Web and Networks: Status in liaison organizations and W3C

Web & Networks Interest Group

Song XU, China Mobile

01 **Web and networks relevant technologies in W3C**

02 **Activities in other organizations**

Network Information API

- **Status** : Draft
- **Link** : http://wicg.github.io/netinfo/
- **Working group** : WICG
- **Last modified** : 2019/02/20
- **Brief**: The Network Information API enables web applications to access information about the network connection in use by the device, for example, wifi, Bluetooth connection and cellular connection etc. Additionally, users can listen for the changes of network when the network is switched. It is not supported for the 5g right now, but it is looking at for the future upgrades.

Web XR

- **Status**: Draft
- **Link**: https://immersive-web.github.io/webxr/
- **Working group**: Immersive Web Working Group
- **Last modified**: 2019/08/15
- **Brief**: this API is designed to provide an interface for the developers with an immersive application, which is allowed to develop an web-based, comfortable and attractive immersive application. Moreover, Immersive Web application has high requirements for broadband and latency, which is a typical usage scenario of W&N

Web RTC

- **Status**: Candidate recommendation (CR)
- **Link**: https://www.w3.org/TR/webrtc/
- **Working group**: Web Real-Time Communications Working Group
- **Version 1.0**: 2018/09/27
- **Brief**: this document defines a set of protocols APIs, which allowed the multimedia data from one browser or device to another in real-time, This specification is being developed in conjunction with a protocol specification developed by the IETF RTCWEB group and an API specification. currently, it is only an API for end-to-end real-time media data transmission , 5G edge computing can be extended to implement real-time data transmission from end to edge node.

Web of Things (WoT) Architrcture

- **Status**: Candidate recommendation (CR)
- **Link**: https://www.w3.org/TR/2019/CR-wot-architecture-20190516/
- **Working group**: Web of Things Working Group
- **Last modified**: 2018/05/16
- **Brief**: The WoT was created to enable interoperability across IoT platforms and application domains. WoT provides mechanisms to formally describe IoT interfaces to allow IoT devices and services to communicate with each other, independent of their underlying implementation, and across multiple networking protocols. In addition WoT offers a standardized way to define and program IoT behavior. IOT is a typical usage scenario of 5G, which has high requirements for large-scale machine communication network.

