



Berkeley Extensible Software Switch (BESS)

Sangjin Han – UC Berkeley

Christian Maciocco – Intel

DPDK US Summit - San Jose - 2016



BESS



BESS:

A Virtual Switch Tailored for NFV

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

SPAN: Software Principle for Advanced Networking

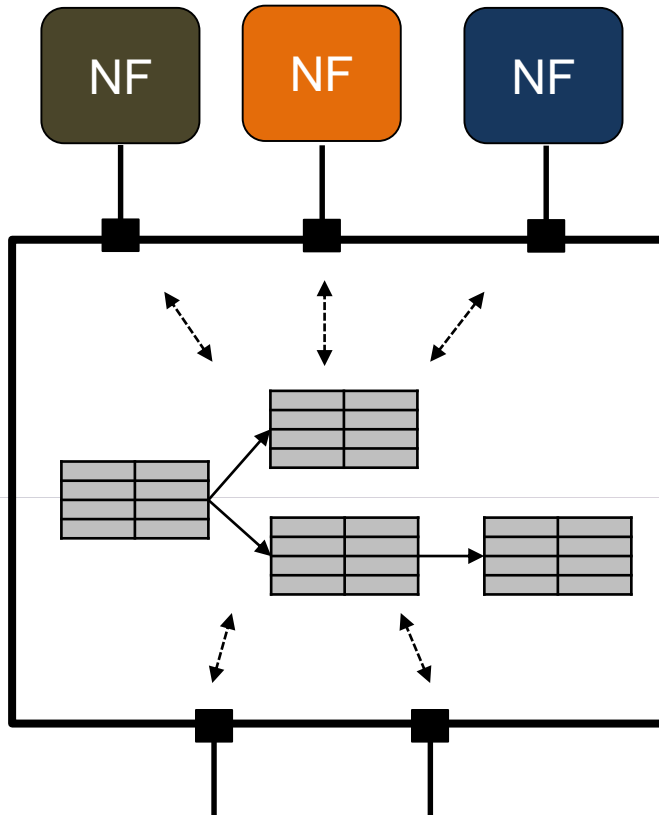
<http://span.cs.Berkeley.edu>



Why Another Virtual Switch ?



- Does Open vSwitch meet all the requirements for NFV?

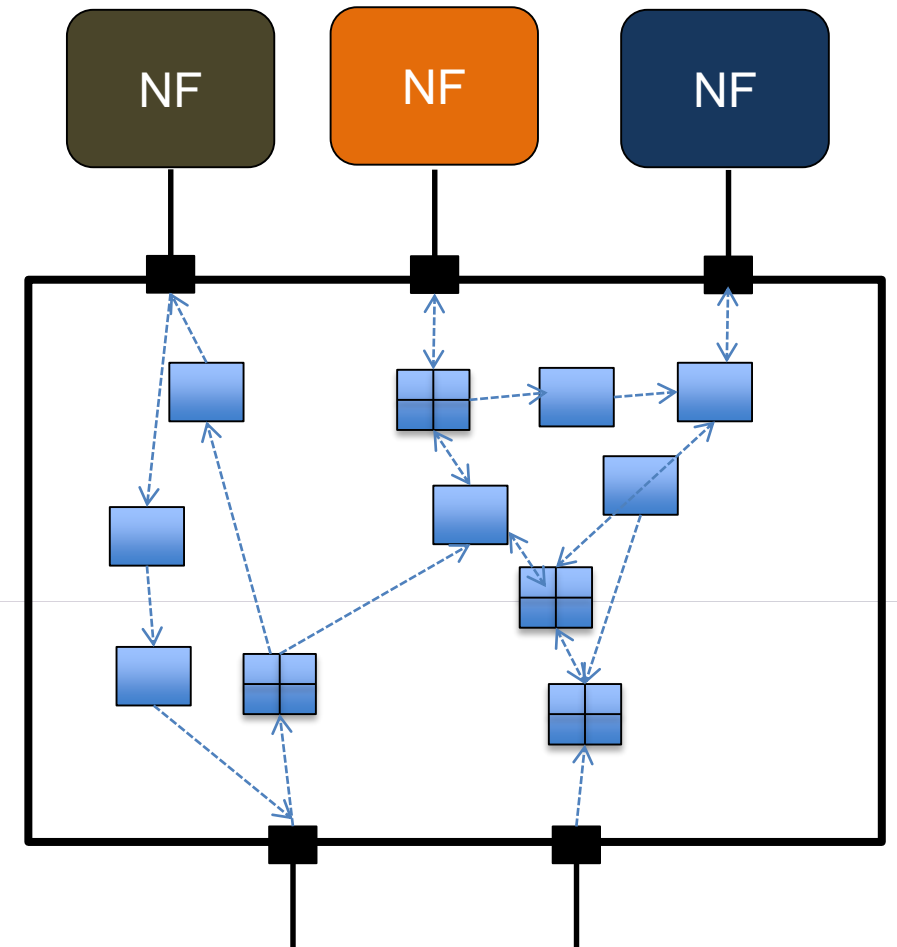


1. Performance
 - OVS (~1Mpps) → OVS-DPDK (~15Mpps)
 - Vanilla DPDK (~59 Mpps/core)
 - Packet I/O is only half of the problem
2. Flexibility
 - Custom actions ?
 - Stateful packet processing ?
3. Extensibility
 - Must enable NFV controller evolution
 - Easily add support for new/niche protocols

Alternative Approach with BESS



- Modular pipeline as a dataflow graph
 - 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



Berkeley Extensible Software Switch (BESS)



- ▶ 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 existing virtual switches)

What can you Build on BESS ?

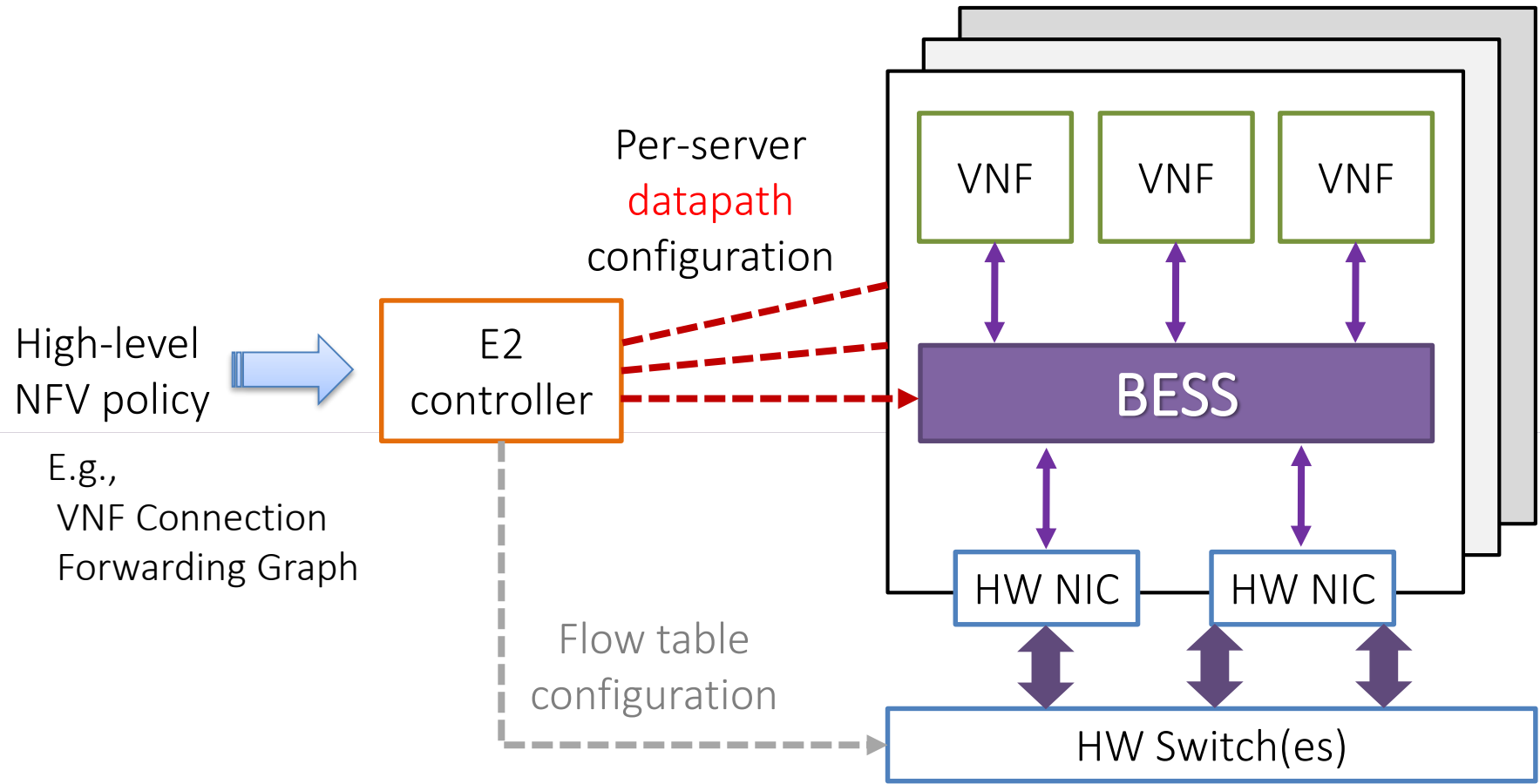


- ▶ NFV virtual switches
- ▶ Other possible usages:
 - ▶ Network virtualization for multi-tenant datacenters
 - ▶ Network functions (firewall, VPN, ADC, etc.)
 - ▶ Traditional L2/L3 switches
 - ▶ “Smart” NICs
 - ▶ ...

Our use Case: Elastic Edge (E2)



- E2 is a research prototype of our NFV platform
 - “E2: A Framework for NFV Applications”, Palkar et al., In *ACM SOSP*, 2015



Performance – Minimum Framework Overhead



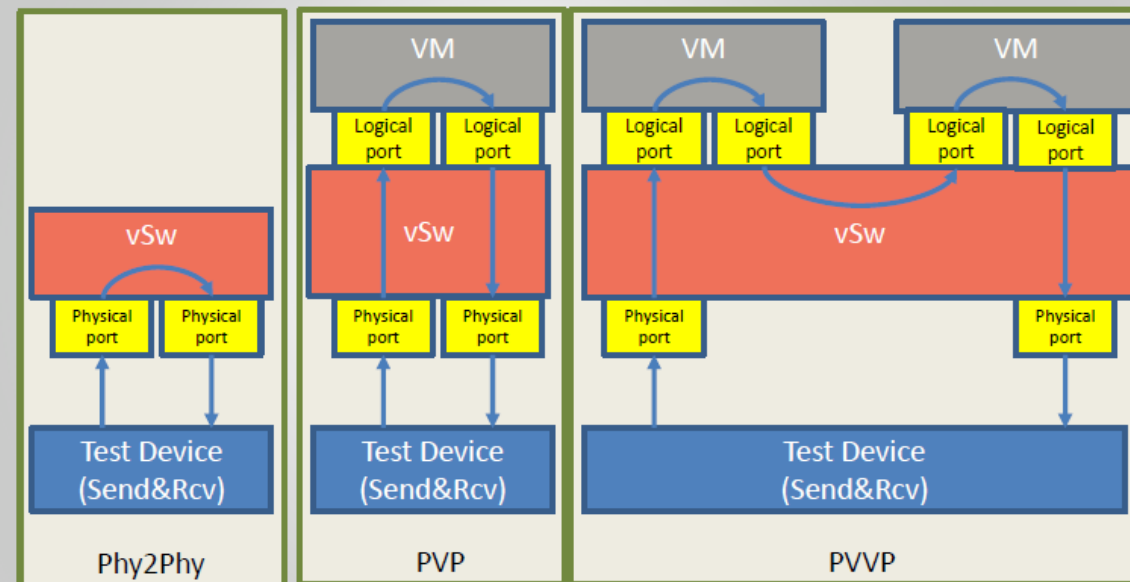
- ▶ Packet buffer allocation/deallocation
 - ▶ ~10 CPU cycles per packet
- ▶ CPU scheduling
 - ▶ ~50 CPU cycles per round
 - ▶ Scales well with thousands of traffic classes
- ▶ Dynamic per-packet metadata attributes
 - ▶ Zero instruction overhead for access
 - ▶ Optimal CPU cache-line usage

