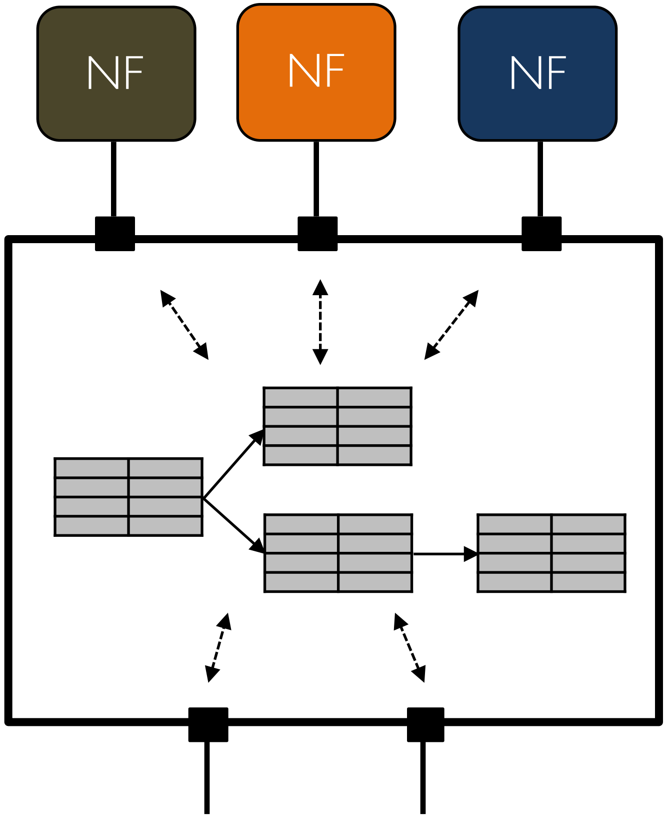
BESS:

A Virtual Switch Tailored for NFV

Sangjin Han, Aurojit Panda, Brian Kim, Keon Jang, Joshua Reich
Sai Edupuganti, Christian Maciocco, Sylvia Ratnasamy, Scott Shenker

Why Another Virtual Switch?

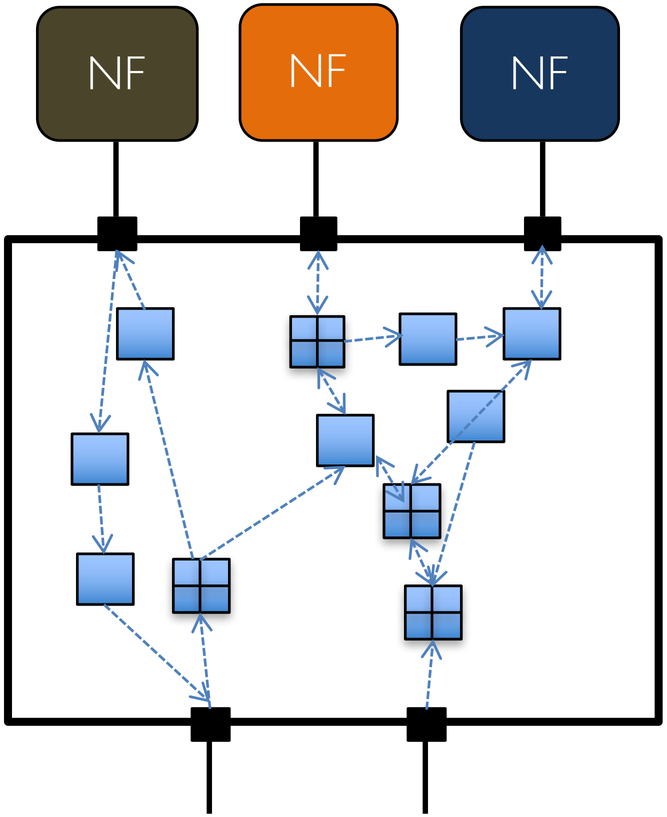
- Does OpenVSwitch meet all the requirements for NFV?



1. Performance
   - OVS (~1Mpps) → OVS-DPDK (~15Mpps)
   - cf. Vanilla DPDK (~59Mpps/core)
   - Packet I/O is only half of the problem
2. Flexibility
   - Custom actions?
   - Stateful packet processing?
3. Extensibility
   - Must enable NFV controller evolution
   - Easily add support for new/niche protocols

Alternative Approach with BESS

- Modular pipeline as a dataflow graph
- Each module can run arbitrary code
  - Not limited by Match/Action semantics
  - Independently extensible & optimizable
- Everything is programmable, not just flow tables.
- You pay only for what you use.
  - No performance cost for unused features

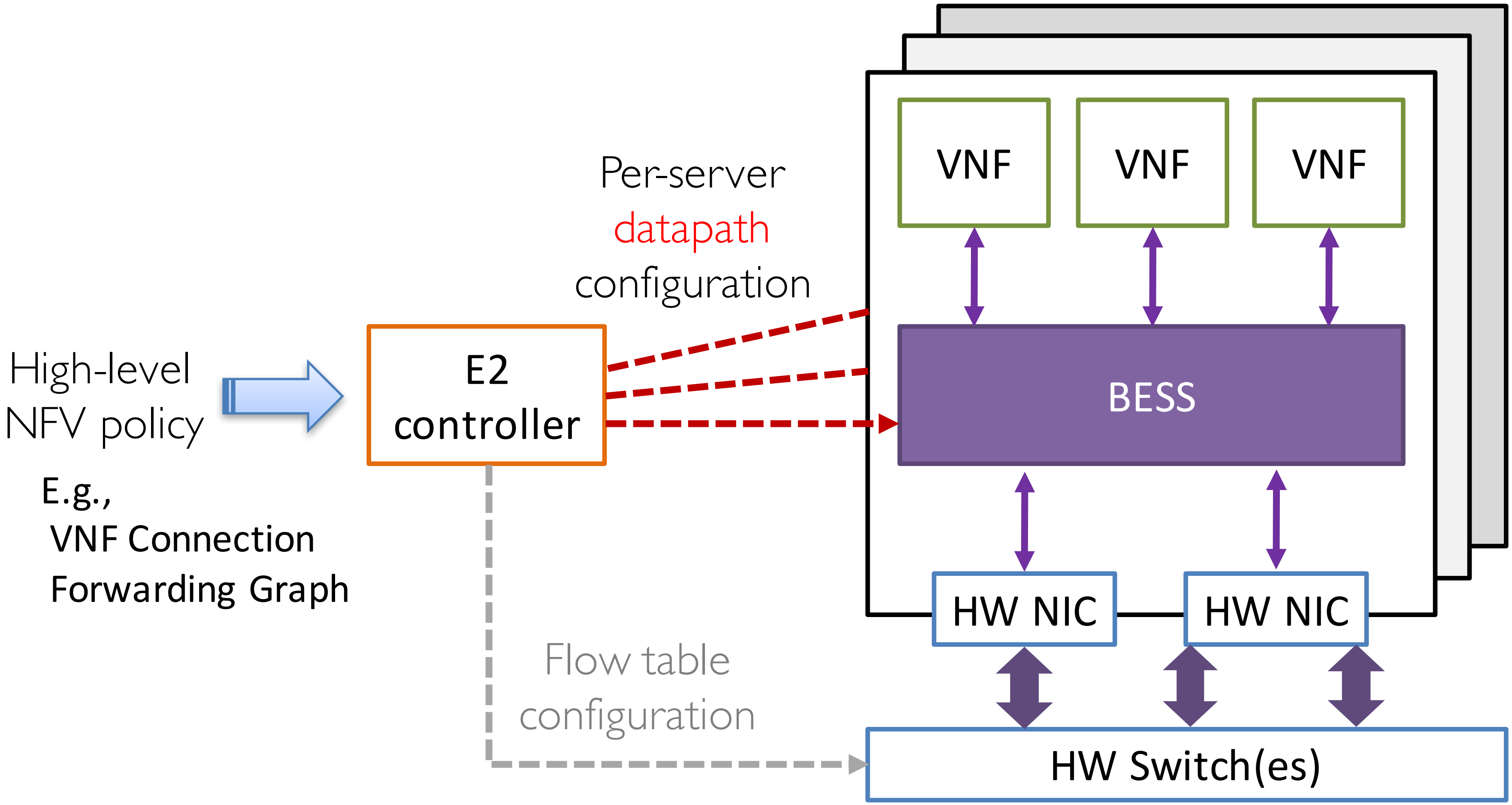


BESS: Berkeley Extensible Software Switch

- BESS is a **programmable platform** for vSwitch dataplane

- Clean-slate internal architecture with NFV in mind
  - Highly extensible & customizable
  - Readily deployable with backward compatibility
  - … all with extreme performance:
    - Sub-microsecond latency
    - Line-rate 40Gbps with min-sized packets on two cores
    - (> 2x faster than other virtual switches)