01 Web and networks relevant technologies in W3C

02 **Activities in other organizations**

Edge computing

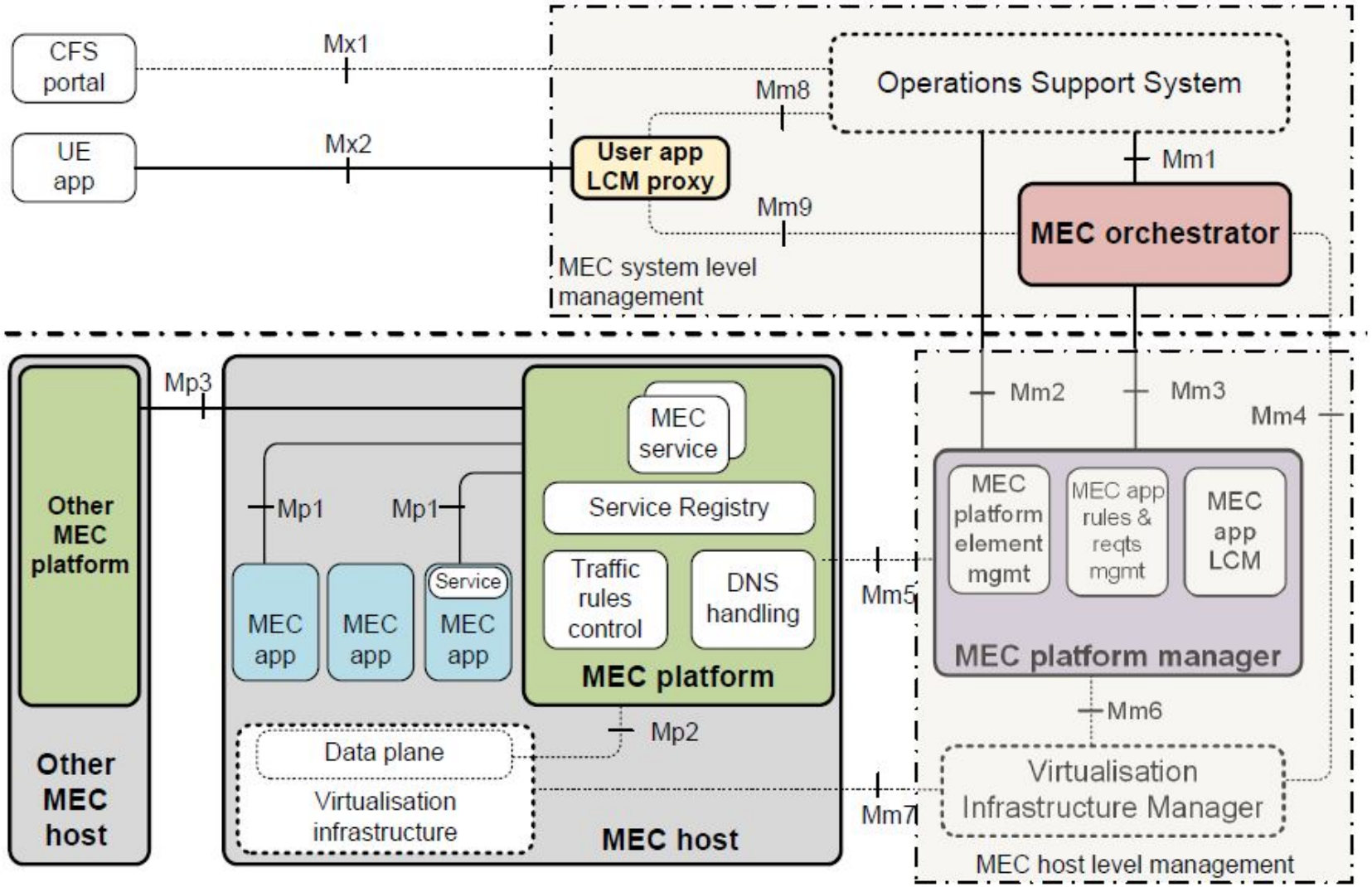
- **Development**

Similar as the function of CDN, edge computing is the marginalization of application computing layer, while the CDN is the marginalization of static resources. mobile edge computing(MEC) is used to access link from end to edge nodes, to achieve faster response and more efficient computing, and can define a web-friendly API or framework that similar to MEC.

- **Status in ETSI/3GPP**

ETSI MEC ISG have already defined a series of API to perform edge computing easily (Location API，Bandwith Mgmt API，Radio Network Info API etc.). It's worth mentioning that theses APIs are exposed by using REST HTTP. So web technology regulation organization implements or defines new API for user agent(browsers), we can refer **ETSI GS MEC 09-12** framework.

