configuration of MEC apps with Simu5G

Alessandro Noferi, Giovanni Nardini, Antonio Virdis, Giovanni Stea

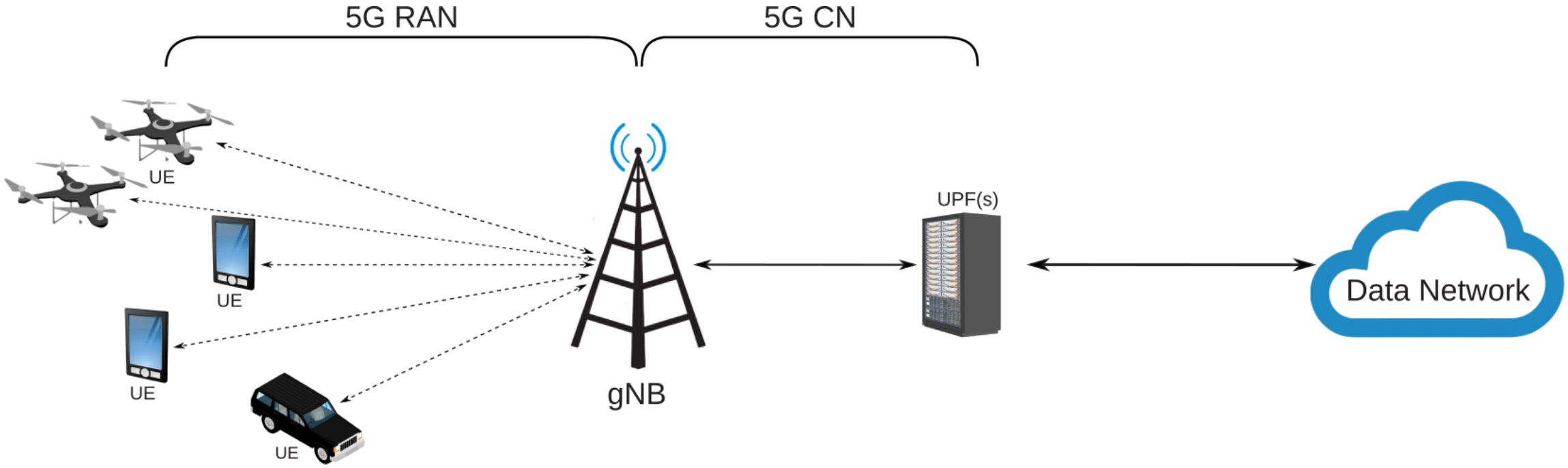
Department of Information Engineering, University of Pisa, Italy

OMNeT++ Community Summit 2021

Outline

- 5G mobile networks
- Multi-access Edge Computing (MEC)
- Tools for MEC apps prototyping
- 5G networks simulation with OMNeT++: Simu5G
- Modeling MEC within Simu5G
- Deployment scenarios
- Conclusion

