Porting an Open Source 5G Core Software into a Time-controlled Linux Container Environment

Dr. Mike Chih-Che Lin, President of EstiNet Technologies
Our 2018 Work with Open Source Networking Projects

- Developed an IoT SDN Controller Platform based on Open DayLight for Smart City and edge networks

- Enhanced from Open DayLight Controller
  - Hydrogen → Lithium → Carbon → Nitrogen → Oxygen

- Bugs fixed and new bundles (modules) added into the original open-sourced package

- Provide Restful North-bound API (NBI) for upper-layer Management System

- Provide both OpenFlow and SNMP Southbound APIs for SDN devices (SDN Switches and IoT WiFi AP)

- Integrating common campus-level network services as NFVs
  - DHCP Server
  - Radius Server
Overview of Our Changes to ODL
Implement OpenFlow Table Features on Domestic Realtek IC

- 24 1-Gbps Ethernet RJ-45 port
- POE port supported by RT188P
- 4 1-Gbps SFP port

Major Features
- Realtek RTL838-series ASIC inside
- Provide 1K flow entries
- Provide hardware-based meters
- Support OpenFlow 1.3 (47.8% compliance)
- Support OVS/OVSDB
- Support Legacy protocol suite (STP/RSTP/VLAN/QiniQ/IGMP/QoS)
Our Alpha Testbed for 100+ Device Scale

- Comprises 45 Raspberry Pi to build an “IoT Device Access Testbed”

- Every Pi emulates 3-6 IoT device or end hosts. The whole testbed is capable of emulating up to 270+ hosts being simultaneously online.

- Unified control by a central script, allowing quick launch of experiments
Our Beta Testbed for 100,00+ IoT Devices

- Located in Si-Soft Research Center, Hsinchu Science Park.
- 84-100 real RT188T switch, which uses Realtek’s 838 series IC.
- 4 high-speed servers, emulating 40 virtual switches being online.
- 100 Raspberry Pi emulating 100,00+ IoT devices
- Capability: emulating a large network composed of 100 switches, connected with 10,000 IoT devices
Problem of Testing with Real-world Testbed

- **Complexity of Deployment**
  - Deployment usually requires 1-2 engineers to work half-to-one day.

- **Complexity of Launching Script**
  - Requires 1 engineer to spend one day to write and test the start script

- **Complexity of Changes**
  - Difficult to change → So, we don’t change it!

In the past, we used “mininet” to do pre-testbed test.
What we learned from VT-Mininet

This inspired us!

What do we want?

- Container $\rightarrow$ clean and convenience name space isolation
- test real-life programs
- No try-and-error for time dilation setting $\rightarrow$ adaptive virtual time control
OUR WORK IN 2019
We Modified EstiNet Network Simulator to Achieve This!

- IP-level routing-table manipulation approach to run real-life programs
- Already use the virtual time notion
- Event-driven approach for time advancing → a type of adaptive virtual time control
Patented Approach to Control & Sync System Times of Linux Containers

Modification Summary:

- **Process Table**
  - A new field denoting the invocation of the IO system calls that may not generate timer events

- **Kernel**
  - CPU occupation checking logic
  - New notification events for SE

- **SE**
  - Add a new syscall so that it can check if a child process is calling specific IO system calls.

Taiwan Patent: I709308 “網路模擬器平台上之時間控管方法及系統”, 2020/11/01
# Comparison among Time Control Methods for Linux Containers

<table>
<thead>
<tr>
<th>Rationale</th>
<th>ASAP Event Scheduling</th>
<th>Loading Monitor</th>
<th>Time Dilation</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Event-rized” packets with timestamp sorting and execution</td>
<td>Use a fixed coefficient TDF and check the CPU load to properly shorten the physical elapsed time</td>
<td>Use a fixed coefficient TDF to shorten the physical elapsed time</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Representatives</th>
<th>EstiNetX</th>
<th>MiniNet-HiFi</th>
<th>VT-MiniNet</th>
<th>Diecast</th>
<th>SVEET (Event+Time Dilation)</th>
<th>SliceTime (NS3+VM, loose time sync.)</th>
<th>TimeJails/NETBalance</th>
<th>TimeKeeper</th>
<th>SELENA (VM-based + Time Dilation)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Download Availability</th>
<th>Yes (Official Website)</th>
<th>Yes (GitHub)</th>
<th>VT-MiniNet (GitHub)</th>
<th>Diecast (NA)</th>
</tr>
</thead>
</table>
Incoming 5G Era

Source: 陈文村教授．AnNet_2018_KEYnote.pdf
Objective of 5G Spec.