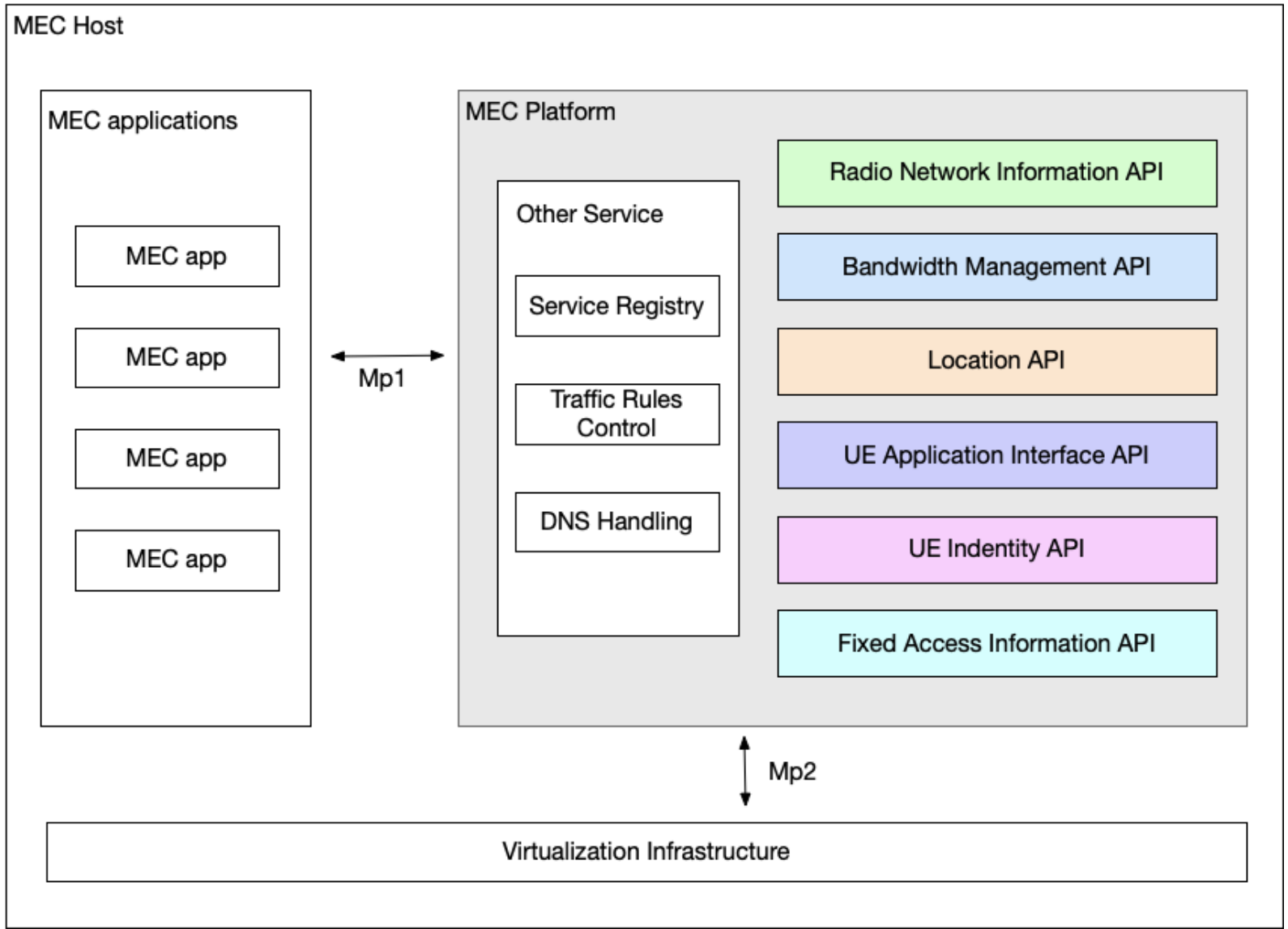
Edge computing – ETSI MEC Framework



Goal of ETSI MEC:

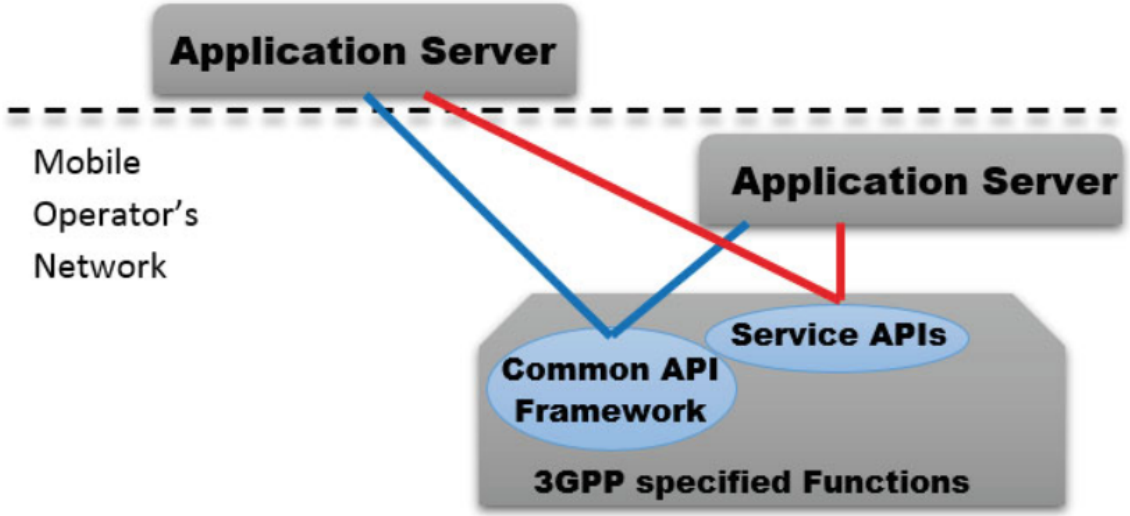
Standardized edge computing service interfaces ensure the versatility and interoperability of networks and services in different environments.

