TOWARDS FAST IP FORWARDING

IP FORWARDING PERFORMANCE IMPROVEMENT AND MEASUREMENT IN FREEBSD

Nanako Momiyama

Keio University

25th September 2016

EuroBSDcon 2016

OUTLINE

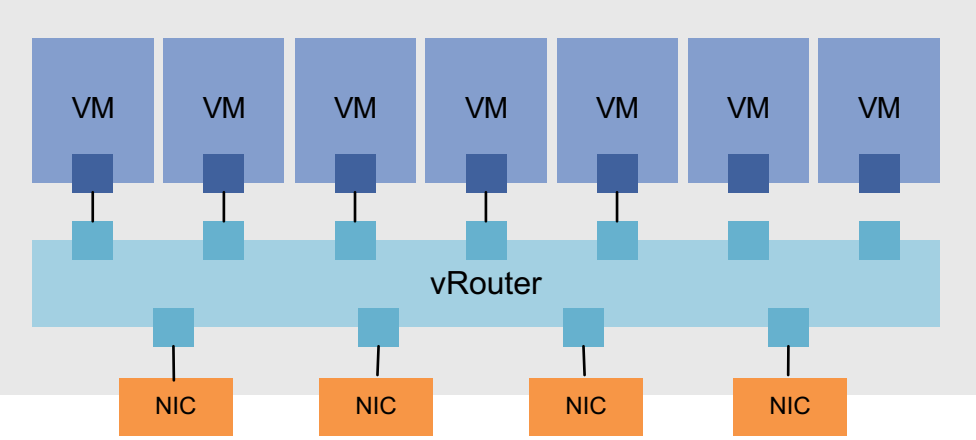
- Motivation
- Design and implementation
  - Applying fast packet I/O and fast IP lookup into FreeBSD network stack
- Measurement results
- Problem analysis
- Approach (ongoing work)
- Conclusion

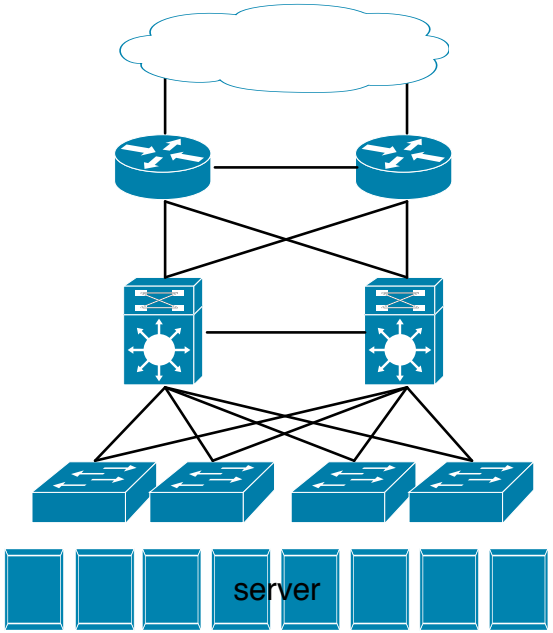
MOTIVATION

- Software packet forwarding has played an important role in general-purpose OSes
  - L2 bridging, IP Routing, Firewall etc
- Increasing network capacities (10GbE, 40GbE...) pushed people out of the kernel
  - user-space packet forwarding on top of netmap[1], DPDK[2]
- Stresses using them in production are beginning to arise
  - APIs/CLIs compatibility, port scalability (NICs, VMs), features and isolation
- It's time for bridging a performance gap between kernel-based packet forwarding (1-2 Mpps) and user-space one (> 10 Mpps)

STARTING POINT

- L3 IP forwarding
  - Support the Internet
  - Useful for datacenter and VM back-end
    - L2 network doesn't scale





WHERE IS THE PERFORMANCE BOTTLENECK?

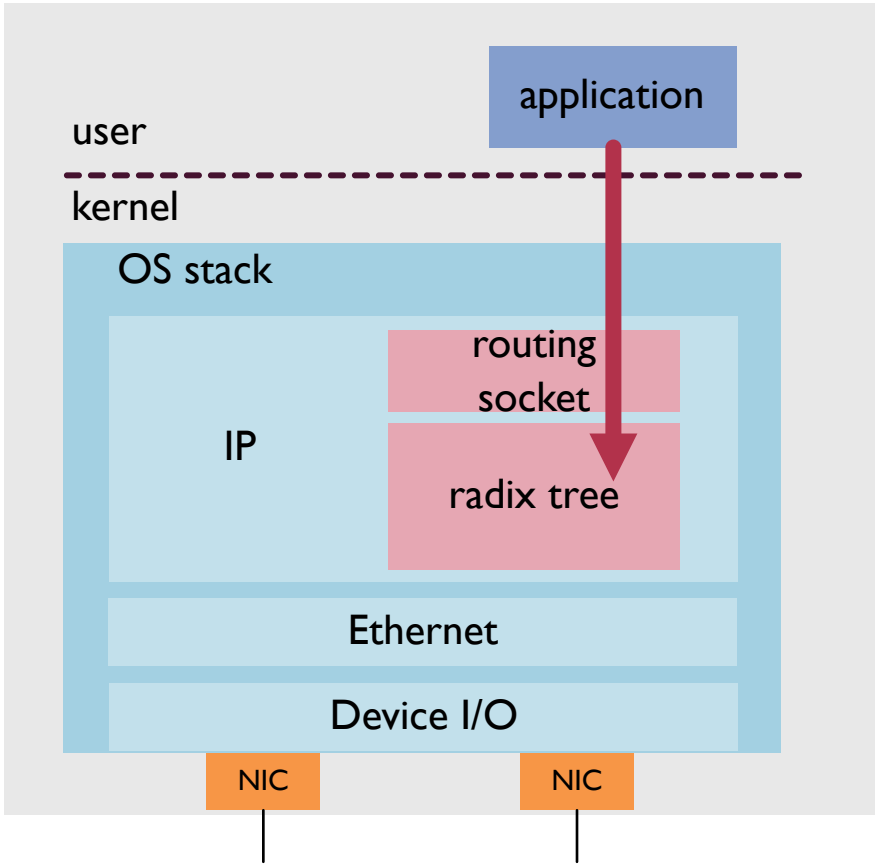
- Default FreeBSD can forward packets only at 1.4 Mpps (10GbE line rate is 14.88 Mpps)
- Packet I/O?
  - Was a main bottleneck for packet forwarding
  - Now several solutions to achieve the 10GbE line rate
    - netmap, DPDK
- IP routing table lookup?
  - Hardware appliance has TCAM for fast lookup
  - Now several fast routing lookup algorithms for software
    - SAIL[3], DXR[4], Poptrie[5]

