5G Mobile Platform with P4-enabled Network Slicing and MEC

Wilson Wang
National Chiao Tung University
About Me

• Wilson Wang
• NCTU PhD candidate
• ITRI engineer
• Cat person
• ONOS deployment brigade member
Outline

• What We Have Done
• 5G Mobile Platform with free5GC
• Reduce Loading in MEC with P4 Switch
• P4-enabled Network Slicing
ONS 2016

- Build an **SDN-IP** site in NCTU
- Connect with KREONET and AmLight
- Live DEMO

### 42u Rack Cabinet

| L2 ToR Switch (1G) | 42U
| L2 ToR Switch (1G) | 41U
| NCTU-of1 Switch (10G) | 38U
| NCTU-of2 Switch (10G) | 36U
| NCTU-of3 Switch (10G) | 34U
| NCTU-of4 Switch (10G) | 32U
| ONOS cluster | 30-27U
| NCTU Quagga 2 (AS 65113) | 26U
| NCTU Quagga 1 (AS 65113) | 25U
| NCTU BGP 1 (AS 65110) | 24U
| NCTU BGP 2 (AS 65120) | 23U

![Diagram of ONOS and Quagga clusters connected through L2 ToR Switches](image)
• Integrate OAI-as-a-Service into M-CORD
• Presentation and live DEMO
ONF Connect 2018

- OAI M-CORD Platform with P4-enabled Network Slicing
- Live DEMO
ONF Connect 2019

- 5G Mobile Platform with P4-enabled Network Slicing and MEC
- Compliant with ETSI MANO
- NCTU free5GC
- Loading Reduction in MEC with P4 Switch
- P4-enabled network slicing
Introduction and Motivation for Our 5G Mobile Platform

• 5G need **Virtualized Network Functions (VNFs)**
  • Flexible and efficient network

• **Cloud-Native VNFs (CNF)**
  • VNFs based on Cloud-Native containerization technology
  • Lower overhead and higher performance

• ETSI proposes **NFV Management and Orchestration (NFV-MANO)** architecture
• Many existing NFV-MANO projects
  • Complex service development
  • Insufficient support of CNF orchestration
  • High resource usage, e.g. CPU, memory, disk ...

• Need a **5G Lightweight NFV-MANO platform**
So We Want to

• Propose a 5G Lightweight NFV-MANO Mobile Platform
  • Utilize SDN, NFV, Cloud to provide 5GC flexibility and scalability
  • All open sources
    • Kubernetes, ONOS, free5GC …
  • NFV functionality
    • Scalable free5GC CNFs
  • Cloud functionality
    • Agile orchestration
  • SDN functionality
    • Flexible underlay network
ETSI NFV-MANO Architecture

 NFVI
- Virtual Computing
- Virtual Storage
- Virtual Network

 NFV-MANO functional blocks

 NFVO
- Os-Ma-nfvo
- Ve-Vnfm-em
- Ve-Vnfm-vnf

 VIM
- Nf-Vi

 VNFM
- Or-Vnf
- Vi-Vnfm

 VNF
- Vn-Nf

 EM
- OSS/BSS
ETSI NFV-MANO Functional Blocks

- **NFVO: NFV Orchestrator**
  - Management of the instantiation of VNFMs where applicable
  - Network Services (NSs) lifecycle management

- **VNFM: VNF Manager**
  - Manage lifecycle of VNF instances
  - Creates, maintains and terminates VNF instances

- **VIM: Virtualized Infrastructure Manager**
  - E.g. OpenStack, Kubernetes, ONOS

- **VNF: Virtualized Network Function**
  - free5GC

- **NFVI: NFV Infrastructure**
  - Provide the infrastructure resources

- **EM: Element Management**

- **OSS/BSS: Operation/Business System Support**
What is free5GC

• The free5GC is an **open-source** project code for **5G generation mobile core network** created by NCTU

• Based on the Rel-13 EPC and migrates into Rel-15 5GC
  • Focus enhance Mobile Broadband (eMBB) feature
  • Ultra-Reliable Low Latency Connection (URLLC) and Massive Internet of Things (MIoT) are not supported yet
free5GC CNFs