Edge computing – ETSI MEC API



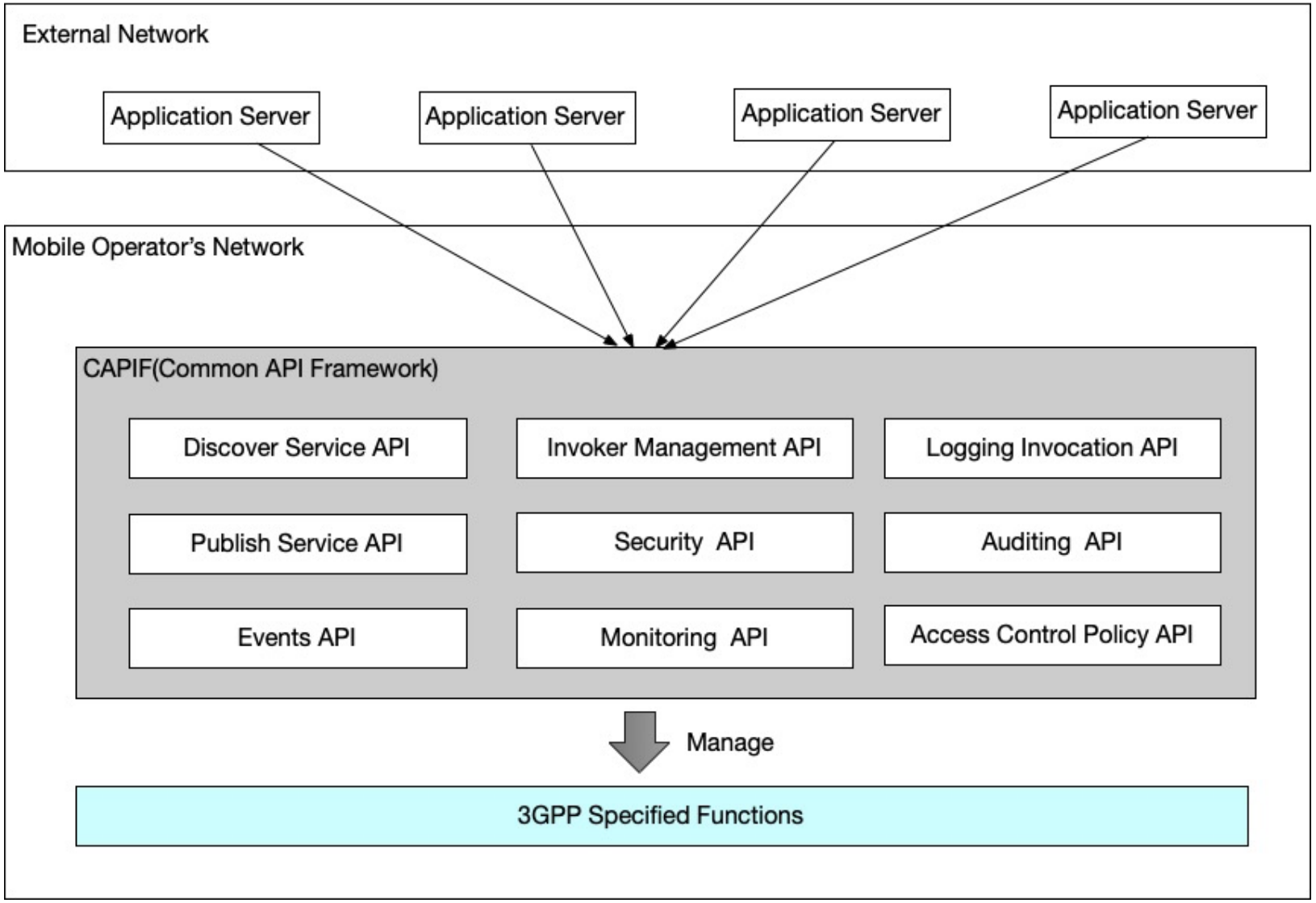
- The MEC API is located in the MEC Platform and contains the following API interfaces:
- Radio Network Information API- users get wireless network-related information to optimize edge application services
- Bandwidth Management API - primarily for bandwidth management services
- The details in specification: 《ETSI GS MEC 013》