What if we bring these techniques into FreeBSD?

DESIGN AND IMPLEMENTATION

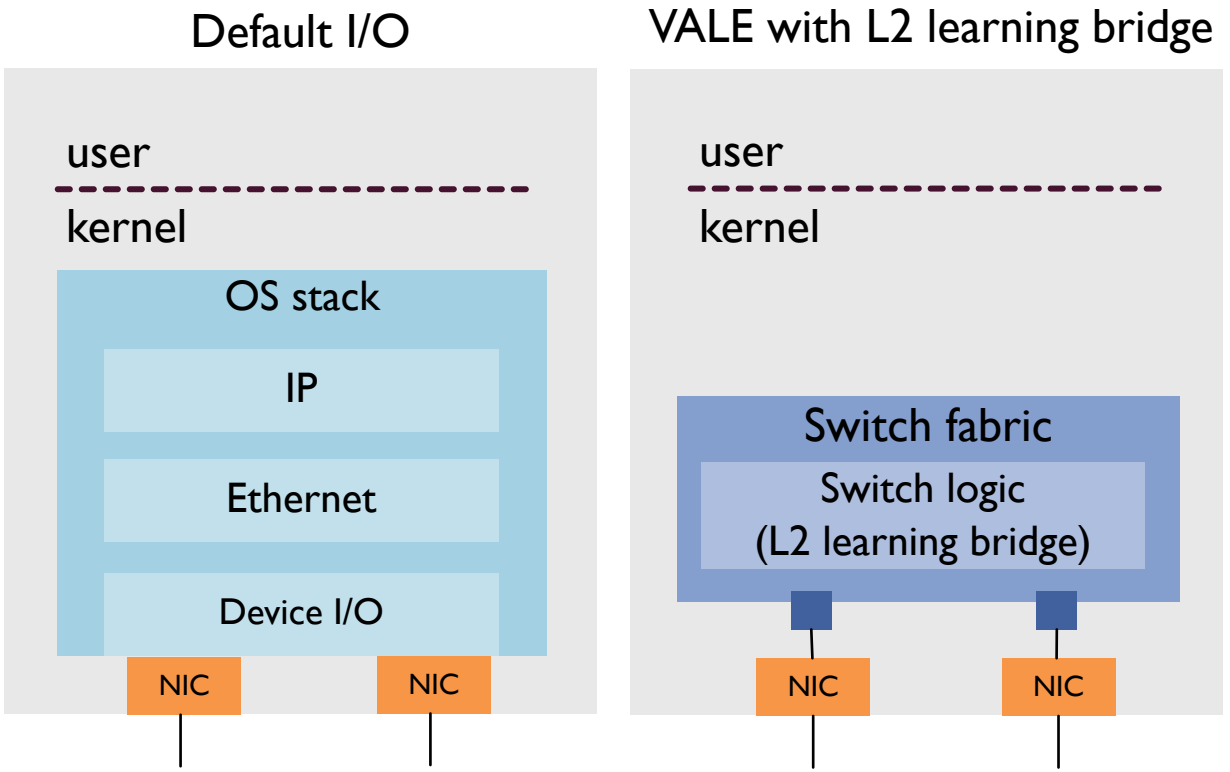
- Design overview
  - FreeBSD for Control Plane
    - The OS network stack to preserve existing APIs
  - VALE[6] + DXR for Forwarding Plane
    - VALE for fast, scalable packet I/O
    - DXR for fast IP route lookup