- CNFs
  - AMF: Access Management Function
  - SMF: Session Management Function
  - HSS: Home Subscriber Server
  - PCRF: Policy and Charging Rules Function
  - UPF: User Plane Function
- All CNFs are containerization and running on K8s cluster
Design Concept of 5G Lightweight NFV-MANO Mobile Platform

- Each NF of free5GC is a CNF (Base on SBA)
- May install/update a group of designated free5GC CNFs (Using Helm)

- Create Custom Resource Definition for free5GC CNFs
- Introduce a free5GC Operator (free5GC-Op) as VNFM for free5GC CNFs CR
- Employ OLM Operator (OLM-Op) as NFVO to manage VNFM (e.g. free5GC-Op)
- Create Custom Resource Definition for C-Op
  - Treat Custom Operator (C-op) as CR allowing dynamic C-Op installation/update
Operator Lifecycle Manager (OLM)

• Open source project hosted by Red Hat
• Create Custom Resource Definition for **Custom Operators (C-Ops)**
  • Treat Custom Operator (C-Op) as CR in K8s
• Employ two operators to manage C-Op CR:
  1. **OLM Operator (OLM-Op):**
     • Watch C-Op CR update request on K8s API Server
     • Perform C-Op installation/modification
  2. **Catalog Operator** (optional)
     • Cache of C-Op custom resource
Design Concept of 5G Lightweight NFV-MANO Mobile Platform (Cont.)

- Use **ONOS** as SDN controller to manage underlying SDN network

- Two approaches to interact with ONOS (through **ONOS northbound REST API**)
  1. Modify **OLM-Op** to interact with ONOS Controller directly
  2. Introduce **ONOS-Op** to interact with ONOS Controller (On behalf of OLM-Op)
     - Need not to modify OLM-Op!

- **Introduce a ONOS Operator (ONOS-Op) as VNFM**
  - Create CRD for **ONOS REST API**
    - Treat ONOS REST API as CR
  - Implement a ONOS-Op as **VNFM for ONOS REST API**
Architecture of 5G Mobile Platform

- **NFVO: OLM-Op**
  - Watch CR update requests of **C-Op** on Kube-apiserver
  - Install / update C-Op

- **VNFM: C-Ops**
  - free5GC Operator (**free5GC-Op**)  
    - Watch CR update requests of **free5GC** on Kube-apiserver
    - Install / update free5GC CNFs
  - ONOS Operator (**ONOS-Op**)  
    - Watch CR update requests of **ONOS REST API** on Kube-apiserver
    - Call ONOS northbound REST API
Design of free5GC Operator (free5GC-Op)

• Responsible for installing/updating free5GC CNF by using Helm
• Three components:
  • Helm Chart of free5GC CNFs (template file)
    • Template for K8s resources definitions of free5GC CNFs
  • watches.yml (file)
    • CR name: free5GC
    • Helm Chart path
  • helm-operator (free5GC-Op core)
    • Get CR name specified in watches.yml
    • Watch CR update requests of free5GC on kube-apiserver
    • On receiving request, transform config of request content into Helm Config
    • Send Helm Chart and Config to Tiller Server
Design of ONOS Operator (ONOS-Op)

- Responsible for interacting with ONOS Controller
- Two components:
  - Spec of ONOS REST API
  - ONOS-Op core:
    - Watch CR update requests of ONOS REST API on kube-apiserver
    - On receiving request, transform ONOS REST API CR content into REST API format
    - Call ONOS REST API

1: Create (ONOS, ONOS REST API)

2: forward

3: Call REST API
free5GC CNFs Installation / Modification

1. free5GC-Op watches **free5GC CR** update request on Kube-apiserver
2. User requests Kube-apiserver to create/update **free5GC CR**
3. Kube-apiserver forwards request to free5GC-Op
4. free5GC-Op transforms **config of request content** into Helm **Config**
5. free5GC-Op sends Helm **Chart** and **Config** to Tiller server (TS)
6. TS combines Helm **Chart** and **Config**, sends data to Kube-apiserver
7. Kube-apiserver deploys **free5GC CNFs**
Interact with ONOS

