A Cloud-Native Based Access and Mobility Management Function Implementation in 5G Core

Keliang Du*†, XiangMing Wen*†, Luhan Wang*†, Tien-Thinh Nguyen‡
*Beijing Laboratory of Advanced Information Networks, BUPT, Beijing, China
† Beijing Key Laboratory of Network System Architecture and Convergence, BUPT, Beijing, China
‡ Eurecom, Biot, France

***Abstract*—The fifth generation of mobile communication network(5G) core network(5GC) adopts microservice architecture, overturning the original core network architecture, making it easier for 5G networks to realize network slicing, supporting large planning and high-speed terminal access. Many studies have been constructed on 5G technologies, but it is difficult to validate these theories and algorithms in a real-world environment. Therefore, we aim to build the open-source end-to-end(E2E) 5G mobile communication system platform. In this paper, we develop a cloud-native based implementation of Access and Mobility Management Function(AMF) under the OAI 5G Core project. We firstly review the design principles of cloud-native applications and its relationship with serviced 5G core network functions. And then, the designed application architecture for the cloud-native based AMF and detailed implementation of system procedures, e.g. initial registration procedure, are described. In the end, the AMF is tested in a real E2E 5G system for its functionalities. Also, it's deployed in a cloud environment, tested with simulated gNB/UEs, for its stability and concurrency capacity.**

***Index Terms*—5G Core Network; AMF; cloud-native; 5G-AKA; E2E 5G SA system; modular design**

I. Introduction

5G is defined as the converged network, supporting diverse access technologies, heterogeneous networks, and services for various need[1]. As the 5G standards continue to mature, the implementation of 5G technologies has aroused great interests of scholars either in academia or industry field. Recent years, many studies have been constructed on 5G technologies, e.g. Network Slicing. And many of these studies need to be verified and corrected in the real environment. Nowadays, the idea of open source begins to affect the mobile communication field, which helps to provide some open-source platforms for developers and researchers.

In this context, researchers can further validate their theoretical results and apply that into real applications, supported by the open-source tools. However, the real open-source end-to-end 5G mobile communication system in standalone(SA) mode is still not avaliable. OpenAirInterface(OAI) Software Alliance(OSA) is a large open-source community with many contributors and consumers. It focuses on the E2E protocols implementation of the mobile communication network and has developed the E2E Long Term Evolution(LTE) system in the past few years[2]. Now, they are working on the open-source 5G Next Radio(NR) and 5G Core Network(5GC)[3]. And the AMF implementation mentioned in this paper is one of the open-source 5G Core Network projects.

In the 5G/B5G(Beyond 5G) era, due to the convergence of CT(Communication Technology), IT(Information Technology), and DT(Data Technology) technologies, the Internet-related mature technologies are becoming the basis of 5G technology research and implementation. In the Internet field, cloud-native technology is considered to support micro-service architecture naturally in a cloud environment. The concept of cloud-native was proposed by Matt Stine in 2013, which means an application designed for "cloud". And cloud-native applications can naturally support deployment in a cloud environment, leveraging cloud resources. Referring to 3GPP TS 23.501[4], the 5GC is designed as a micro-service architecture, which is borrowed from the concept of servitization of Internet. Hence, the cloud-native based 5GC network functions(NFs) allows for flexible scheduling of resources of cloud, thus enabling self-healing/self-extending/slicing.

In this paper, we develop the cloud-native based AMF, which is the combination of ServiceMesh and service-oriented application. Key functions of AMF in 5G Core are developed, including *Initial registration procedure*, *PDU session establishment procedure*, and so on. Due to the lack of Unified Data Management(UDM) and Authentication Server Function(AUSF), we design the all-in-one 5G-AKA authentication model. Besides, all related algorithms, e.g. Kausf/Kamf derivation and so on, are implemented. And several examples of necessary parameters/test data for the algorithms are given. Moreover, the ITTI-based procedure processing skeleton is designed to improve the concurrency capacity of the system. And detailed protocol stack implementation and message procedures are depicted. In the end, we build one E2E 5G SA system using our own *SMF* and *UPF*, which will also be open source in future, and commercial *Huawei CPE*, *Amari soft gNB* to verify the AMF's functionalities. Moreover, extensive experiments are done to verify the AMF's stability and concurrency capacity supported by simulated UE/gNB(only supports NGAP/NAS protocol, no Air interface protocols).

Our contributions in this paper are as follows:

(1) A cloud-native based AMF architecture is designed and implemented, which can be deployed in the cloud environment, leveraging cloud resources naturally.

(2) Modular design is applied in the development, including data modules(e.g. ue context), functional modules(e.g. authentication algorithms), and two libraries(libnas/libngap).