FreeBSD default network stack



VALE OVERVIEW

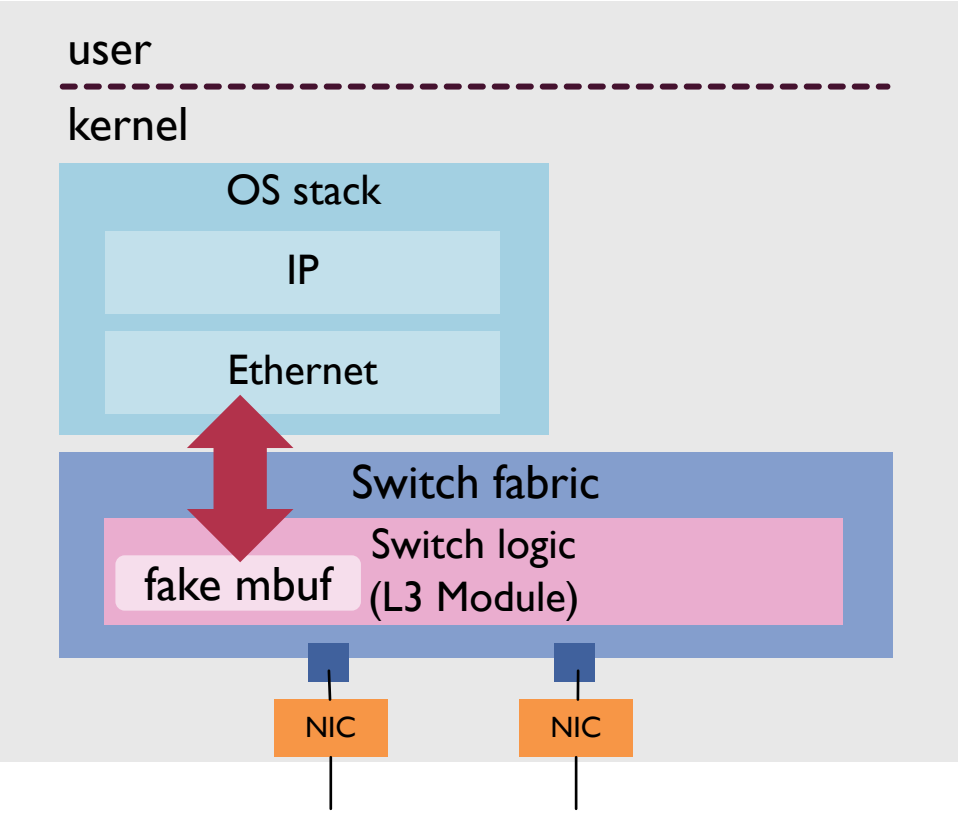
- VALE is a software switch
  - Run in the kernel
  - Part of the netmap framework
- Netmap is a fast packet I/O framework which enables applications to send and receive packets at 10 GbE line rate
- VALE works as a L2 learning switch by default
- Packets do NOT go through the OS network stack
  - just forwarding packets from one port to another port
- L2 switch logic can be replaced with a different module



NEW SWITCH LOGIC IMPLEMENTATION

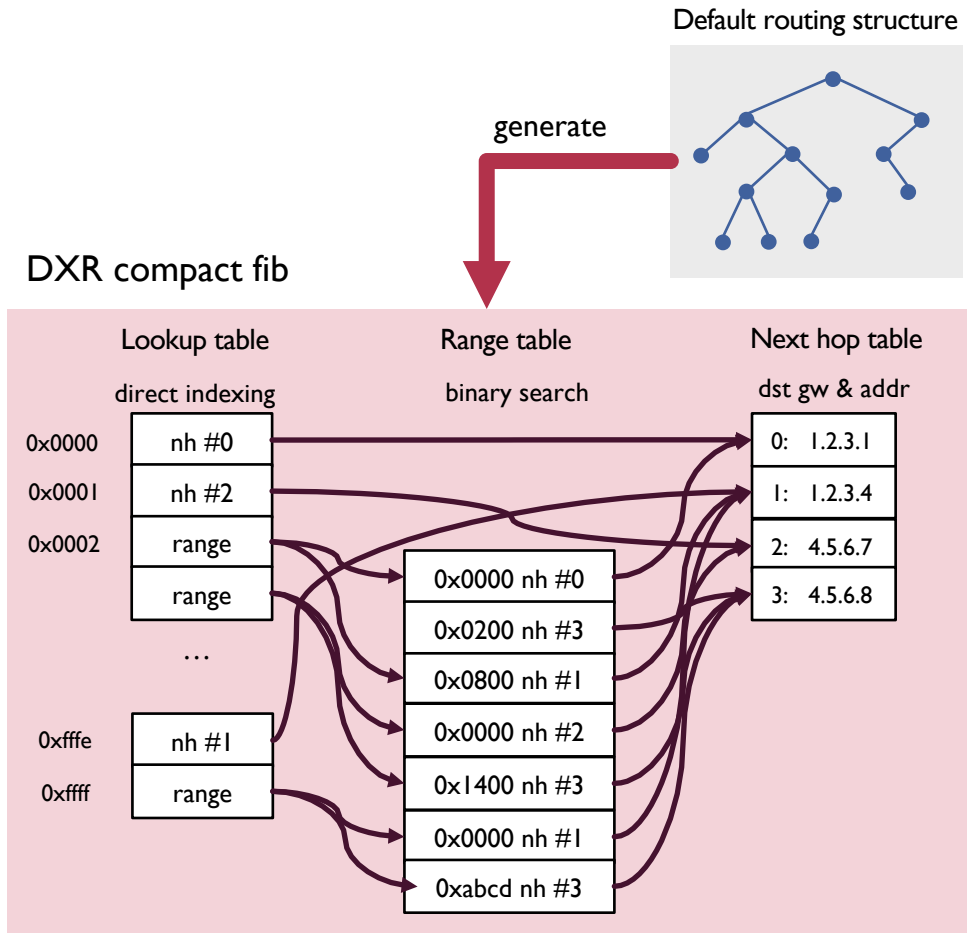
- Create a new function as a new switch logic (L3 module) in VALE
- Use VALE for packet I/O and the OS network stack for L2/L3 processing
- Make a fake mbuf in VALE and pass it to the OS network stack
- The OS stack embeds a route lookup result in an unused mbuf field
- Before if_transmit(), force return to have VALE transmit packets