Common API Framework – 3GPP CAPIF



- Increasingly devices and network elements
- The interface is not unified
- 3GPP defines the Common API to standardize security, authorization, logging, auditing and others for devices
- All devices and network elements can call a unified interface to exchange data.

Common API Framework – 3GPP CAPIF



ETSI CAPIF contains the following interfaces

- Discover Service API, primarily for discovery services

5G network slicing

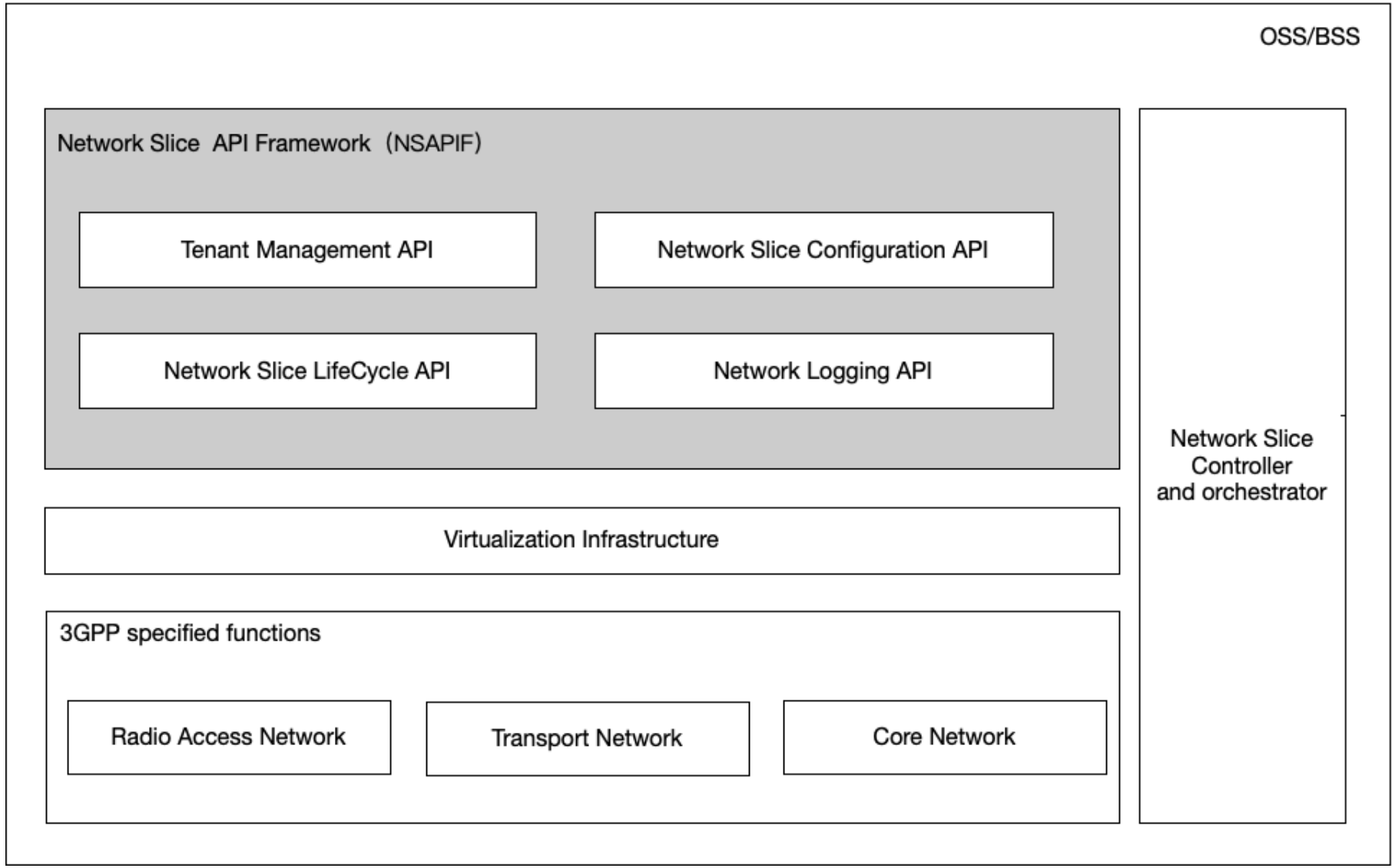
- **Thoughts**

Network slicing is to divide the network into multiple virtual networks for different requirements, for example: IOT(mMTC), high definition live and video(eMBB), low-latency(uRLLC), unmanned and telemedicine etc.

- **Development**

An API can be defined to get the current information of network slice, such as NS identity, bandwidth, delay and capacity, etc. So it is developer-friendly to detect the running environment of mobile network.

5G network slicing – Proposed API



The Network Slice API contains the following interfaces:

- Tenant Management API - primarily for tenant management
- Network Slice Configuration API - mainly used for slice configuration management, custom configuration of SLAs, such as upstream rate, downstream rate, UE type, etc.
- Network Slice LifeCycle API - primarily used for lifecycle management of tiles, including the creation and destruction of tiles
- Network Logging API - primarily for logging

5G network slicing –usages of API (Example only)

```
axios.post('/applyNetworkSlice', {
    NST:'',
    NSName:'',
    NSSCompany:'',
    Version:'',
    NSType:'',
    NSLocation:'',
    timeDelay: '',
    churn: '',
    NetworkRate:''
})..then((req) => {
        console.log(`Apply Network slice success. Slice info: $
}).catch(error => {
    console.log(`Apply Network slice failed: ${error}.`);