BESS in E2



High-level NFV policy

E.g.,
VNF Connection
Forwarding Graph

E2 controller

Per-server datapath configuration

Flow table configuration

VNF

VNF

VNF

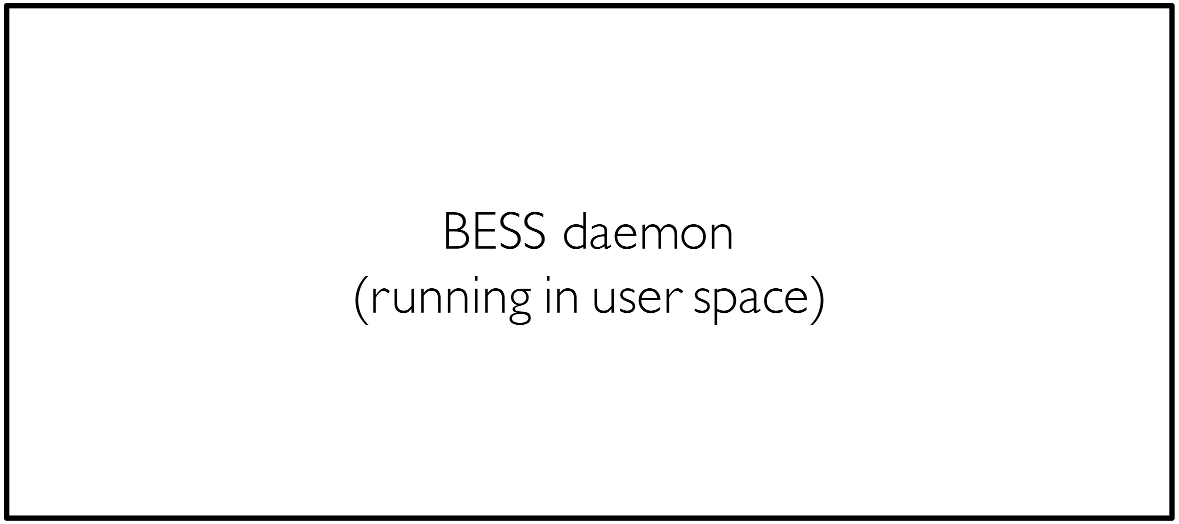
BESS

HW NIC

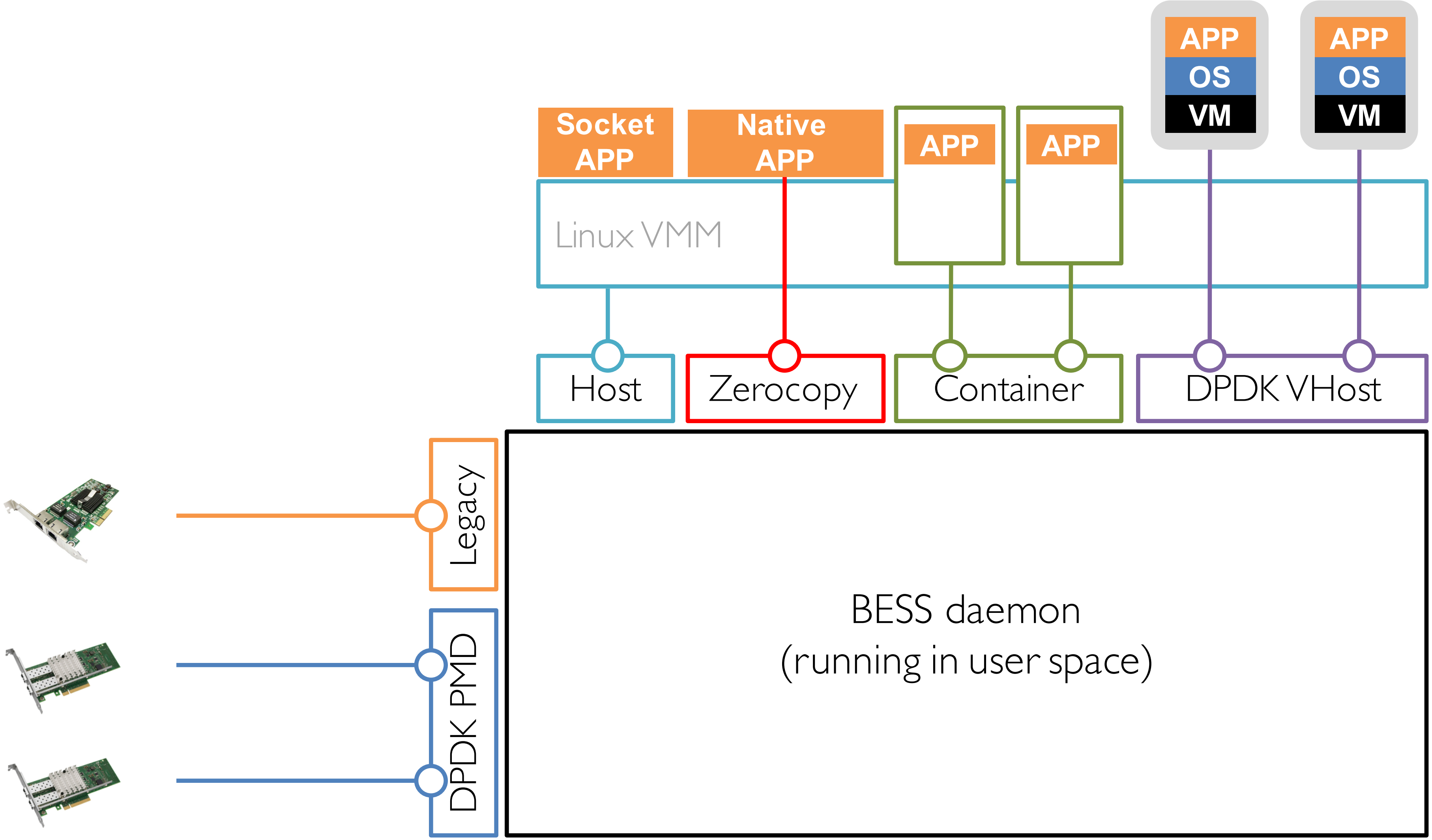
HW NIC

HW Switch(es)