Performance Evaluation



► OPNFV VSPERF usage models

VSPERF LTD Supported Deployment Scenarios



System Configuration



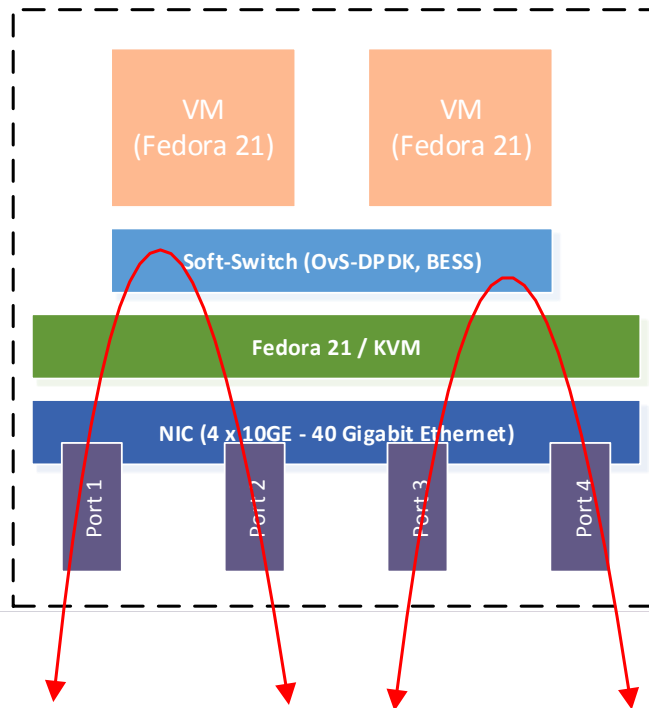
Hardware	
Platform	Wildcatpass S2600WT2
CPU	Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz
Chipset	Intel® C610 series chipset (Wellsburg)
No of CPU	1
Cores per CPU	14 (HT Enabled. Total: 28)
LL CACHE	35840K
QPI/DMI	Auto
PCIe	Port3A and Port3C(x8)
MEMORY	Micron 16GB 1Rx4 PC4-2133MHz, 16GB per channel, 4 Channels, 64GB Total
NIC	2 x Intel® Ethernet X710-DA2 Adapter (Total: 4 Ports)
NIC Mbps	10000
BIOS	Version: SE5C610.86B.01.01.0008.021120151325 & Date: 02/11/2015

Software	
OS	Fedora 23
Kernel version	4.2.3-300.fc23.x86_64
Host Machine boot setting	Hugepage size = 1G ; No. of Hugepages = 16 Hugepage size=2MB; No. of Hugepages = 2048 isolcpus=1-9,21-29
Software version	DPDK 16.07 BESS (Commit id: 940b2f114dcb9989b4fd2d3a24a4612454c21840)
Host settings	firewall, iptables, Selinux, network Manager disabled ip_forward = 0 set uncore frequency to the max ratio kill -9 dhclient rmmod ipmi_si ipmi_devintf ipmi_msghandler lpc_ich bridge sepci -s 00:03.0 184.l=1408 sepci -s 00:03.2 184.l=1408
IXIA TEST	RFC 2544 0% PACKET LOSS, 2 flows total/two ports
BIOS settings	P-state Disabled, C-State Disabled, HT ON and Turbo Boost Disabled
Fortville NIC FW Version	FW 4.33 API 1.2 NVM 04.04.02 eetrack 8000191c
IXIA TEST	RFC 2544 0% PACKET LOSS, 4 total flows/4 ports

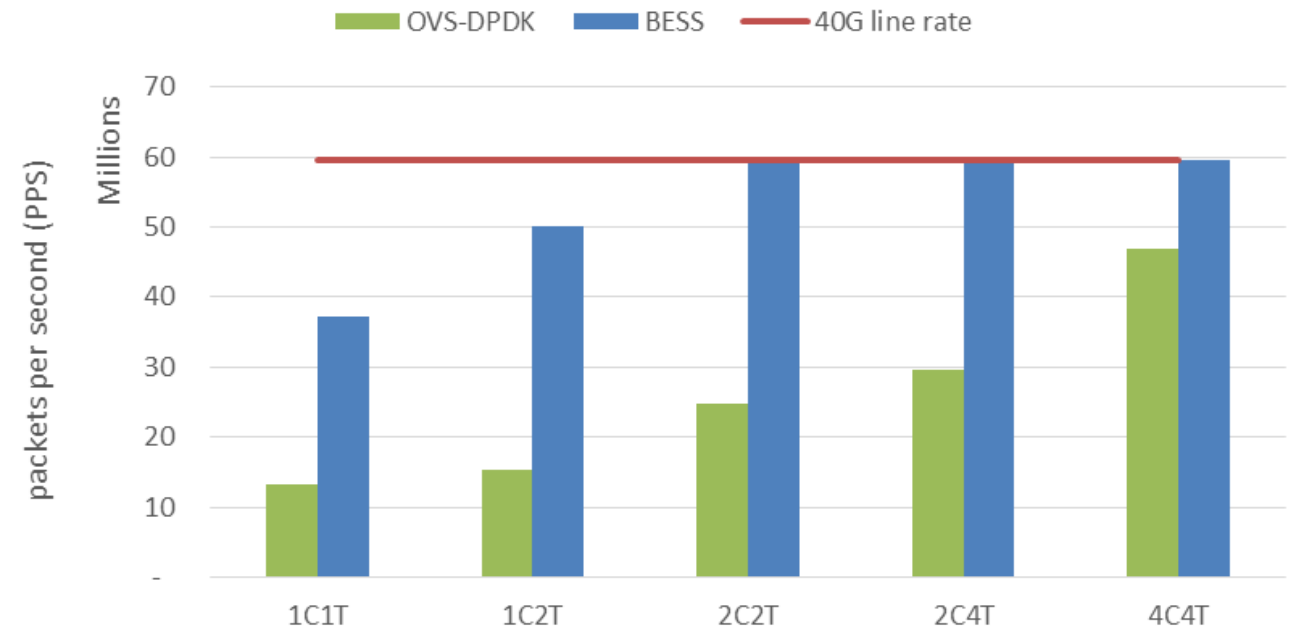
Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804

Phy-to-Phy Performance (1/2)