(3) A simualted UE/gNB is developed, supporting N-

TABLE I: The Cloud-based Software Development Model: 12 *Factor*

	Base Code	Dependence	Configuration	Backend Service	Build-Release-Run	Process	Port Binding	Concurrency	Easy Handling	Environmental Equivalence	Log	Management Process
Cloud-Native Applications	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Cloud-Native Based AMF	✓	✓	✓			✓	✓	✓		✓	✓	
Service Mesh or Kubernetes	✓	✓	✓	✓	✓	✓			✓	✓		✓

GAP/NAS protocols, to simulate multi-terminal registration scenarios.

The remainder of the paper is organized as follows. Section II describes the cloud-native based AMF overall architecture. Section III describes the detailed implementation of the AMF's functionalities. Section IV describes two testbeds, the actual E2E 5G SA system and virtualized environment based on simulated UE/gNB, to verify its features and performance. Section V concludes this paper and future research directions.

II. Cloud-Native Based AMF Architecture

The main design principle of the cloud-native based AMF is the decouple between service and management, allowing functional logic to evolve and optimize independently. Hence, the features of 5G core network micro-service architecture shall be supported by some common software infrastructures, e.g. ServiceMesh. In 5GC, the network functions can be deployed using dockers or virtual machines over the cloud infrastructure. These are similar to the cloud-native based model, which is composed of IaaS(Infrastructure as a Service), PaaS(Platform as a Service), and SaaS(Software as a Service). For the design of cloud-native applications, Heroku collated and proposed the *12 Factor* to help developers design cloud-native applications. From TABLE I, the combination of ServiceMesh and service-oriented AMF design also meets the *12 Factor*.

In this context, we design the cloud-native based AMF architecture referring to the principles for cloud-native applications over the Internet, which is depicted in Fig. 1. The cloud infrastructure(IaaS) can be provided by some public clouds, e.g. Tencent Cloud, and/or private cloud built with OpenStack. The management platform(PaaS) can be supported by kubernetes(K8s), providing/managing basic dockers and containers for each micro-service. ServiceMesh, one software infrastructure(SaaS), can provide the features like service registration/discovery/load balance for micro-services.

In such a software environment, we design the internal architecture of procedures' processing. Pistache[5] is used to implement service-based interface(SBI) protocol s-tack. It's a high-performance REST toolkit written in C++ and can provide user-friendly interfaces of endpoints. In this paper, service operations, e.g. Namf_Communication_-N1N2MessageTransfer, are implemented in the Pistache way.

For N1/N2 protocol stack implementation, InterTaskInterface(ITTI) is used as the main skeleton of system procedures' processing. Each protocol instance has its own task, which is implemented by one thread. For example, TASK_AMF_N1 is the implementation of NAS protocol.

Besides, the modular design is applied to the implementation of the functional and data modules, which can make each module evolve independently. Contexts for recording communication status, database for storing user subscription information, and configuration file for setting necessary parameters of the network are included in the common data layer. And functional modules in the uppermost layer are responsible for some logical processing of NAS or NGAP procedures.

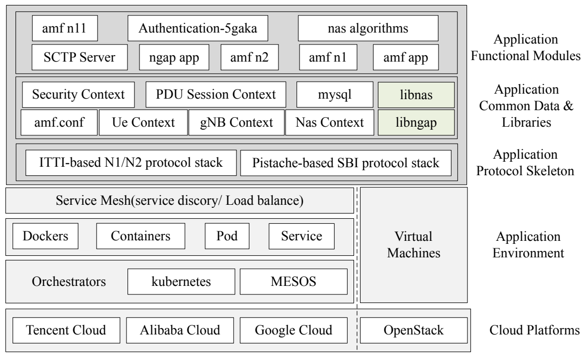


Fig. 1: The Cloud-Native Based AMF Architecture

III. Functional Procedures Implementation

In order to further decouple common modules from functional procedures, we firstly design a set of libraries, including libnas and libngap. Then, we implement the procedure processing skeleton using ITTI and Pistache technologies. Besides, we re-design the primary authentication and key agreement model to authenticate the UEs in an all-in-one manner. And we provide an example of setting the parameters of the algorithms involved in the authentication procedure.

A. Pistache-Based AMF Services Generation

Referring to the 3rd Generation Partnership Project(3GPP) standards, the AMF's services are described by yaml files, e.g. *TS29518_Namf_Communication.yaml*, whose syntax follows the openapi3.0 specification[6]. And OpenApiTools[7] is such a tool to generate different server stubs from openapi3.0 specifications. In this way, the skeleton of the AMF, Pistache Server, can be generated with yaml files using OpenApiTools.

B. Itti-Based Procedure Processing Skeleton

Itti(InterTaskInterface) is a lightweight middleware which can provide services to the application. Services are Timer facilities, Asynchronous Inter-task message facilities and I/O events facilities. Each protocol instance or interface adapter has been assigned with its own itti task.