VALE with L3 module



DXR OVERVIEW

- DXR is a fast IPv4 route lookup algorithm
- Create compact data structures based on a large routing table (radix tree)
  - Fit into CPU caches
- See the DXR paper for more details

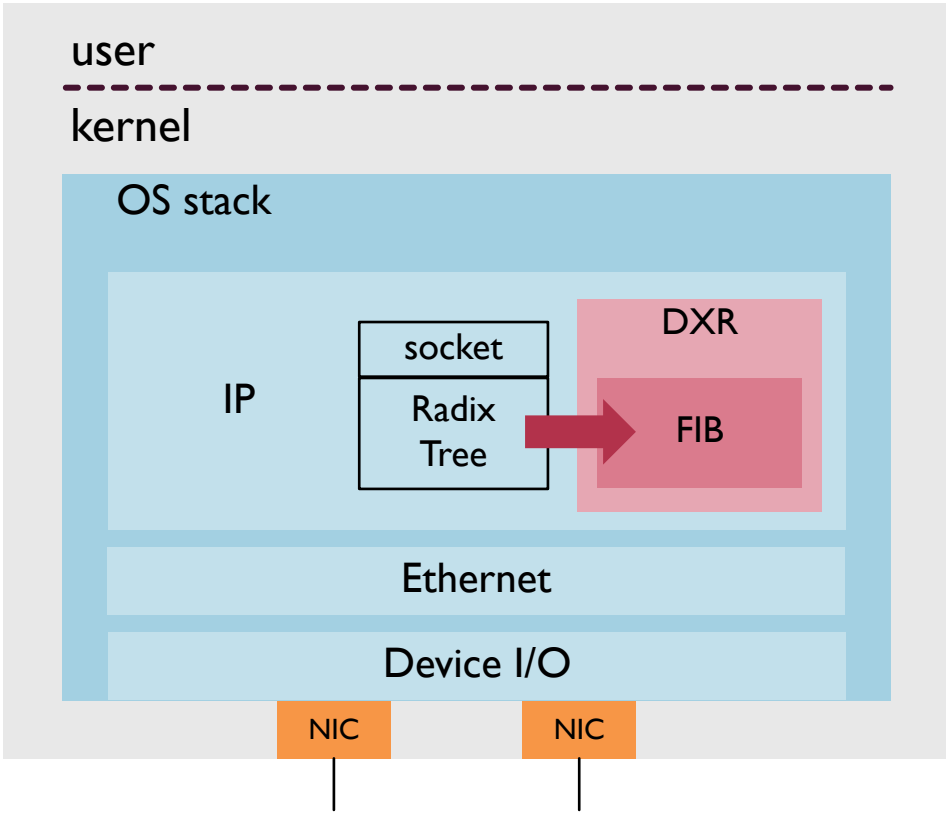


Ref. Modified from Figure 1 of Zec, Marko, Luigi Rizzo, and Miljenko Mikuc. "DXR: towards a billion routing lookups per second in software." *ACM SIGCOMM Computer Communication Review* 42.5 (2012): 29-36.

DXR IMPLEMENTATION

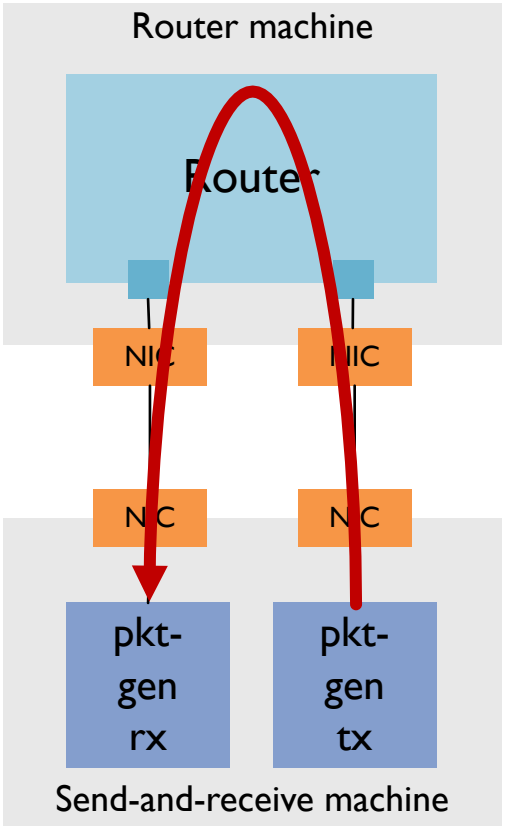
- Porting DXR patch for FreeBSD 8.0 to FreeBSD 12.0-CURRENT
- DXR builds and uses new compact data structures based on the OS radix tree
- DXR-specific lookup function is called instead of ip_findroute()

