Challenges in software packet switching

P4 Workshop - June 2015

Luigi Rizzo

Università di Pisa

Summary

- **Motivation**
- **Processing stages**
- **Tools and data points**

Motivation

once upon a time…

Motivation

now it is all datacenters and VMs

Motivation

- With VMs and containers, a host switch is always the first and last hop
- can be hardware or software
- can play interesting tricks
  - no checksums, no segmentation, …

Options for VM interconnect

Hardware passthrough

- **+ lower CPU load**
- **- higher cost**
- **- lower aggregate bandwidth (PCIe)**

Software switches

- **+ lower cost**
- **+ higher aggregate bandwidth (memory)**
- **- higher CPU load**
- **- possibly lower performance**

Flexibility ? P4 might handle it for both cases

Switch components

- I/O
- parsing
- matching / classification
- packet manipulation

Performance depends on the slowest component

Software Packet I/O

Well known set of tricks to improve speed

- **simplified data structures**
- **batching**
- **process to completion**
- **aggressive use of caching**

Each of them can become a weakness

- **large difference between best and worst case**

Several OS/stack bypass solution

- **user or kernel, distinction is almost irrelevant**
- **likely to converge to better network stacks in the OS**

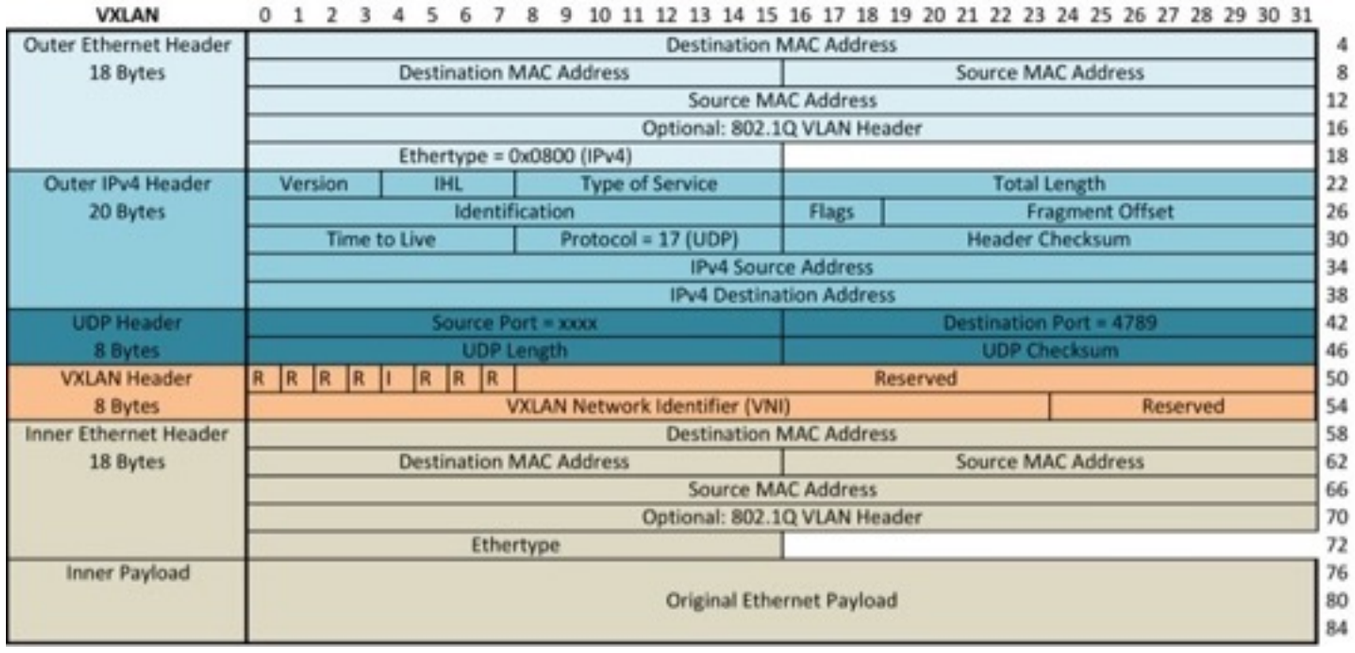
Parsing

Goal:

- Identify headers fields according to specification
- complexity is in the hands of the protocol designer (TLV fields, odd bit sizes etc.)