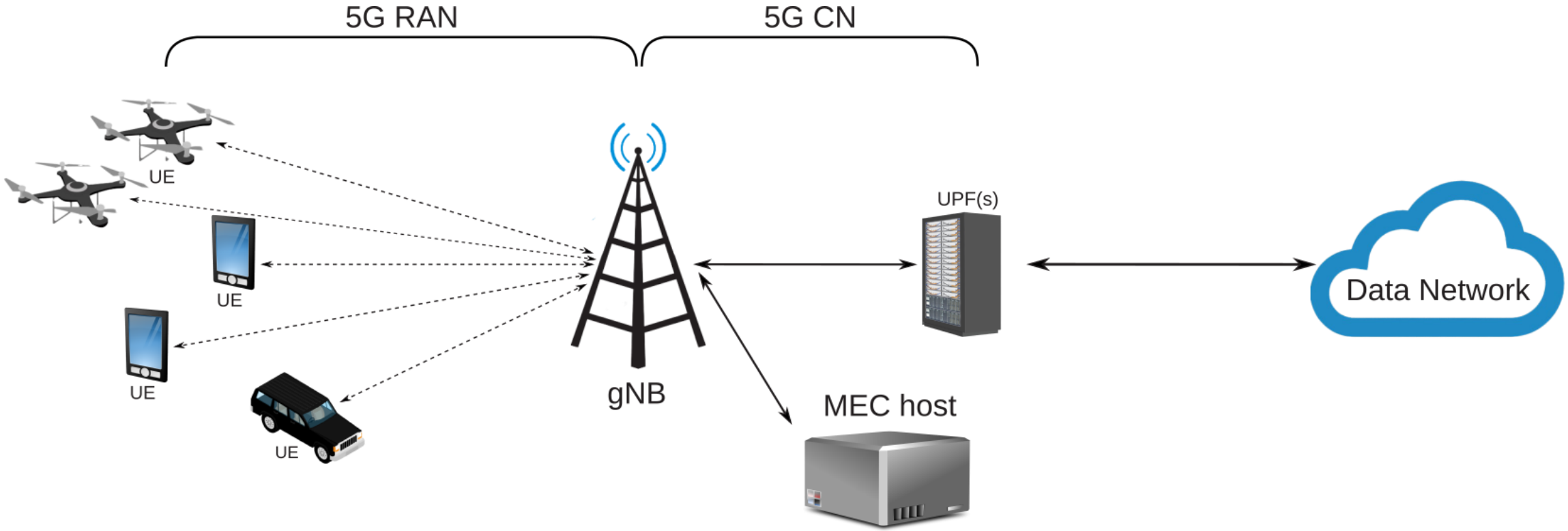
5G communication technology



- Increasing amount of data
- Stringent QoS constraints

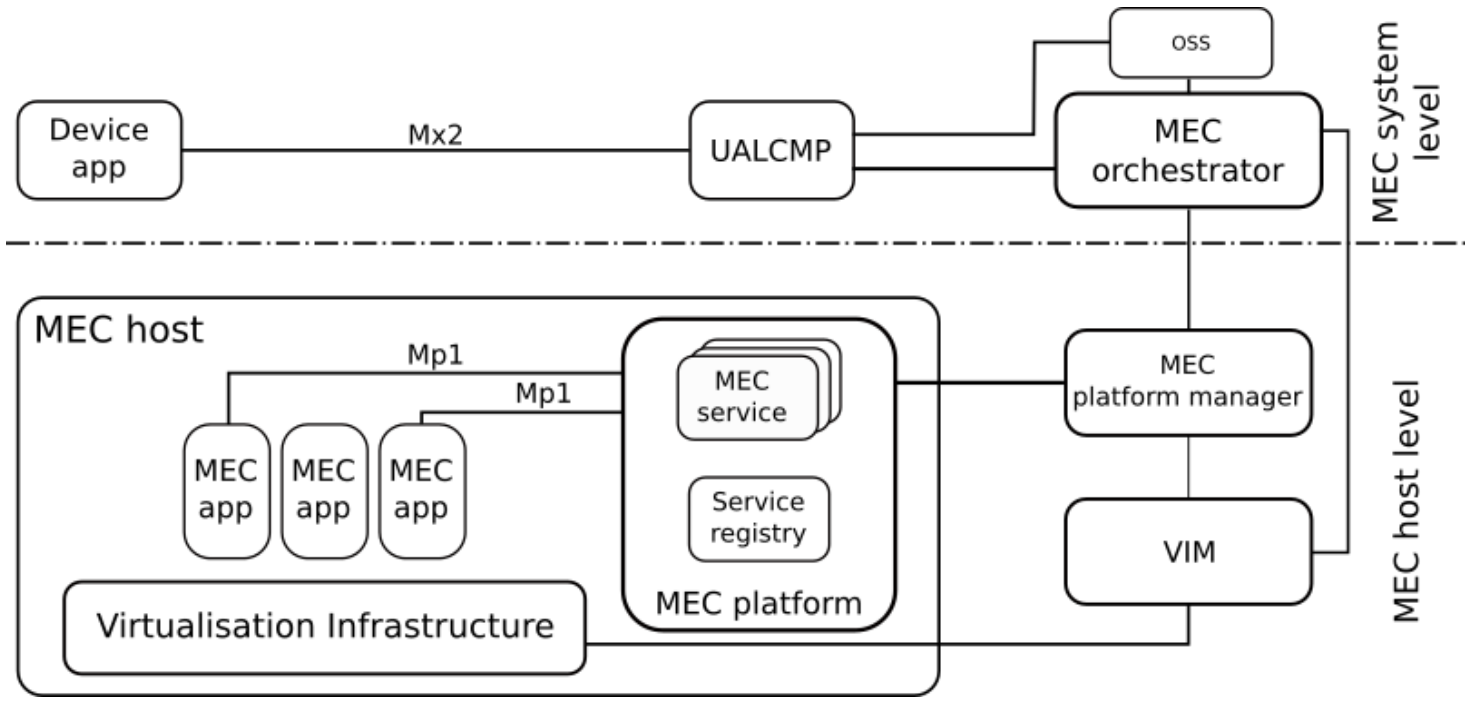
Edge Computing

Multi-access Edge Computing (MEC)



Computing and **storage** capabilities as close as possible to the **edge of the network**

ETSI MEC Architecture



MEC system entities communicate through standardized reference points (e.g Mx2, Mp1)

MEC orchestrator

- Maintains an overall view of the MEC system
- Selects appropriate MEC host(s) for application instantiation based on constraints

Device app

Application on the UE device that has the capability to interact with the MEC system (via the UALCMP)

MEC app

- Edge application requested by the UE to accomplish a task
- Deployed as VM or container on top of the Virtualisation Infrastructure
- Can interact with the MEC platform to consume (or provide) MEC service

MEC platform

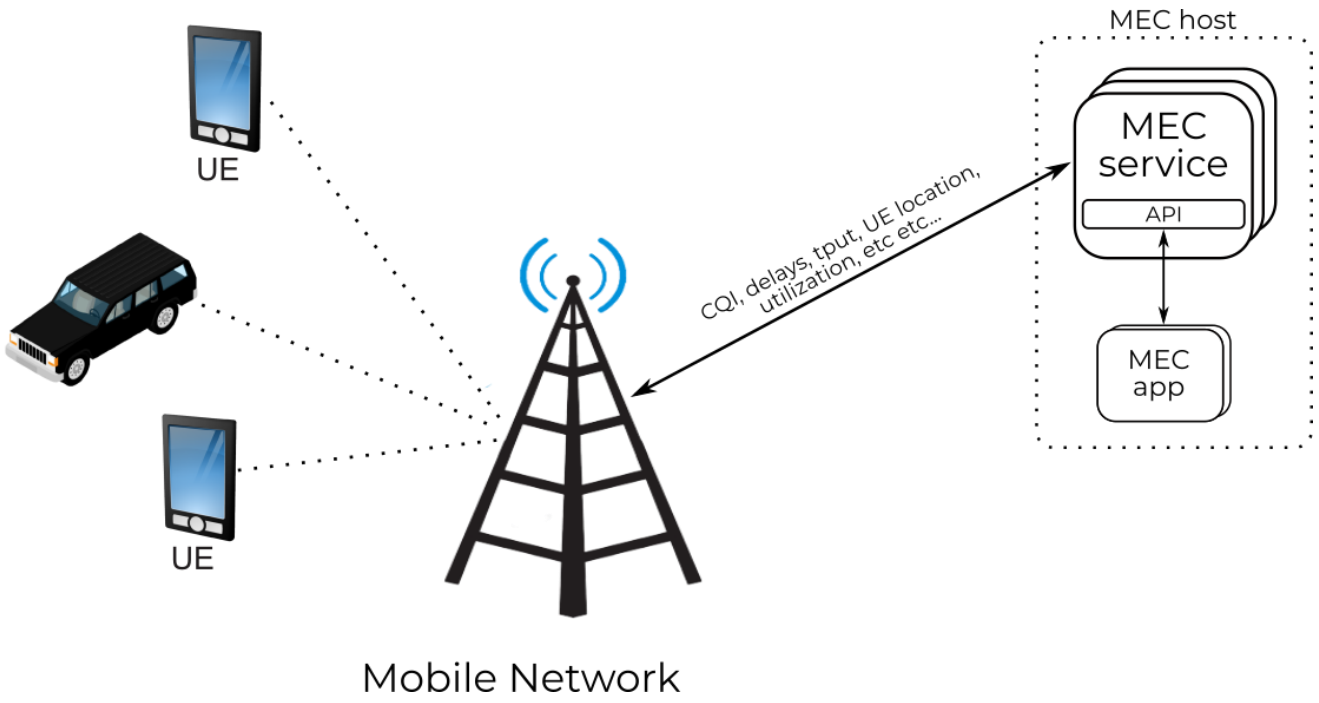
- Contains the MEC services, consumed by the MEC app
- MEC services are discovered by querying the Service registry

