OMEC over the Berkeley Extensible Software Switch

Muhammad Asim Jamshed, Saikrishna Edupuganti and Christian Maciocco
Intel Labs
Outline

• OMEC Overview
• Motivation: The need for an SPGW revamp
• BESS
• Current Status
• Summary
OMEC Overview

Mobile Network Core or Evolved Packet Core (EPC)

- Home Subscription Server (HSS)
- Serving Gateway Control (SGW-C)
- Packet Gateway Control (PGW-C)
- Subscriber Database
- Policy Charging Rules Function (PCRF)

- Mobility Management Entity (MME)
- Serving Gateway User Data (SGW-U)
- Packet Gateway User Data (PGW-U)

- Control
  - S11
  - S6a

- Data
  - S1U
  - S5/S8

- Internet
  - SGI

Legend:
- MME: Mobility Management Engine (Control)
- HSS: Home Subscriber Services (Authentication)
- PCRF: Policy and Charging Rules Function
- SGW-C: Serving Gateway Control
- SGW-U: Serving Gateway User
- PGW-C: Packet Gateway Control
- PGW-U: Packet Gateway User
OMECA Overview

- **Home Subscription Server (HSS)**
- **Mobility Management Entity (MME)**
- **Subscriber Database**
- **Policy Charging Rules Function (PCRF)**
- **Serving/Packet Gateway Control (SPGW-C)**
- **Serving/Packet Gateway User Data (SPGW-U)**

**Network Connections**

1. Access Network
2. Control (S6a)
3. Mobile Network Core or Evolved Packet Core (EPC)
4. Data (S11)
5. Data (S1U)
6. Data (SGI)

**Network Functions**

- **MME**: Mobility Management Engine (Control)
- **HSS**: Home Subscriber Services (Authentication)
- **PCRF**: Policy and Charging Rules Function
- **SPGW-C**: Serving/Packet Gateway Control
- **SPGW-U**: Serving/Packet Gateway User

**Default Configuration**

- Default SPGW-C (CP) + SPGW-U (DP)
Motivation: OMEC SPGW-U Architecture Layout

Current (over-)allocation of Compute Resources

- 4 CPUs
  - Uplink traffic (S1U RX → SGI TX)
  - Downlink traffic (SGI RX → S1U TX)
- CP communication
- Kernel communication
- L3 & L2 updates (route + arp)

- Are separate CPUs needed for
  - CP communication?
  - ARP/Route resolution(s) via the kernel?
Motivation: OMEC SPGW-U Architecture Layout

Is the scale-out too expensive?

• Spin up complete instances (in the worst case)
  • Over-allocation of CPU resources?

(4 x n)
Motivation: OMEC SPGW-U Architecture Layout

Can the base design be improved?

• ARP resolution efficiency
  • $\text{CPU}_{\text{DL/UL}} \rightarrow \text{CPU}_{\text{ARP}} \rightarrow \{\text{KERNEL}\} \rightarrow \text{CPU}_{\text{ARP}} \rightarrow \text{CPU}_{\text{DL/UL}}$
  • $?= 4$ CPU hops
Motivation: OMEC SPGW-U Architecture Layout

Is SPGW-U deployment friendly?

- Containerized solution
  - KNI module is a major hurdle
  - `AF_PACKET` + `veth` pair mode available, but not default
Motivation: OMEC SPGW-U Architecture Layout

SPGWU user configurability

• CPU (re-)configuration needs a process restart, re-compilation or in the worst case, code re-write altogether
  • Hard-coded
    • Single interface / Multi-interfaces
    • Pipelined / Run-to-completion
• Fine-grained CPU scheduling over individual SPGWU pipeline submodules
• Optimizations of individual submodules
  • E.g.: Apply vector operation(s) for processing batch of packets within each submodule of the pipeline
Can we rely on a programmable platform to ease our development/deployment?
BESS
Programmable platform for data plane development

• Clean-slate internal architecture with NFV in mind
  • Highly flexible & customizable
• Creating BESS applications
  • Modular pipeline represented as a directed acyclic graph
  • Each module can run arbitrary code
  • Independently extensible & optimizable
• Configure & control BESS
  • Via NF controller
BESS Architecture Overview

DAG of interconnecting modules

NET_CONTROLLER
Policy updates via CP

HOST_CONTROLLER
Neighbor updates via OS

BESS Daemon (running in user space)

Linux

dpdk pmd

AF_UNIX, PCAP

VFIO, AF_PKT, AF_XDP
BESS: Resource Aware CPU Scheduling

Allows flexible scheduling policies for the data path

- In terms of CPU utilization & bandwidth
BESS: Resource Aware CPU Scheduling

Allows flexible scheduling policies for the data path

- In terms of CPU utilization & bandwidth
BESS: Resource Aware CPU Scheduling

Allows flexible scheduling policies for the data path

- In terms of CPU utilization & bandwidth
BESS: Resource Aware CPU Scheduling

Allows flexible scheduling policies for the data path

- In terms of CPU utilization & bandwidth

Diagram:

- S1U In
- Filter
- GTPDecap
- EtherEncap+Cksum
- Rtr
- SGI Out
- Q1
- VDev (to kernel)

Should not consume > 10% CPU

VDev (to kernel)
BESS: Resource Aware CPU Scheduling
Allows flexible scheduling policies for the data path

- In terms of CPU utilization & bandwidth

Limit by 10 Kbps
OMEC over BESS

Why architecting user-plane with BESS is a good idea: key benefits

• More modular
  • Concentrate only on core business logic (on VNF development) and not the infrastructure development
    • SLOC of individual modules: ~200
  • Mostly rely on built-in BESS modules resulting in a thin stack
  • GRPC-based communication to control daemon
    • Controllers based in python & C++
      • (Route+L2 neighbor) python controller based on pyroute2: SLOC ~350

• Ease of customizing pipeline at runtime
  • e.g. CPU scheduling, adding/removing specific modules

• Configuration ease
  • Multi-workers enable/disable at ease
    • Economical usage of CPU usage
    • Run individual modules on difference CPUs
      • Run to completion vs pipeline become run-time choices (& not compile-time)
  • No need to restart the daemon process for config updates
OMEC over BESS

Why architecting user-plane with BESS is a good idea: key benefits

• Operator friendly
  • Route control (more akin to deployment)
    • Interfacing with the kernel is easier
    • Netlink messages neighbor + route updates
  • KNI support not needed
    • veth pair + AF_PACKET interface
  • AF_PACKET/AF_XDP integration easier (cloud-native friendly) for fastpath
• Monitoring ease at runtime
  • tcpdump
  • Visualization tool
SPGW-U Downlink DAG

FPI: DPDK PMD

CPU Scheduler: CPU 0

Rate limit scheduler: 1000x/sec

Control Plane
- add_session()
- delete_session()
- show_records()

Route Control
- insert_route()
- delete_route()
- add_neighbor()
- delete_neighbor()
<table>
<thead>
<tr>
<th>Runtime model</th>
<th>NGIC/OMEC</th>
<th>SPGWU/BESS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• rtc</td>
<td>• rtc (dynamic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• pipelined (dynamic)</td>
</tr>
</tbody>
</table>
NGIC/OMEC vs SPGWU/BESS

<table>
<thead>
<tr>
<th></th>
<th>NGIC/OMEC</th>
<th>SPGWU/BESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Runtime model</strong></td>
<td>• rtc</td>
<td>• rtc (dynamic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• pipelined (dynamic)</td>
</tr>
<tr>
<td><strong>Monitoring utilities</strong></td>
<td>• shell (basic stats)</td>
<td>• bessctl shell</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• tcpdump</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• GUI</td>
</tr>
</tbody>
</table>
## NGIC/OMEC vs SPGWU/BESS

<table>
<thead>
<tr>
<th></th>
<th>NGIC/OMEC</th>
<th>SPGWU/BESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Runtime model</strong></td>
<td>• rtc</td>
<td>• rtc (dynamic)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• pipelined (dynamic)</td>
</tr>
<tr>
<td><strong>Monitoring utilities</strong></td>
<td>• shell (basic stats)</td>
<td>• bessctl shell</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• tcpdump</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• GUI</td>
</tr>
<tr>
<td><strong>(Re-)configuration ease</strong></td>
<td>• Process restart</td>
<td>• Process reset not needed</td>
</tr>
<tr>
<td></td>
<td>• Code re-write</td>
<td>• Pipeline graph re-set</td>
</tr>
</tbody>
</table>
Preliminary Performance Evaluation

Testbed Specs & Results

• Hardware
  • Intel Xeon Platinum 8170 @ 2.10 GHz (SKX)
  • 98 GB RAM
  • Intel Fortville 10 Gbps (dual port)
• Packet generator
  • ll_trafficgen

![Graph showing processing rate at 0% packet loss with data points for 4 CPUs and 2 CPUs at different packet sizes (128, 512, 1024) for ngic-omec and spgwu-bess]
Implementation

Current Status

• What’s done
  • Encap/Decap
  • CP interfacing via ZMQ bus
  • IP Reassembly
  • IP Fragmentation
  • GTP Echo/Response

• In progress
  • Charging
  • Metering

• All other VNFs (e.g. CP) remain unchanged
Implementation

Contribution to the open source community

• What’s being planned to be upstreamed
  • BESS ported to dpdk-19.08
  • IP fragmentation and reassembly modules
  • Other minor optimizations to existing modules
• SPGWU over BESS is available @:
  • _______________
Thank You

Follow Up Links: XXXX