OMEC over the Berkeley Extensible Software Switch

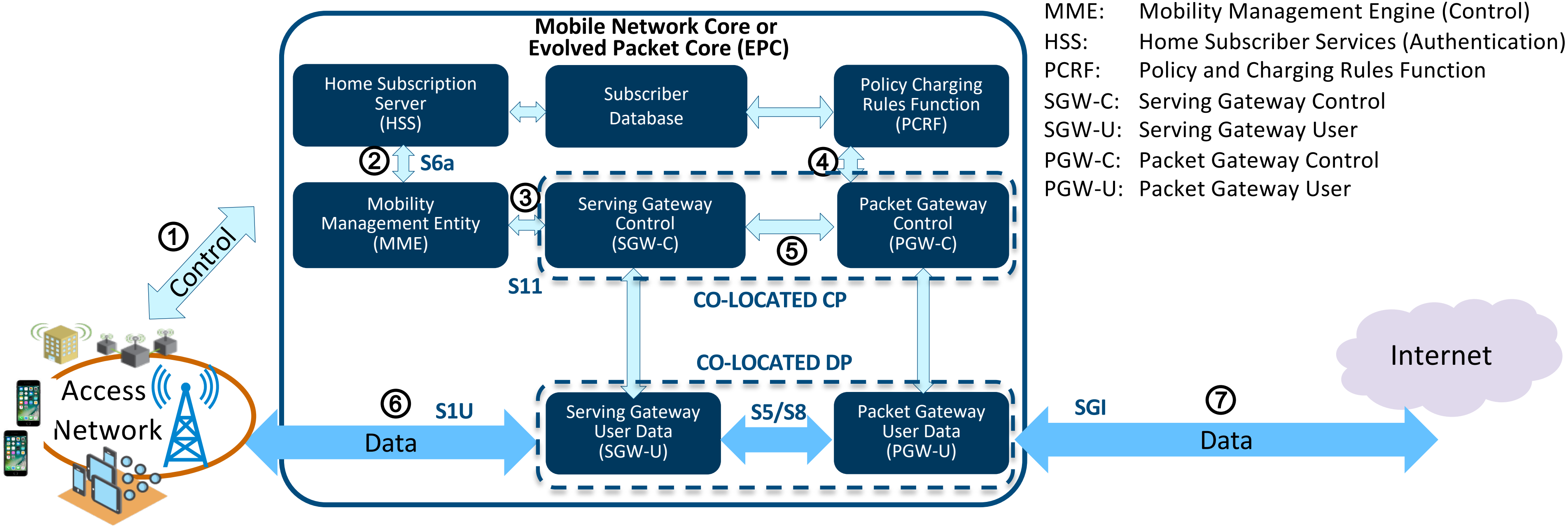
Muhammad Asim Jamshed, Saikrishna Edupuganti and Christian Maciocco
Intel Labs