MEC platform manager and VIM

Management side of the MEC host level

MEC services

With MEC services, MEC enables the monitoring of **real-time network conditions**.

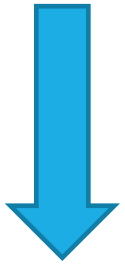


- MEC services provide network information to the MEC app through standardized RESTful APIs
- MEC services are particularly important as they help MEC applications to satisfy the stringent QoS constraints required by the end-users

Idea: tool for MEC apps prototyping

- Implementation of a tool to let MEC app developers to rapidly prototype and test MEC applications in credible and controllable 5G mobile network scenarios.

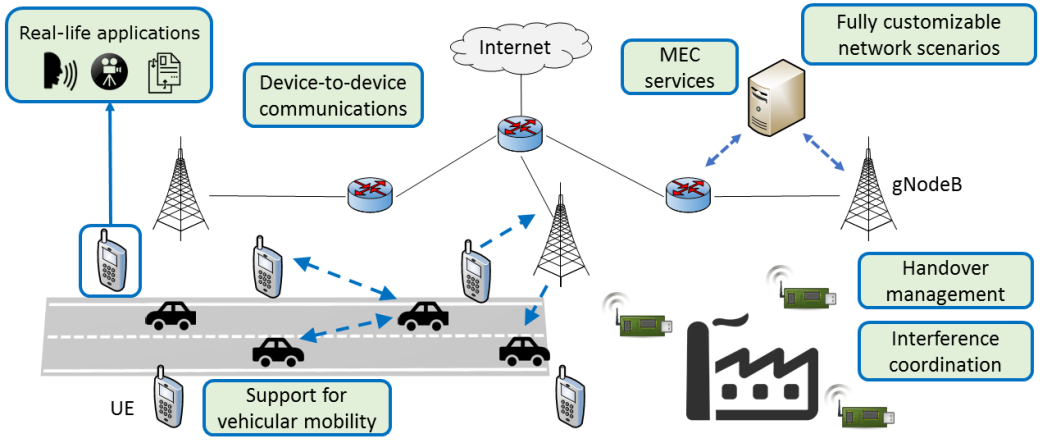
- Provide MEC services able to produce real information from a 5G mobile network and offer them through ETSI compliant API.



