OpenAirCN-5G Project Status and Current Activities

OLIVIER CHOISY, B<>COM
FRÉDÉRIC LEROY, B<>COM
LUHAN WANG, BUPT
TIEN-THINH NGUYEN, EURECOM
Outline

• Introduction to OAI-CN 5G project
  – Objectives
  – Overall plan

• 5GC components implementation status
  – AMF, NRF, UPF, SMF
  – Demo
    ○ A work-in-progress implementation for NRF
    ○ A work-in-progress implementation for AMF

• Roadmap
Introduction

• Assets: OpenAir LTE Core Network
  – Rel10/Rel14 Implementation for EPC components: MME, S-P/GW, HSS (plus CUPS implementation)

• Objective: Open Source 5G Core Network implementation
  – Provide an open source implementation of 5GC as specified by 3GPP
  – Partners: Eurecom, b<>com, Bupt, IT Aveiro, KCL, ng4T, Blackned
**Scope of the Implementation**

**Mandatory**
- Access and Mobility Management Function (AMF)
- Session Management Function (SMF)
- User plane function (UPF)
  (with gNB interfaces)

**Other**
- NF Repository function (NRF)
- Authentication Server Function (AUSF)
- Unified Data Management (UDM)

**Not in scope currently**
- Policy Control Function (PCF)
- Network Exposure function (NEF)
- Network Slice Selection Function (NSSF)
On-going Activities

• Implementation of 5G Core SBA components: https://github.com/OPENAIRINTERFACE/openair5g-cn
  – AMF implementation (b<>com, Bupt, Eurecom)
  – NRF implementation (b<>com)
  – SMF implementation (b<>com)
  – UPF implementation (Eurecom, IT Aveiro)
  – UDM, AUSF implementation
• GIT re-organization to publish NRF source code
• Work with ng4T to bring ng4T tester to Eurecom 5G test bench
5GC Component Implementation Status

Service Based Interface (SBI) implementation:
• NFs Interface APIs and Code generation for interfaces
  – automatic interfaces source code generation
  – based on 3GPP specifications (YAML files), Implementation based on
    3GPP Rel 15.2 (December 2018)
  – Status: code generated client and server side for AMF, NRF and SMF
    available
• Core function implementation languages:
  – AMF in C/C++ (AMF stubbed in Java available)
  – SMF in Java
  – NRF in Java
  – UPF in C++
OpenAir5G-CN Project

- Synchro meeting every 2 weeks
- Trello available
Overall Status and Objectives
OSA Objective

- Have a 5G Core implementation to integrate with OSA 5G newRadio implementation on going
- Have a 5G Core Open Source community to build a shared implementation
- Have a reliable 5G Core proposal to be used for testbeds and experimentations
Overall Context for b<>com

- **Control Plane**: SDN EPC, Kubernetes, OpenStack, Bare-Metal
- **User Plane**: UPF, KVM, ONF bridge, Openflow
- **Data Plane**: Infrastructure Network, Tunnels, NAR Web Server

- **5GNR+LTE(WiFi)**: Proxy, NRF, DHCP-S, AMF, SMF
- **N1/N2 SIM**: 3GPP Control Plane
- **RAN**: 3GPP Control Plane
<1> Code Generation

PROCESS

TOOLS EVALUATION
Code Generation Process Overview

- Have a Skeleton of all NFs
- Be able to quickly take into account 3GPP updated specification
Yaml Code Generation Software Environment

- Java environment
  - java11
  - maven 3.5.x
  - thorntail 2.4.0/Microprofile 2.2
    - uberjar file to be run on CLI
    - HTTP/2 enabled
- C++
  - Pistache framework (server side)
  - cpprest-sdk LIB (client side)
  - C++11, GNU g++ 5.4.0, cmake, make
Code Generation Feedback

• RESULTS

REST facade source code fully generated from yaml files.
- 1st implementations in Java & C++ serve responses with random values based on their types and value constraints (range constraints, regex)
- serving error responses (ProblemDetails) on not valid content receipt (malformed json body, invalid type values, constraint violation...)

Work to do is to add business logic on top of the generated skeleton.