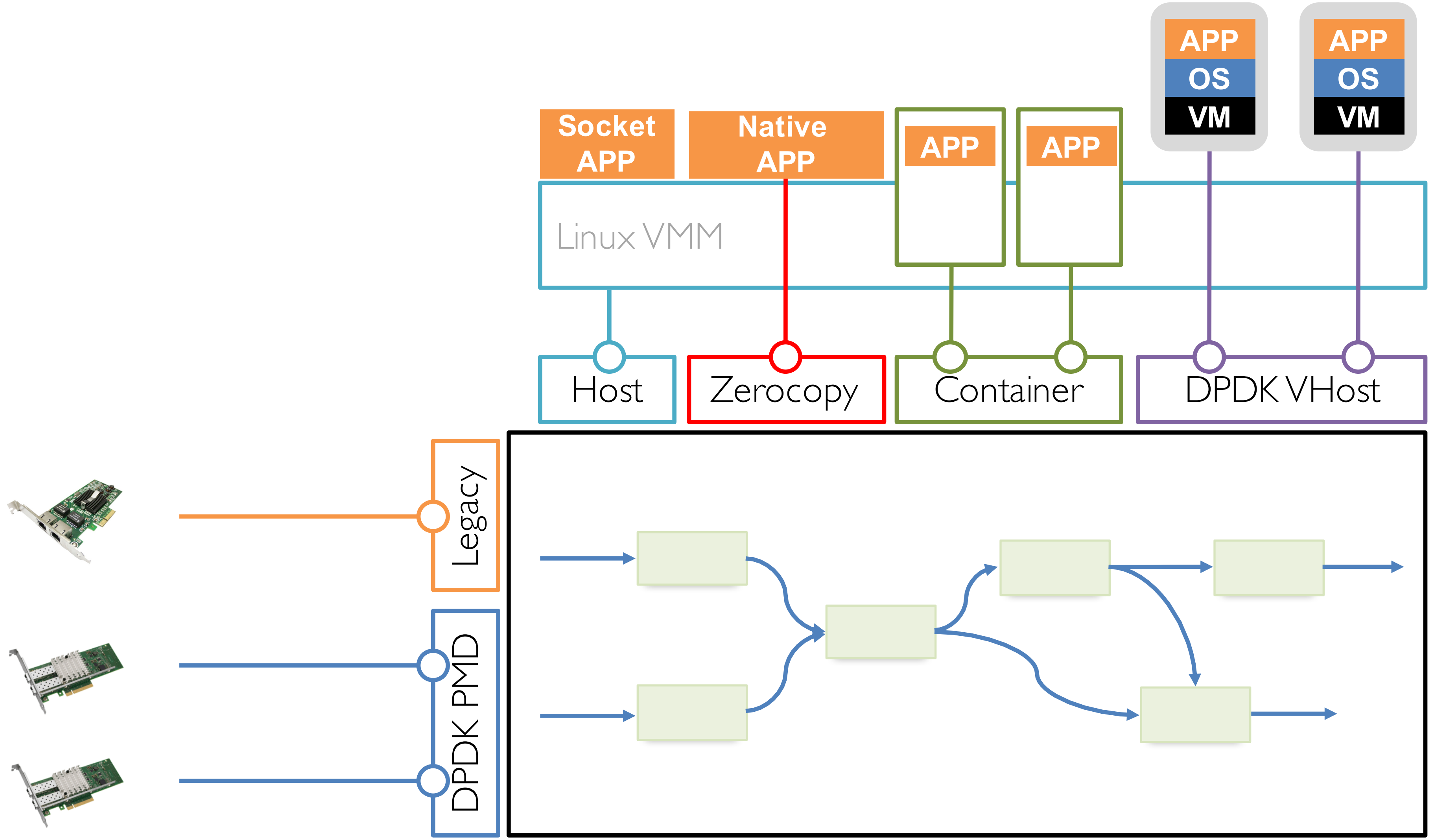
BESS Architecture Overview



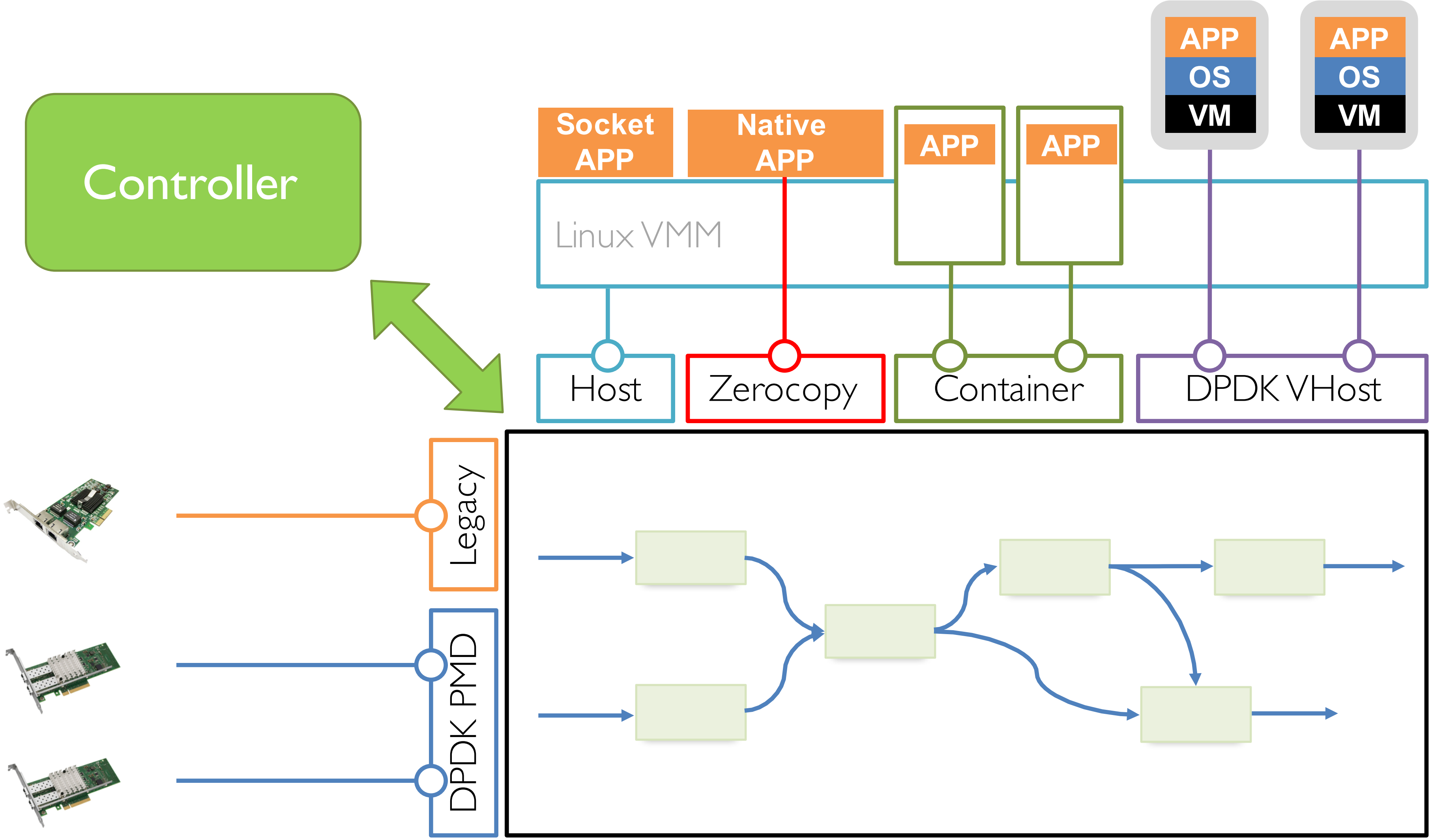
BESS Architecture Overview



BESS Architecture Overview

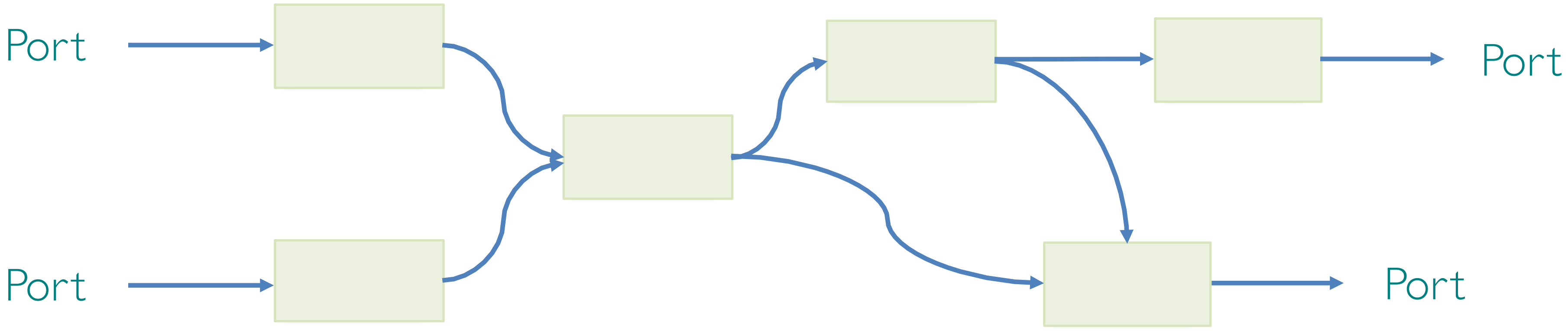


BESS Architecture Overview



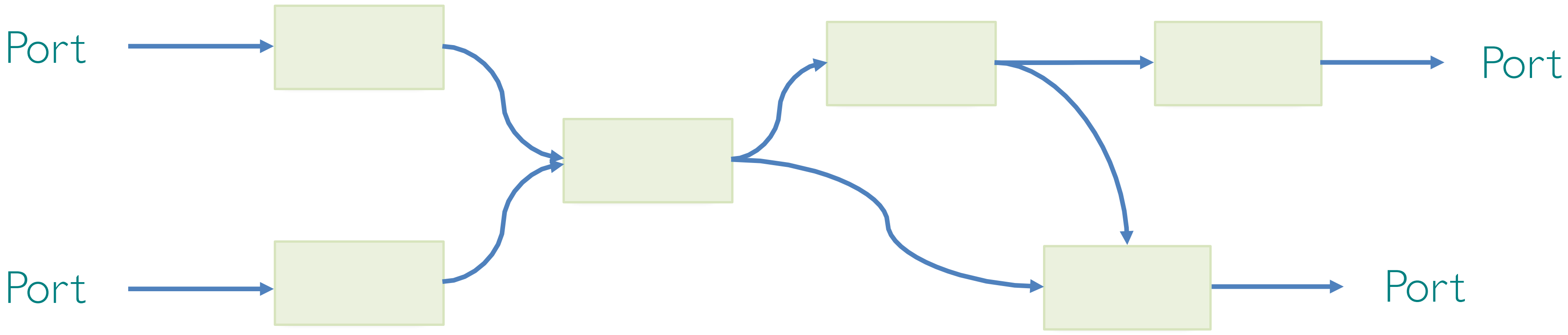
Modular Datapath Pipeline

- External ports are interconnected with "modules" in a dataflow graph (like the Click modular router).
  - You can compose modules to implement your own datapath.
  - Developing a new module is easy.



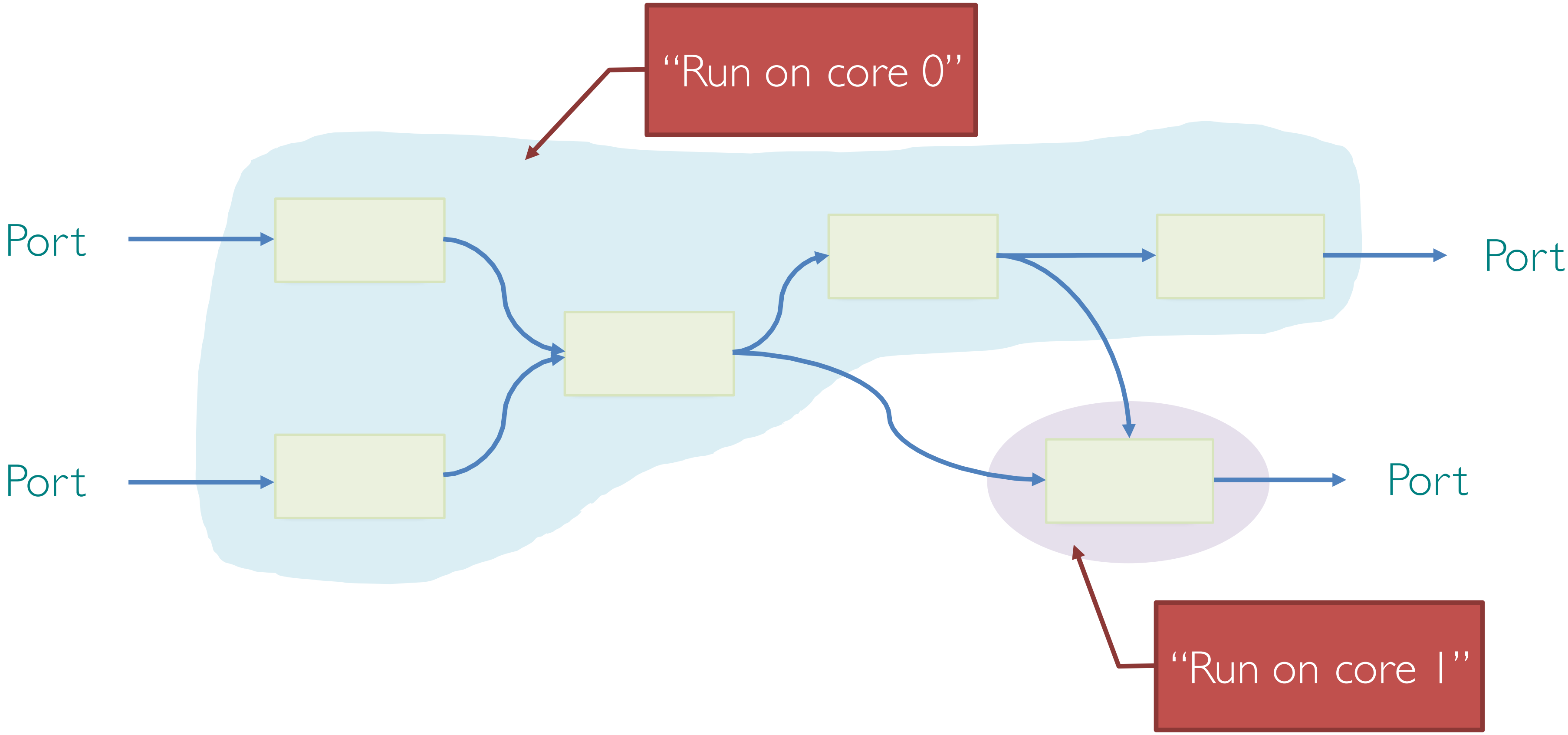
Resource-Aware CPU Scheduler

- BESS allows flexible scheduling policies for the data path.
  - In terms of CPU utilization and bandwidth. Examples:



Resource-Aware CPU Scheduler

- BESS allows flexible scheduling policies for the data path.
  - In terms of CPU utilization and bandwidth. Examples:

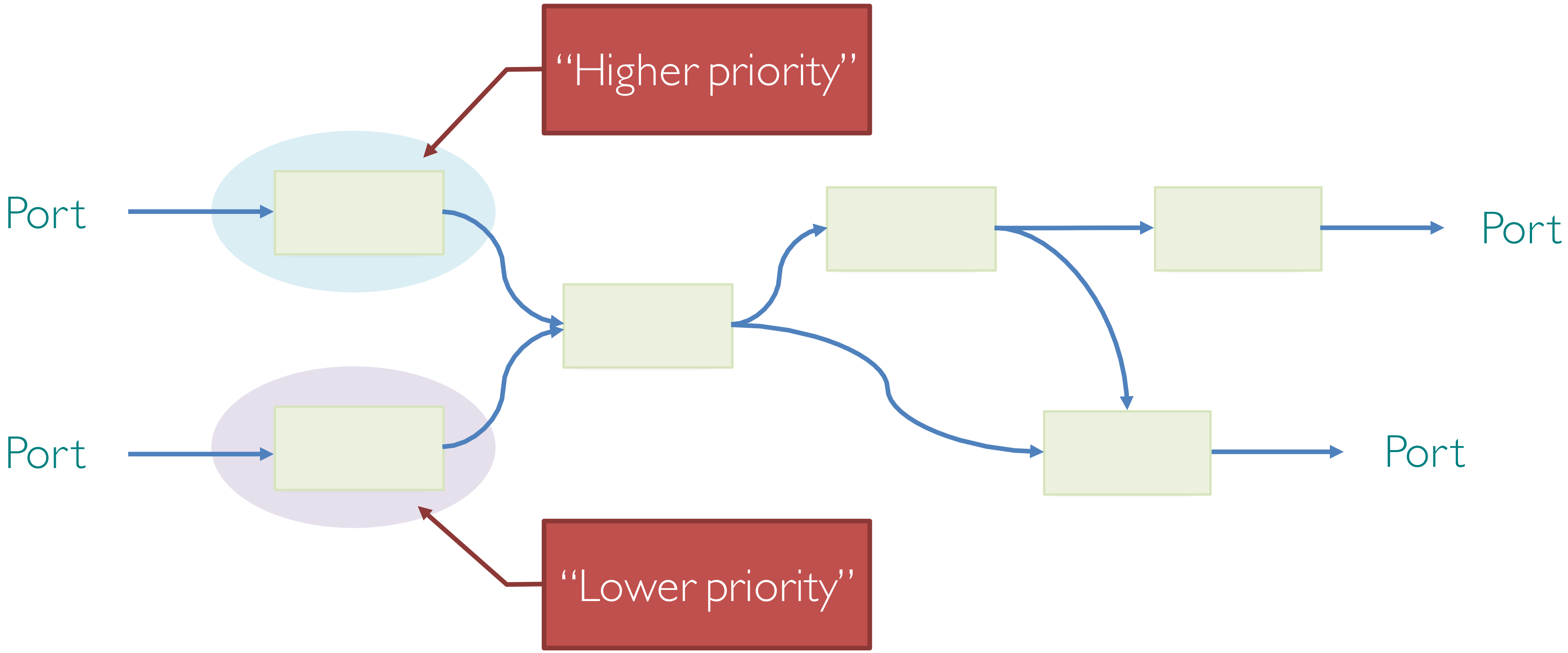


“Run on core 0”

“Run on core 1”

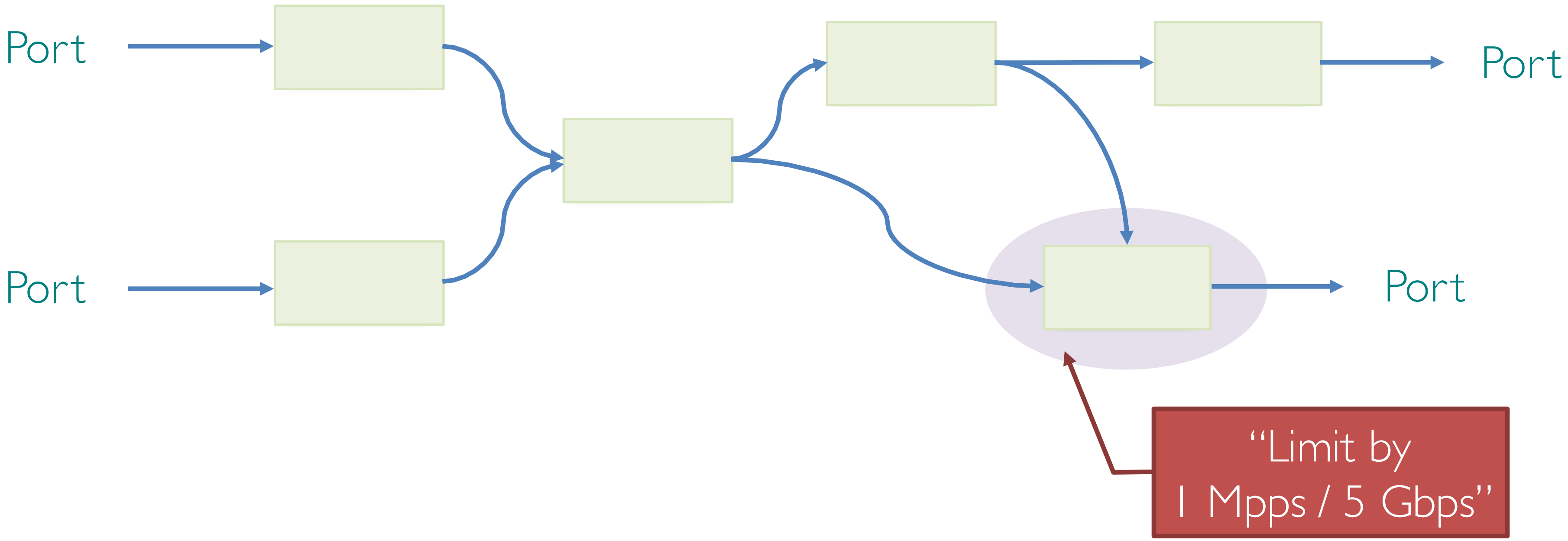
Resource-Aware CPU Scheduler

- BESS allows flexible scheduling policies for the data path.
  - In terms of CPU utilization and bandwidth. Examples:



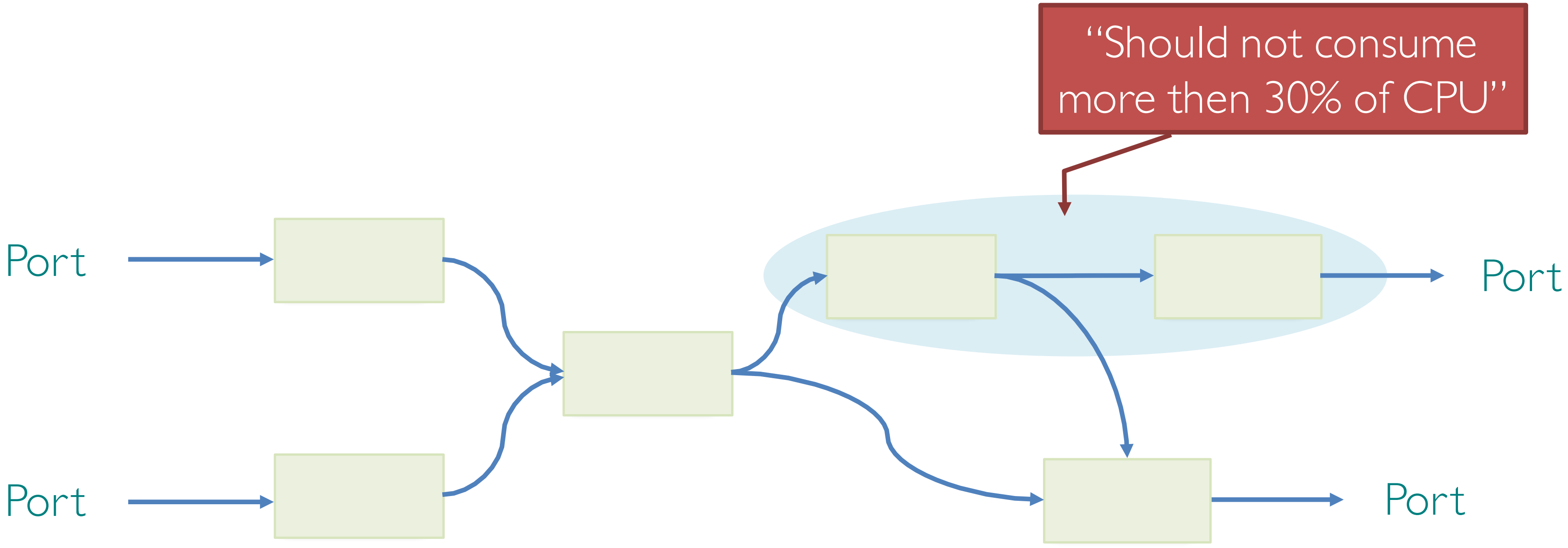
Resource-Aware CPU Scheduler

- BESS allows flexible scheduling policies for the data path.
  - In terms of CPU utilization and bandwidth. Examples:



Resource-Aware CPU Scheduler

- BESS allows flexible scheduling policies for the data path.
  - In terms of CPU utilization and bandwidth. Examples:



“Should not consume more then 30% of CPU”