Implement a MEC architecture within SIMU5G

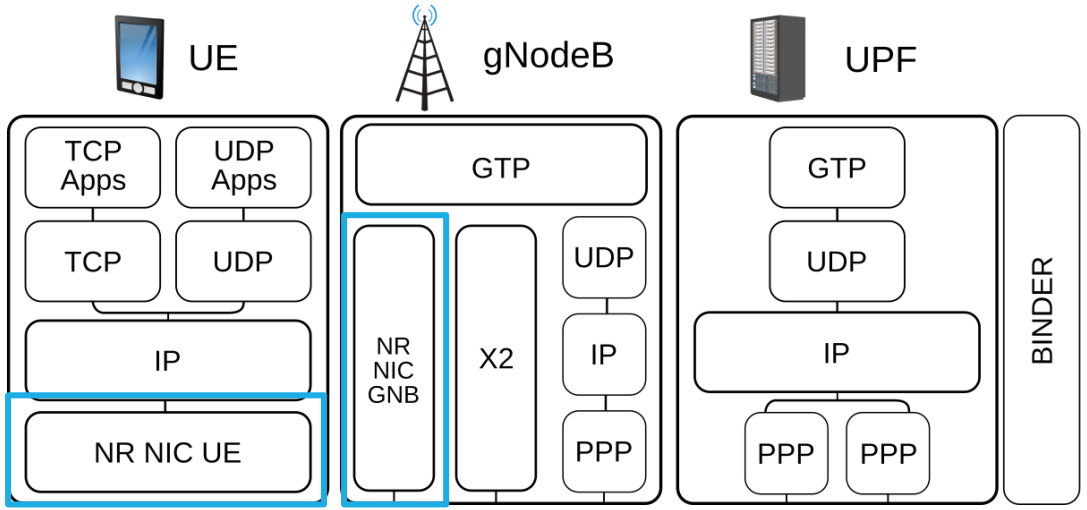
Simu5G overview

www.simu5g.org



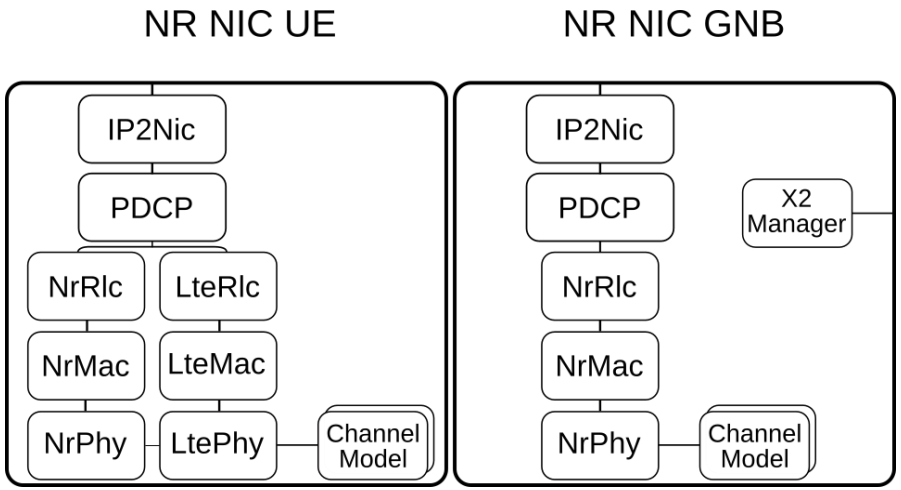
- User-plane simulator of 5G NR and LTE/LTE-A networks (both RAN and CN) based on the OMNeT++ framework
- Full protocol stack of the New Radio Network Interface Card (**NR NIC**) implementation
- X2 communication support
- X2-based handover
- Interoperability with INET's higher-layer protocol modules
- Support for vehicular mobility
- Real-time network **emulation** capabilities

Simu5G main components

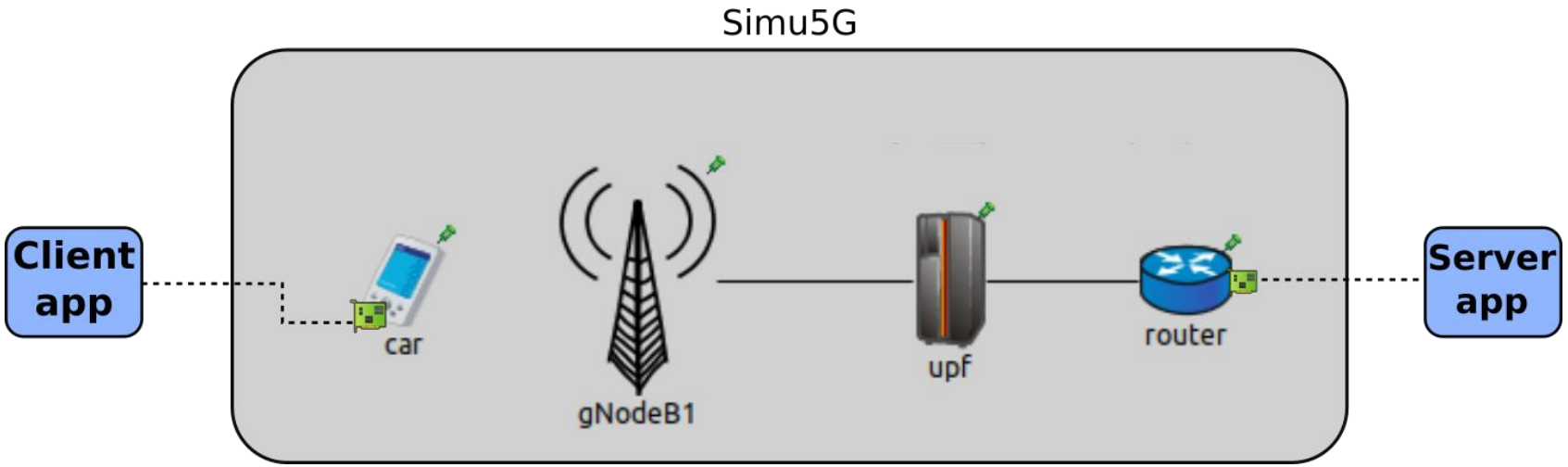


- UPF module implements the GPRS Tunnelling Protocol (GTP) for routing IP datagrams between the gNBs and the data network.
- The Binder module acts as the oracle of the network, accessible by all nodes in the network using direct method calls.

UE and gNBs compound modules are equipped with the NR NIC, containing a simple module for each layer of the NR protocol stack.



Simu5G as an emulator

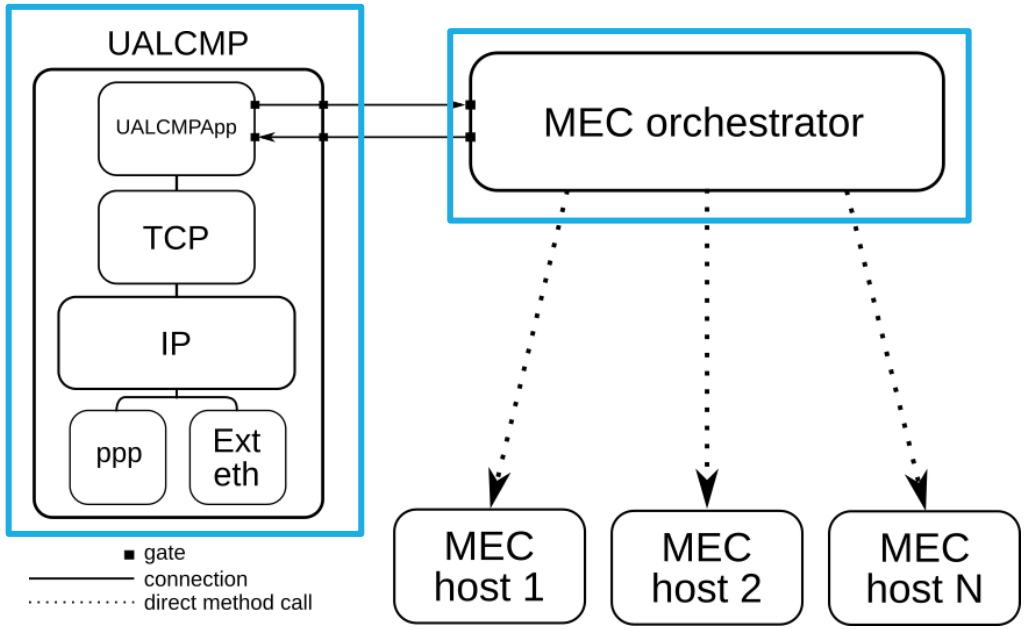


UEs (and any other modules) can be equipped with the ***ExtLowerEthernetInterface*** module provided by the INET library and exchange real package in real time by exploiting the **real-time scheduler** provided by OMNeT++.