DXR integration



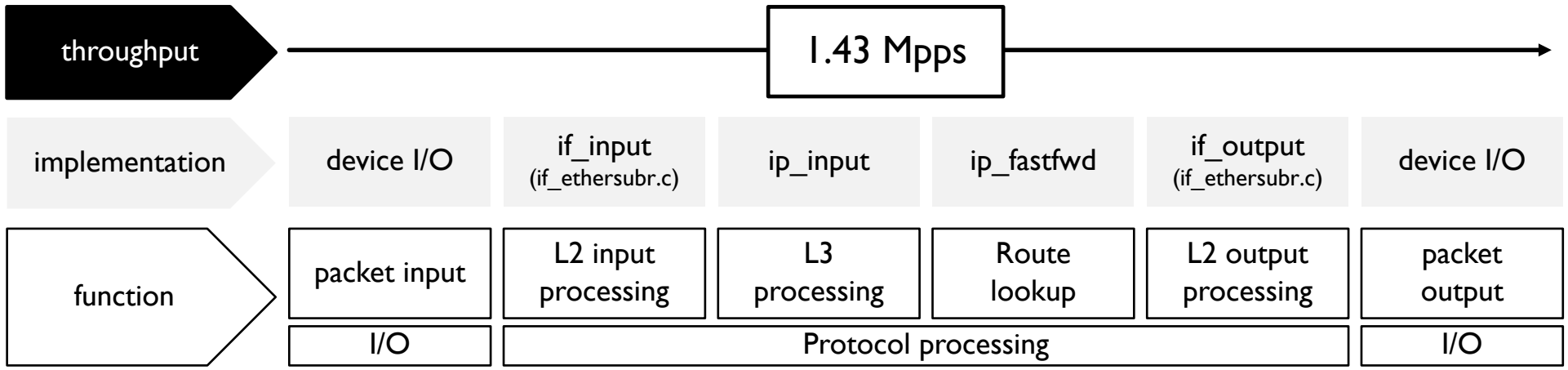
EXPERIMENTAL SETUP

- Machine spec
  - OS: FreeBSD (12.0-CURRENT, 04/08/16 snapshot)
  - CPU: Intel(R) Core(TM) i7-3930K CPU @ 3.20GHz 6 core
  - NIC: Intel X520 10GbE dual-port
- Method
  - Two machines connected back-to-back
  - Generate 10GbE line-rate traffic using pkt-gen application
  - Measure packet rates forwarded by router machine
- Setting
  - Packet size is 64 byte (Incl. Ethernet CRC)
  - Routing table size is minimum(less than 10 entries)



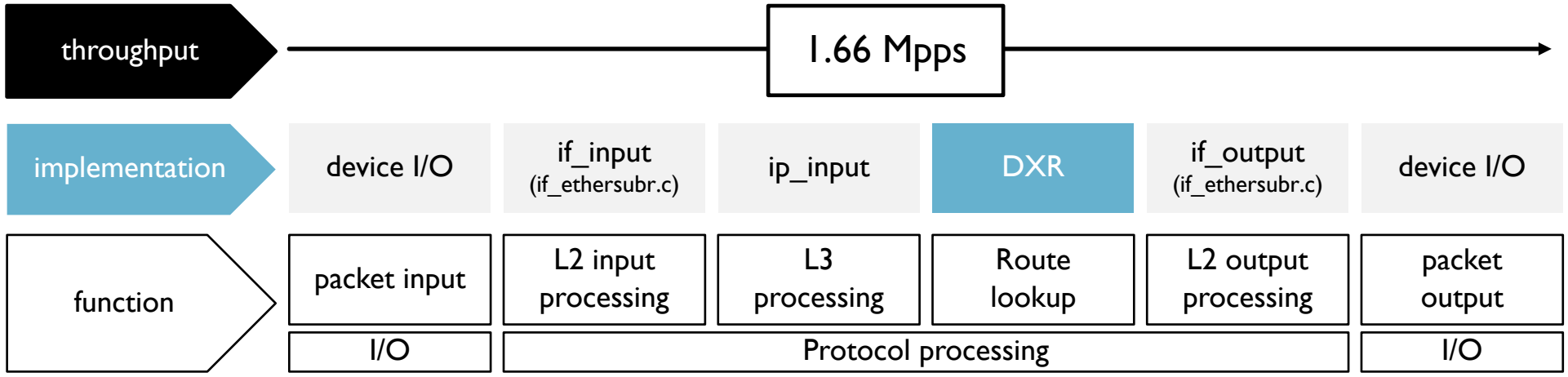
RESULTS

- Default FreeBSD
  - **1.43 Mpps** out of 14.88 Mpps 10GbE line rate



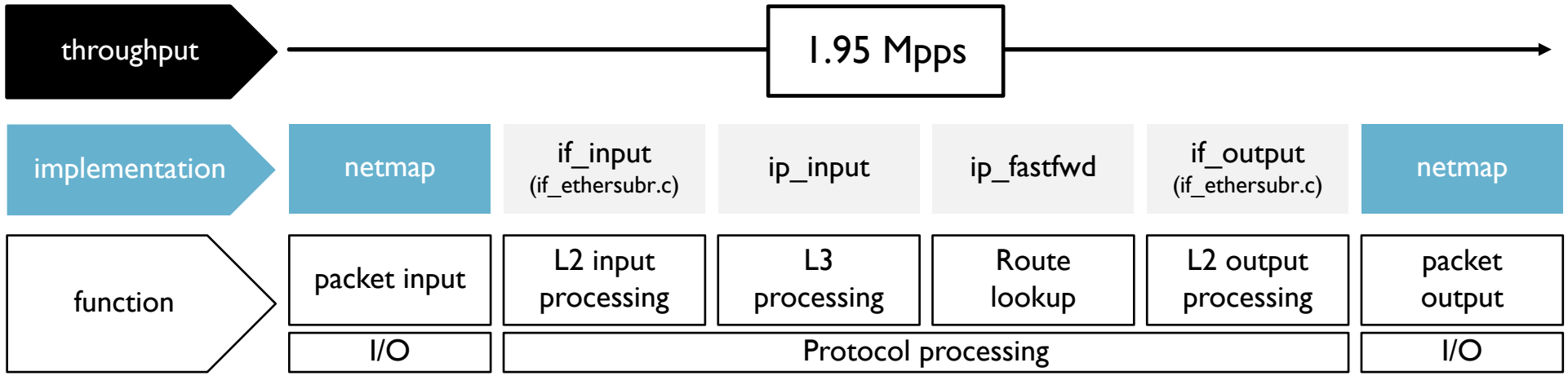
RESULTS

- Default I/O + DXR lookup
  - Using DXR lookup instead of FreeBSD default routing lookup (ip_findroute())
  - **1.66 Mpps** out of 14.88 Mpps 10GbE line rate
  - Replacing lookup part saves 97 ns