Registration management[8] is the most important functionality in the AMF, capable of managing and processing the UE(s) registration requests. Its logic processing can be divided into leveled sub-procedures. For NGAP level, they are *ng setup procedure, initial ue message procedure and uplink/downlink nas transport procedure*[9]. For NAS level, they are *registration procedure, primary authentication and key agreement procedure, identification procedure*[10]. And, not only can itti-based skeleton allow for a clear division of functional modules, but it also allows for the simultaneous processing of multiple UE registration requests, taking advantage of its high concurrency and asynchronous communication mechanism features. Hence, we redesigned the registration management procedure in the way of the itti mechanism. It can be seen in Fig. 2.

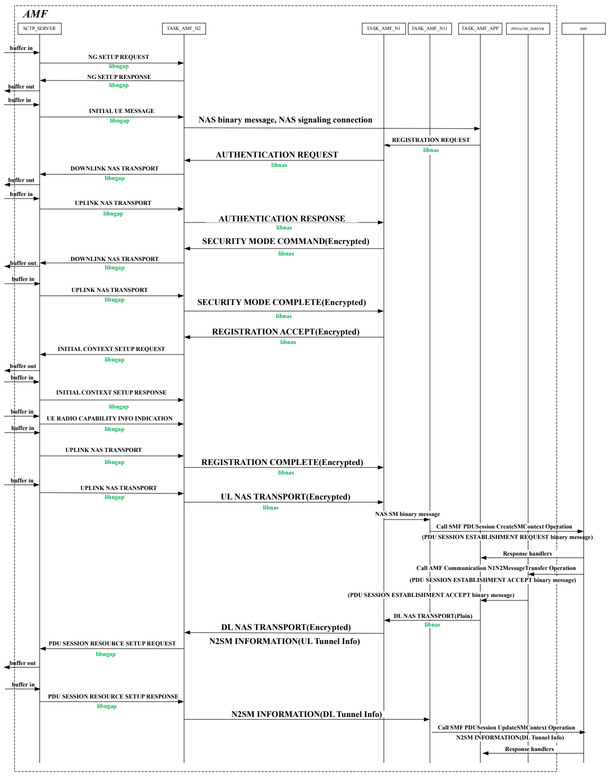

AMF
SCTP_SERVER
TASK_AMF_N2
TASK_AMF_N1
TASK_AMF_N11
TASK_AMF_APP
PISTACHE_SERVER
SMF
buffer in
NG SETUP REQUEST
libngap
NG SETUP RESPONSE
libngap
buffer out
buffer in
INITIAL UE MESSAGE
libngap
NAS binary message, NAS signaling connection
REGISTRATION REQUEST
libnas
AUTHENTICATION REQUEST
libnas
DOWNLINK NAS TRANSPORT
libngap
buffer out
buffer in
UPLINK NAS TRANSPORT
libngap
AUTHENTICATION RESPONSE
libnas
SECURITY MODE COMMAND(Encrypted)
libnas
DOWNLINK NAS TRANSPORT
libngap
buffer out
buffer in
UPLINK NAS TRANSPORT
libngap
SECURITY MODE COMPLETE(Encrypted)
libnas
REGISTRATION ACCEPT(Encrypted)
libnas
INITIAL CONTEXT SETUP REQUEST
libngap
buffer out
buffer in
INITIAL CONTEXT SETUP RESPONSE
libngap
buffer in
UE RADIO CAPABILITY INFO INDICATION
buffer in
libngap
UPLINK NAS TRANSPORT
libngap
REGISTRATION COMPLETE(Encrypted)
libnas
buffer in
UPLINK NAS TRANSPORT
libngap
UL NAS TRANSPORT(Encrypted)
libnas
NAS SM binary message
Call SMF PDUSession CreateSMContext Operation
(PDU SESSION ESTABLISHMENT REQUEST binary message)
Response handlers
Call AMF Communication N1N2MessageTransfer Operation
(PDU SESSION ESTABLISHMENT ACCEPT binary message)
(PDU SESSION ESTABLISHMENT ACCEPT binary message)
DL NAS TRANSPORT(Plain)
libnas
DL NAS TRANSPORT(Encrypted)
N2SM INFORMATION(UL Tunnel Info)
PDU SESSION RESOURCE SETUP REQUEST
libngap
buffer out
buffer in
PDU SESSION RESOURCE SETUP RESPONSE
libngap
N2SM INFORMATION(DL Tunnel Info)
Call SMF PDUSession UpdateSMContext Operation
N2SM INFORMATION(DL Tunnel Info)
Response handlers

Fig. 2: Itti-Based Procedure Skeleton

As shown, the functionalities of each task are to receive/send SCTP buffer, to process NGAP-level procedures, to process NAS-level procedure, to consume other NF services, to manage the whole system, to expose the AMF capabilities from the left to right. Each task runs independently, maintaining one message queue. The external signaling messages, e.g. NAS/NGAP messages, can be encoded/decoded by the support of libnas/libngap. Also, itti messages with unique data struct and id are transported among itti tasks.