Packets exchanged between the two applications will be affected by the conditions of the underlying 5G network

Modelling MEC within Simu5G:

MEC system-level




```
# MEC hosts associated to the MEC system
**.mecOrchestrator.mecHostList = "mecHost1, mecHost2"
```

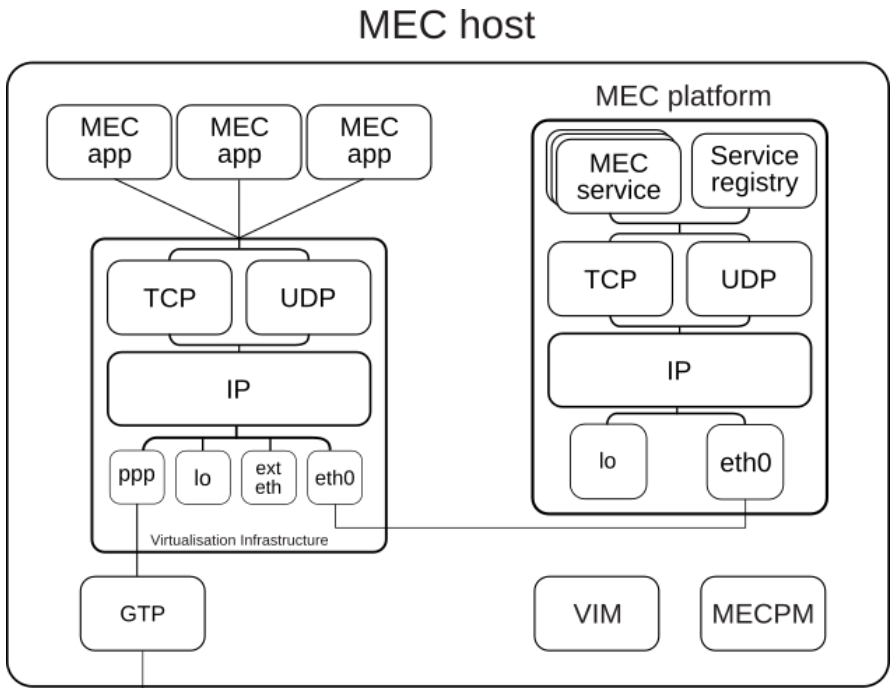
The UALCMP compound module contains an application that provides the RESTful API consumed by the Device app (either internal or external to the simulator) via the Mx2 reference point.

The MEC orchestrator module:

- is connected to the UALCMP via gates
- maintains the list of MEC hosts associated to the MEC system via the ***mecHostsList*** parameter
- selects the most suitable MEC host among those associated with its MEC system
- instantiates and terminates MEC apps modules

Modelling MEC within Simu5G:

MEC host-level



Virtualisation Infrastructure compound module is equipped with the ***ExtLowerEthernetInterface*** module to reach real MEC app running outside the simulator (in emulation mode).

Simulated MEC apps modules are dynamically deployed as applications over the transport-layer of the Virtualization Infrastructure by the VIM.

VIM module keeps trace of the available MEC host resources (RAM, disk, CPU) and of the instantiated MEC apps.

```
# gNBs associated to the MEC Hosts
*.mecHost1.eNBList = "gNodeB1, gNodeB2"
*.mecHost2.eNBList = "gNodeB3"
```

Modelling MEC within Simu5G:

MEC host-level – MEC app implementation

```
moduleinterface IMECApp
{
    parameters:
        @display("i=block/app");
        int mecAppId;

        // service registry endpoint
        string mp1Address;
        int mp1Port;

        int localPort;

        // required resources
        double requiredRam @unit("B");
        double requiredDisk @unit("B");
        double requiredCpu;
}
```

1. Parameters are set by the VIM during instantiation.
   MEC app must be **like IMECApp, IApp**

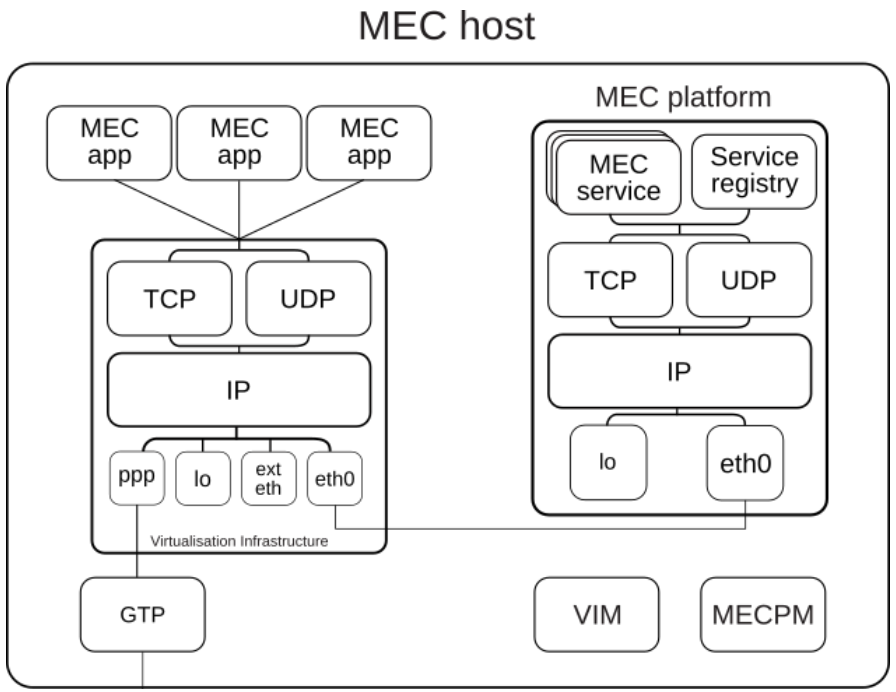
```
{
    "appDId" : "WAMECAPP",
    "appName" : "MECWarningAlertApp",
    "appProvider" : "simu5g.apps.mec.WarningAlert.MECWarningAlertApp",
    "virtualComputeDescriptor" :{
      "virtualDisk" : 10,
      "virtualCpu" : 1500,
      "virtualMemory" :10
    },
    "appServiceRequired" : [{
      "ServiceDependency" :{
            "serName" : "LocationService",
            "version" : "v2",
            "serCategory" : "Location"
      }
    }],
}
```

