OpenFlow Pipeline Design for Efficient Hardware Data Planes

Atri Indiresan
Mel Tsai
Manas Pati
Cisco Systems

FAUCETcon

October 20, 2017

Agenda

- FAUCET vs OpenFlow
- OpenFlow pipeline abstraction
- FAUCET in hardware
  - Hardware resources
    - Processing elements and Tables
    - Efficient pipeline design

FAUCET vs OpenFlow

FAUCET is not OpenFlow

FAUCET is a practical subset of OpenFlow that meets the needs of a wide class of Enterprise networking applications

OpenFlow pipeline abstraction

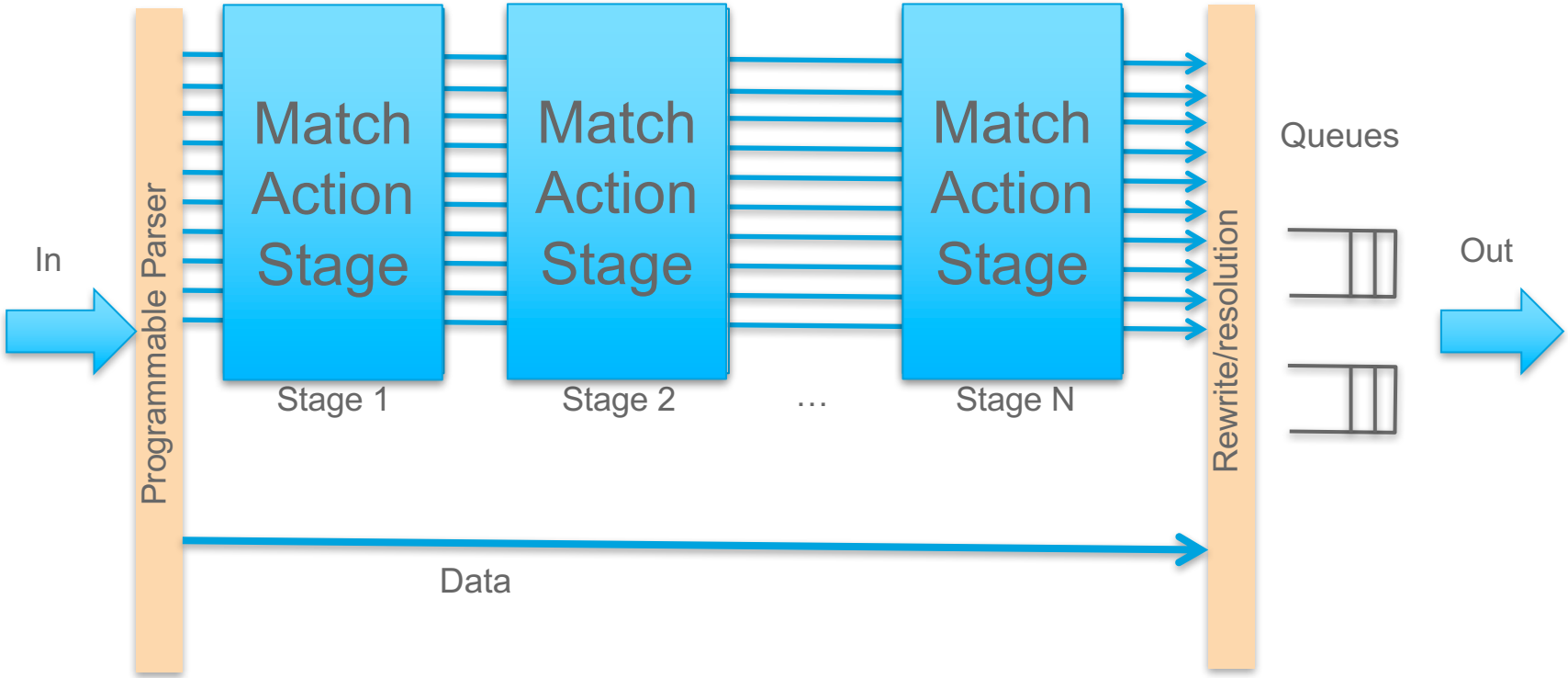
What does the pipeline do?

How is it done?