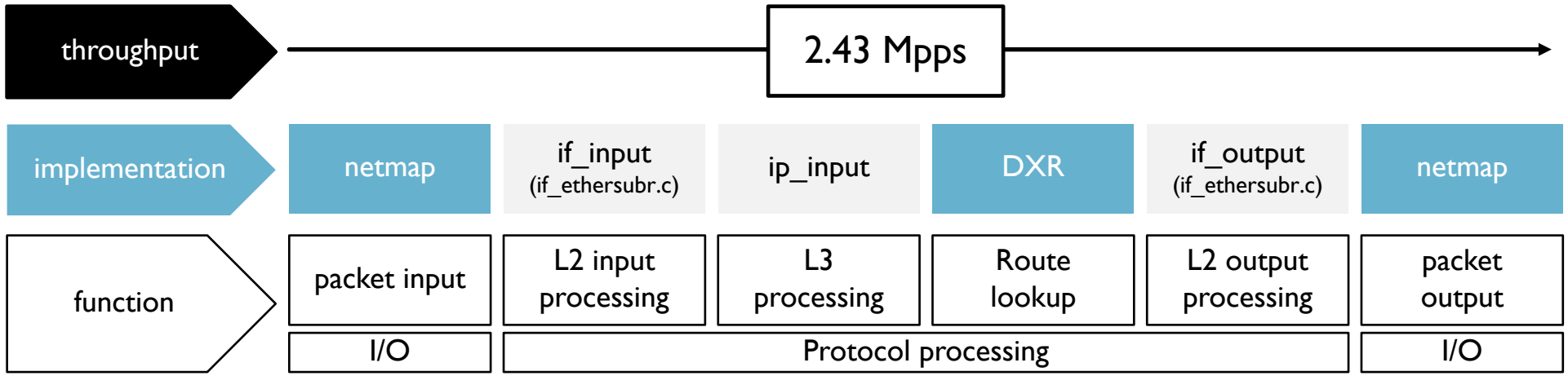
RESULTS

- VALE + default routing lookup
  - Replace FreeBSD default I/O with VALE
  - **1.95 Mpps** out of 14.88 Mpps 10GbE line rate
  - Replacing packet I/O saves 187ns



RESULTS

- VALE + DXR lookup
  - Replace FreeBSD default I/O with VALE and use DXR lookup
  - **2.43 Mpps** out of 14.88 Mpps 10GbE line rate
  - Slightly (1Mpps) faster than default FreeBSD but still SLOW



RESULTS AND TAKEAWAY

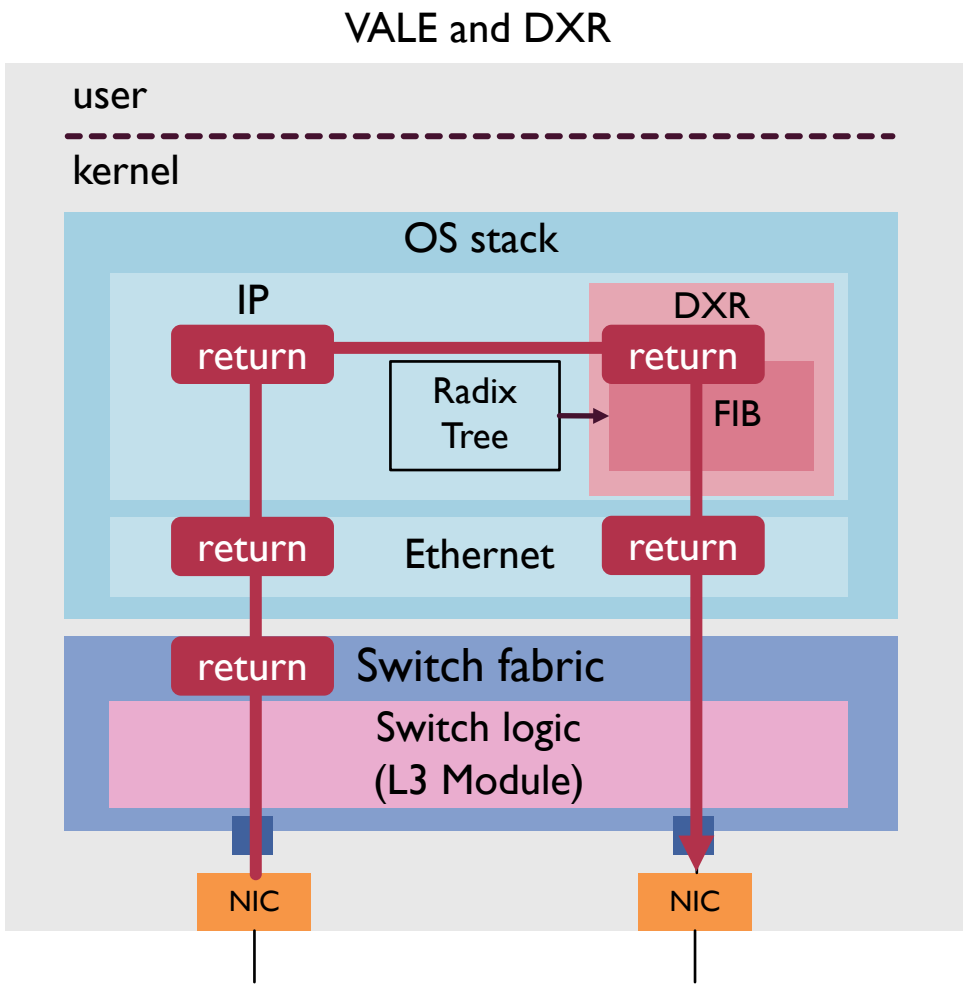
Module	Throughput
Default (baseline)	1.43Mpps
Default I/O + DXR lookup	1.66Mpps
VALE + default lookup	1.95Mpps
VALE + DXR lookup	2.43Mpps
VALE L2 switch	12.39Mpps