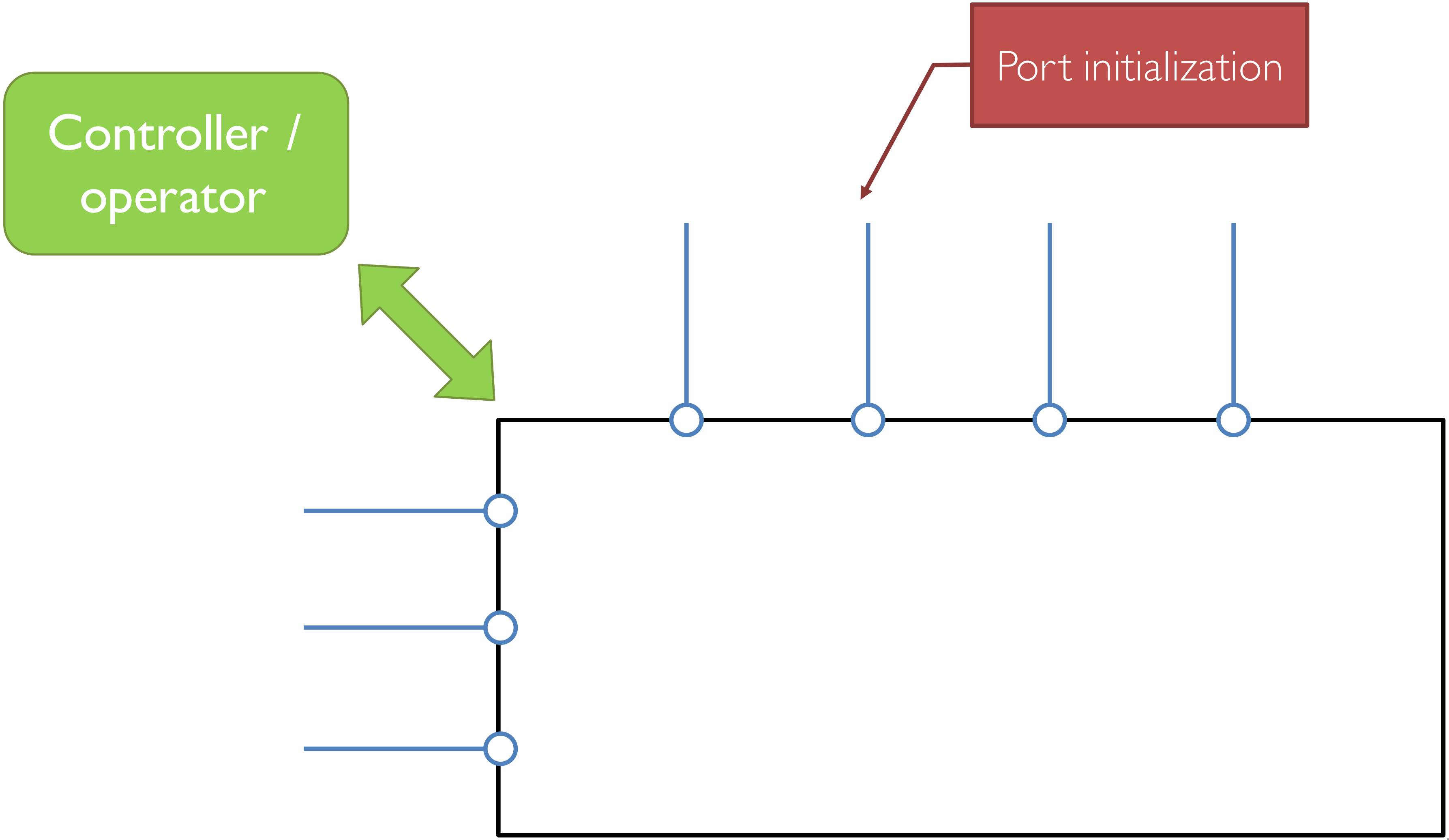
Control Interface

- JSON-like structured messages between BESS and controller

- 3 ways to control the BESS datapath
  - Python/C APIs
  - Scriptable configuration language
  - Cisco iOS-like CLI

- Everything is run-time configurable!

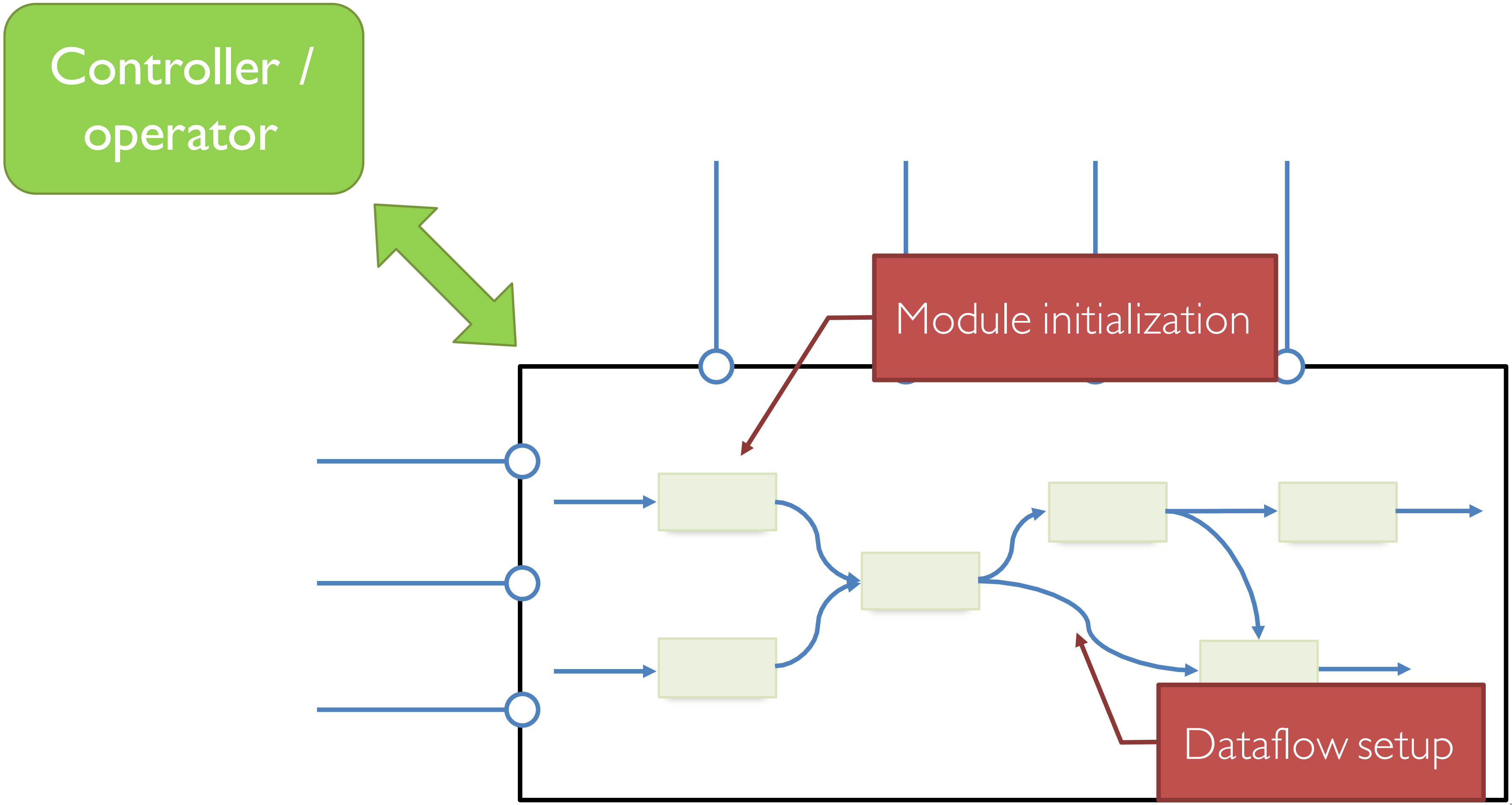
Control Interface



Controller / operator

Port initialization

Control Interface

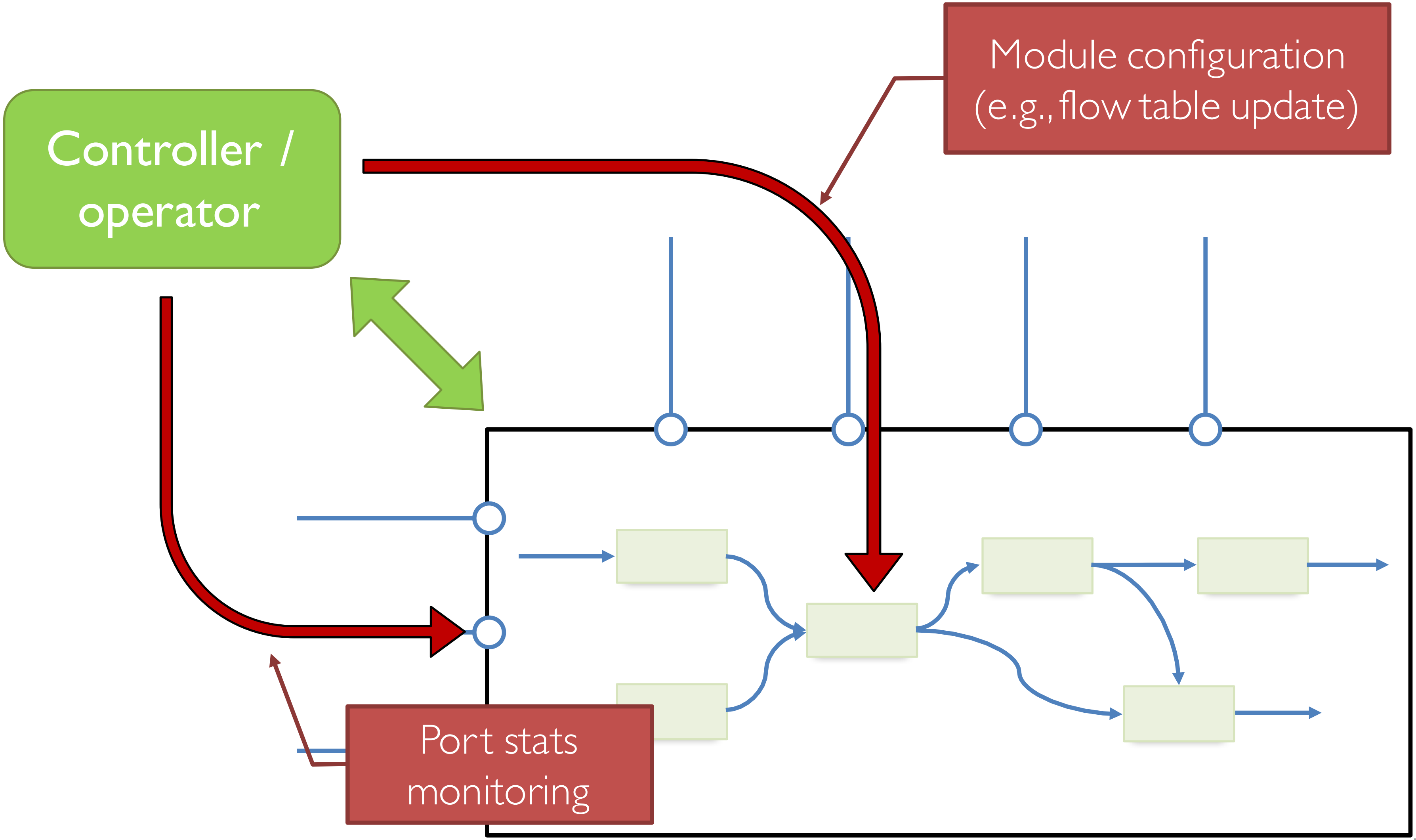


Controller / operator

Module initialization

Dataflow setup

Control Interface



Controller / operator

Module configuration (e.g., flow table update)