Approach:

- may pay to be lazy: only extract fields that are actually used



Matching

match/action is a common abstraction

A variety of match strategies:

(exact, ranges, lpm, probabilistic, …)

- **each maps to one or more possible data structures**
  - **huge amount of literature**
- **can be very time and space consuming**
- **compiled or interpreted not much different**

Packet manipulation

Typically requires simple operations:

- **Field rewrite**
- **Field update**
- **Encapsulation/decapsulation**

Avoid full packet reconstruction

- **mostly affects headers**
- **scatter/gather vs copying ?**

Resources

We have some expertise in

- **I/O frameworks (netmap)**
- **classifiers/schedulers (ipfw/dummynet/qfq/dxr)**

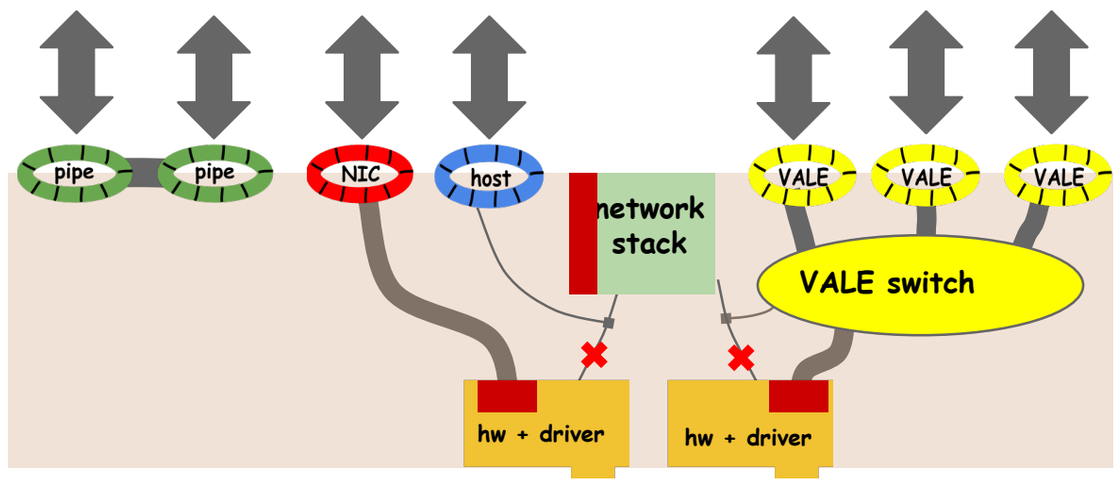
Netmap

Netmap is our framework for packet I/O

- **both userspace and in kernel**
- **FreeBSD, Linux and soon Windows**

Supports NICs, software switches, pipes

- **10 to 50 ns/pkt (tx), 10-20 ns/pkt (rx)**
- **bandwidth limited by memory speed**



Netmap and VMs

Accelerated network path in Qemu and bhyve

- up to 5-8 Mpps and 30-40 Gbit/s per port

Virtual passthrough mode

- both trusted and untrusted peers
- matches netmap speed on bare metal

Classifiers: ipfw

ipfw is the FreeBSD firewall/classifier

- **microinstruction based**
- **field and metadata matching, various tables**
- **runs on all OSes (FreeBSD, Linux, OSX, Windows)**
- **also runs on top of netmap**

Performance:

- **interpreter overhead: 5ns/microinstruction**
- **simple field match: in the noise**
- **LPM lookup: 30ns (trie of depth 32)**

Schedulers

Dummynet includes various schedulers:

- **DRR, QFQ are O(1)**
- **WF2Q+ is O(log N)**
- **cost ranges from 20 to 150ns/pkt**

Conclusions

- Some components are already available for high performance software packet switching
- avg vs worst case very dependent on protocol specification and architectural features of the underlying hw