Phy-Phy Throughput Performance @ 64B on Intel®
Xeon® CPU E5-2697 v3

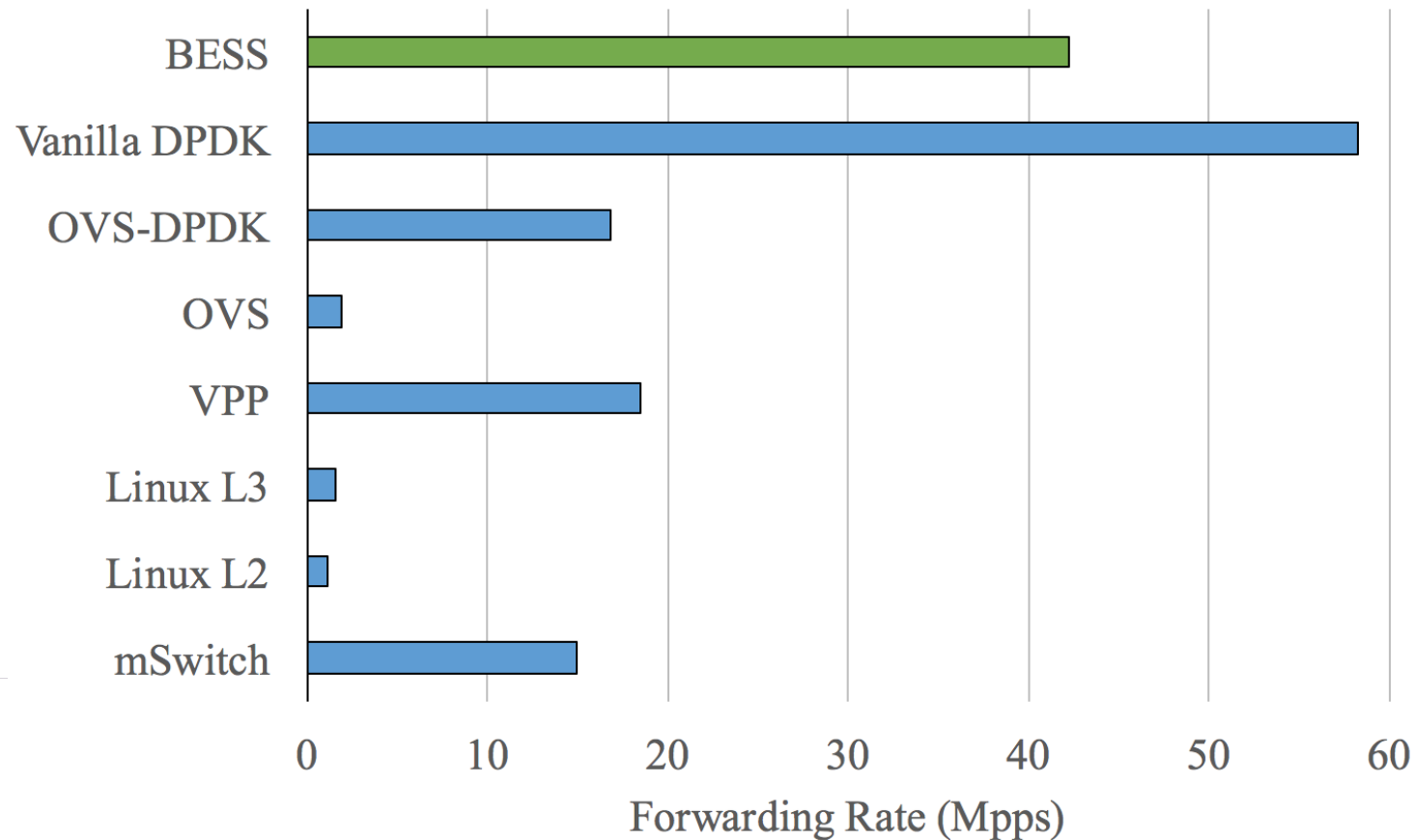


Source: Irene Liew, Anbu Murugesan – Intel Network Packet Group

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804

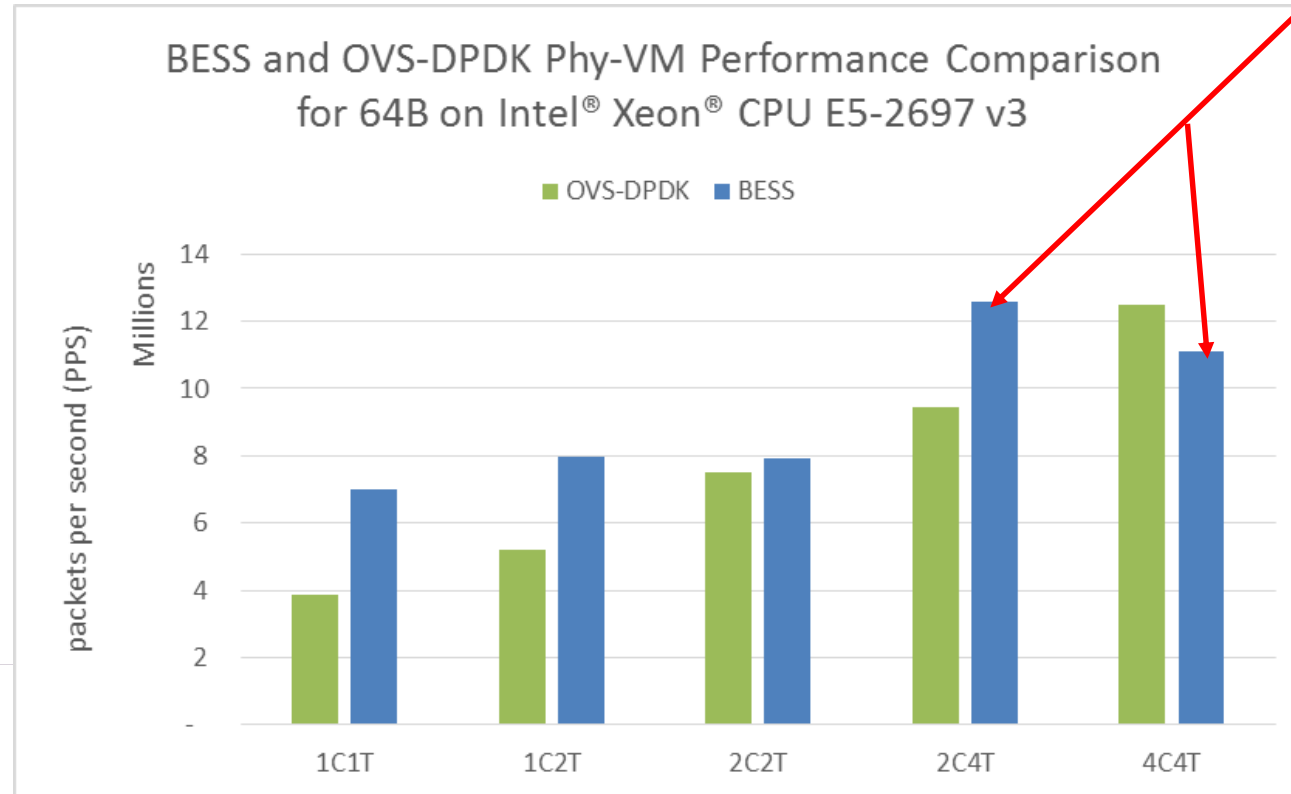
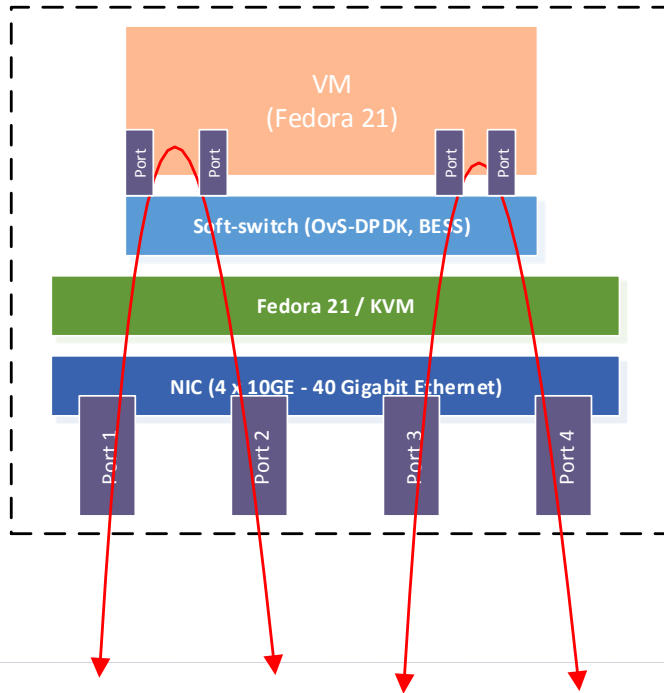
Phy-to-Phy Performance (2/2)



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

Phy-NF-Phy Performance



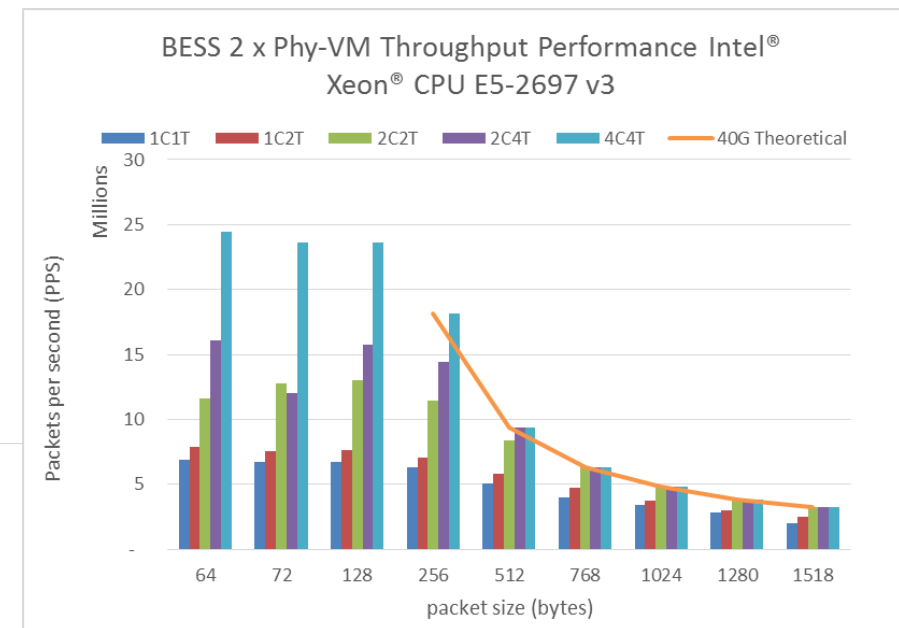
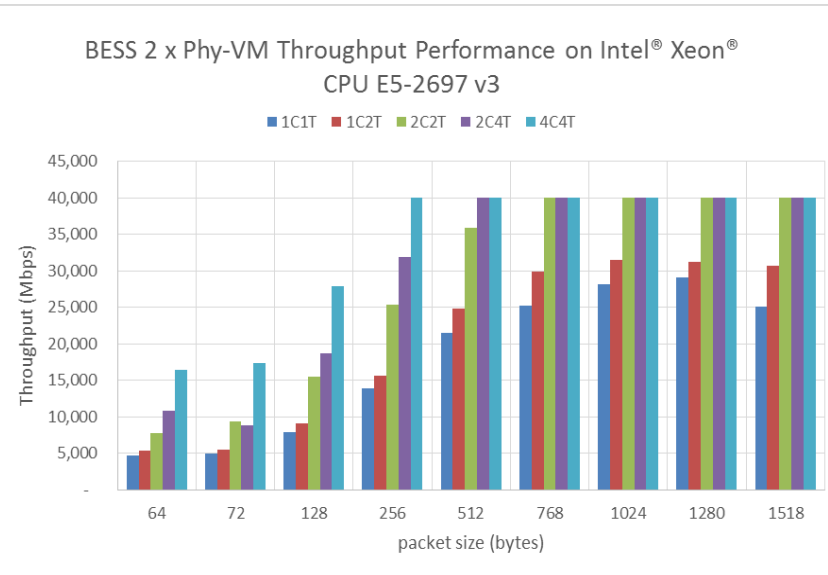
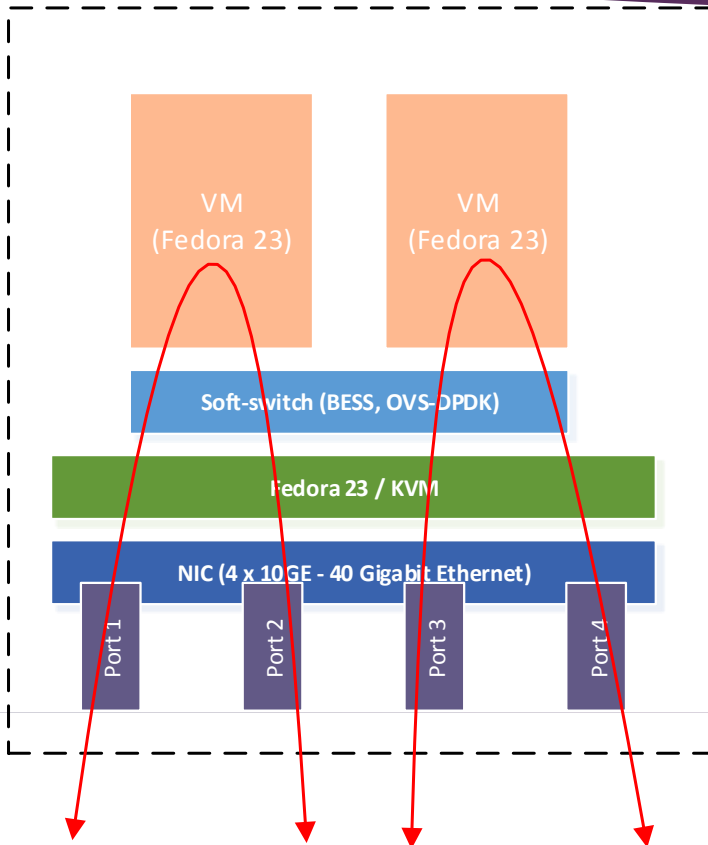
VM
bottleneck

Source: Irene Liew, Anbu Murugesan – Intel Network Packet Group

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804

2 x Phy-NF-Phy Performance



Source: Irene Liew, Anbarasan Murugesan – Intel NPG Architecture

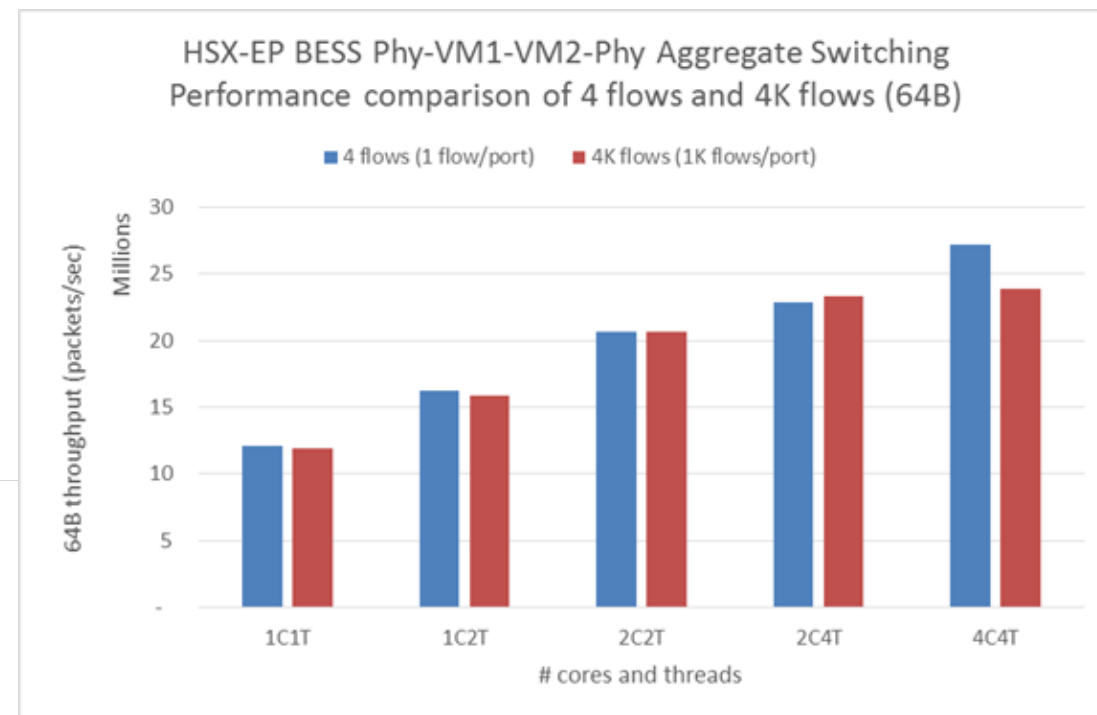
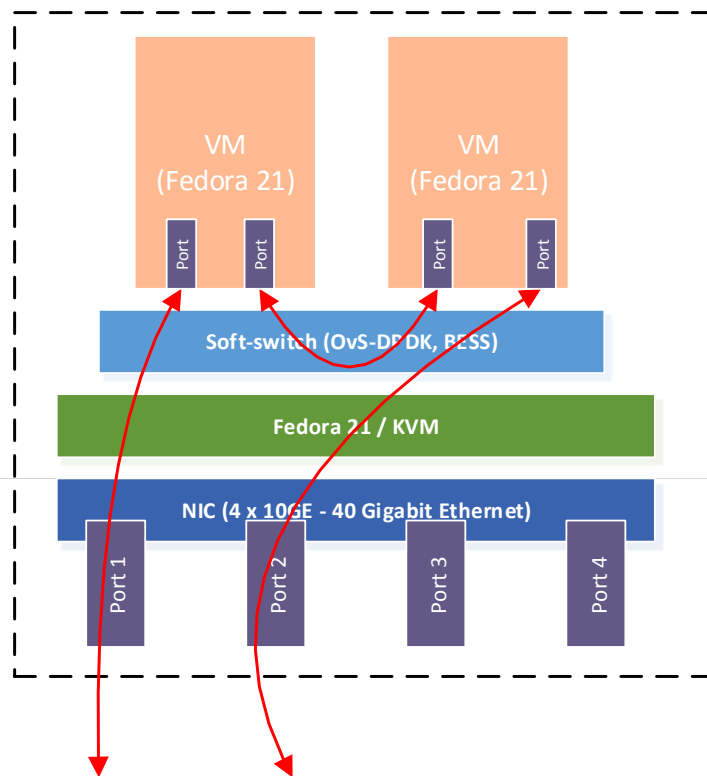
Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804