C. Modular Design: Libngap and Libnas

Modularly designed libngap and libnas are developed to encode/decode binary information, providing external interfaces for the procedure skeleton. Referring to 3GPP TS 38.413[9], NGAP messages are defined by Abstract Syntax Notation One(ASN1), which can be generated by the ASN1 compiler(ASN1C). For NAS messages, 3GPP TS 24.501[10] describes the buffer format for all messages. Each message is designed as one object, providing core-type information elements(IEs) and external interfaces.

D. All-In-One 5G-AKA Authentication Procedure

The AMF implements the primary authentication and key agreement procedure in an all-in-one manner, which should have been related to the AMF, AUSF, and UDM. For 5G cases, this AMF supports the 5G-AKA authentication procedure[11].

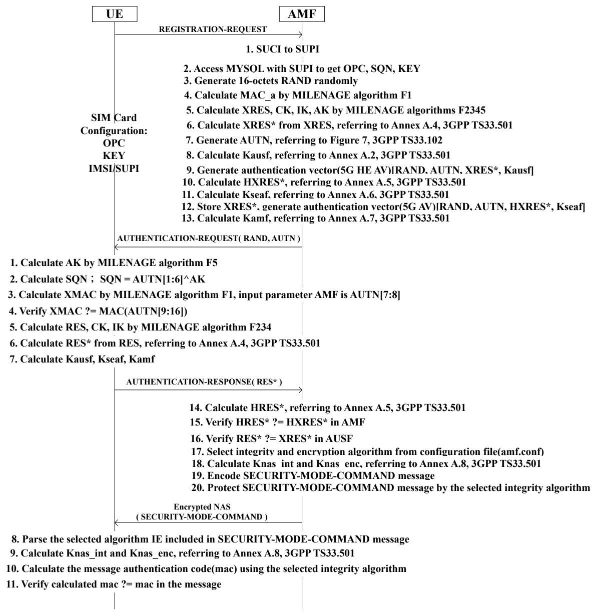

UE
AMF
REGISTRATION-REQUEST
1. SUCI to SUPI
2. Access MYSQL with SUPI to get OPC, SQN, KEY
3. Generate 16-octets RAND randomly
4. Calculate MAC_a by MILENAGE algorithm F1
5. Calculate XRES, CK, IK, AK by MILENAGE algorithms F2345
6. Calculate XRES* from XRES, referring to Annex A.4, 3GPP TS33.501
7. Generate AUTN, referring to Figure 7, 3GPP TS33.102
8. Calculate Kausf, referring to Annex A.2, 3GPP TS33.501
9. Generate authentication vector(5G HE AV)[RAND, AUTN, XRES*, Kausf]
10. Calculate HXRES*, referring to Annex A.5, 3GPP TS33.501
11. Calculate Kseaf, referring to Annex A.6, 3GPP TS33.501
12. Store XRES*, generate authentication vector(5G AV)[RAND, AUTN, HXRES*, Kseaf]
13. Calculate Kamf, referring to Annex A.7, 3GPP TS33.501
SIM Card Configuration:
OPC
KEY
IMSI/SUPI
AUTHENTICATION-REQUEST(RAND, AUTN)
1. Calculate AK by MILENAGE algorithm F5
2. Calculate SQN； SQN = AUTN[1:6]^AK
3. Calculate XMAC by MILENAGE algorithm F1, input parameter AMF is AUTN[7:8]
4. Verify XMAC ?= MAC(AUTN[9:16])
5. Calculate RES, CK, IK by MILENAGE algorithm F234
6. Calculate RES* from RES, referring to Annex A.4, 3GPP TS33.501
7. Calculate Kausf, Kseaf, Kamf
AUTHENTICATION-RESPONSE(RES*)
14. Calculate HRES*, referring to Annex A.5, 3GPP TS33.501
15. Verify HRES* ?= HXRES* in AMF
16. Verify RES* ?= XRES* in AUSF
17. Select integrity and encryption algorithm from configuration file(amf.conf)
18. Calculate Knas_int and Knas_enc, referring to Annex A.8, 3GPP TS33.501
19. Encode SECURITY-MODE-COMMAND message
20. Protect SECURITY-MODE-COMMAND message by the selected integrity algorithm
Encrypted NAS (SECURITY-MODE-COMMAND)
8. Parse the selected algorithm IE included in SECURITY-MODE-COMMAND message
9. Calculate Knas_int and Knas_enc, referring to Annex A.8, 3GPP TS33.501
10. Calculate the message authentication code(mac) using the selected integrity algorithm
11. Verify calculated mac ?= mac in the message

Fig. 3: All-in-one 5G-AKA authentication procedure and algorithms

Fig. 3 shows detailed information about 5G-AKA authentication procedure. F1/F2/F3/F4/F5 are the key algorithms for the whole procedure, which shall be consistent both in UE and UDM. It is up to the operator to decide exactly how these algorithms will be implemented. In this paper, we choose "MILENAGE" as the kernel function[13]. Using F12345, MAC, CK, IK, AK, RES/XRES can be calculated[12]. During the procedure, the AMF will derive several keys, e.g. Kausf, and calculate the expected authentication response parameter(XRES*). HMAC-SHA-256[14] is used as the Key Derivation Function(KDF).

