Next Gen Infrastructure Core *(NGIC)* Hands-On Demo

Presented by **Intel Labs**: Saikrishna Edupuganti, Jacob Cooper, Karla Saur

CORD Build Event - November 7-9th, 2017 – San Jose, CA

Intel Labs

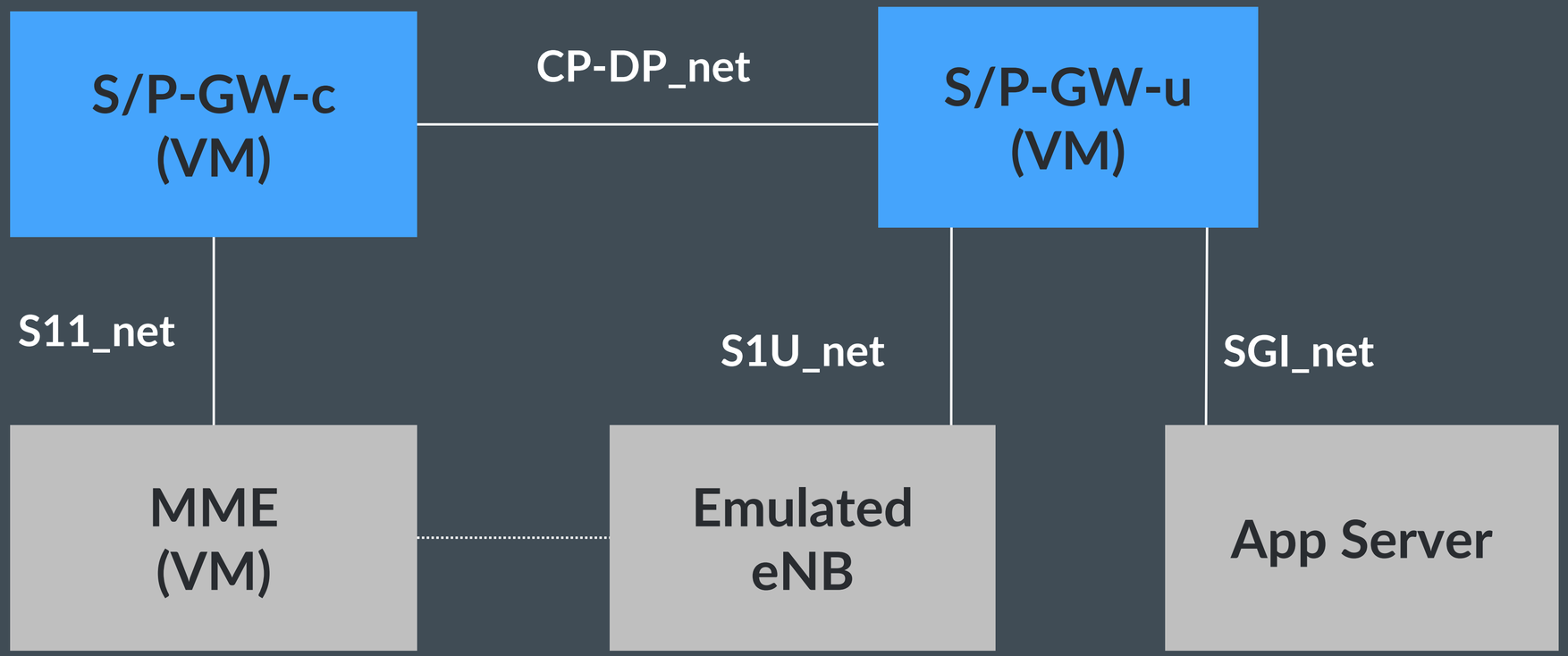
Legal Disclaimer

- This presentation contains the general insights and opinions of Intel Corporation (“Intel”). The information in this presentation is provided for information only and is not to be relied upon for any other purpose than educational. Use at your own risk! Intel makes no representations or warranties regarding the accuracy or completeness of the information in this presentation. Intel accepts no duty to update this presentation based on more current information. Intel is not liable for any damages, direct or indirect, consequential or otherwise, that may arise, directly or indirectly, from the use or misuse of the information in this presentation.
- Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel, the Intel logo and Xeon are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
-