Phy-NF-NF-Phy Performance

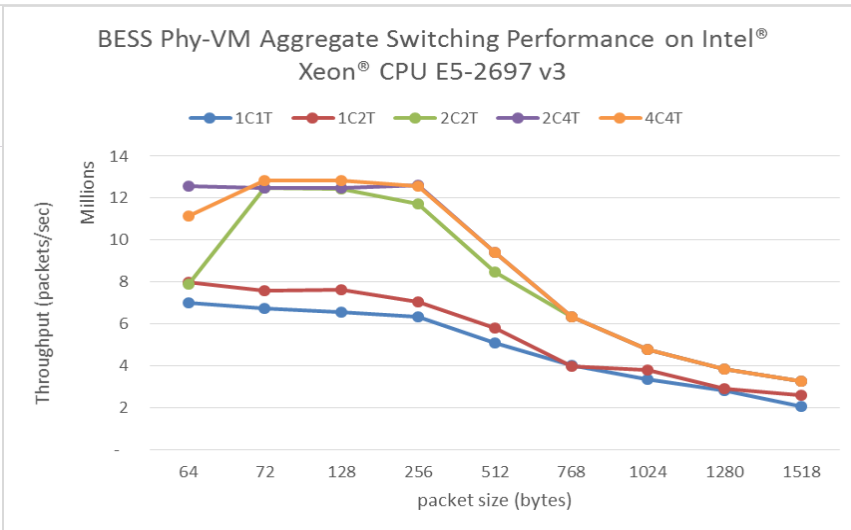
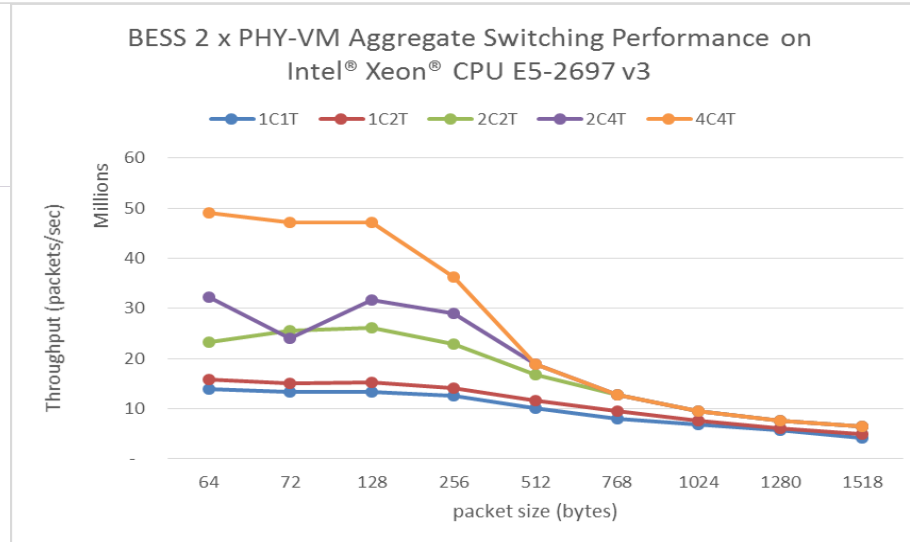
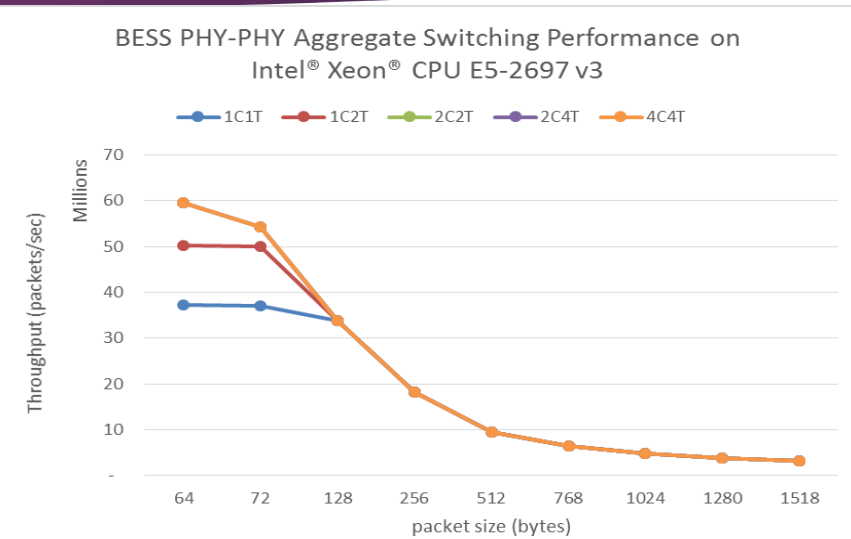
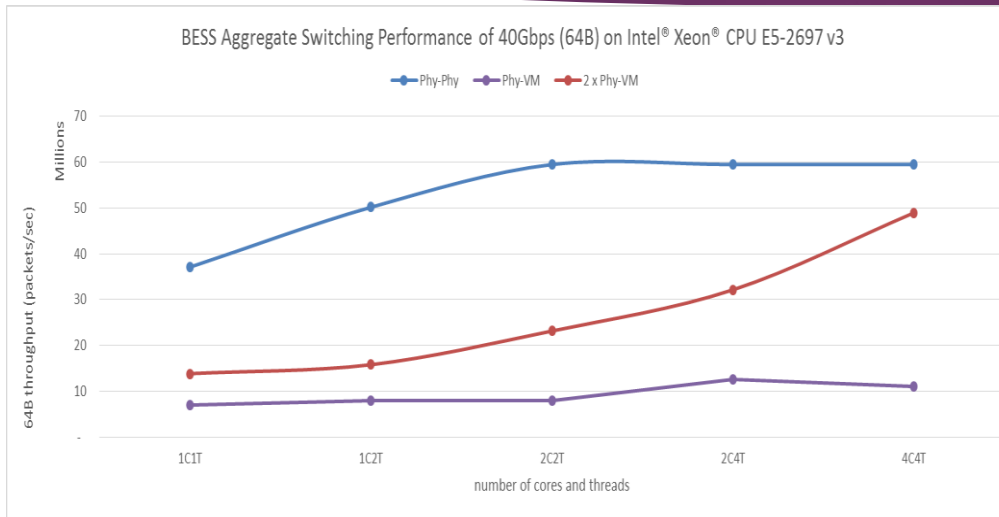