- VALE L2 switch itself can achieve 12.39 Mpps
- Why does the 10 Mpps gap between L2 and L3 module exist?
- We should investigate which parts of processing take time

Packet I/O and route lookup are not very expensive anymore

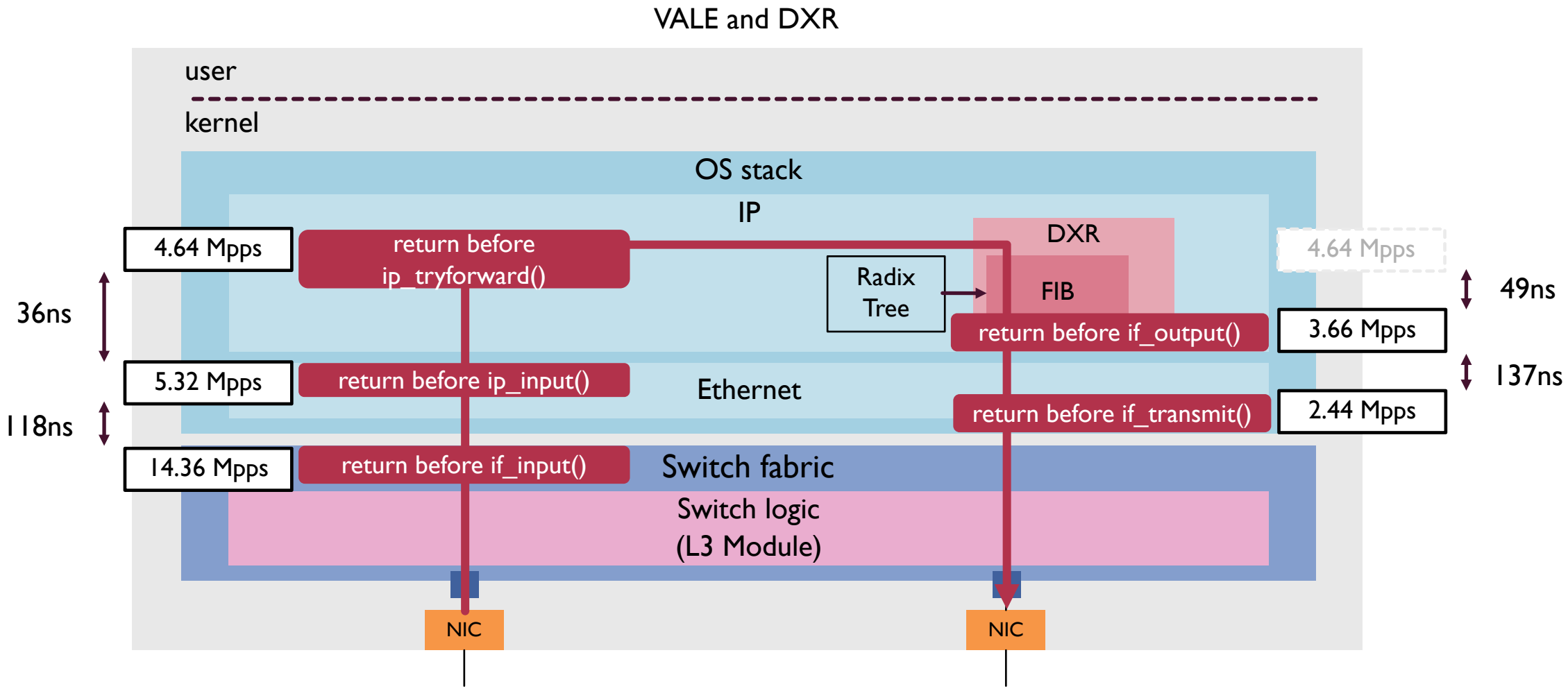
MEASUREMENT METHODOLOGY

- Hardcode the output interface in VALE in advance
- Force to return at the several vantage points
- Receive the packets on the send-and-receive machine and measure rates



VALE + DXR lookup

- Which part does consume time?