5G-AKA is a dual-authentication procedure and the UE firstly decides whether to trust the network after receiving the RAND and AUTN parameters. The UE may reject the network by including "synch failure" or "MAC failure" in the AuthenticationFailure message(NAS message). It means either the authentication token(AUTN) is not fresh, or the calculated MAC is not equal to that received. In this case, the input parameters for MILENAGE algorithms or algorithms themselves are to be checked.

After the authentication procedure, one security context has been established both in the UE and AMF. To take this context into use, another procedure, sercurity mode control(SMC) procedure, is triggered. During the SMC procedure, all transported NAS meesages shall be protected. The chosen integrity protection and encryption algorithms shall match the UE's security capabilities. This AMF supports 128-5G-NIA0/128-5G-NIA1/128-5G-NIA2 and 128-5G-NEA0 algorithms[15].

There is an example of some important parameters set during the whole 5G-AKA and SMC procedure. Given that binary SUCI is $0x64F0110000000001032547698$. Then the SUPI is the IMSI format[16], which is "460110123456789". And the serving network name(SNN) shall be "5G:mnc011.mcc460.3gppnetwork.org"[10]. Both SUPI and SNN shall be encoded into OCTET-STRING using ASCII code. And for integrity protection algorithms input parameters, the BEARER shall be 0x01 and the message shall be the concatenation of sequence number and plain NAS message buffer. **DATA 1** is an overall test data for the whole procedure, which can be used to correct each algorithm.

DATA 1 Test Data for 5G-AKA and SMC procedure

```
(1) Given OPC, KEY, RAND and SQN $\bigoplus$ AK
OPC: $0x000102030405060708090a0b0c0d0e0f$
KEY: $0x00112233445566778899aabbccddeeff$
RAND: $0x6a8959fb188c73308d679f7bc8313d65$
SQN $\bigoplus$ AK: $0x97779b305686$
(2) XRES* Derivation
Input S: $0x6b35473a6d6e633031312e6d63633436302e$
$336770706e6574776f726b2e6f726700206a8959fb188$
$c73308d679f7bc8313d6500106283ace5e894a0ad0008$
Key: $0x12757da1c0747d4a88b7d8b86446244b7f64a6$
$ccf95b98b25e9a41f007037d86$
XRES*: $0x03f8627a004484086f35398f7c56df32$
(3) Kausf Derivation
Input S: $0x6a35473a6d6e633031312e6d63633436$
$302e336770706e6574776f726b2e6f72670020$
$97779b3056860006$
Key: $0x12757da1c0747d4a88b7d8b86446244b7f64a6$
$ccf95b98b25e9a41f007037d86$
Kausf: $0x117cc3da749fb0b92c6fc4f4547a1e7af9$
$499391028d80d75bfe88eb813ead4c$
(4) Kseaf Derivation
Input S: $0x6c35473a6d6e633031312e6d636334$
$36302e336770706e6574776f726b2e6f72670020$
Key: $0x117cc3da749fb0b92c6fc4f4547a1e7af9$
$499391028d80d75bfe88eb813ead4c$
Kseaf: $0x4c888c8df5c4e76f7400f14503276a8a3$
$e35bfd9ac8b136c2768189a5a116d10$
(5) Kamf Derivation
Input S: $0x6d343630313130313233343536373839$
$000f00000002$
Kamf: $0x20583705e4710990f71de4dbfcc2309c7d$
$baf2019bd3aae407c8749ba5a72155$
(6) Knas_int Derivation for 128-5G-NIA1
Input S: $0x69020001010001$
Knas_int: $0x9af45e60d402b8674c4f4767bd737e0f$
(7) 128-5G-NIA1 Algorithm Parameters
bearer: $0x01$
direction: $0x0$ for uplink, $0x1$ for downlink
message: $0x007e005d010102f0f0e1360102$
Output mac: $0x33054cb4$
```

E. Function Set and Common Data

Aiming to improve the portability, the contexts and logical functions are designed as modules. Each module is equipped with one or more related functionalities. For example, *class amf_n2* is designed to encode/decode NGAP messages and process the corresponding message procedures. The modular contexts are used to manage the system procedures clearly, each with unique keys. The mapping of the contexts and keys are listed in TABLE II. Besides, some supportable features of the AMF are configured in *amf.conf*. And the UEs' subscription information, e.g. Key and OPc, is stored in the local MYSQL database, OAI_DB.