- ▶ BESS outperforms OVS-DPDK by a factor of 4-5x*

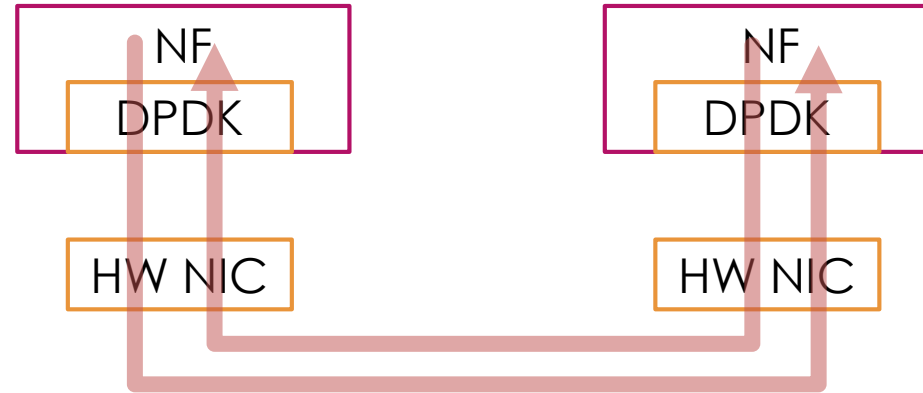


* Source: Intel ONP 2.1 Performance Test Report

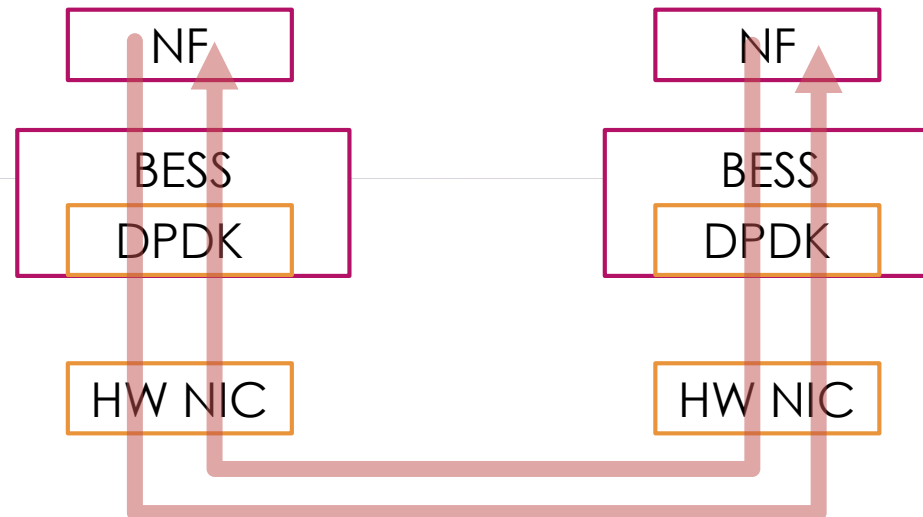
Multi-Core / Thread Scalability



Round-Trip Latency



RTT: 8.22us



RTT: 8.82us

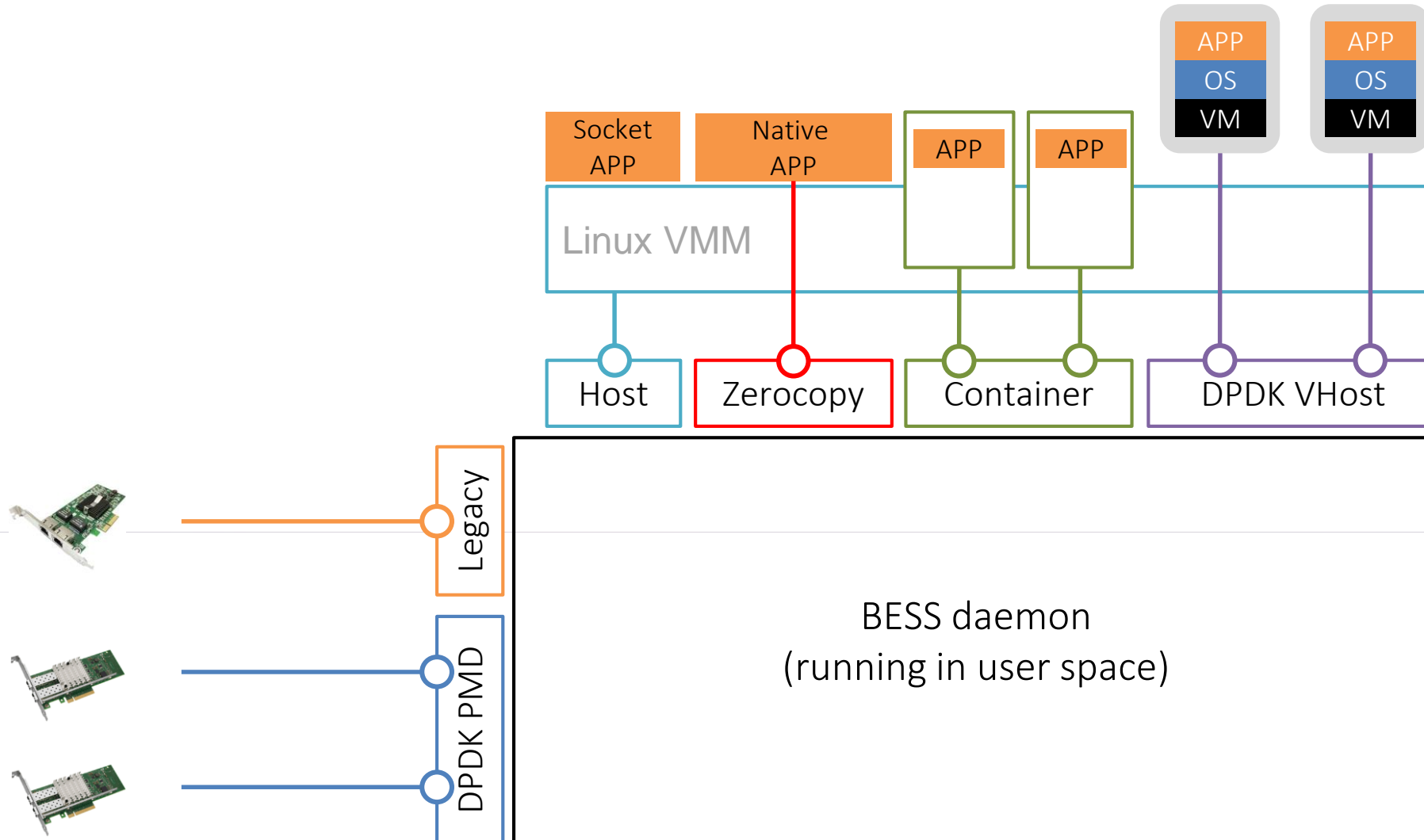
- Increase of 0.60us
(0.15us per
BESS traverse)

BESS Architecture Overview (1/3)

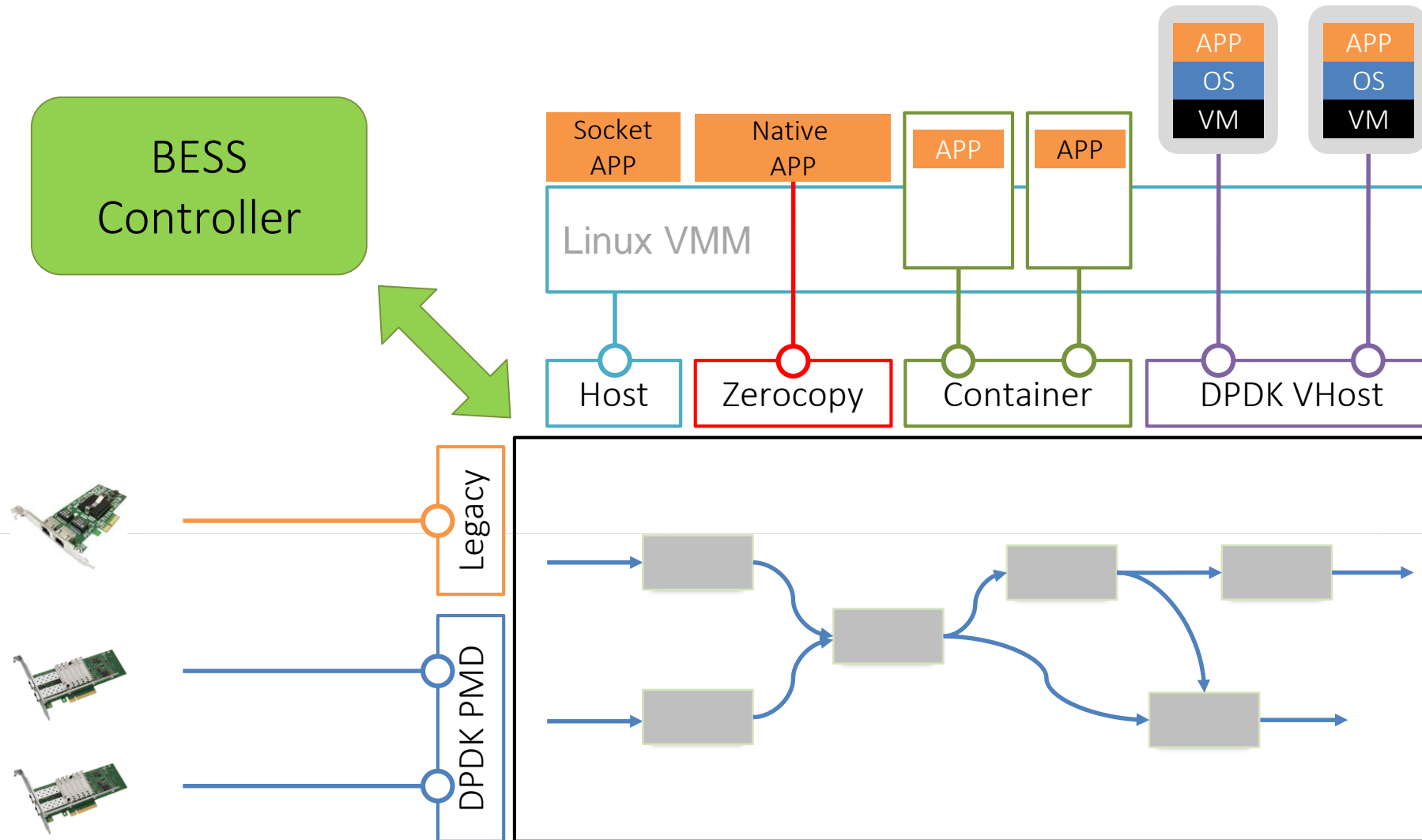


BESS daemon
(running in user space)

BESS Architecture Overview (2/3)



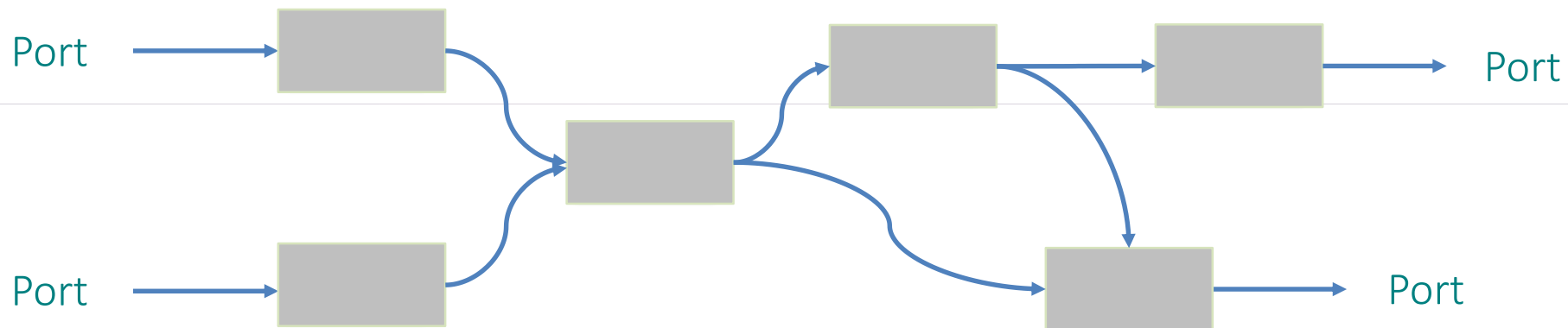
BESS Architecture Overview (3/3)



Modular Datapath Pipeline



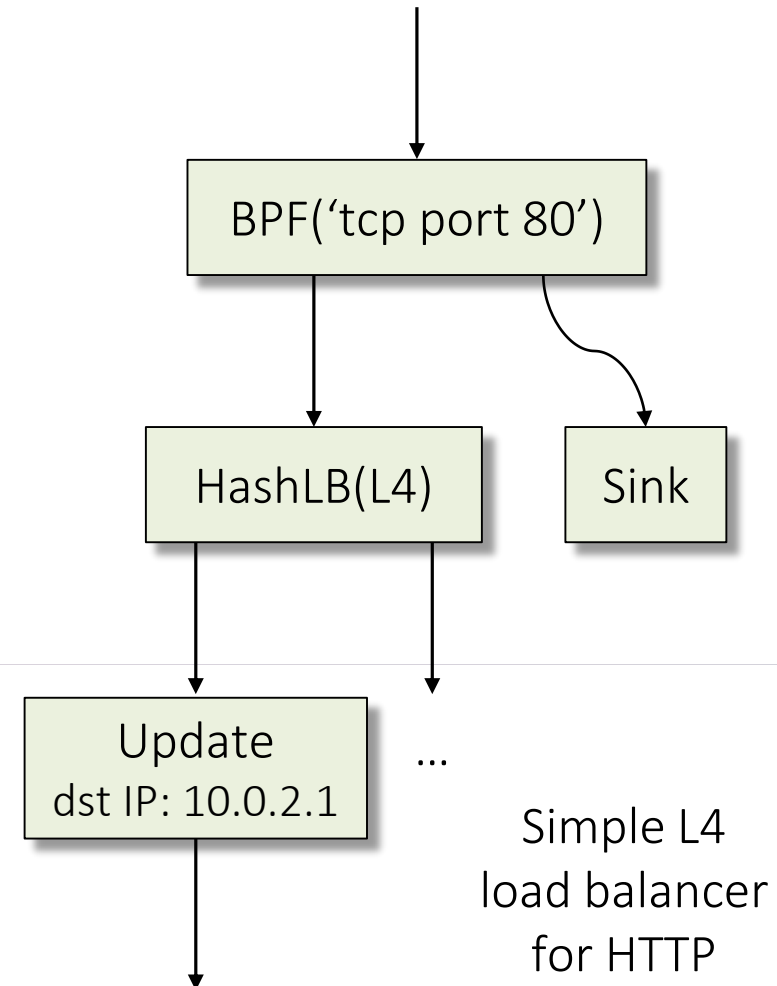
- External ports are interconnected with “modules” in a dataflow graph (like the Click modular router).
 - You can compose modules to implement your own datapath.
 - Developing a new module is easy.