Port stats monitoring

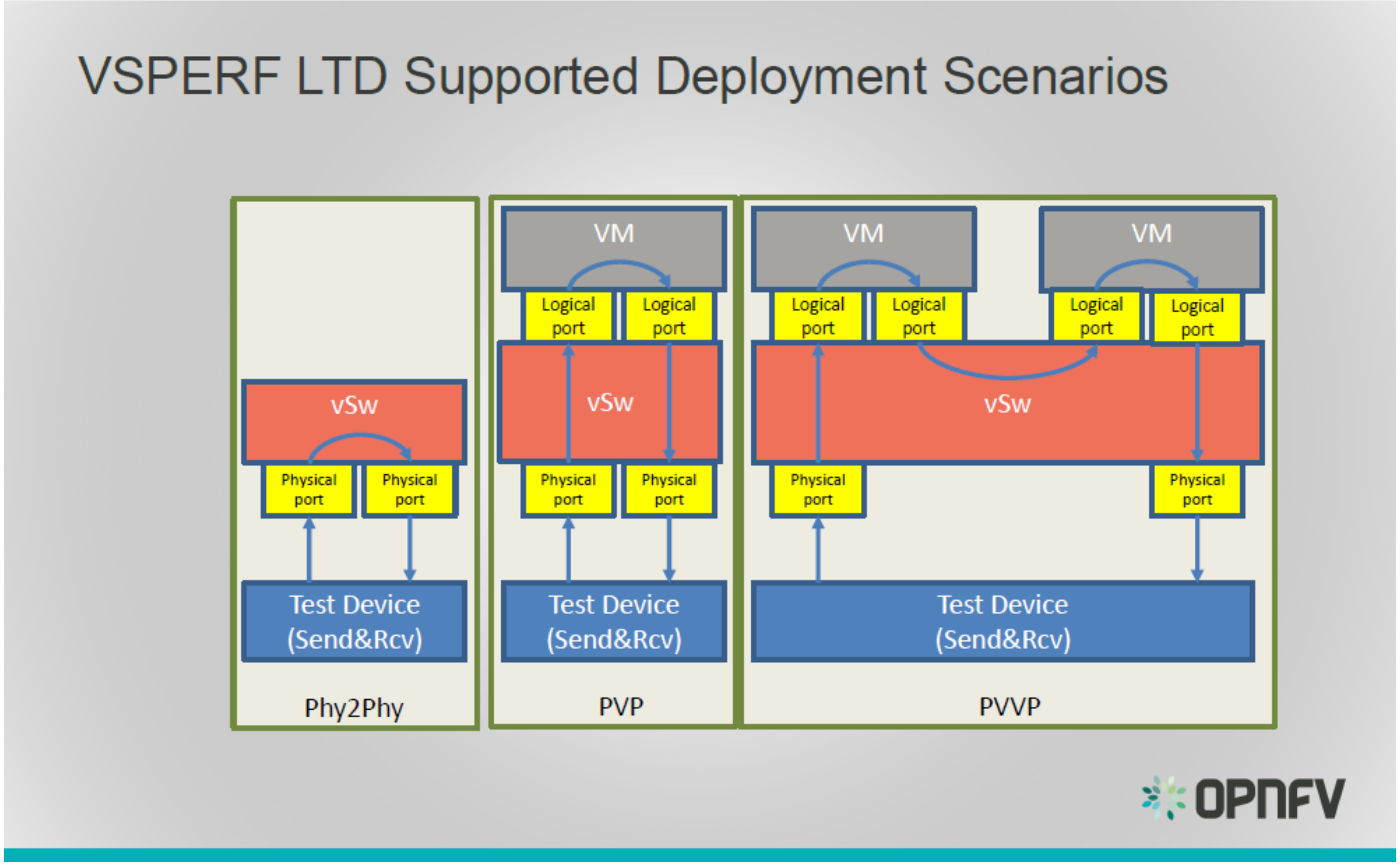
Performance?

Minimum Framework Overhead

- Packet buffer allocation/deallocation
  - ~10 CPU cycles per packet
- CPU scheduling
  - ~50 CPU cycles per round
  - Scales well with thousands of traffic classes
- Dynamic per-packet metadata attributes
  - Zero instruction overhead for access
  - Optimal CPU cache-line usage