2. ETSI compliant MEC app descriptor
   Loaded by the MEC orchestrator and used to choose the best MEC host

3. Implement the logic of the MEC app – A *MECAppBase* module is already present in the framework

Modelling MEC within Simu5G:

MEC host-level



MEC platform can also involve real MEC apps when Simu5G runs as an emulator

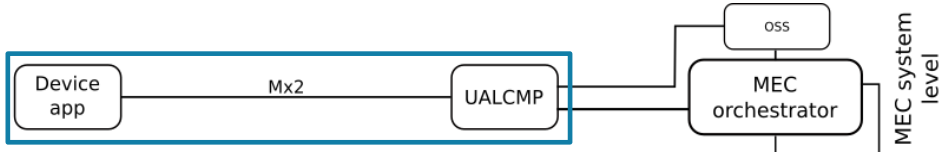
MEC services and Service Registry are implemented as **ETSI compliant** RESTful HTTP server

The framework provides a module called ***MecServiceBase*** and it implements all the non-functional requirements needed for running an HTTP server.

rapidly deploy an ETSI-compliant MEC service by only implementing the HTTP methods

Location Service
Radio Network Information Service

Device App



- Communications to the UALCMP are via HTTP messages according to the ETSI specifications of the Mx2 interface

- Communications with the UE app are not defined by ETSI, but depend on the app developer.

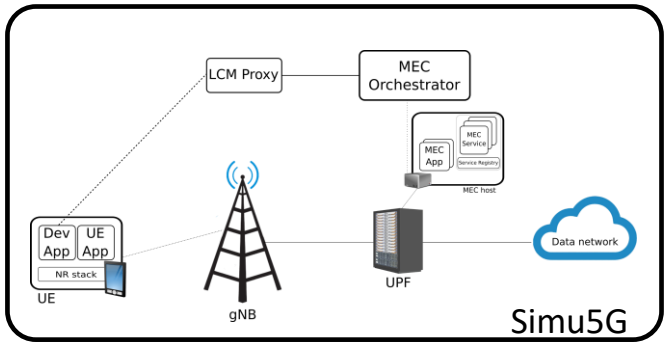
Our basic Device app communicates with the UE app over an UDP socket by means of a simple interface.

DeviceAppPacket
string type

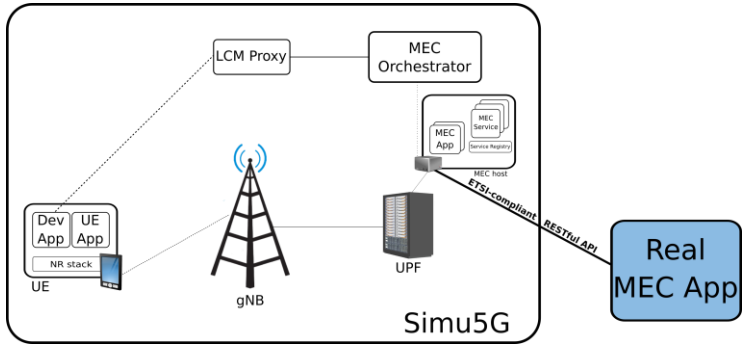
"MecAppStart"	"DeviceAppStartPacket"
"MecAppStop"	"DeviceAppStopPacket"
"MecAppStartAck"	"DeviceAppStartAckPacket"
"MecAppStartAck"	"DeviceAppStartAckPacket"

```
class DeviceAppStartPacket extends DeviceAppPacket
{
    string mecAppDId;
    string mecAppName;
    string appPackageSource;
}
```