Building Modules



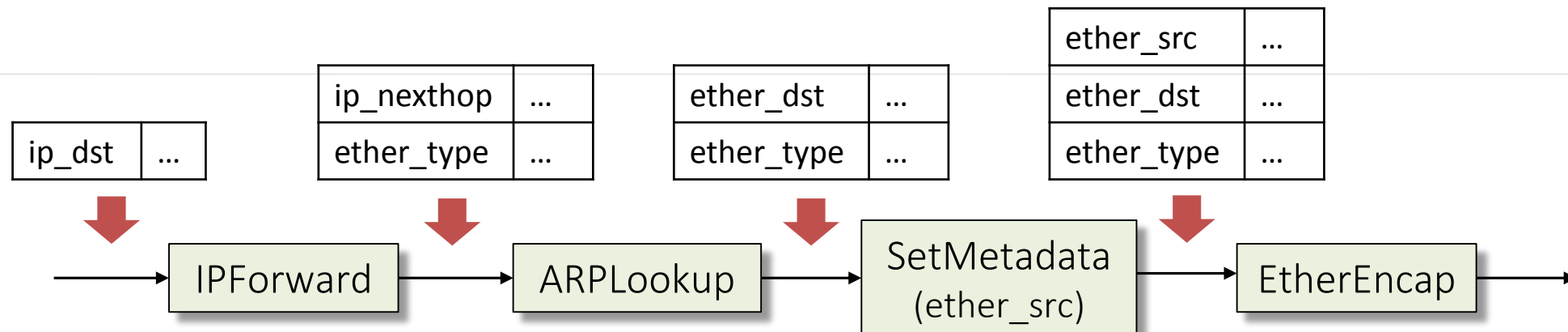
- BPF filter
- Exact match table
- Wildcard match table
- Load balancer
- Encapsulation / decapsulation
- L2 forward
- IP Lookup
- 802.1q / 802.1ad
- Metadata
- ...



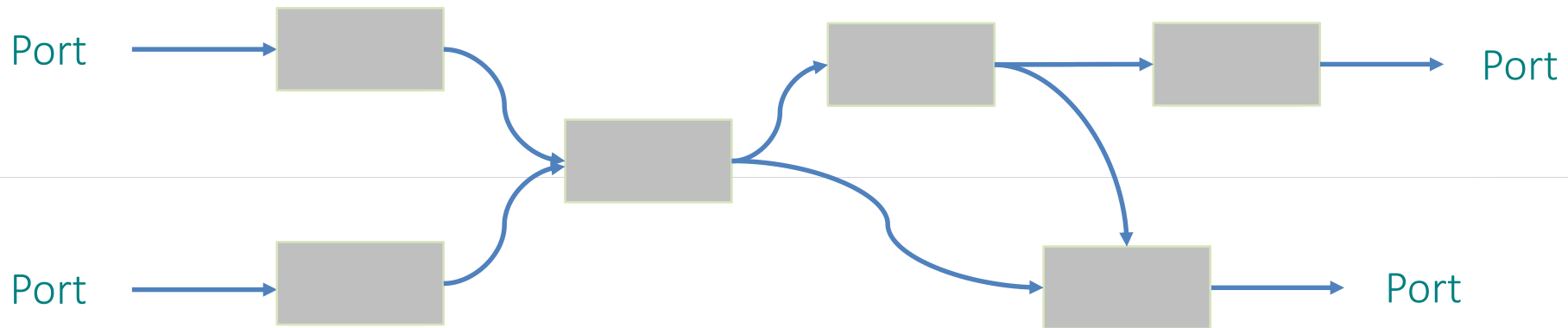
Dynamic Packet Metadata



- Modules can tag metadata attributes to packets
 - Dynamic key-value table for each packet
 - 0-instruction access overhead, as compared to static “C struct”
 - Optimal usage of memory (cf. metadata bloat in Linux sk_buff)
 - Enables more decoupled and coherent design of modules



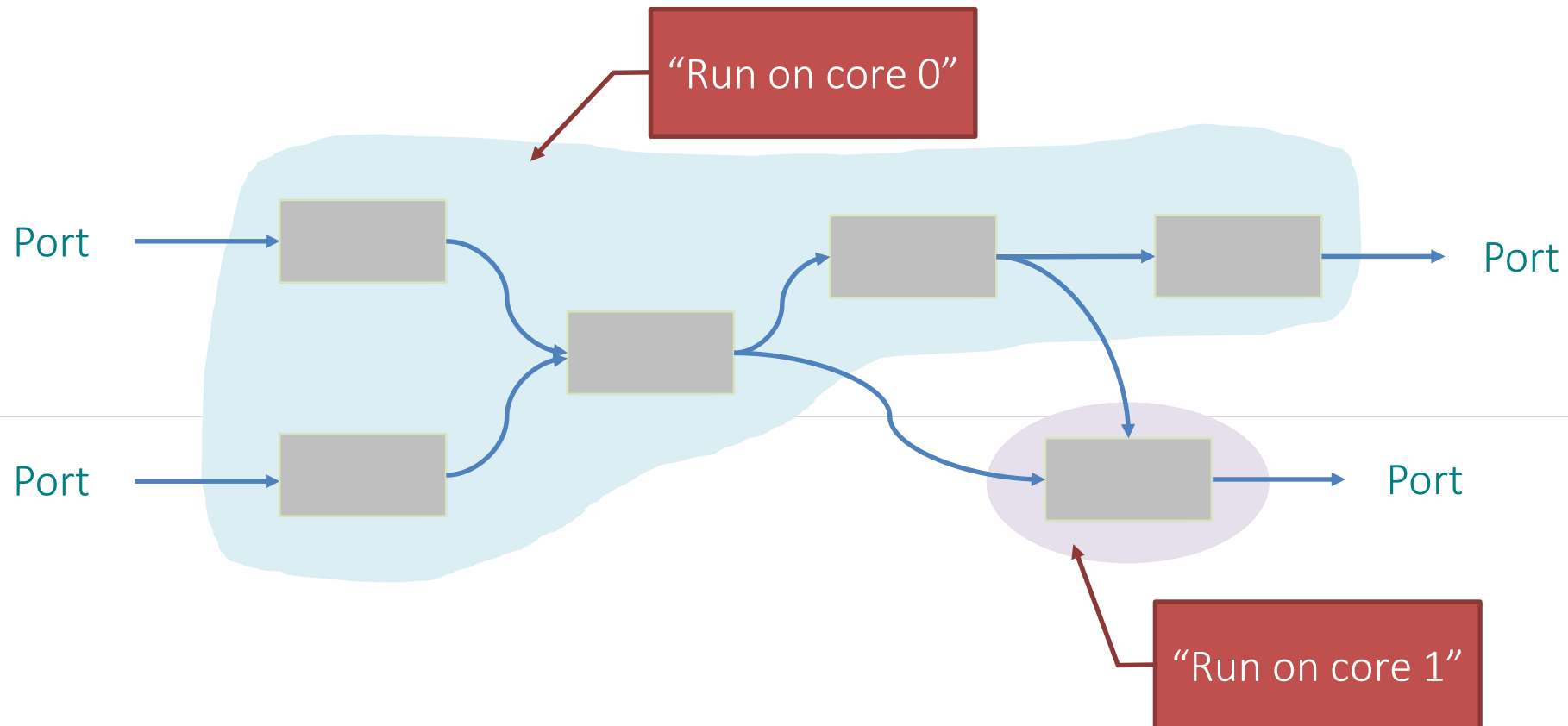
- BESS allows flexible scheduling policies for the data path
 - In terms of CPU utilization and bandwidth. Examples:



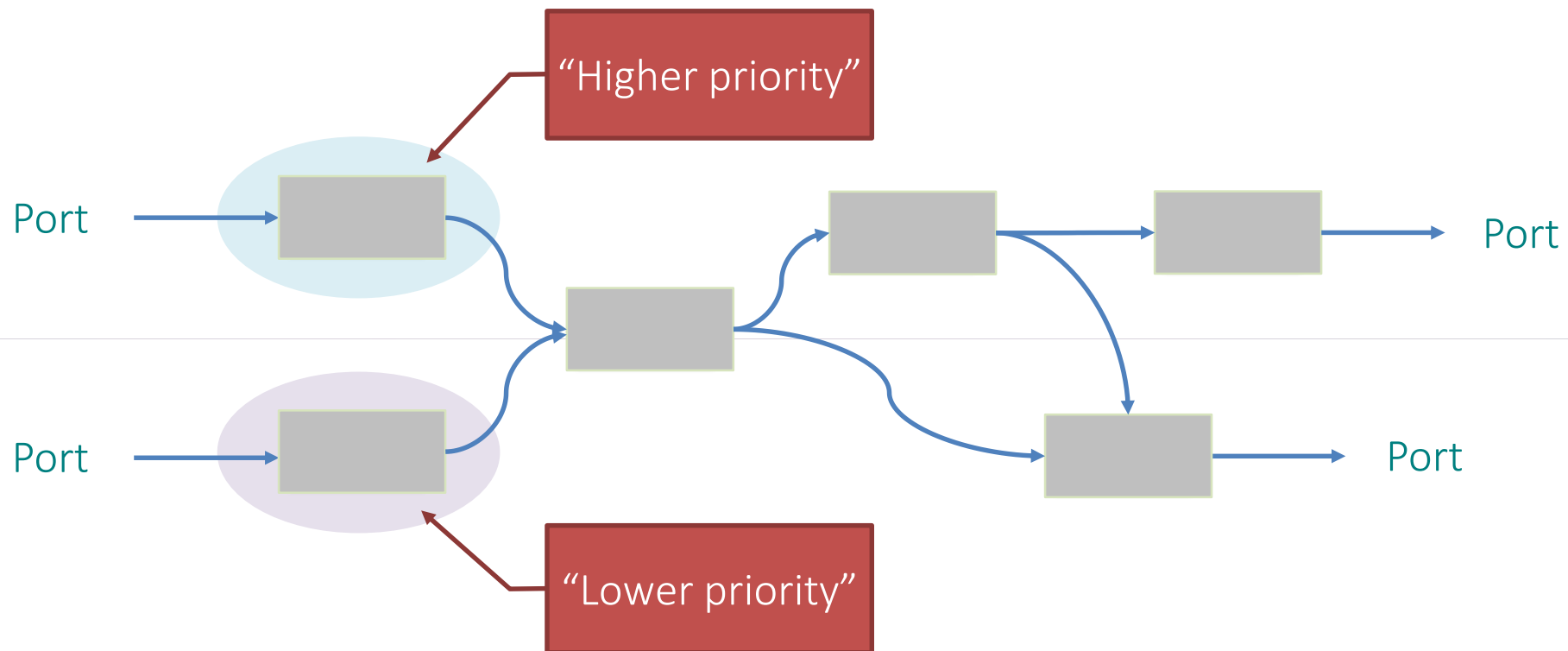
Resource-Aware CPU Scheduler (2/5)



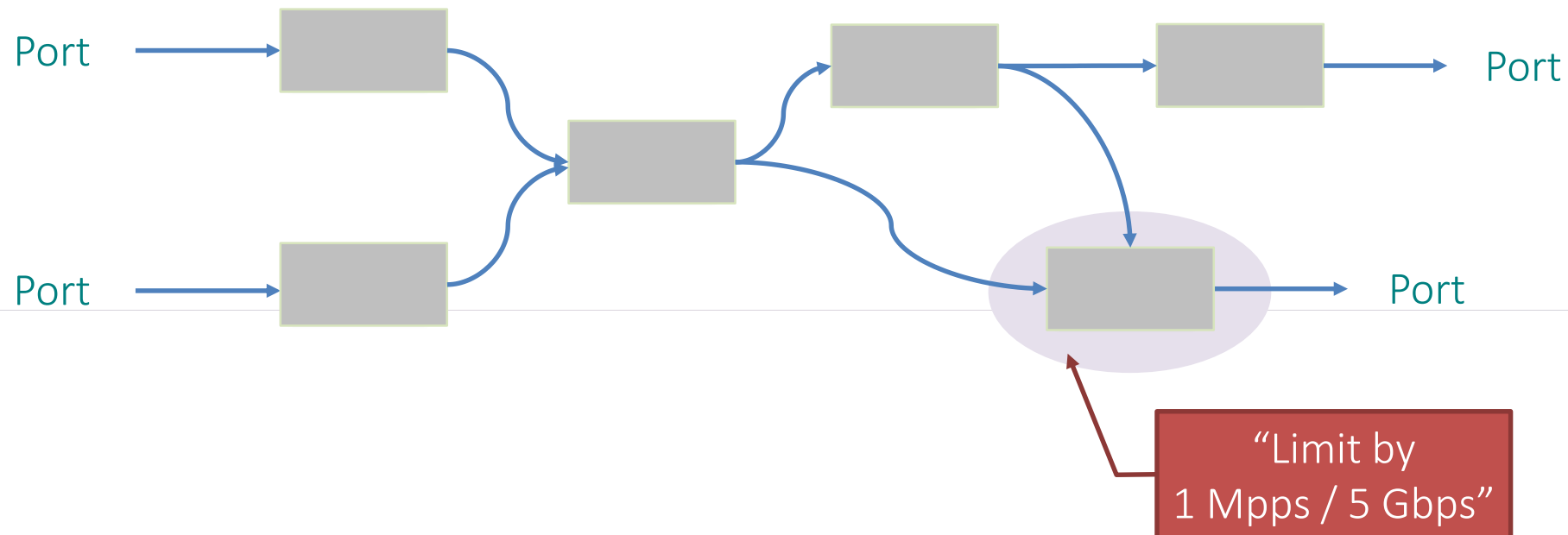
- ▶ BESS allows flexible scheduling policies for the data path.
- ▶ In terms of CPU utilization and bandwidth. Examples:



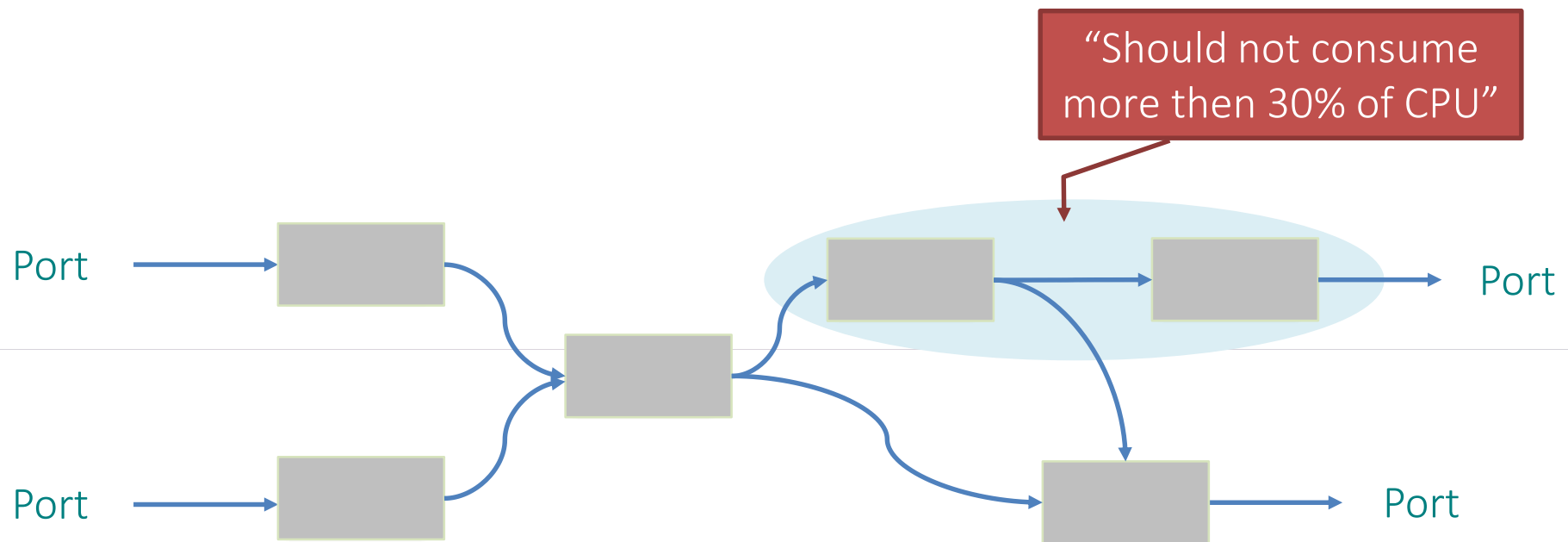
- BESS allows flexible scheduling policies for the data path.
 - In terms of CPU utilization and bandwidth. Examples:



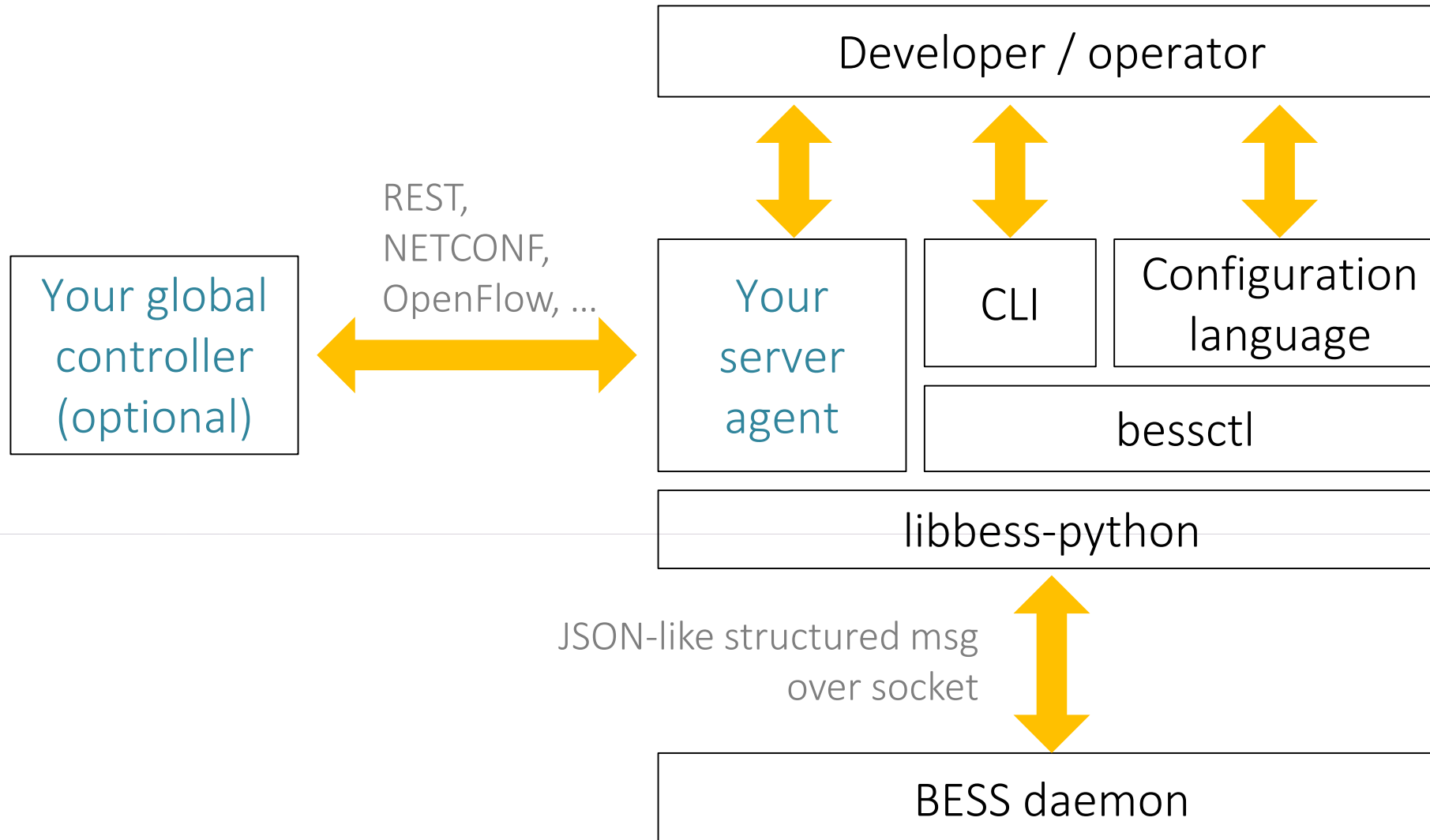
- BESS allows flexible scheduling policies for the data path.
 - In terms of CPU utilization and bandwidth. Examples:



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



BESS Control Interface (1/5)



BESS Control Interface (2/5)



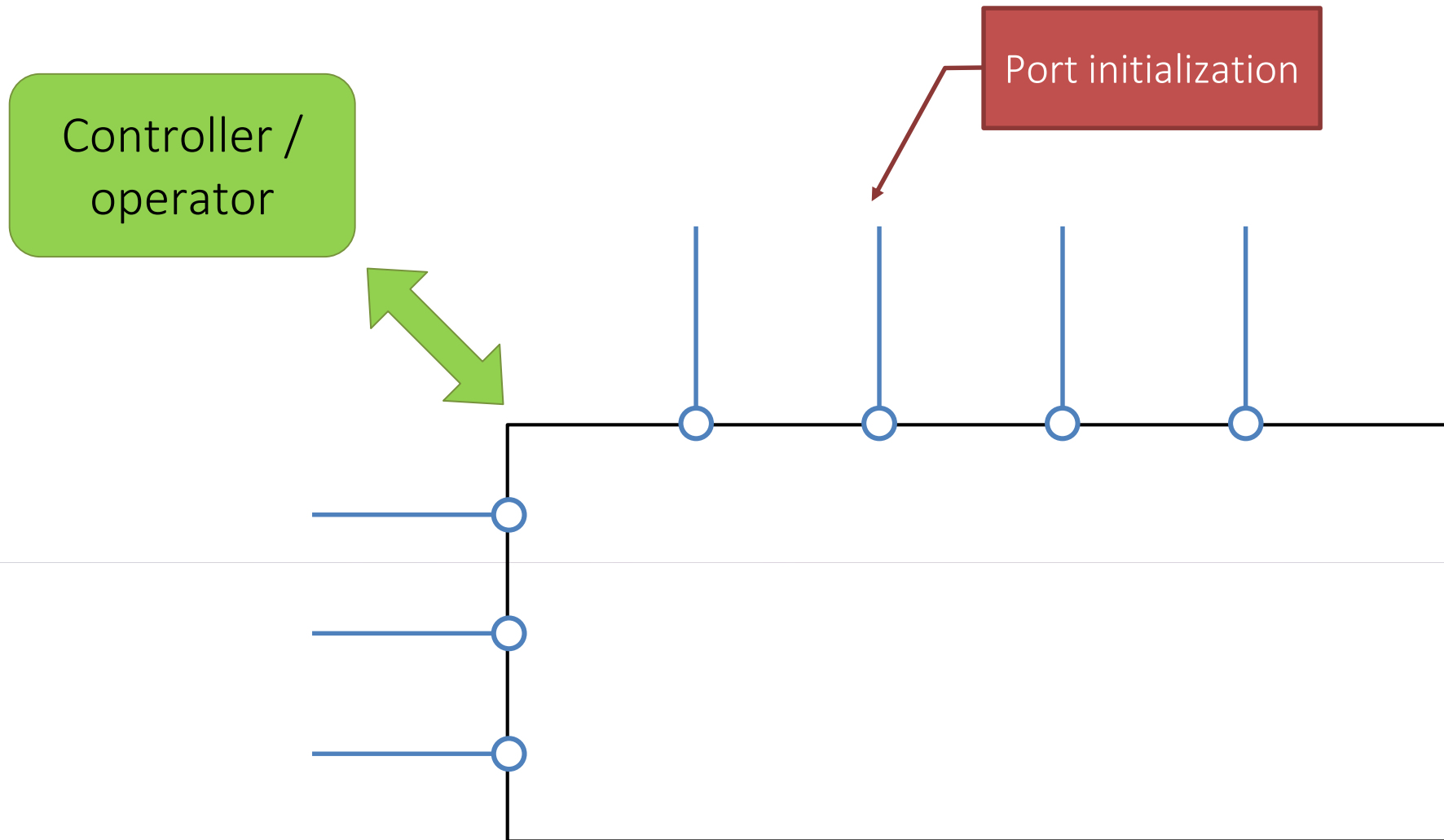
```
[2007] sangjin@c6:~/bess/bessctl/conf/samples [develop] $ cat update.bess
import scapy.all as scapy

eth = scapy.Ether(src='02:1e:67:9f:4d:ae', dst='06:16:3e:1b:72:32')
ip = scapy.IP(src='192.168.1.1', dst='10.0.0.1')
udp = scapy.UDP()
payload = scapy.Raw()
pkt_bytes = len(payload)

Source(
    show module MODULE...          Show the status of specified modules
    show mclass                    Show all module classes
    show mclass MCLASS...          Show the details of specified module classes
    monitor pipeline                Monitor packet counters in the datapath pipeline
    monitor pipeline batch          Monitor batch counters in the datapath pipeline
    monitor port                    Monitor the current traffic of all ports
    monitor port PORT...            Monitor the current traffic of specified ports
    monitor tc                      Monitor the statistics of all traffic classes
    monitor tc TC...                Monitor the statistics of specified traffic classes
    tcpdump MODULE [OGATE] [TCPDUMP_OPTS...] Capture packets on a gate

# src M interactive localhost:10514 $ tcpdump rupdate0 0 -nex | less
rr:1 \ localhost:10514 $ shc Running: tcpdump -r /tmp/tmpSQpu9b -nex | less
      Worker ID   Statu reading from file /tmp/tmpSQpu9b, link-type EN10MB (Ethernet)
->      0         RUNNIN 14:33:35.788275 02:1e:67:9f:4d:ae > 06:16:3e:1b:72:32, ethertype IPv4 (0x0800), length 52: 192.168.1.1.27082 > 10.0.0.1.42002: UDP, length 10
-> localhost:10514 $ shc
      +-----+
      | source0 |
      | Source  | :0 54592
      +-----+
      | 0x0000: 4500 0026 0001 0000 4011 af1c c0a8 0101
      | 0x0010: 0a00 0001 69ca a412 0012 c5dc 6865 6c6c
      | 0x0020: 6f77 6f72 6c64
# UDP s | - | ----- 14:33:35.788275 02:1e:67:9f:4d:ae > 06:16:3e:1b:72:32, ethertype IPv4 (0x0800), length 52: 192.168.1.1.23810 > 10.0.0.1.47340: UDP, length 10
rr:2 \ +-----+
      | 0x0000: 4500 0026 0001 0000 4011 af1c c0a8 0101
      | 0x0010: 0a00 0001 5d02 b8ec 0012 c5dc 6865 6c6c
      | 0x0020: 6f77 6f72 6c64
->
      | 14:33:35.788275 02:1e:67:9f:4d:ae > 06:16:3e:1b:72:32, ethertype IPv4 (0x0800), length 52: 192.168.1.1.28106 > 10.0.0.1.46999: UDP, length 10
      | 0x0000: 4500 0026 0001 0000 4011 af1c c0a8 0101
      | 0x0010: 0a00 0001 5d02 b8ec 0012 c5dc 6865 6c6c
      | 0x0020: 6f77 6f72 6c64
->
      | 14:33:35.788275 02:1e:67:9f:4d:ae > 06:16:3e:1b:72:32, ethertype IPv4 (0x0800), length 52: 192.168.1.1.22699 > 10.0.0.1.45448: UDP, length 10
      | 0x0000: 4500 0026 0001 0000 4011 af1c c0a8 0101
      | 0x0010: 0a00 0001 58ab b188 0012 c5dc 6865 6c6c
      | 0x0020: 6f77 6f72 6c64
localhost:10514 $
      | 14:33:35.788275 02:1e:67:9f:4d:ae > 06:16:3e:1b:72:32, ethertype IPv4 (0x0800), length 52: 192.168.1.1.24023 > 10.0.0.1.47417: UDP, length 10
      | 0x0000: 4500 0026 0001 0000 4011 af1c c0a8 0101
      | 0x0010: 0a00 0001 5dd7 b939 0012 c5dc 6865 6c6c
      | 0x0020: 6f77 6f72 6c64
```