Outline

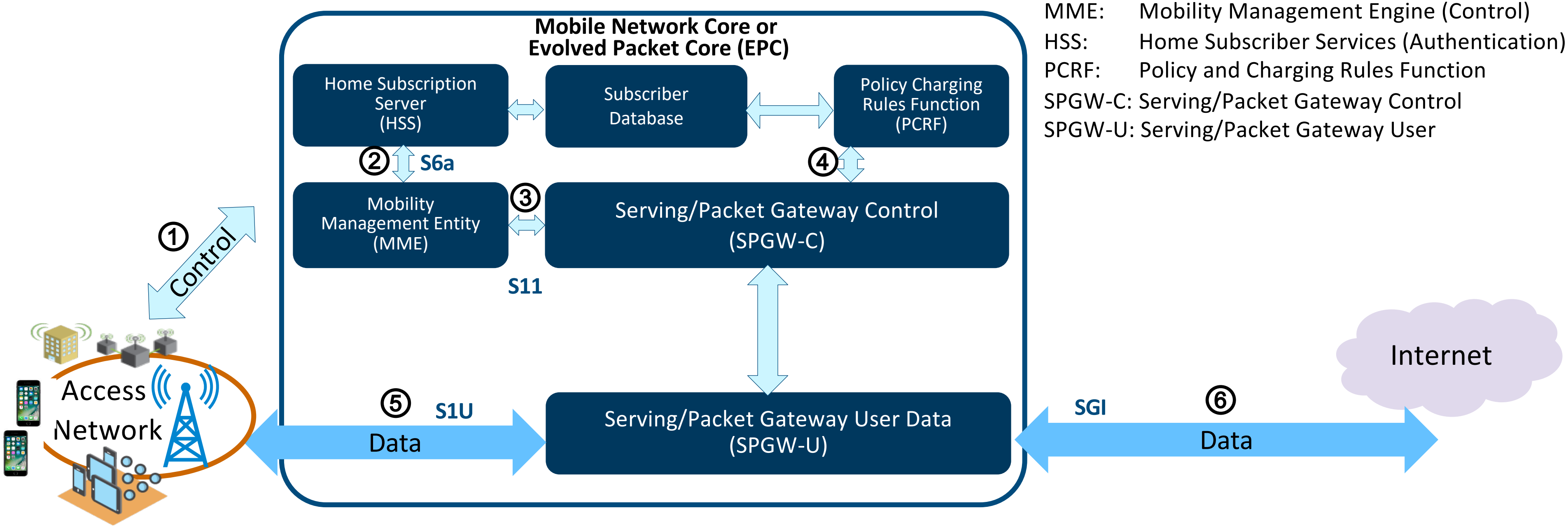
- OMEC Overview
- Motivation: The need for an SPGW revamp
- BESS
- Current Status
- Summary

OMEC Overview

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



OMEC Overview



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

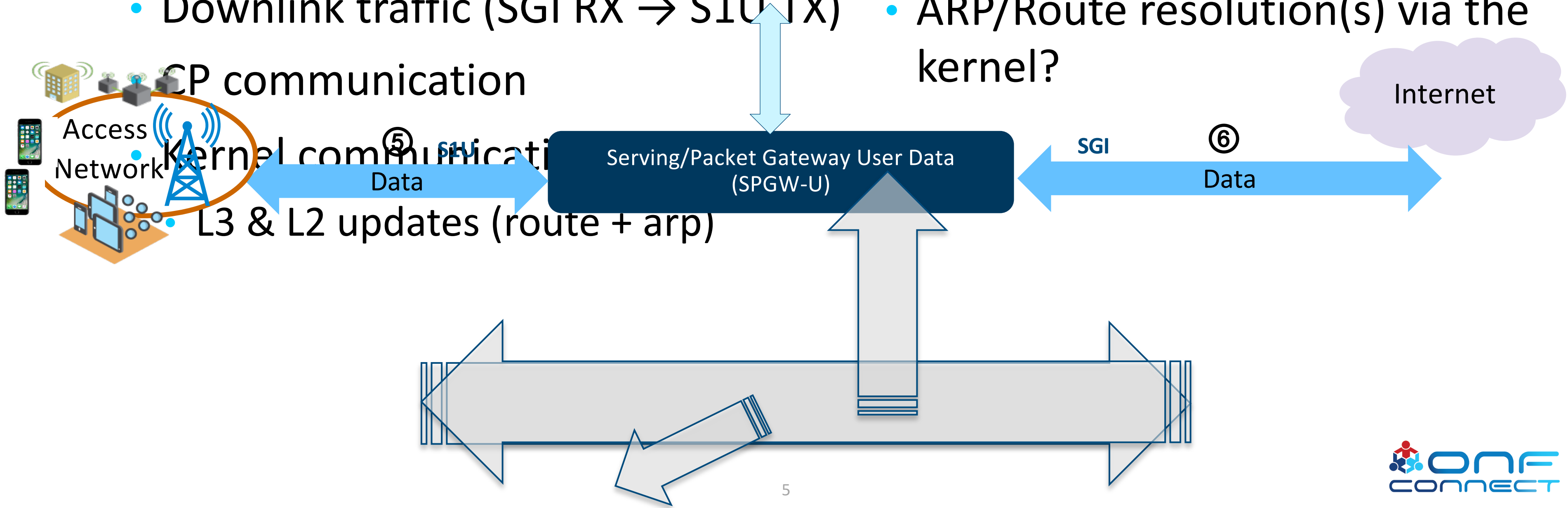
- Default SPGW-C (CP) + SPGW-U (DP)