})
```

1. applyNetworkSlice

Apply to the network slice marketplace portal for network slices, return to the successful status of the application, and so on.

IN:

NST (identity, network slice template ID)

NSName (network slice Name)
Company (Manufacturer)
version (version)

NSType (network slice types: eMBB., uRLLC, mMTC)

NSLocation (Network Deployment Locations, e.g. CentralDC, Edge DC)

timeDelay e.g. 1ms, 5ms, 10ms

churn (jitter parameter, e.g. 1us, 1ms)

NetworkRate (Rate can be divided into upstream and downstream rates)

5G network slicing –usages of API
(Example only)

```
axios.post('/networkSliceConfig', {
    NST:'',
    uploadRate:'',
    downloadRate:'',
    UEType:'',
    NSType:'',
})..then((req) => {
        console.log(`Network slice Config success. Slice info:
}).catch(error => {
    console.log(`Network slice Config failed: ${error}.`);
})
```

2. networkSliceConfig
Designed for NS configuration management and configuration customization of SLA's, such as upstream rate, downlink rate, UE type.

IN:
uploadRate: Upstream rate
downloadRate: Downlink rate
UEType: Type of mobile device
NST: Template ID of Network slice

5G network slicing –usages of API
(Example only)

```
axios.post('/networkSliceManage', {
    lifeType: 'create', // 创建切片
    NST:'',
    uploadRate:'',
    downloadRate:'',
    UEType:'',
    NSType:'',
})..then((req) => {
        console.log(`Network slice Mange success. Slice info: $
}).catch(error => {
    console.log(`Network slice Mange failed: ${error}.`);
})
```

3. networkSliceManage
Designed for NS lifecycle management, including the construction and destruction of network slices.

IN:
lifeType: type of Lifecycle (create: construction, destory:destruction)
uploadRate: Upstream rate
downloadRate: Downlink rate
UEType: Type of mobile device
NST: Template ID of Network slice