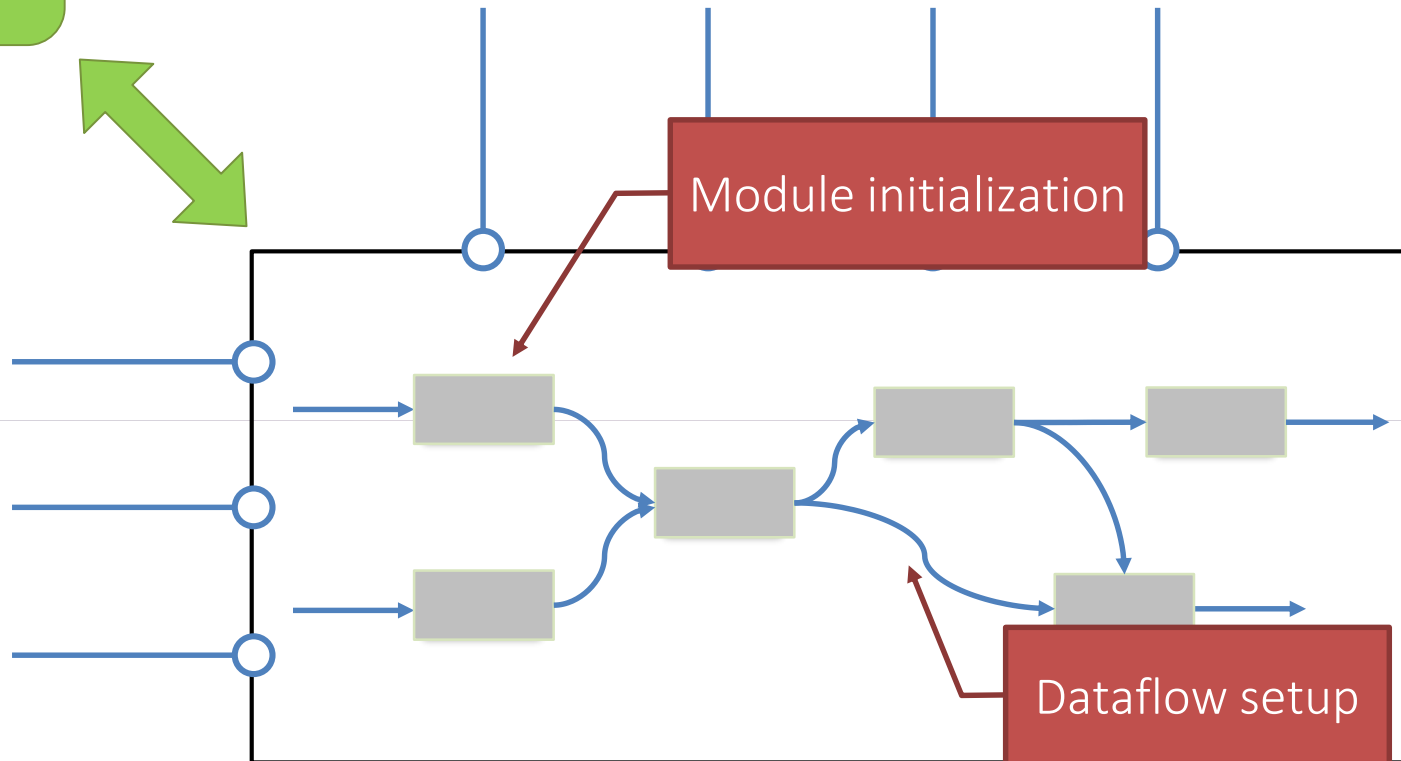
BESS Control Interface (3/5)



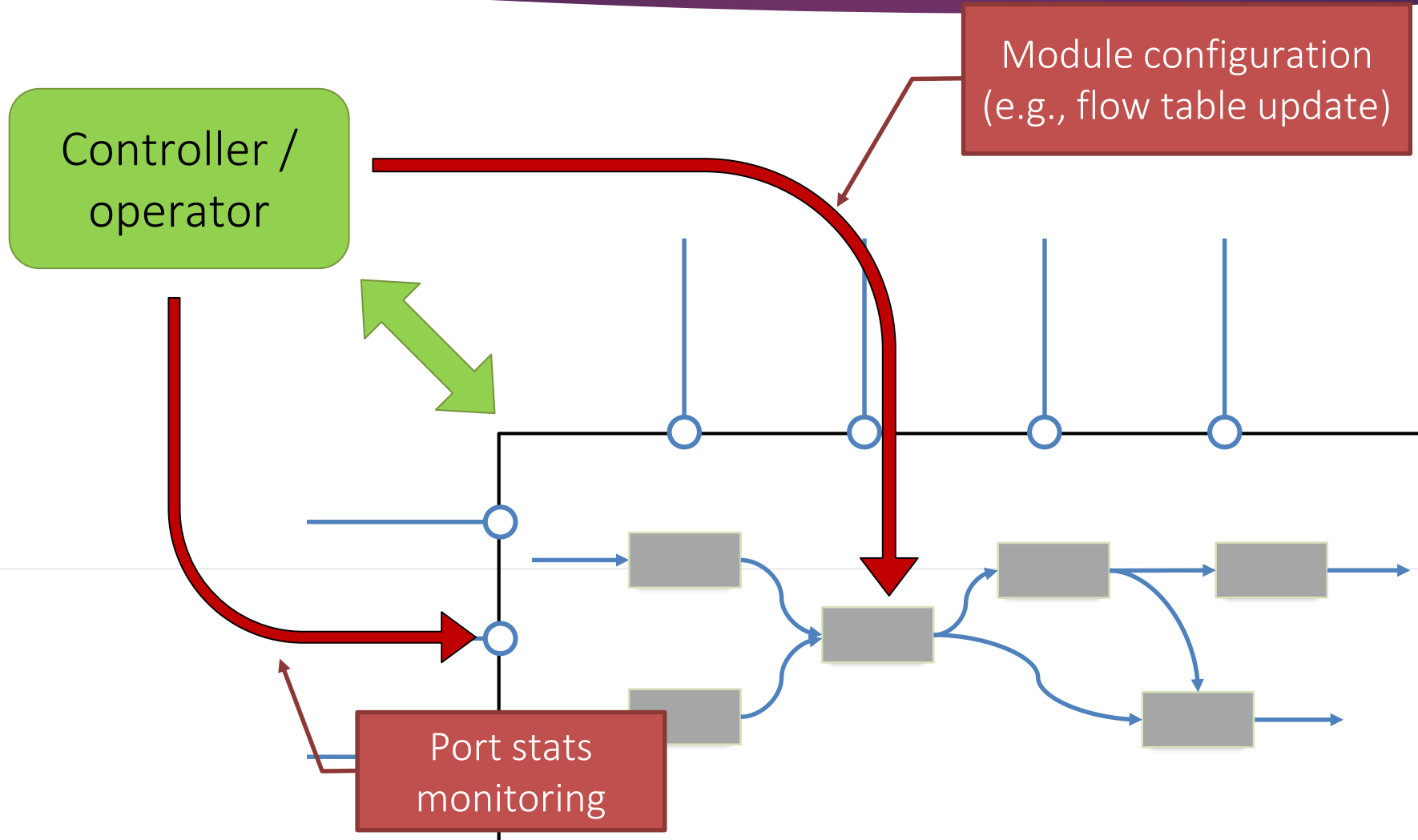
BESS Control Interface (4/5)



Controller /
operator



BESS Control Interface (5/5)



- ▶ 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
 - ▶ ~34k lines in C and Python, supporting
 - ▶ Linux 3.x / 4.x (x86_64), DPDK 16.04
 - ▶ QEMU/KVM virtual machines, Docker/LXC containers

Legal Disclaimers



No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. **No computer system can be absolutely secure.** Check with your system manufacturer or retailer or learn more at intel.com.

© 2016 Intel Corporation. Intel, the Intel logo, Intel. Experience What's Inside, and the Intel. Experience What's Inside logo are trademarks of Intel. Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Questions?

Sangjin Han

sangjin@eecs.berkeley.edu

Christian Maciocco

christian.maciocco@intel.com