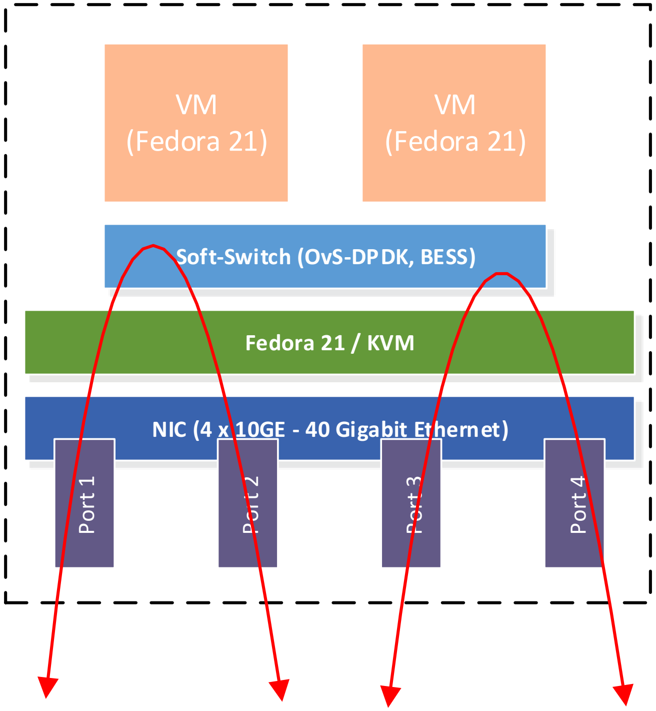
Performance Evaluation

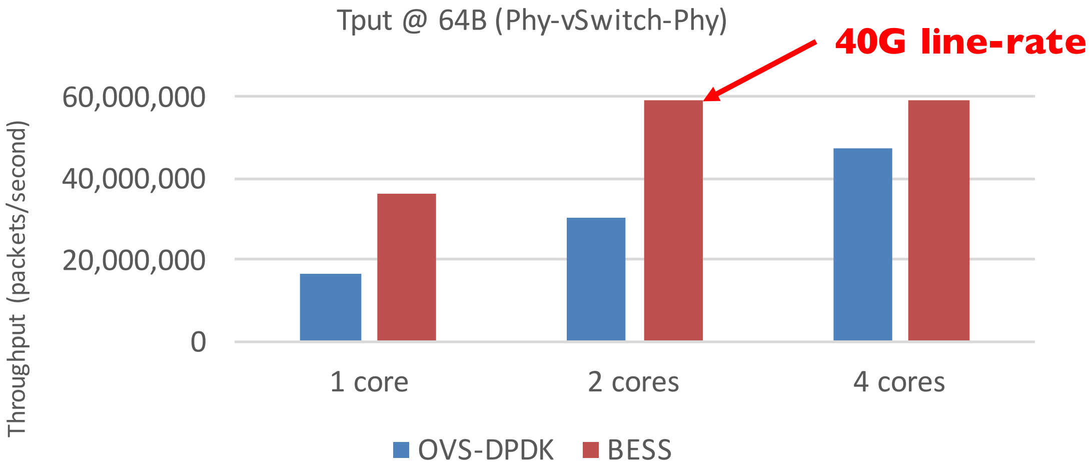
- OPNFV VSPERF usage models



VSPERF LTD Supported Deployment Scenarios

1. Phy-Phy Performance

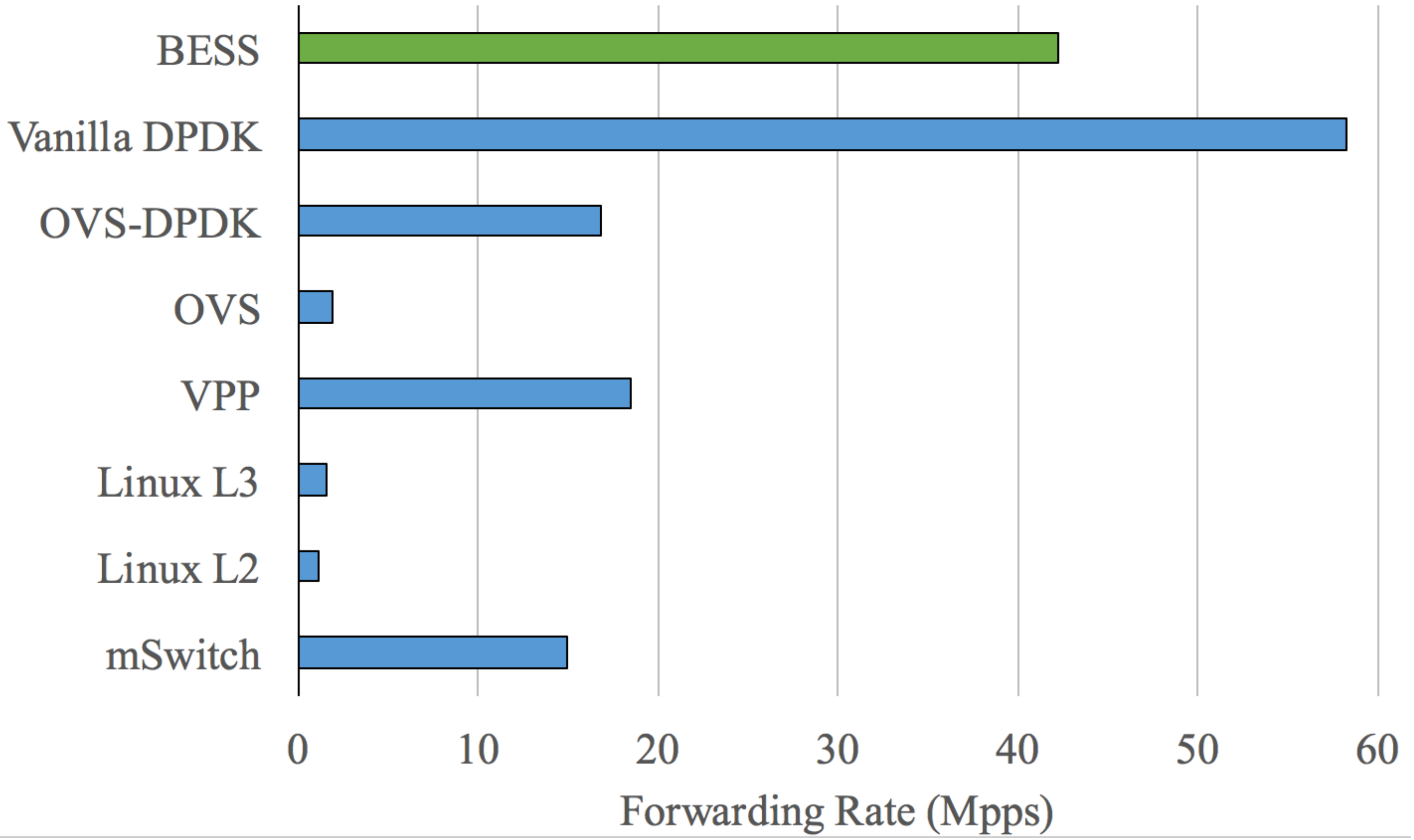




40G line-rate

Source: Irene Liew, Anbu Murugesan – Intel IOTG/NPG

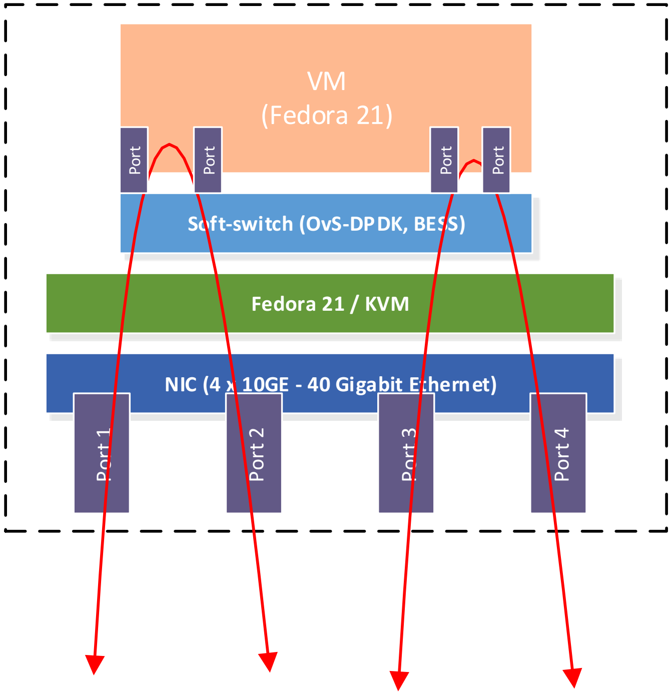
1. Phy-Phy Performance



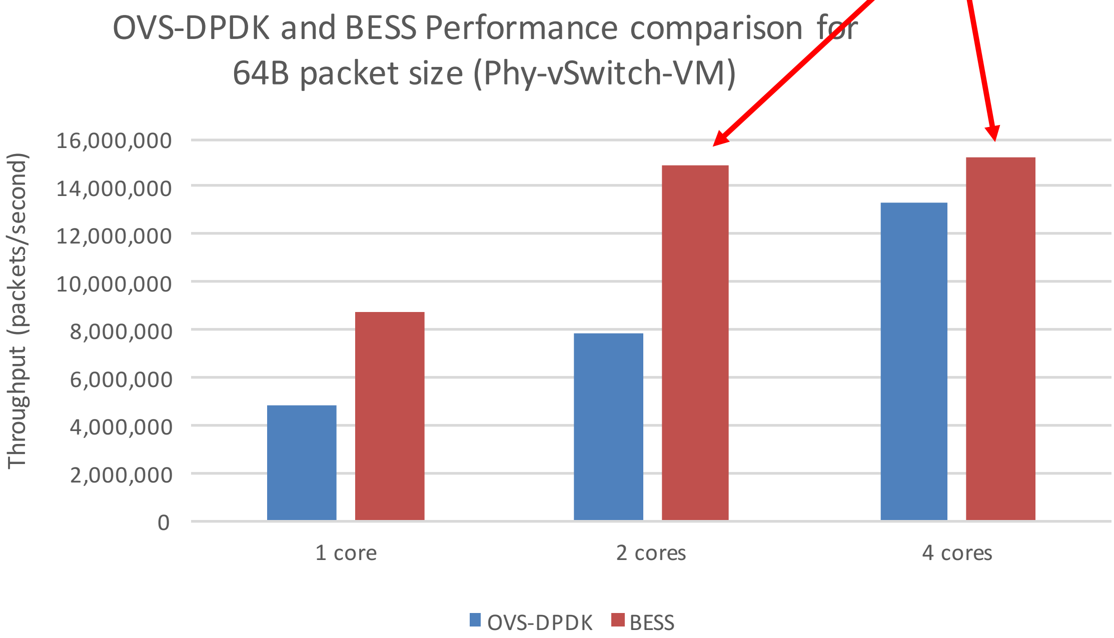
Data sources:

- BESS, Vanilla DPDK, VPP: measured on a 2.6GHz Xeon E5-2650 v2 machine
- OVS, Linux L2/L3: Emmerich et al. "Performance Characteristics of Virtual Switching", CloudNet 2014
- OVS-DPDK: Intel ONP 2.1 Performance Test Report
- mSwitch: (link bottlenecked w/ large batch sizes @ 3.2GHz) Honda et al. "mSwitch: A Highly-Scalable, Modular Software Switch", SOSR 2015