1. ONOS-Op watches **ONOS REST API CR** update request on Kube-apiserver
2. User requests Kube-apiserver to create/update **ONOS REST API CR**
3. Kube-apiserver forwards request to ONOS-Op
4. On receiving request, ONOS-Op transform request content into **ONOS REST API format**
5. ONOS-Op call **ONOS northbound REST API**
Data Network: Multus + Calico + SR-IOV

- Multi-interface of free5GC CNFs
  - eth0 of CNFs: for 5G Core Network functions interaction
  - eth1 of AMF and UPF: for connect to eNodeB

Why Multus + Calico + SR-IOV?
- Multus: Enabling attach **multiple network** interfaces to PODs
- Calico: Good performance for deliver native Linux networking dataplane
  - No packets encapsulation, direct packets natively by BGP routing mechanism
  - Minimize overall CPU usage and occupancy by Calico’s control plane and policy engine
- SR-IOV: **Lowers latency** and boosts **throughput** to satisfy CNF data plane needs
  - Hardware based virtualization technology that improve performance and scalability
Design of Data Network

- free5GC-Op
- Tiller Server
- K8s Control Plane
  - Scheduler
  - Controller Manager
  - etcd
- API Server

Nodes
- PCRF POD (eth0)
- HSS POD (eth0)
- SMF POD (eth0)
- AMF POD (eth0, SR-IOV)
- UPF POD (eth0, SR-IOV)

Plugins
- Felix
- Bird

Network Components
- iptables
- Routing Table
- cali-X
- tun0
- eth0
- eth1

Connections
- DN (Data network)
- External Network

K8s Cluster
- kubelet

Onf Connect
Reduce Loading in MEC with P4 Switch

• Propose a P4-based MEC network
  • Network feature
    • Provide better packet I/O with P4 switch
    • Reduce MEC loading from packet encapsulation and decapsulation
    • Redirect DNS
Stateful GTP packet tracking

- Decapsulate GTP-U header before sending it to MEC
- Encapsulate packet with GTP-U header before sending it to UE
- Tracking mapping between UE IP and downlink TEID

```
Payload
TCP/UDP
IP
GTP-U(TEID=XXX)
UDP
IP
Ethernet
```

Diagram:
- UE
- eNodeB
- P4 Switch
- MEC
- Core network
- Internet
- EPC
Reduce Loading in MEC with P4 Switch

• Two approaches
  • Packet-in downlink GTP-U packets
  • Packet-in SCTP packets
Packet-in downlink GTP-U packets

Switch

<table>
<thead>
<tr>
<th>Payload</th>
<th>TCP/UDP</th>
<th>IP</th>
<th>GTP-U</th>
<th>UDP</th>
<th>IP</th>
<th>Ethernet</th>
</tr>
</thead>
</table>

**UE session state**

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.3.2</td>
<td>NoAction</td>
</tr>
</tbody>
</table>

**UE encapsulation**

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.3.2</td>
<td>set_gtp_header(777)</td>
</tr>
</tbody>
</table>

Controller

<table>
<thead>
<tr>
<th>UE Addr.</th>
<th>DL TEID</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.3.2</td>
<td>777</td>
</tr>
</tbody>
</table>

UE session state

Match  Action
192.168.3.2  NoAction

UE encapsulation

Match  Action
192.168.3.2  set_gtp_header(777)
Packet-in downlink GTP-U packets

Switch

- Initial Context Setup
- Response
- S1-AP
- SCTP
- IP
- Ethernet

Controller

<table>
<thead>
<tr>
<th>MME-UE-ID</th>
<th>SGW Addr.</th>
<th>ENB Addr.</th>
<th>DNS Addr.</th>
<th>UE Addr.</th>
<th>DL TEID</th>
<th>UL TEID</th>
</tr>
</thead>
<tbody>
<tr>
<td>112233</td>
<td>10.0.9.2</td>
<td>10.0.9.100</td>
<td>8.8.8.8</td>
<td>192.168.3.2</td>
<td>777</td>
<td>1</td>
</tr>
</tbody>
</table>
DNS traffic redirection