TABLE II: Mapping of the contexts and keys

Contexts	**Keys**	**Descriptions**
gnb_context	*sctp_assoc_id*	During the NG Setup procedure
	Global gNB ID	After the NG Setup procedure
nas_context	*5G-GUTI*	After RegistrationAccept message
	5G-S-TMSI	After the initial registration procedure
	AMF_UE_NGAP_ID	During the initial registration procedure
nas_security_context	*ngKSI*	During the authentication and SMC procedures
pdu_session_context	*SUPI/IMSI and pduSessionId*	During the UlNasTransport procedure for N1 SM message

IV. Funtionalities and Performance Analysis

In this section, the real end-to-end(E2E) 5G standalone(SA) testbed is discussed for testing the AMF's functionalities. Moreover, extensive experiments are designed to analyze the system performance. In this case, the AMF is deployed in a cloud-native environment and simulated UE/gNB is used.

A. Testbed: Deployment and Functional Testing

As shown in Fig. 4, the E2E system is composed of UE, gNB, AMF, SMF, UDM, and UPF. Huawei CPE, one commercial 5G user equipment, supports SA/NSA mode. And the USIM card is programmable. In this testbed, we use the SA mode and the card is configured the same information with that in MYSQL, in the AMF. The soft gNB supports 5G NR and is equipped with two Network Interface Controllers(NICs) for signaling and data transportation respectively. The SMF and UDM are the open-source projects of OSA, deployed in the virtual machines. The UPF is also an element of the open source 5G core project, supporting full-stack PFCP protocols and data plane establishment. In this testbed, Huawei CPE can be registered in the network with the AMF. And also, the AMF can transport NAS session management(SM) messages from the CPE to the SMF transparently.

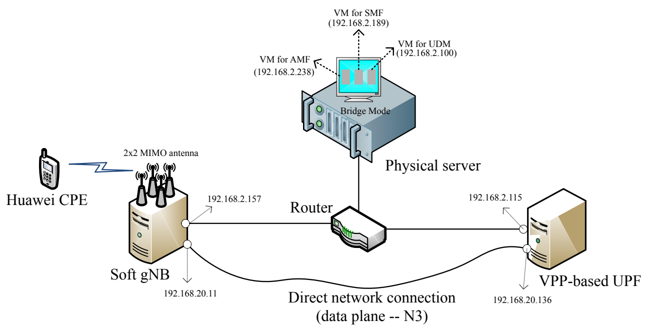
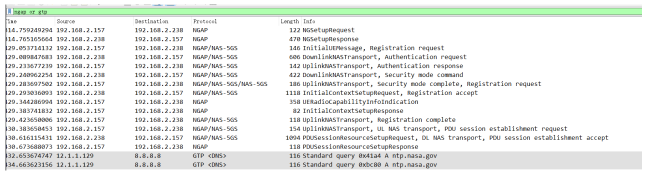


Fig. 4: Testbed for the AMF Functionalities



```
ngap or gtp
Time         Source         Destination    Protocol                Length Info
314.759249294 192.168.2.157 192.168.2.238  NGAP                       122 NGSetupRequest
314.765165664 192.168.2.238 192.168.2.157  NGAP                       470 NGSetupResponse
429.053714132 192.168.2.157 192.168.2.238  NGAP/NAS-5GS               146 InitialUEMessage, Registration request
429.089847683 192.168.2.238 192.168.2.157  NGAP/NAS-5GS               606 DownlinkNASTransport, Authentication request
429.233677239 192.168.2.157 192.168.2.238  NGAP/NAS-5GS               142 UplinkNASTransport, Authentication response
429.240962254 192.168.2.238 192.168.2.157  NGAP/NAS-5GS               422 DownlinkNASTransport, Security mode command
429.283697502 192.168.2.157 192.168.2.238  NGAP/NAS-5GS/NAS-5GS       186 UplinkNASTransport, Security mode complete, Registration request
429.293036093 192.168.2.238 192.168.2.157  NGAP/NAS-5GS              1118 InitialContextSetupRequest, Registration accept
429.344286994 192.168.2.157 192.168.2.238  NGAP                       358 UERadioCapabilityInfoIndication
429.383741832 192.168.2.157 192.168.2.238  NGAP                        82 InitialContextSetupResponse
429.423650006 192.168.2.157 192.168.2.238  NGAP/NAS-5GS               118 UplinkNASTransport, Registration complete
430.383650453 192.168.2.157 192.168.2.238  NGAP/NAS-5GS               154 UplinkNASTransport, UL NAS transport, PDU session establishment request
430.616115431 192.168.2.238 192.168.2.157  NGAP/NAS-5GS              1094 PDUSessionResourceSetupRequest, DL NAS transport, PDU session establishment accept
430.673688073 192.168.2.157 192.168.2.238  NGAP                       118 PDUSessionResourceSetupResponse
432.653674747 12.1.1.129    8.8.8.8        GTP <DNS>                  116 Standard query 0x41a4 A ntp.nasa.gov
434.663623156 12.1.1.129    8.8.8.8        GTP <DNS>                  116 Standard query 0xbc80 A ntp.nasa.gov
```