2. Phy-NF-Phy Performance



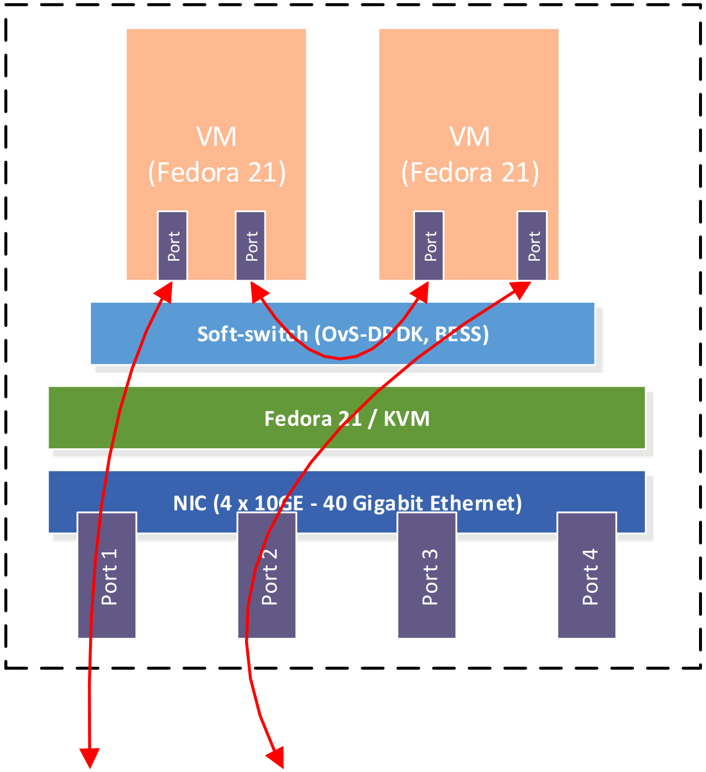
VM bottleneck

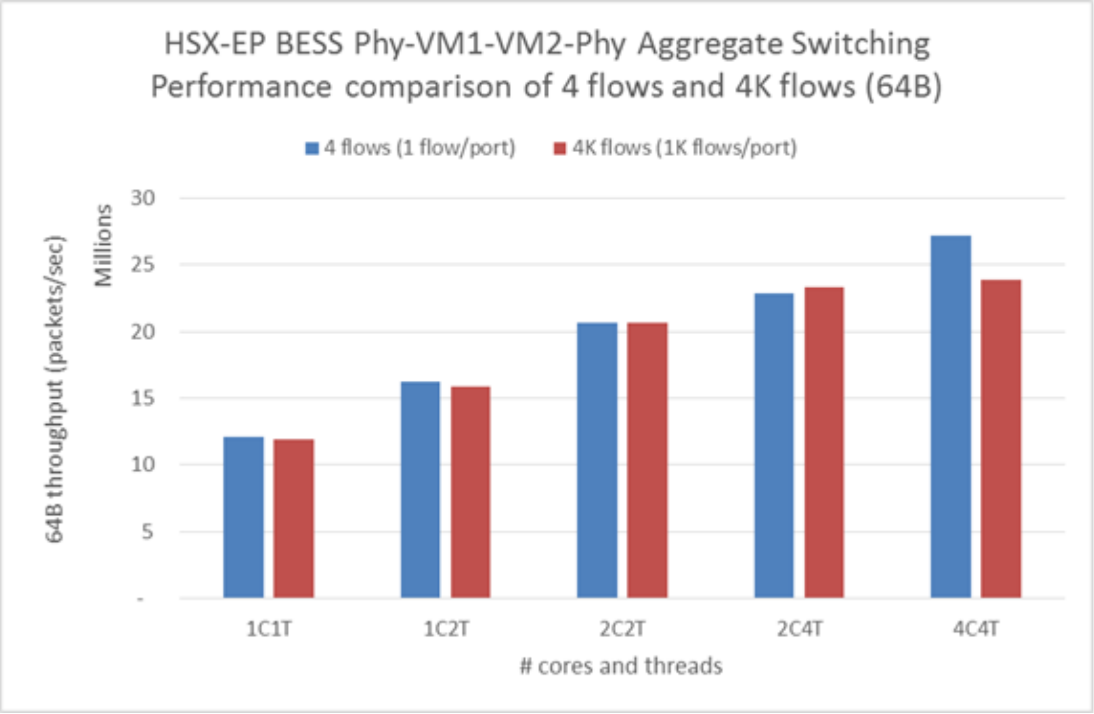


Source: Irene Liew, Anbu Murugesan – Intel IOTG/NPG

3. Phy-NF-NF-Phy Performance

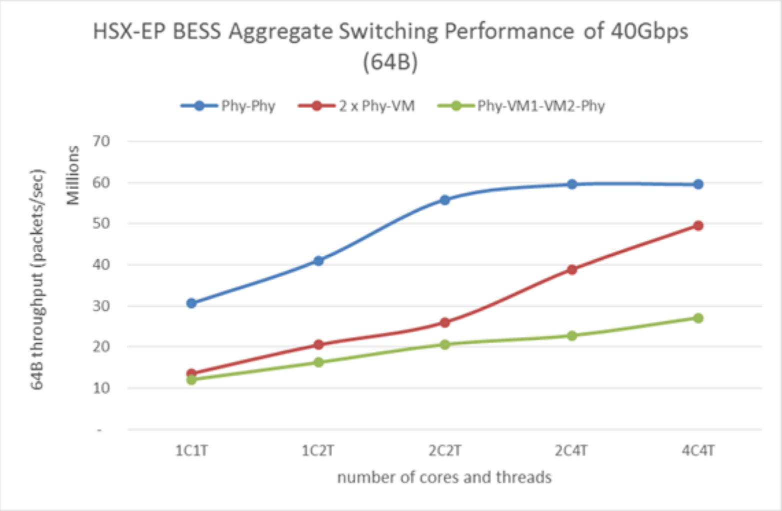
- BESS outperforms OVS-DPDK by a factor of 4-5x*

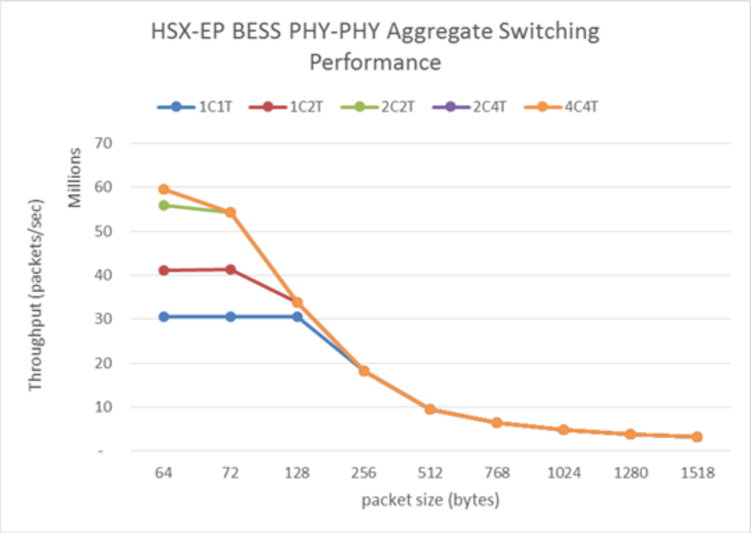


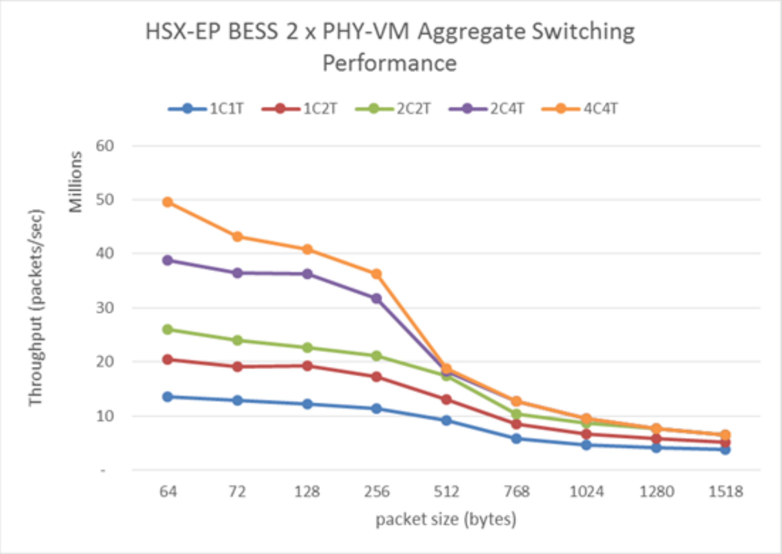


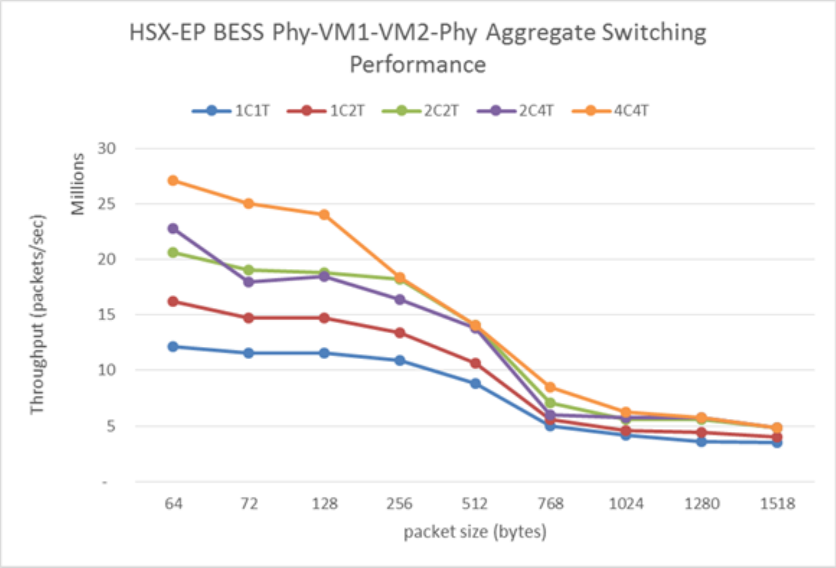
* Source: Intel ONP 2.1 Performance Test Report

Multi-Core/Thread Scalability

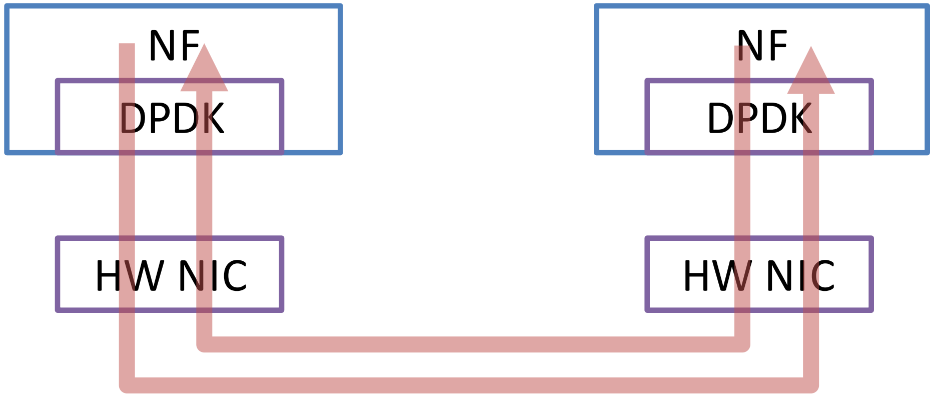

HSX-EP BESS Aggregate Switching Performance of 40Gbps (64B)


HSX-EP BESS PHY-PHY Aggregate Switching Performance

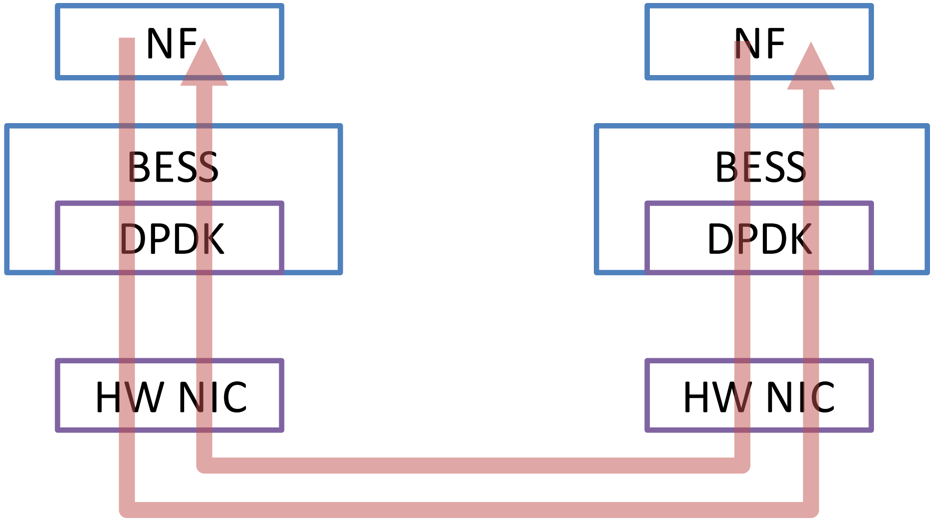

HSX-EP BESS 2 x PHY-VM Aggregate Switching Performance


HSX-EP BESS Phy-VM1-VM2-Phy Aggregate Switching Performance

Round-Trip Latency



RTT: 8.22us



RTT: 8.82us

- Increase of 0.60us (0.15us per BESS traverse)

Summary

- BESS is an ideal vSwitch platform for NFV
  - High performance
    - Sub-microsecond latency/jitter
    - Small packet 40Gbps throughput with only 1-2 cores
  - Full flexibility and extensibility

- Available on GitHub: https://github.com/netsys/bess
  - Under BSD3 License
  - ~30k lines in C and Python, supporting
    - Linux 3.x / 4.x (x86_64), DPDK 16.04
    - QEMU/KVM virtual machines, Docker/LXC containers