• RESTRICTIONS

Need to update 3GPP yaml files because of openapi generator restrictions
- $ref path must start with ./
- not fully OpenApi 3.0 compliant (anyOf, allOf, OneOf & callbacks)
- rewrite definitions to get around generator issues

https://github.com/OPENAIRINTERFACE/openair5g-cn/wiki/Openapi-generator-CPP-starter-kit
NRF Status
NRF Architecture & Functions
NRF Architecture & Functions

Functional implementation available with:
- Error Management implemented
- Tracing with Jaegger implemented
- Metrics in progress (with Prometheus, grafana for rendering)

Next Steps are to move from throntail to quarkus.io, to implement notification management.
**NRF Demo**

Available on internet to be tested

https://5g.labs.b-com.com/

- NRF services (swagger)
  - NNRF-NFM [https://5g.labs.b-com.com/swagger/nnrf-nfm/v1](https://5g.labs.b-com.com/swagger/nnrf-nfm/v1)
  - NNRF-DISC [https://5g.labs.b-com.com/swagger/nnrf-disc/v1](https://5g.labs.b-com.com/swagger/nnrf-disc/v1)

- Jaeger [https://5g.labs.b-com.com:16686/search](https://5g.labs.b-com.com:16686/search)

- AMF services (swagger)
  (Java SBA evaluation)
  - NAMF-MT [https://5g.labs.b-com.com/swagger/namf-mt/v1](https://5g.labs.b-com.com/swagger/namf-mt/v1)
  - NAMF-LOC [https://5g.labs.b-com.com/swagger/namf-loc/v1](https://5g.labs.b-com.com/swagger/namf-loc/v1)
  - NAMF-EVTS [https://5g.labs.b-com.com/swagger/namf-evts/v1](https://5g.labs.b-com.com/swagger/namf-evts/v1)
  - NAMF-COMM [https://5g.labs.b-com.com/swagger/namf-comm/v1](https://5g.labs.b-com.com/swagger/namf-comm/v1)

All exposed API are available programmatically

Implemented SBA compliant NFs may be used to trigger these functions
<3> AMF Status

CONTEXT & FOCUS

2 DEMOS
AMF Implementation Status (1/2)

• Bupt focus on protocols implementation (N1, N2 interfaces from NewRadio), one process target
• B<>com focus on SBA code generation and to provide a micro services based architecture evaluation for AMF, **Cloud Native** (deployed on Kubernetes)

Currently, common activities are:
  – reuse of libraries,
  – share tools
AMF Implementation Status (2/2)

• Available
  – Full SBA Skeleton based
  – Implementation of gNodeb attach (Bupt, b<>com, Eurecom)
  – Implementation of UE register
    ○ Bupt (demo), from gUE to AMF
  – 1 binary for AMF (Bupt)
  – 2 microservices (kubernetes) b<>com

• Next Steps
  – UE register integrated with SMF and UPF (b<>com)
  – Integration with NRF (SMF discover, …)
  – SBA callflows (Bupt)
Future: Enhance software reusability for projects

- **Libngapcodec**
  - Library for coding decoding values of protocol NGAP
  - Based on code generated by modifier asn1c generator
    - [https://github.com/velichkov/asn1c](https://github.com/velichkov/asn1c)
    - 1 bug identified: encoded message is bigger than necessary (lots of 0’s trailing)

- **Libnas**
  - Library for coding/decoding NAS messages from UE.
  - Need to make a fully autonomous library like libngapcodec

- **Libngmessage**
  - Implements higher functions to manipulate NGAP messages
  - C++ implementation
<3.1> AMF Status BUPT Work
To implement AMF, we need:
- Implement N1/N2 interface protocol stack
- Implement AMF SBI interface protocol stack
- And detailed message handling mechanism according to 3GPP specifications

More detailed, we need:
- Build the whole architecture and corresponding system environment
- Build N2/N1 interface protocol stack (SCTP/NGAP/NAS MM)
- Build SBI interface protocol stack
- NGAP messages encoding/decoding library
- NAS messages encoding/decoding library
- Integrate SCTP (open source project) into AMF project
- Create gNB context, UE context, NAS MM context ... to support signaling management
AMF Implementation and Demo

- **Tools**
  - Language: C/C++
  - Linux distribution: Ubuntu 16.04
  - OpenSource projects: OAILOG (from OAI LTE EPC), libconfig, itti, libsctp, libcrypt
- **Architecture**
AMF Implementation and Demo