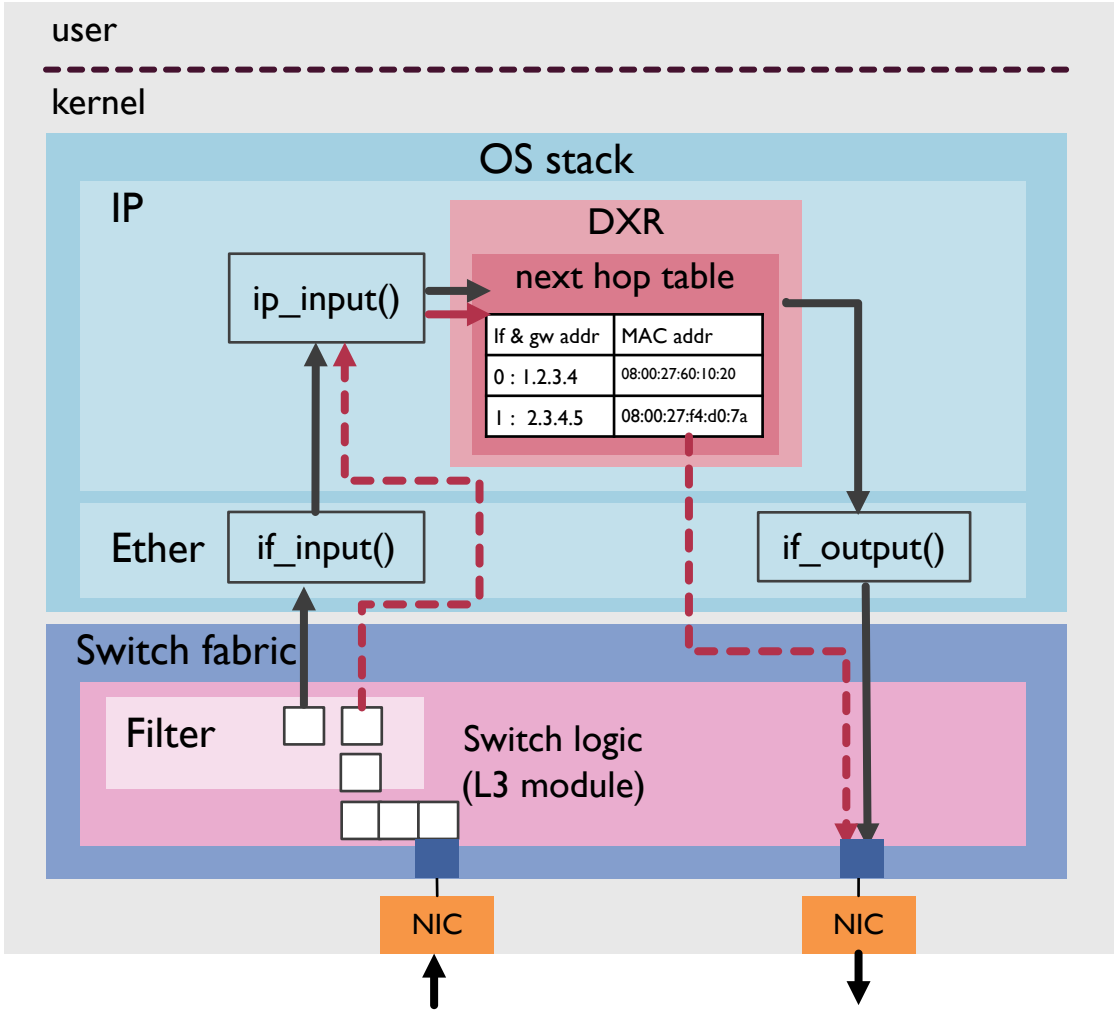
MEASUREMENT CONCLUSION

- Packet I/O is fast enough and the cost of route lookup is negligible
- L2 protocol processing has become a new performance bottleneck

How can we solve this problem?

BASIC DESIGN(ONGOING WORK)

- if_input() bypass
  - Filtering packets in VALE
    - if the packet has protocol type of IPv4(0x0800) and the destination MAC address of the input interface, it directory goes to ip_input()
- if_output() bypass
  - Add a new field in DXR's FIB to cache the destination MAC address of the next hop
    - Avoid if_output() (incl. ARP resolve) for subsequent packets



CONCLUSION

- FreeBSD can forward packets only at 1.43 Mpps
- By replacing packet I/O with VALE, and route lookup with DXR, we can forward packets at 2.43 Mpps
- Ethernet layer processing remains expensive
- We have to bypass it for further speed up

THANK YOU

- Questions? Comments?

- Mail nanako@sfc.wide.ad.jp

- Code https://github.com/nanakom/freebsd/tree/dxr

REFERENCES

- [1] L. Rizzo. netmap: A novel framework for fast packet i/o. In *Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12)*, June 2012.
- [2] DPDK: http://dpdk.org
- [3] T. Yang, G. Xie, Y. Li, Q. Fu, A. X. Liu, Q. Li, and L. Mathy. Guarantee IP Lookup Performance with FIB Explosion. In ACM SIGCOMM, pages 39–50, 2014.
- [4] M. Zec, L. Rizzo, and M. Mikuc. Dxr: Towards a billion routing lookups per second in software. *SIGCOMM Comput. Commun. Rev.*, 42(5):29–36, Sept. 2012.
- [5] H. Asai and Y. Ohara. Poptrie: A compressed trie with population count for fast and scalable software IP routing table lookup. In *ACM SIGCOMM*, pages 57–70, 2015.
- [6] M. Honda, F. Huici, G. Lettieri, and L. Rizzo. mswitch: A highly- scalable, modular software switch. In *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, SOSR '15, pages 1:1–1:13, New York, NY, USA, 2015. ACM