Fig. 5: Wireshark packets for the AMF functional test

For the AMF's functional test, the results are given as a *pcap* file. We can see from Fig. 5, the AMF supports the NG Setup procedure, exchanging application-level data needed with the gNB node. For the UE-requested registration procedure, the AMF supports the authentication and security mode control sub-procedures. After successful registration, the UE will request to establish a PDU session for data traffic. That will be supported by the DL/UL NAS transport procedure of the AMF. *UL-NGU-UP-TNLInformation* and *dLQosFlowPerTNLInformation* can be found in PDUSessionResourceSetupRequest and PDUSessionResourceSetupResponse messages respectively. In this case, the UE can enjoy downlink/uplink data traffic.

B. System Performance: Experiments and Analysis

Unlike the functional test in the real E2E system, we analyze its performance with simulated gNB and UEs. The simulated gNB/UE supports only NGAP and NAS protocols, without air signalings. And the AMF is deployed in a cloud-native environment with the ability to customize host configurations on demand. In this context, extensive experiments are done to check the AMF's capabilities.

(1) High Stability

In this experiment, the AMF is deployed in a virtual machine with one-core CPU. And 1000 UEs initiated registration requests(RRs) at 10us intervals(equally, RR rate is $10^5/s$). Fig. 6 depicts the life cycle for one UE's registration request. There are 16 signaling interactions and each one shall be processed successfully to finish the registration procedure. The AMF can handle 16×1000 signalings under 1000 RRs with basic hardware configuration(one-core CPU), showing its high stability.

(2) Concurrency Capacity V.S. Virtual Machine Configuration

In this experiment, the AMF is deployed in virtual machines with different number of CPU cores. Also, the RR rate is $10^5/s$ and total RRs are 1000. We are using the time interval after one RR is sent and one registration accept message is

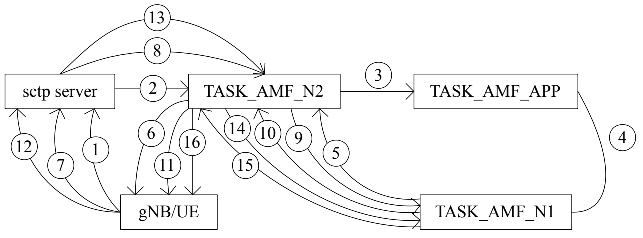


Fig. 6: Life cycle for one UE's registration request

received at the simulated gNB/UE as one registration time. Fig. 7 shows the impact on the registration time with different number of CPU cores. As shown, 1) registration time trends are consistent and it will peak in the central portion of UEs; 2) with more CPU cores, the arrival of the peak registration time will be delayed and overall registration times are much lower; 3) no significant difference in registration time beyond eight-cores CPU. From the results, we can improve the concurrency capacity by adding more CPU cores for the virtual machine.

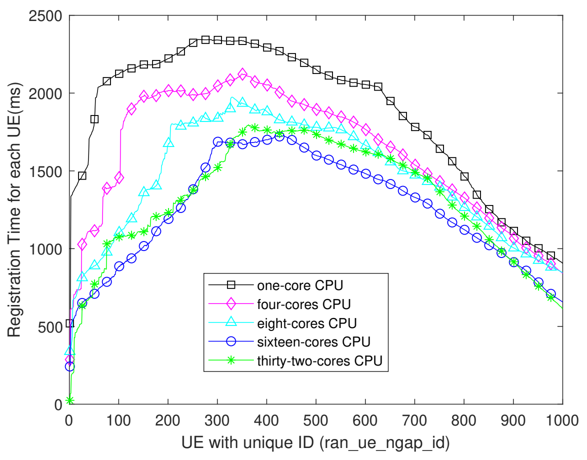


Fig. 7: Registration Time Distribution Under different number of CPU cores

(3) Registration Time Consumption V.S. RR rate

In this experiment, the AMF is deployed in a virtual machines with one-core CPU. We analyzed the effect of different registration request(RR) rates on the registration time under 1000 registration requests. From the analysis in Fig. 8, the registration request rate greatly affects the registration time. When the request rate drops, the registration time drops significantly.

(4) Average Registration Time Consumption V.S. Total Registration Requests at unit time interval

In this experiment, we deploy the AMF in a virtual machine with 32-cores CPU. We count the average registration time for different RR rates and RRs, as shown in Fig. 9. This experiment simulates a population distribution scenario with different densities, e.g. the high-density crowd scenes in Tiananmen Square. When the RRs is the same, the shorter the request time interval, the longer the average registration time. However, when the request interval is around 500us($2 \times 10^3 RRs/s$), the average registration time under different RRs is almost the same.