• N1/N2 interface protocol stack by 4 TASKs (threads)
  - TASK_SCTP: Create listener on port 36412 and send/recv sctp payload
  - TASK_NGAP: Decode/encode NGAP msgs and handle NGAP procedures
  - TASK_NAS_AMF: Decode/encode NAS msgs and handle NAS procedures
  - TASK_AMF_APP: Contexts management and handle time related events

• Layers message brokers
  - For SCTP layer, SCTP_INIT_MSG, SCTP_DATA_REQ
  - For NGAP layer, SCTP_NEW_ASSOCIATION, SCTP_DATA_IND, NGAP_NAS_DL_DATA_REQ
  - For NAS MM layer, NGAP_DATA_IND:
  - For AMF app layer, TIMER_HAS_EXPIRED
AMF Implementation and Demo

- Libngap implementation (Bupt, B<>com, Eurecom)
  - asn1 files from 3GPP TS38.413 specification
  - asn1 compiler (v0.9.29)
  - patch file (b<>com) to make generated c files support aper

- Several NGAP messages using libngap
  - NG SETUP REQUEST, NG SETUP RESPONSE, INITIAL UE MESSAGE, UPLINK NAS TRANSPORT, DOWNLINK NAS TRANSPORT
AMF Implementation and Demo

- Libnas implementation
  - NAS buffer format and corresponding functions

Extended protocol discriminator: octet1
Security header type: octet2
Message authentication code: octet3~6
Sequence number: octet7
Extended protocol discriminator: octet8
Security header type: octet9
Message type: octet10
Other information of one nas msg: octet10~n

Encrypt/decrypt algorithm: NEA1 & NEA2
Integrity algorithm: NIA1 & NIA2
AMF Implementation and Demo

- Libnas implementation
  - Specific NAS MM msgs structure define / encode to buffer / decode from buffer
  - NAS msg define/encode/decode example: REGISTRATION REQUEST

```c
typedef struct registration_request_msg_tag{
  ExtendedProtocolDiscriminator extendedprotocoldiscriminator;
  SecurityModeType securitymode;
  MessageType messagetype;
  5GRegistrationType _5gregistrationtype;
  NASkeySetIdentifier naskeysetidentifier;
  //5GNASIdentity _5gnasidentity;
  uint32_t presenc;
  NASkeySetIdentifier non_current_native_nas_key_set_identifier;
  //5GMMCapability _5gmmcapability;
  //UESecurityCapability uesecuritycapability;
  //NASSAI nsaai;
  //5GTrackingAreaIdentity _5gtrackingareaidentity;
  //SIBUENetworkCapability sibuenetworkcapability;
  UplinkDataStatus uplinkdatastatus;
  PDUSessionStatus pduSessionstatus;
  MECIndication mecindication;
  UEStatus uestatus;
  //5GMultiIdentity AdditionalQCI;
  AllowedPDUSessionStatus allowedpdusessionstatus;
  UEUsageSetting uesettingsetting;
  //5G5GPDOParameters _5gsdpparameters;
  EPSNASMessageContainer epsnasmessagecontainer;
  //LADNIndication ladnindication;
  PayloadContainerType payloadcontainerType;
  PayloadContainer payloadcontainer;
  NetworkSlicingIndication networkSlicingIndication;
  //5G5UpdateType _5gs5updateType;
  NASMessageContainer nasmessagecontainer;
}registration_request_msg;
```

Header, not used here

**All possible Information Elements (IE):**
- `encode_nas_key_set_identifier()`
- `encode_5gmm_capability()`
- `encode_ue_security_capability()`
- `encode__5gs_tracking_area_identity()`
- `encode_s1_ue_network Capability()`
- `encode_uplink_data_status()`
- `encode_pdu_session_status()`
- `encode_registration_request()`
AMF Implementation and Demo

- Libnas implementation
  - NAS MM message IEs define/encode/decode example: NASKeySetIdentifier

```
typedef struct nasKeySetIdentifier {
    #define NAS_KEY_SET_IDENTIFIER_NATIVE 0
    #define NAS_KEY_SET_IDENTIFIER_MAPPED 1
    uint8_t tsc; // type of security context tag
    #define NAS_KEY_SET_IDENTIFIER_NOT_AVAILABLE 0b111
    uint8_t naskeysetidentifier;
} NASKeySetIdentifier;

int encode_u8_nas_key_set_identifier (NASKeySetIdentifier * naskeysetidentifier) {
    uint8_t bufferReturn;
    uint8_t encoded = 0;
    uint8_t tsc = (naskeysetidentifier->tsc & 0x1) << 3 |
                  (naskeysetidentifier->naskeysetidentifier & 0x7);
    encoded++; return bufferReturn;
}

int decode_u8_nas_key_set_identifier (NASKeySetIdentifier * naskeysetidentifier, uint8_t le, uint8_t t, uint32_t value, uint32_t len ) {
    int decoded = 0;
    uint8_t tsc = (*buffer + decoded) >> 3 & 0x1;
    naskeysetidentifier->tsc = (*buffer + decoded) & 0x7;
    decoded++; return decoded;
}
```