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)

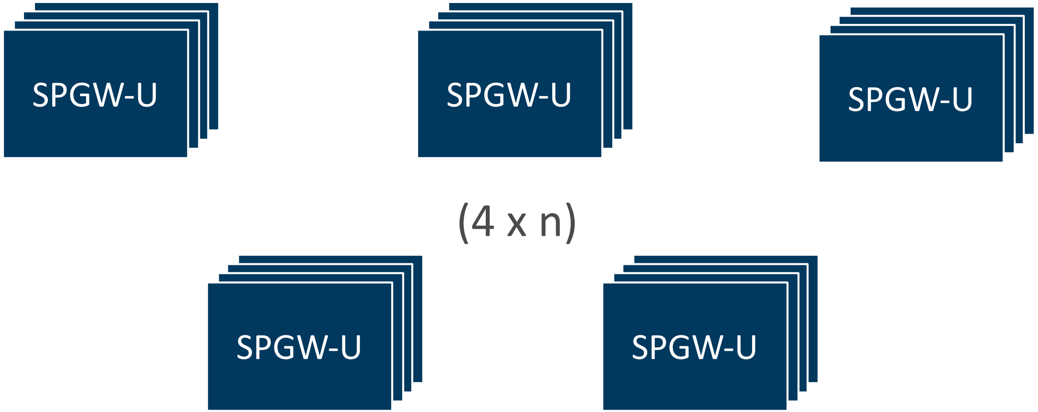
- Are separate CPUs needed for
  - CP communication?
  - ARP/Route resolution(s) via the kernel?



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?



Motivation: OMEC SPGW-U Architecture Layout

Can the base design be improved?

- ARP resolution efficiency
  - $CPU_{DL/UL} \rightarrow CPU_{ARP} \rightarrow \{KERNEL\} \rightarrow CPU_{ARP} \rightarrow CPU_{DL/UL}$
  - ?= 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?