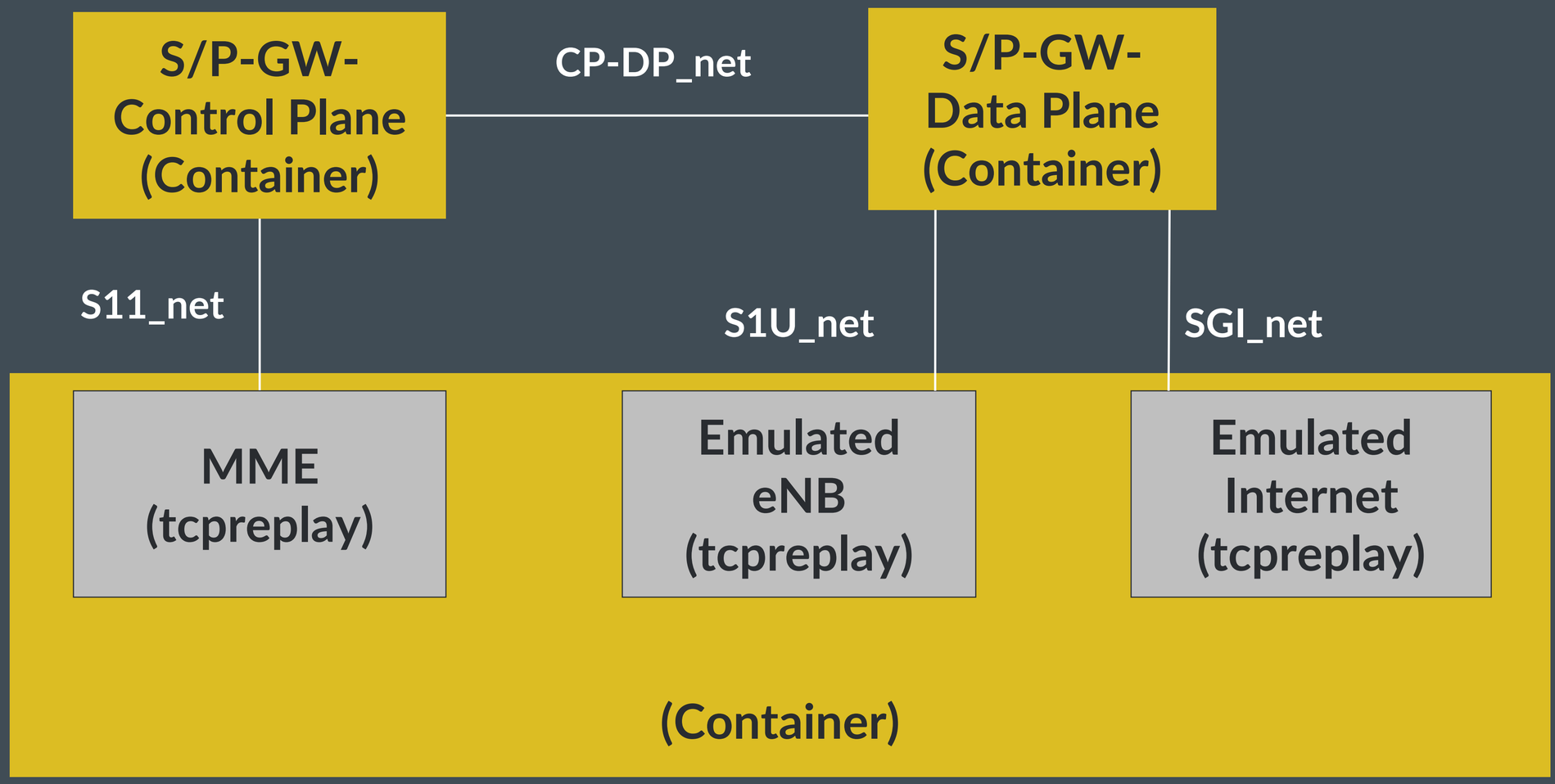
Agenda

- Brief vEPC/NGIC Background
- Hands-on Demo

LOGICAL NETWORK CONNECTIVITY



DEMO of NGIC using 3 containers



Setup

- Prerequisites:
  - Install Docker (1.13 or higher) and Docker Compose
  - Docker images for NGIC control and data plane

```
docker pull ngiccorddemo/ngic-cp
docker pull ngiccorddemo/ngic-dp
docker pull ngiccorddemo/ngic-traffic
```

  - Demo folder

```
git clone https://github.com/ngiccorddemo/cordbuild2017.git
```

Setup

For later: The NGIC Code is available at https://gerrit.opencord.org/#/q/project:ngic
`Commit-id: a9e05`

For now: In this short 45 minute demo, we will be using prebuilt Docker images

Open 3 terminals and change directories to your Demo folder

Step 1: Start the Data Plane

In terminal #1:

```
docker-compose -p epc up dp
```

Wait for DP to print stats

Step 2: Start the Control Plane

In terminal #2:

```
docker-compose -p epc up cp
```

You will see a large table of stats printing periodically

Step 3: Start the Traffic Container

In terminal #3: Bring up traffic container in daemon mode (-d)

```
docker-compose -p epc up -d traffic
```

Enter the container

```
docker exec -it epc_traffic_1 /bin/bash
```

Step 4: Start the traffic

First, get the interface names by running the following commands:

```
S11_IFACE=$( netstat -ie | grep -B1 10.1.10 | head -n1 | awk '{print $1}' | tr --d : )
S1U_IFACE=$( netstat -ie | grep -B1 11.1.1 | head -n1 | awk '{print $1}' | tr --d : )
SGI_IFACE=$( netstat -ie | grep -B1 13.1.1 | head -n1 | awk '{print $1}' | tr --d : )
```

Step 5: Start the Control traffic

Play the S11 (control plane) traffic to set up the flows

```
tcpreplay  --pps=200 -i $S11_IFACE s11.pcap
```

Look at the Control Plane (Screen #2) and make sure that the packets appear

There should be 2000 packets sent/2000 received (from the 1000 CreateSession and 1000 ModifyBearer packets)

Step 6: Start the Data Plane traffic

Now start the S1U (Data Plane uplink) traffic

```
tcpreplay  -i $S1U_IFACE s1u.pcap
```

Check the Data Plane (Screen #1). You should see ~6500 packets received on the S1U and ~6500 packets transmitted on the SGi

Step 6: Start the Data Plane traffic (cont.)

Now start the SGi (Data Plane downlink) traffic

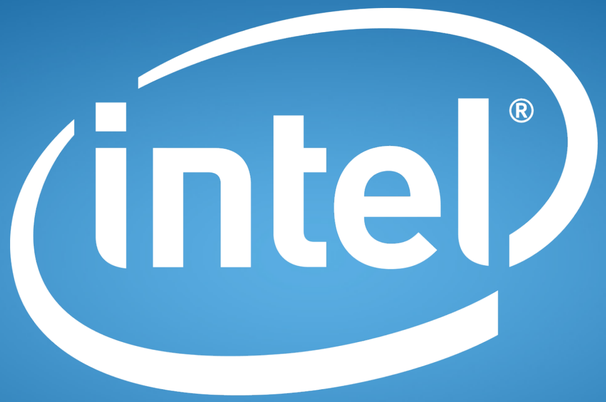
```
tcpreplay  -i $SGI_IFACE sgi.pcap
```

Check the Data Plane (Screen #1). You should see ~6500 packets received on the SGi and ~6500 packets transmitted on the S1U

Step 7: Clean Up

From the traffic terminal, type `'exit'` and then type:

```
docker-compose -p epc down
```



experience
what's inside™