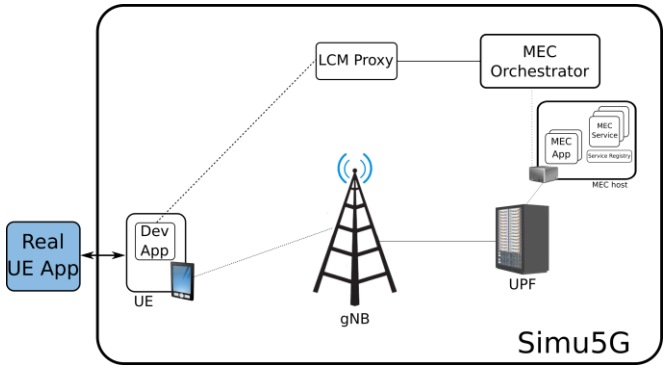
Deployment scenarios



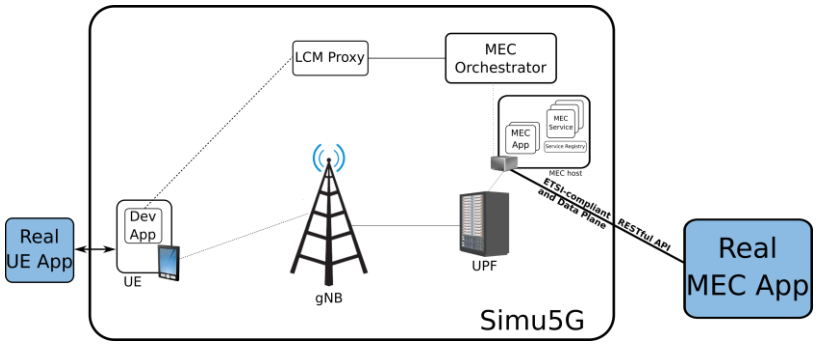
Fully simulated scenario



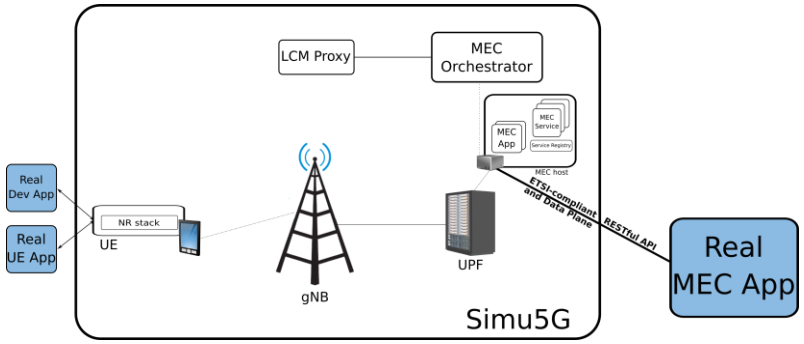
Real MEC app



Real UE app



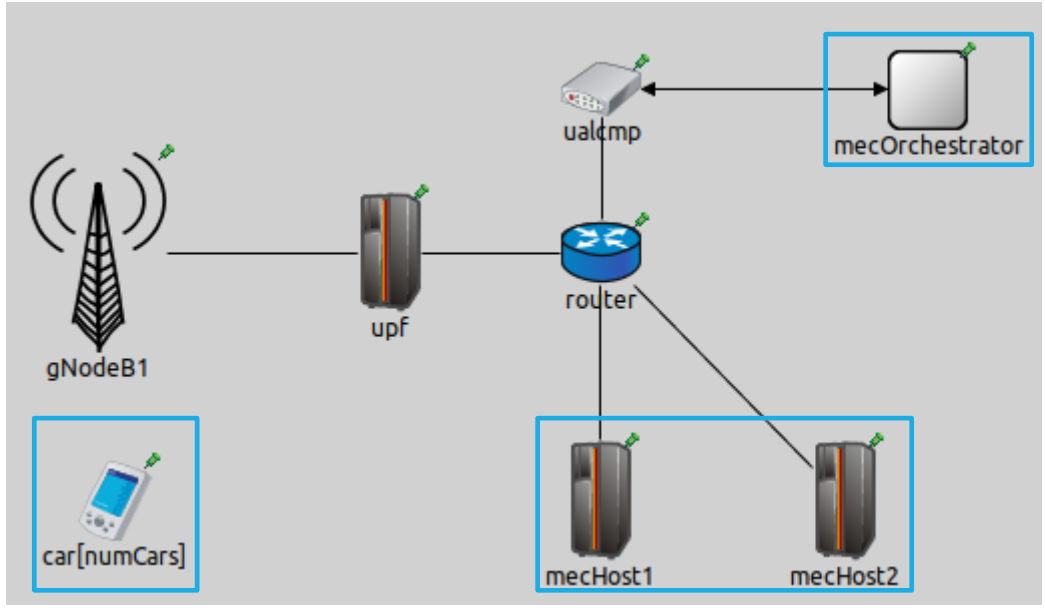
Real UE app – real MEC app



All app endpoints are real

Deployment scenarios:

Running a fully simulated MEC system

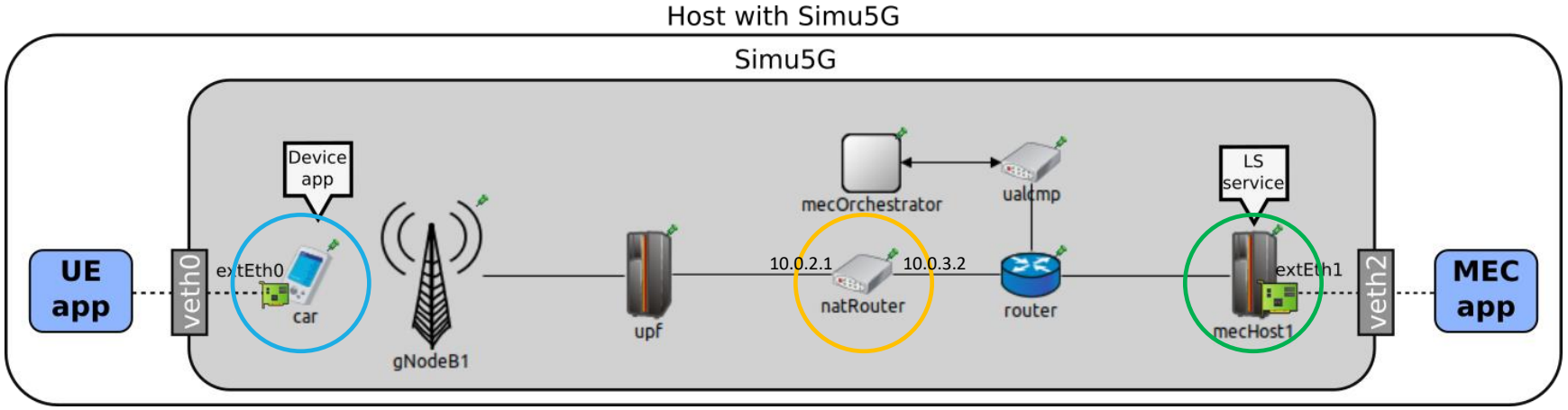



```
#-----------UEWarningAlertApp--------------
*.car[*].numApps = 2

# app[0] is the DeviceApp
*.car[*].app[0].typename = "DeviceApp"
*.car[*].app[0].localPort = 4500
*.car[*].app[0].UALCMPAddress = "ualcmp"
*.car[*].app[0].UALCMPPort = 1000
# app[1] is the UE app
*.car[*].app[1].typename = "UEWarningAlertApp"
# Device App address
*.car[*].app[1].destAddress = "car["+string(ancestorIndex(1))+"]"
# Device App port
*.car[*].app[1].destPort = 4500
```

```
#####################_MEC Hosts Side_####################
# available resources
*.mecHost*.maxMEApps = 100   # max ME Apps to instantiate
*.mecHost*.maxRam = 32GB           # max Ram
*.mecHost*.maxDisk = 100TB   # max Disk Space
*.mecHost*.maxCpuSpeed = 400000     # max CPU
#-------------------------------------
# gNBs associated to the MEC Hosts
*.mecHost*.eNBList = "gNodeB1"

#------REST Services:--------------
# MEC host 2 services configurations
*.mecHost2.mecPlatform.numMecServices = 1
*.mecHost2.mecPlatform.mecService[0].typename = "LocationService"
*.mecHost2.mecPlatform.mecService[0].localAddress = "mecHost2.mecPlatform"
*.mecHost2.mecPlatform.mecService[0].localPort = 10020

*.mecHost2.mecPlatform.serviceRegistry.localAddress = "mecHost2.mecPlatform"
*.mecHost2.mecPlatform.serviceRegistry.localPort = 10021

# MEC hosts associated to the MEC system
**.mecOrchestrator.mecHostList = "mecHost1, mecHost2"
# the MECPM needs to know its MEC orchestrator
**.mecHost*.mecPlatformManager.mecOrchestrator = "mecOrchestrator"
# List of MEC app descriptors to be onboarded during initialization
**.mecOrchestrator.mecApplicationPackageList = "WarningAlertApp"
```