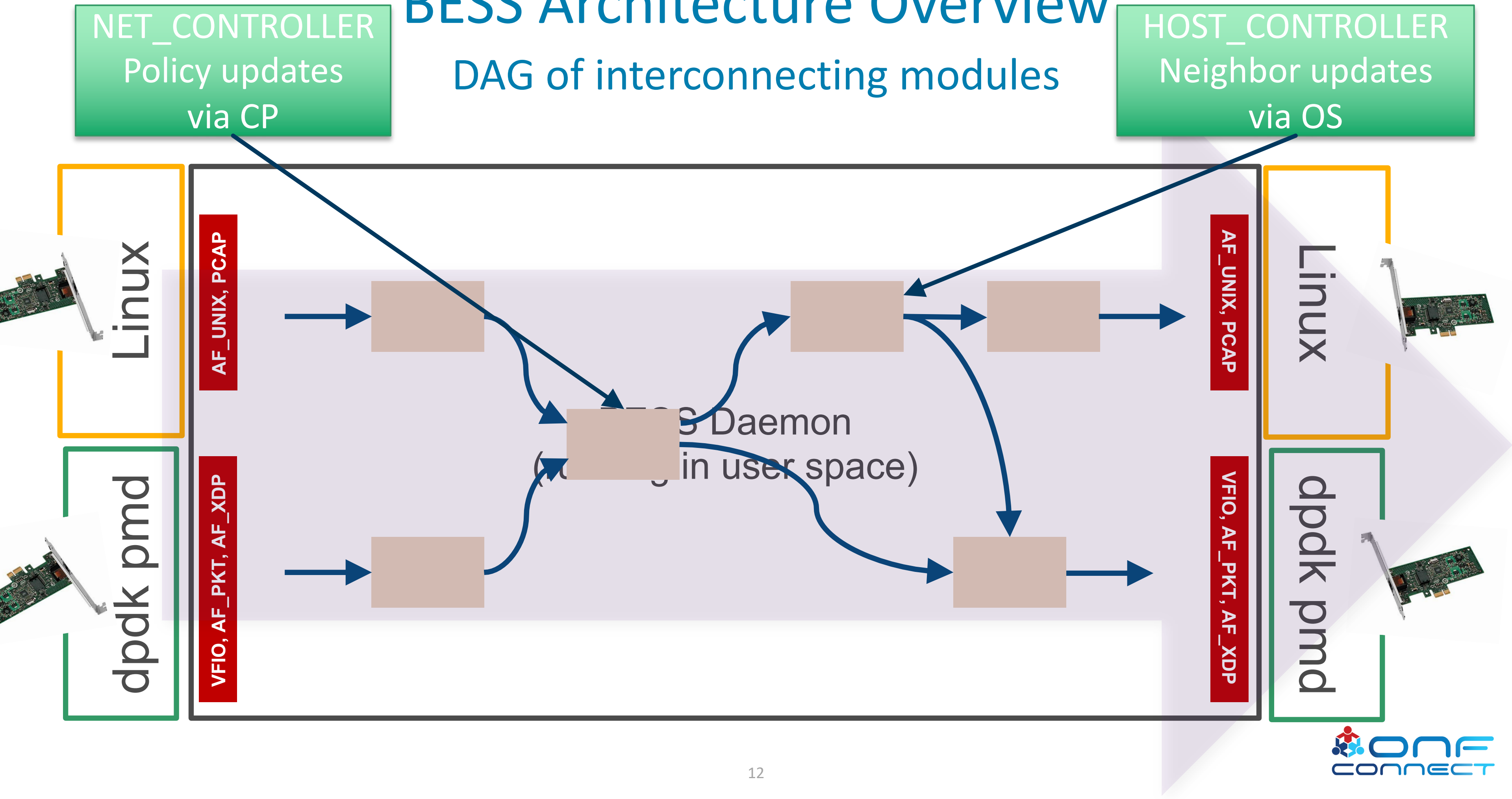
BESS

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 <u>extensible</u> & <u>optimizable</u>
- Configure & control BESS
  - Via NF controller