- **eMBB**: Enhanced Mobile Broadband
  - Downlink: 20Gbps
  - Uplink: 10Gbps

- **URLLC**: Ultra-reliable and Low Latency Communications
  - Control-plane Latency: 10ms
  - Data-plane Latency: 0.5ms
  - Mobility unavailability: 0ms
  - Reliability: 99.999%

- **mMTC**: massive Machine Type Communications
  - 106 device/km² (under certain QoS req.)

New Architecture of 5G Core:

- **Micro-Service**

- **Service-based Interface**
  - Realize Service Function Chain in SDN
  - **HTTP RESTFUL System**
    - REST (Representational State Transfer)
CONTAINER-BASED 
5G SIMULATION PLATFORM
Difference Between Container and VM

VM Architecture

Container Architecture
Introduction to Free5GC

- Developed by Prof. Jyh-Cheng Chen, NCTU, Taiwan
- Free to download and use
- Most of source codes are open (a small portion of codes are in binary form)
- Operation with members’ fee and supported by Governmental R&D projects
Free5GC Stage-3 Release

- Release in April, 2020
- Support Service-based Interface
  - Micro-service
  - Restful HTTP API Arch.
- Support N3IWF function
Concept of Container

- Conceptually isolated unit for a group of software package
  - Application
  - Library
  - CPU+MEM+Storage+Network

- Linux *namespace* technology
  - CPU namespace
  - Memory namespace
  - Storage namespace
  - Network interface namespace

Source: https://blog.trendmicro.com.tw/?p=60814
**Difference between Container and VM**

**Translation Approach**

**VM Architecture**

- VM
  - App 1
  - Bin/Lib
  - Guest OS
  - Hypervisor
  - Host OS
  - Host (Physical Server)

**Container Architecture**

- Container
  - App 1
  - Bin/Lib
  - Host OS
  - Host (Physical Server)

**ID (Data Structure) Approach**

- VM
  - App 2
  - Bin/Lib
  - Guest OS
  - Hypervisor
  - Host OS
  - Host (Physical Server)

- Container
  - App 2
  - Bin/Lib
  - Host OS
  - Host (Physical Server)

- Container
  - App 3
  - Bin/Lib
  - Host OS
  - Host (Physical Server)

- Container
  - App 4
  - Bin/Lib
  - Host OS
  - Host (Physical Server)
Micro-service Arch. Fits into Container Technology

---

[Diagram showing service-based interface and various components such as NSSF, NRF, UDM, PCF, AMF, SMF, AUSF, RAN, UPF, and DN, with annotations like N1, N2, N3, N4, N5, N6, and N7 for connections and interactions.]
In our EstiNet11 environment, we created 9 network nodes which are ready to load Docker images.
Our Porting Approach

② We use Free5GC source codes to create a single Docker image which contains all 5GC micro services
3. Load the Docker image onto each node

4. Configure a specific microservice for each chosen node
   - e.g., the AMF node runs the AMF micro service on its starting script
   - Follow the same way that one runs Free5GC in the real world
• In our environment, we change to the type of the micro service process to “Real-time Process” in Linux kernel.
• We change the SCTP connections used in Free5GC from non-blocking mode back to blocking mode, in order to prevent AMF from occupying too much CPU resource. This will cause the issue that an UE cannot attach to the network.
• Fixed the issue that SMF always returns the same Network Slicing ID.
All micro-services of 5GC on one machine is realized by the Linux Container Technology. Next Question: How to achieve repeatable results and to do observations?
• Have done the 5GC part some patches

• Created UE+RAN simplified implementation

• Each component is run by a container and time synchronized/controlled by the patented SE and kernel modifications
Video Demo

- Run a 5G network simulation with Free5GC and synchronized multiple visual packet analyzers
User List of This Platform

Taiwan

• National ChiaoTung University
• National Sun Yat-sen University
• National Taipei University of Education
• National Yunlin University of Science and Technology
• National Kaoshung University of Science and Technology
• FuJen Catholic University
• National Formosa University
• TungHai University
• ChaoYang University of Technology

Overseas

• India
  • IIT Bhubaneswar
  • Manipal University, Jaipur

• France
  • Université Paris-Est Créteil (UPEC, Paris 12th Univ.)

• Slovak
  • Slovak University of Technology in Bratislava

• China
  • BeiJing JiaoTong University
  • ChongQing JiaoTong University
  • XiDian University
  • GuiLin University of Aerospace Technology
Thanks very much for your listening!

Q&A Time