What? Abstract Functional Model

- Baseline functions
  - L2/L3 forwarding
  - Statistics
- Extended functions
  - ACL
  - Protocol processing
  - Punt
  - Mirror
- Future feature flexibility
  - Meters
  - Tunnels

Match-Action Table Pipeline



How? Abstract data plane model

- Programmable/Flexible Parser
  - Parse packet header data and store in packet state vector
- Match-Action Stages
  - Table lookups based on combinations of packet fields
  - Lookup results can influence packet rewrite as well as subsequent lookup stages
- Rewriter
  - Modify packet based on flexible lookup results
- Transmit
  - Send packet to destination(s) from flexible lookup results

What's in a pipeline?

- Match stages
- Match tables

Hardware Resources

- Programmable/Flexible parser
- Match-Action Stage
  - Lookup tables
    - TCAM
    - Longest Prefix Match
    - Hash
  - Actions
- Rewriter
  - At the Match-Action Stage?
  - At the end of the pipeline?

Some rules …

- Hardware resources are scarce
- Flexibility can be expensive
  - Hardware is more rigid than software
  - TCAMs are much more expensive than hash
- Inefficient use of resources can negatively impact:
  - Functionality
  - Performance
  - Scale

What's in FAUCET?

We have a problem …

Flexible Pipeline Stages

- 9/8 logical pipeline stages in 1.6.7
- Are more needed?
- Can we do with fewer?
- Partial order or total order?
- Recirculation is expensive

Match Tables

The FIB table got better …

```
1.3.2:
   icmp,dl_vlan,nw_src,nw_dst → VIP
   ip,dl_vlan,nw_dst
1.6.7
   ip,dl_vlan
   ip,dl_vlan,nw_dst
```

… but ETH_DST table got worse, and …

```
1.3.2:
   dl_vlan,dl_dst
1.6.7
   dl_vlan,dl_dst
   in_port,dl_vlan,dl_dst ← From FLOOD
```

More Match Tables

… ETH_SRC table is really a mess!

```
1.3.2/1.6.7:
   arp,dl_vlan,arp_tpa
   dl_vlan
   dl_vlan,dl_src
   icmp6,dl_vlan,dl_dst,icmp_type
   icmp6,dl_vlan,icmp_type,nd_target
   in_port,dl_vlan,dl_src
   ip,dl_vlan,dl_dst
   ipv6,dl_vlan,dl_dst
```

Fast Forward to Future FAUCET

… can we make it better?

Preferred: Fixed keys for large scale tables

- SrcMac + port, vlan (L2 learn)
- DestMac + vlan (L2 forward)
  - Hash tables
  - Limited TCAM for destMac with mask (can be in FLOOD table)
- SrcIp + vlan/vrf (uRPF, RPF)
- DstIp + vlan/vrf (L3 forward)
  - Longest-prefix-match (TCAM or tree)
  - Use hash for host routes
  - Narrower keys compared to flexible tables
- ~~Multicast ((S, G), (*, G))~~

Flexible tables

- Variable L2/L3/L4 fields
- IPv4 with SingleWide (256b) TCAM
- IPv6 with DoubleWide (512b) TCAM
- L2/L4 matches with optional L3 requires DoubleWide

Efficient table usage

- Select the large scale tables and assign them optimally in the flexible pipeline
- Separate fixed and flexible functions
  - presence of a flexible function can require that the fixed function is also implemented flexibly (at cost of silicon and scale)
- If possible, make fixed and mask tables into separate logical tables
  - E.g. DestMac lookup key is (vlan, DestMac)
    - All exact match MAC addresses can use hash lookup
      - All masked match can use TCAM
      - Logically, as a single table, all masked match are lower priority than exact match
    - FIB tables use the same principles

Better pipeline stages

- Separate (more?) logical tables with distinct functions
- Parallel tables for mutually exclusive packet types
- Limited rewrite in the pipeline stages
- More aggressive use of groups
  - Shared adjacencies
  - Complex rewrites

Conclusions

This is not the end …

… but just the beginning

- Building flexible pipelines is not easy
- Fast, cheap, flexible … pick any two
- With some changes in how we do things, perhaps all three?
- Application developers and switch vendors get together to make the world a better place

- Let's talk!

Q & A


Your Time Is Now