BESS Architecture Overview

DAG of interconnecting modules



BESS: Resource Aware CPU Scheduling

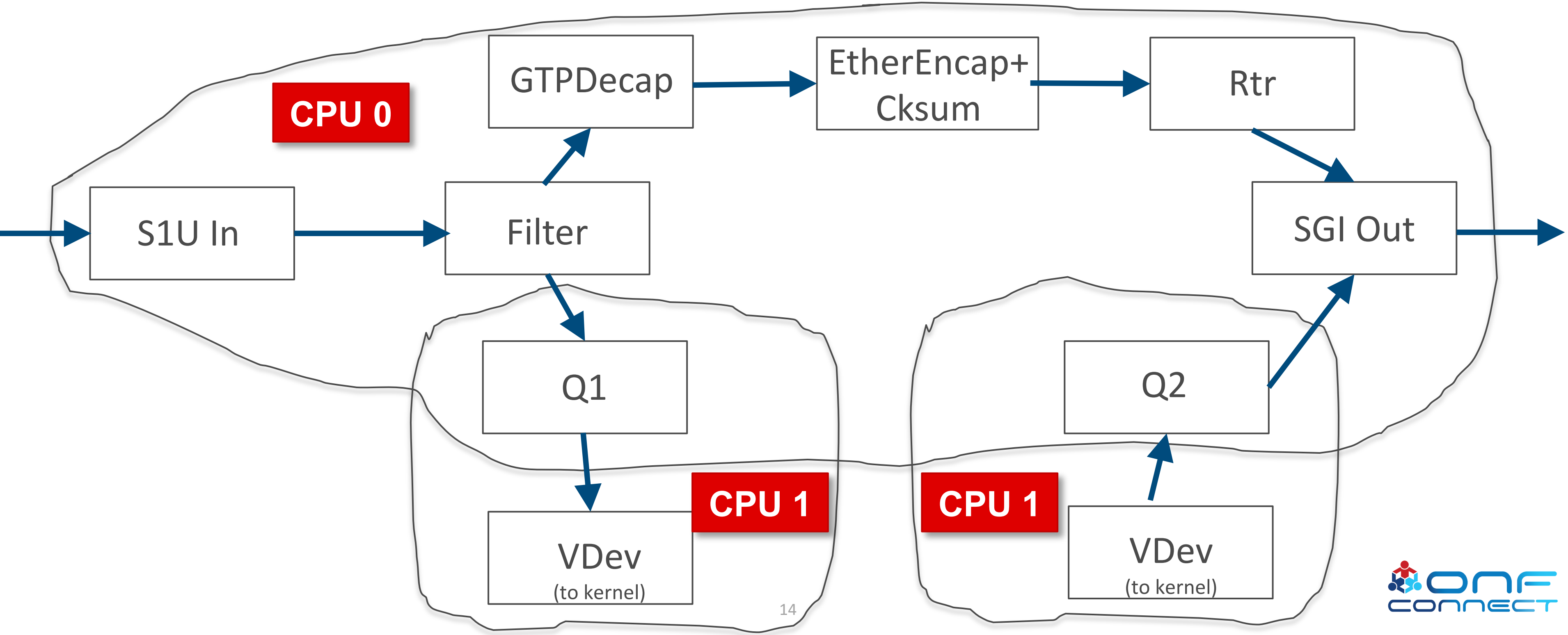
Allows flexible scheduling policies for the data path