Deployment scenarios:

Running a MEC system in emulation mode




```
*.car.numEthInterfaces = 1
*.car.eth[0].typename = "ExtLowerEthernetInterface"
*.car.eth[0].device = "veth0"
*.car.extHostAddress = "192.168.3.2"
*.car.ipv4.forwarding = true
```

```
############## Ext Interfaces configuration ##############
*.mecHost1.virtualisationInfrastructure.numExtEthInterfaces = 1
*.mecHost1.virtualisationInfrastructure.extEth[0].typename = "ExtLowerEthernetInterface"
*.mecHost1.virtualisationInfrastructure.extEth[0].device = "veth2"
*.mecHost1.virtualisationInfrastructure.ipv4.forwarding = true
```

```
*.natRouter.ipv4.natTable.config = xml("<config> \
                        <entry type='prerouting' \
                        packetDataFilter='*Ipv4Header and destAddress=~10.0.2.1' \
                        srcAddress='10.0.3.2' destAddress='192.168.2.2'/> \
                        <entry type='prerouting' \
                        packetDataFilter='*Ipv4Header and destAddress=~10.0.3.2' \
                        srcAddress='10.0.2.1' destAddress='192.168.3.2'/> \
                        </config>")
```

Deployment scenarios:

Running a MEC system in emulation mode

```
# create virtual ethernet link: veth0 <--> veth1,
#                               veth2 <--> veth3
sudo ip link add veth0 type veth peer name veth1
sudo ip link add veth2 type veth peer name veth3

# veth0 <--> veth1 link uses 192.168.3.x addresses
sudo ip addr add 192.168.3.2 dev veth1
# veth2 <--> veth3 link uses 192.168.2.x addresses
sudo ip addr add 192.168.2.2 dev veth3

# bring up both interfaces
sudo ip link set veth0 up; sudo ip link set veth1 up
sudo ip link set veth2 up; sudo ip link set veth3 up

# UE's eth interface is 192.168.4.1, route for Device app
sudo route add -net 192.168.4.0 netmask 255.255.255.0 dev veth1
# natRouter left interface address is 10.0.2.1
sudo route add -net 10.0.2.0 netmask 255.255.255.0 dev veth1
# natRouter right interface address is 10.0.3.2
sudo route add -net 10.0.3.0 netmask 255.255.255.0 dev veth3
# mecPlatform interface address is 10.0.5.2 (for MEC services)
sudo route add -net 10.0.5.0 netmask 255.255.255.0 dev veth3
```

```
{
    "appDId" : "WAMECAPP",
    "appName" : "MECWarningAlertApp",
    "appProvider" : " ",
    "virtualComputeDescriptor" :{
     "virtualDisk" : 10,
     "virtualCpu" : 1500,
     "virtualMemory" :10
    },
    "appServiceRequired" : [{
     "ServiceDependency" :{
          "serName" : "LocationService",
          "version" : "v2",
          "serCategory" : "Location"
     }
    }],
  "emulatedMecApplication" :{
    "ipAddress": "10.0.2.1",
    "port": 4022
  }
}
```

Conclusions

- Model of a MEC architecture within Simu5G simulator
  - Provide a tool for rapid prototyping of MEC app and MEC services
  - MEC-related app communications occur via ETSI standard APIs
- Running Simu5G in emulation mode allows MEC app developers to test and evaluate real MEC apps running on real hosts
- Configurations needed to run an MEC system — and MEC-related applications — with Simu5G, in both simulation and emulation modes.

The source code can be found on GitHub: www.github.com/Unipisa/Simu5G
For more information: www.simu5g.org/MEC.html

Future work

- Add more ETSI MEC services (e.g. V2X Information Service API – ETSI GS MEC 030)
- Implement other ETSI standard reference points (e.g. Mm3 between MEC system and MEC host level)
- Add MEC apps **migration** support between MEC hosts (Application Mobility Service API - ETSI GS MEC 021)
- Integration with external frameworks like Intel OpenNESS[1]

The source code can be found on GitHub: www.github.com/Unipisa/Simu5G
For more information: www.simu5g.org/MEC.html

[1]www.openness.org

Thank you!

Alessandro Noferi – University of Pisa