Figure 9.11.3.32.1: NAS key set identifier information element
AMF Implementation and Demo

- **Libnas implementation**
  - All implemented NAS MM msgs

<table>
<thead>
<tr>
<th>AUTHENTICATION REQUEST</th>
<th>AUTHENTICATION RESPONSE</th>
<th>AUTHENTICATION FAILURE</th>
<th>AUTHENTICATION REJECT</th>
<th>AUTHENTICATION RESULT</th>
<th>SECURITY MODE COMPLETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGISTRATION REQUEST</td>
<td>REGISTRATION ACCEPT</td>
<td>REGISTRATION COMPLETE</td>
<td>REGISTRATION REJECT</td>
<td>SECURITY MODE COMMAND</td>
<td>SECURITY MODE REJECT</td>
</tr>
</tbody>
</table>
For complete UE registration procedures
- Implemented messages interactions
AMF Implementation and Demo

- For complete UE registration procedures
  - non-UE TNL association
AMF Implementation and Demo

- For complete UE registration procedures
  - NG SETUP procedure
AMF Implementation and Demo

- For complete UE registration procedures
  - UE registration (NAS procedures)
**AMF Implementation and Demo**

- To run this DEMO, we need
  - Source code: [https://github.com/OPENAIRINTERFACE/openair5g-cn](https://github.com/OPENAIRINTERFACE/openair5g-cn)
  - bupt-test branch for amf, bupt_fake_gnb for simulated gNB and UE
  - Install necessary libraries (.build_amf -i)
  - Compile source code (.build_amf -c)
  - To run AMF (sudo build/amf/build/amf)
  - To run gNB & UE (1. scripts/run_gnb 2. scripts/run_ue)

- Running result
  - See Demo presentation
<3.2> AMF Status

b<>com work
SBA NF as CNF

Evolving from VNFs to CNFs
Functional Infrastructure

WEF 2.1 Service based architecture
AMF Microservices Architecture
Microservices Involved for NGSetup
Currently, 3 services implemented

- **N2Link microservice**
  - Keep connection with Amf (optimized for load and speed)
  - Dispatch N1/N2 traffic to other services
  - Docker image size ~10 MB

- **NGSetup microservice**
  - Manage NGSetup procedure for gNB (minimal implementation to do the call flow regardless of NGSetup content, can do NGSetupResponse or NGSetupFailure)
  - Docker image size ~10 MB

- **pygNB simulator** (b<>com private)
  - Minimal simulator for the NGSetup procedure use case (NGSetup implemented only, multi-amf not tested)
  - Based on python pycrate GPL library (hence b<>com won’t redistribute)
  - Docker image size ~80MB
To run this Demo, we need

- Source code: [not yet published to OpenAir5G-CN](git in discussion)
- Kubernetes infrastructure
- b<>com tools and scripts to build dockers images and deploy them

Running result

- See Demo presentation
Roadmap
Roadmap

• First version available Q4 2019/Q1 2020
• Focusing on Functional parts:
  - gNB Attach/Detach (AMF)
  - UE Register/DeRegister without authentication (AMF)
  - UE PDUSessioncreate (AMF, SMF, UPF)
  - NRF main features available

  ➤ gNB interfaces with 5G-Core
  ➤ SBA implementation (UE registered, NRF Features)
  ➤ Tests with ng4T simulator, OAI-5G-RAN
Help Wanted

Implementation end of 2019 will be mostly a POC level

• To increase software maturity
  – Log, persistency, metrics, tracing, HTTP2

• To implement missing NFs
  – Authentication (AUSF)
  – Persistency (UDM, UDR)

• To take into account new 3GPP spec (from Rel 15.3)

• To implement missing Functions
  – Handover
  – Non-Standalone mode (4G Radio)
  – ...