- 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

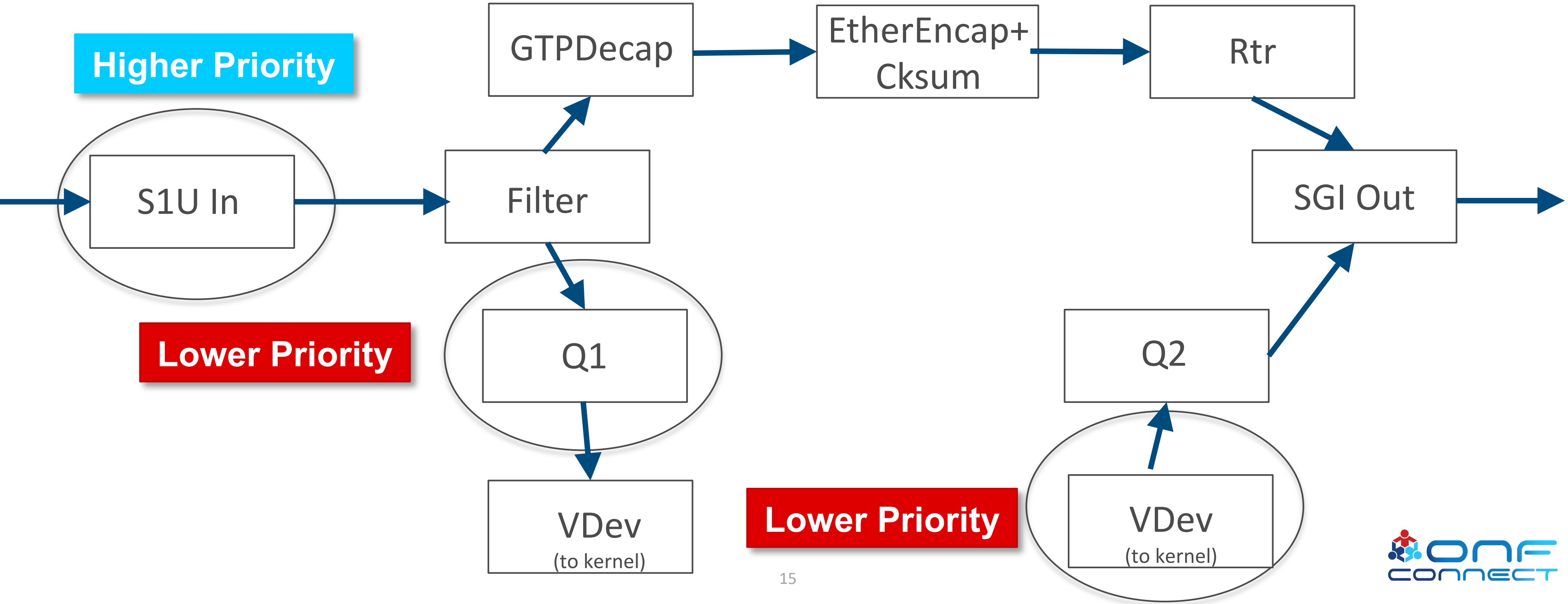


ONF CONNECT

BESS: Resource Aware CPU Scheduling

Allows flexible scheduling policies for the data path

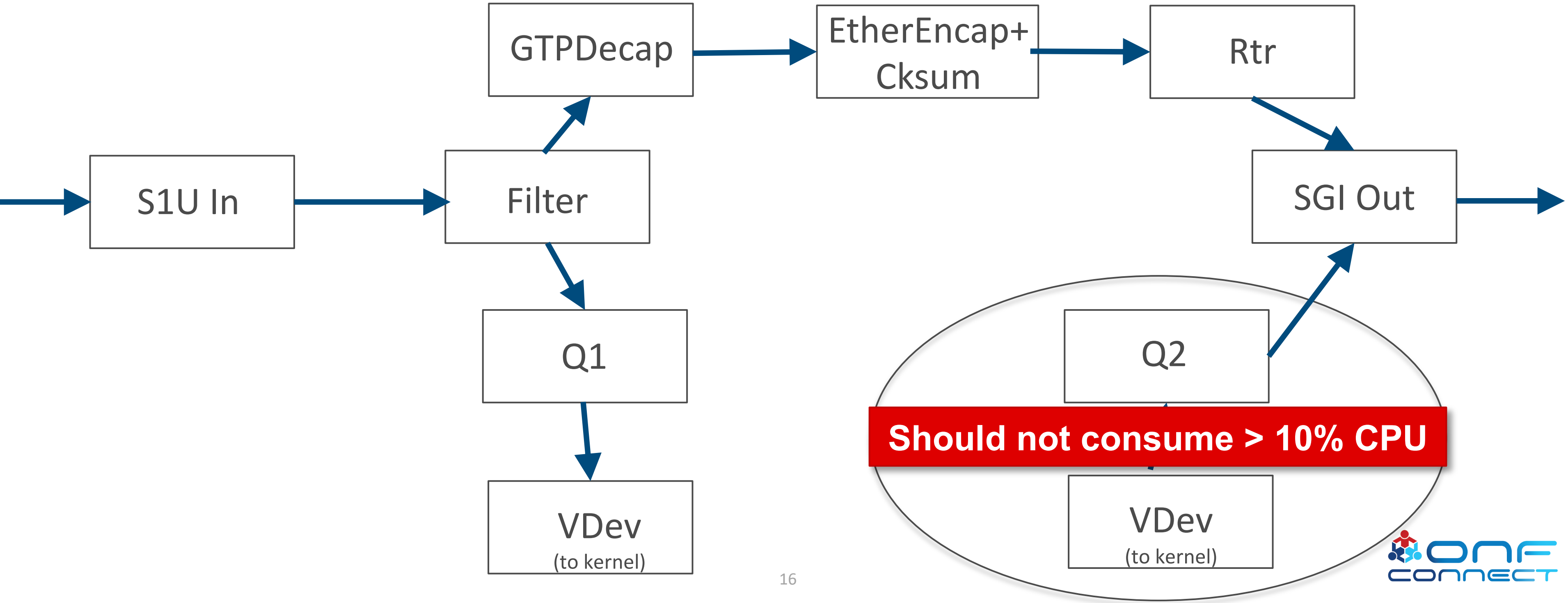
- In terms of CPU utilization & bandwidth