1. UE send DNS requests to ask for a specific service on the internet
2. Switch redirect the DNS query to MEC
   • Target service can be provided by MEC
     • Response the request by MEC address
   • Target service cannot be provided by MEC
     • Response the request by real service address
3. UE send normal traffic to service
Implementation of mec-spGW.p4

GTP Header

- S1U uplink session lookup
  - Match: Outer Source IP
  - Action: no

- S1U downlink session lookup
  - Match: Outer Source IP
  - Action: no

  - Drop

- MEC uplink session lookup
  - Match: Inner Dest. IP, Inner Dest. Port
  - Action: Set Egress Port

- UE TEID match table
  - Match: Inner Dest. IP
  - Action: Packet in TEID

- UE Filter table
  - Match: Destination IP

- MEC downlink session lookup
  - Match: Src. IP

- MEC TEID enc table
  - Match: Dst. IP
  - Action: Set downlink session info.
Design Concept for Bandwidth Slice Management

• Bandwidth slice
  • Contain disjoint traffic flows identified from user-defined field
  • Reach isolation of bandwidth resources by priority forwarding
• Aggregated traffic flow in a slice will share the bandwidth resource
Policy of Bandwidth Management

- Slice Traffic (aggregated traffic flows)
  - Guarantee minimum bandwidth
  - Best effort delivery without any guarantee
  - Limit maximum bandwidth

- Unspecified Traffic
  - Best effort delivery without any guarantee
Packet Classification

• P4 Meter with Two Rate Three Color Marker classification
  • minimum bandwidth: Committed Information Rate (CIR)
  • maximum bandwidth: Peak Information Rate (PIR)

• Color result
  • Green: Guarantee traffic
  • Yellow: Best Effort traffic
  • Red: Abandon traffic
Priority Forwarding

- Guarantee traffic
  - Request bandwidth cannot exceed link available bandwidth
- Best Effort traffic
  - Contain unspecified packets
  - Deliver by residual bandwidth
    - Maximize bandwidth utilization
- Abandon traffic
Priority Forwarding - Two-Level Priority Queue

• For example:

  Slice 1
  10 Mpbs
  min=6, max=10 trTCM
  6 Mbps
  4 Mbps

  Slice 2
  4 Mpbs
  min=4, max=5 trTCM
  4 Mbps

• Single Queue: Best effort packet interference

  Pipeline
  
  1 2
  
  high priority
  1 2 2 .... 1 2
  
  out

• Two-Level Priority Queue

  Pipeline
  
  1
  
  low priority
  1 .... 1 1
  
  out
Implementation of BW-Slicing.p4

- Extension from ONOS Basic pipeline
- Provides fundamental data-plane functionalities of the switch

<table>
<thead>
<tr>
<th>Slicing Table</th>
<th>Classifier Table</th>
<th>Policer Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match</td>
<td>Match</td>
<td>Match</td>
</tr>
<tr>
<td>Whatever</td>
<td>tag_slice_id</td>
<td>packet_color</td>
</tr>
<tr>
<td>Match</td>
<td>slice_id</td>
<td>set_priority</td>
</tr>
<tr>
<td>Fields</td>
<td></td>
<td>drop</td>
</tr>
<tr>
<td>You</td>
<td>set_color</td>
<td></td>
</tr>
<tr>
<td>Like</td>
<td>set_uncolor</td>
<td></td>
</tr>
</tbody>
</table>

(Unspecified fields treated as wildcard)
ONF Connect 2019

- 5G Mobile Platform with P4-enabled Network Slicing and MEC
  - Compliant with ETSI MANO
  - NCTU free5GC
  - Loading Reduction in MEC with P4 Switch
  - P4-enabled network slicing
Thank You

Follow Up

willson6688@gmail.com