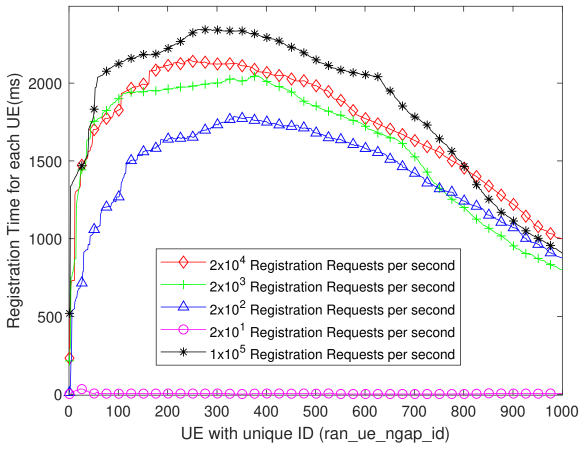


Fig. 8: Registration Time Distribution Under different Registration Request Rates

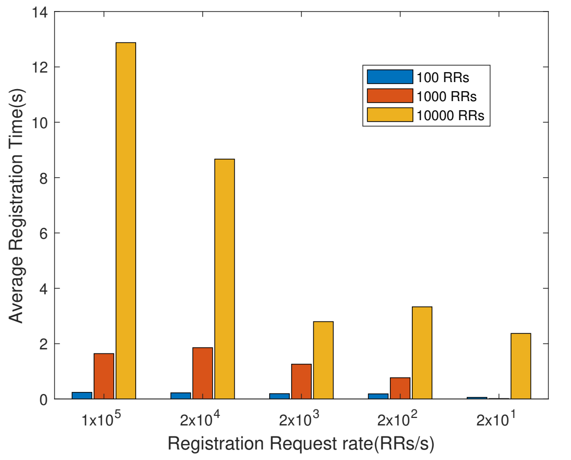


Fig. 9: Average Registration Time Distribution Under 100/1000/10000 Registration Requests

In summary, the AMF is capable of registring network-unknown UEs into the 5G core network and transporting NAS SM(session management) message to the SMF transparently. Besides, the AMF has high stability and high concurrency capacity, and can be suitable for many service scenarios.

V. Conclusion

In this paper, the minimum functions of 5GC AMF are implemented, and procedures/performances are also verified in either simulated environment or real environment with commercial UE and gNB. Specifically, the ITTI-based procedure processing skeleton is of high concurrency capacity and stability, which can process signallings effectively. Besides, the all-in-one 5G-AKA authentication model enables the AMF to authenticate registering UEs locally, simplifing the 5G-AKA procedure. Examples regarding the parameter settings of authentication algorithms during the 5G-AKA procedure provide explicit parameter details for the validation of 5G-AKA-related algorithms. In the end, the E2E 5G system in standalone(SA) mode, built with other open-source and/or commercial components, confirms the AMF's functionalities. Moreover, several experimental environments for simulating multi-UE registration requests scenario, built with simulated gNB and UE, confirm the AMF's high stability and concurrency capacity. However, current AMF still lacks of many important functions, such as handover, mobility registration, roaming, etc. In our future work, we will focus on improving the completeness of AMF.

VI. Acknowledgement

This work is supported by National Key R&D Program of China, under grant 2019YFB1803301, Intergovernmental international science and technology innovation cooperation key project under Project 2017YFE0121300-2, and Beijing Nova Program of Science and Technology under grant Z191100001119028. Meanwhile this work is supported by WG:OAI-5G-Core, OSA(OpenAirInterface Software Alliance).

References

[1] N. Alliance, "5G White Paper," Next Generation Mobile Networks, white paper, 2015.
[2] OpenAirInterface 4G: An Open LTE Network in a PC.
[3] Florian Kaltenberger, Aloizio P. Silva, Abhimanyu Gosain, Luhan Wang and Tien-Thinh Nguyen, "OpenAirInterface: Democratizing innovation in the 5G Era", Computer Networks, 2020.
[4] 3GPP TS 23.501:"System Architecture for the 5G System".
[5] "https://github.com/oktal/pistache".
[6] "https://swagger.io/specification/".
[7] "https://github.com/OpenAPITools/openapi-generator".
[8] 3GPP TS 23.502:"Procedures for the 5G System".
[9] 3GPP TS 38.413:"NG-RAN; NG Application Protocol (NGAP)".
[10] 3GPP TS 24.501:"Non-Access-Stratum (NAS) protocol for 5G System (5GS)".
[11] 3GPP TS 33.501:"Security architecture and procedures for 5G system".
[12] 3GPP TS 33.102:"3G Security; Security architecture".
[13] 3GPP TS 35.206:"key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 2: Algorithm specification".
[14] 3GPP TS 33.220:"Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)".
[15] 3GPP TS 33.401:"3GPP System Architecture Evolution (SAE); Security architecture".
[16] 3GPP TS 23.003:"Numbering, addressing and identification".