BESS: Resource Aware CPU Scheduling

Allows flexible scheduling policies for the data path

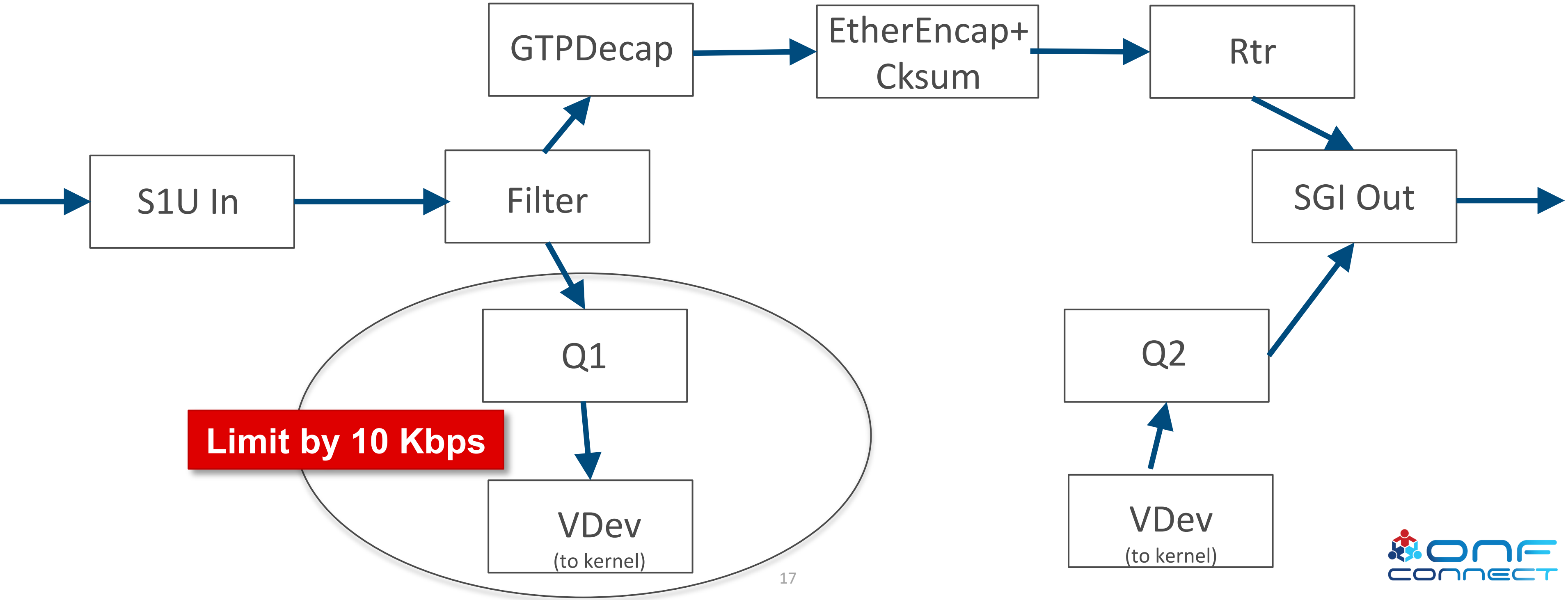
- In terms of CPU utilization & bandwidth



BESS: Resource Aware CPU Scheduling

Allows flexible scheduling policies for the data path

- In terms of CPU utilization & bandwidth



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

ONF CONNECT

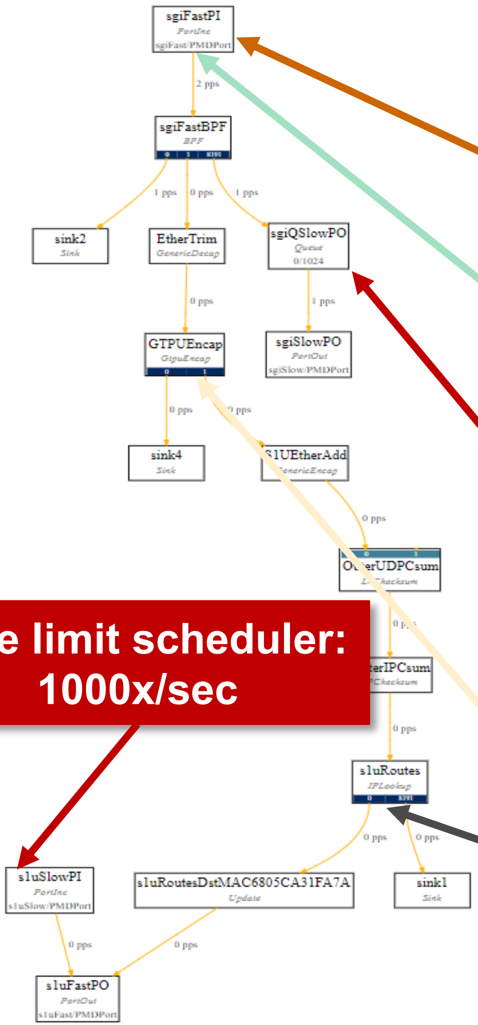
OMEC over BESS

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 <u>at runtime</u>
  - `tcpdump`
  - Visualization tool

DEMO

SPGW-U Downlink DAG



FPI: DPDK PMD

CPU Scheduler: CPU 0

Rate limit scheduler: 1000x/sec

Rate limit scheduler: 1000x/sec

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

Route Control
- `insert_route()`
- `delete_route()`
- `add_neighbor()`
- `delete_neighbor()`

NGIC/OMEC vs SPGWU/BESS

-	NGIC/OMEC	SPGWU/BESS
Runtime model	• rtc	• rtc (dynamic) • pipelined (dynamic)

NGIC/OMEC vs SPGWU/BESS

-	NGIC/OMEC	SPGWU/BESS
Runtime model	• rtc	• rtc (dynamic) • pipelined (dynamic)
Monitoring utilities	• shell (basic stats)	• bessctl shell • `tcpdump` • GUI

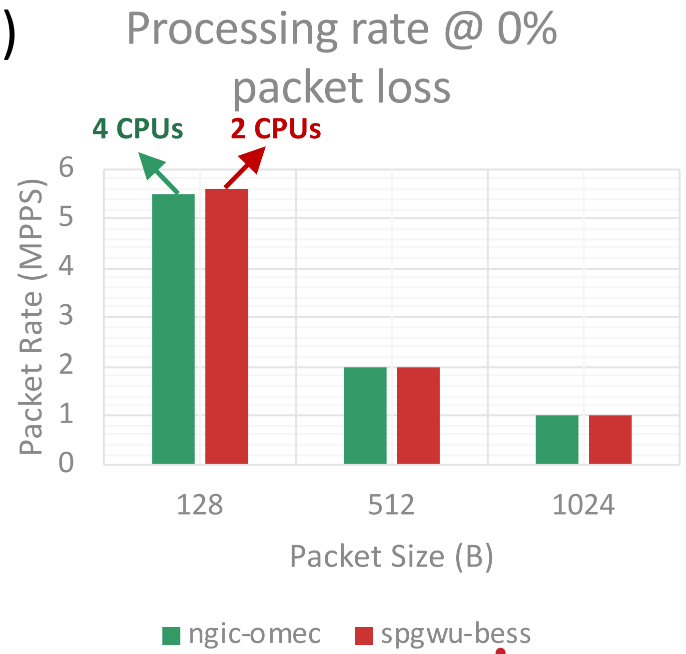
NGIC/OMEC vs SPGWU/BESS

-	NGIC/OMEC	SPGWU/BESS
Runtime model	• rtc	• rtc (dynamic) • pipelined (dynamic)
Monitoring utilities	• shell (basic stats)	• bessctl shell • `tcpdump` • GUI
(Re-)configuration ease	• Process restart • Code re-write	• Process reset not needed • Pipeline graph re-set

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
  - Il_trafficgen



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