Programmable Forwarding Planes at Terabit/s Speeds

Patrick Bosshart

CHIEF TECHNOLOGY OFFICER, BAREFOOT NETWORKS

And the entire Barefoot Networks team

Hot Chips 30, August 21, 2018
Barefoot Tofino™: A Domain Specific Processor for Networking

Signal Processing
- Matlab Compiler
- DSP

Graphics
- OpenCL Compiler
- GPU

Machine Learning
- TensorFlow Compiler
- TPU

Networking
- P4 Compiler
- Tofino
Performance Penalty for Programmable Packet Processing

- CPU - 100x
- Network Processor - 10x
- FPGA - 10x
- These ratios have been constant for many years
Presentation Outline

• Introduction to Tofino
• Basics of switching
• Programmable switching
• Tofino details
• Applications

• What is the processing paradigm for programmable switches?
• How does Tofino process packets?
• How does it avoid a performance penalty?
• What can you do with its programmability?
• 6.5Tb/s switch
• 260 lanes of 25G SerDes
• 260 x 25G Ethernet ports, 130 x 50G, 65 x 100G, or combinations
• 11B transistors
• 16nm technology
• 1250 MHz
• Equivalent in area and power to fixed function chips
What’s in a packet? – A simple example

- MAC
  - Dest. Addr.
  - Src. Addr.
  - Eth. Type = IPv4

- IPv4
  - Src. IP
  - Dst. IP
  - IP Proto.
  - Time To Live (TTL) = TCP

- TCP
  - Src. Port
  - Dst. Port
  - Seq. Num.

- Payload
Match and Action

MAC Dest. Addr. → MATCH → ACTION → Packet Header Output

Set Output Port

#
Sequence of Match Tables

MAC TABLE

Exact Match
MAC Dest. Addr.

Longest Prefix Match
IPv4 Dest. Addr.

Ternary Match
IPv4 Dest. Addr.
TCP Src. Port

IP TABLE

ACL TABLE
## Generalizing OpenFlow

<table>
<thead>
<tr>
<th>Components</th>
<th>OpenFlow</th>
<th>Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Fields</td>
<td>Defined standard packet fields</td>
<td>User Defined</td>
</tr>
<tr>
<td>Matching</td>
<td>Defined exact and ternary matching on those fields</td>
<td>Same</td>
</tr>
<tr>
<td>Action</td>
<td>Defined actions on those fields - specific, complex</td>
<td>Generalized, simple actions, VLIW with one action per word</td>
</tr>
<tr>
<td>Multiple Tables</td>
<td>Defined sequence of match-action tables</td>
<td>Same</td>
</tr>
<tr>
<td>Programming the Forwarding Plane</td>
<td>Defined API, e.g. for table, flow entry setup/modification/deletion</td>
<td>P4.Runtime</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Developed by Google, Barefoot, ONF and contributed to P4.org</td>
</tr>
</tbody>
</table>
Define MAC header

```c
header ethernet_t {
    bit<48> dstAddr;
    bit<48> srcAddr;
    bit<16> etherType;
}
```

Define table matching on MAC header word

```c
table mac {
    key = {
        ingress_metadata.bd : exact;
        l2_metadata.lkp_mac_da : exact;
    }
    actions = {
        dmac_hit;
        dmac_miss;
        dmac_redirect_to_cpu;
    }
    default_action = dmac_miss;
    size = MAC_TABLE_SIZE;
}
```

Define table actions

```c
action dmac_hit(bit<16> ifindex, bit<16> port_lag_index) {
    ingress_metadata.egress_ifindex = ifindex;
    ingress_metadata.egress_port_lag_index = port_lag_index;
    l2_metadata.same_if_check = l2_metadata.same_if_check, ^ ifindex;
}
```

P4 Code Example

- Open Source
- Reconfigurable
- Protocol Independent
- Target Independent: s/w, FPGAs, NICs, fixed and programmable switches
- Vendor Independent
- P4 Tutorial at Hot Chips 2017

P4 Runtime

- Open Source
- P4 compiler -> P4 Runtime -> Switch
Tofino Hardware
16 Parsers Per Pipe, Each handles 100G Traffic
Match-Action Unit (MAU)
Table graph mapping to Match-Action Stages

- Multiple small tables per stage
- Large tables spread over multiple stages
Table Predication

- Multiple tables per stage
- Each table produces next-table pointer
- Predication invalidates skipped tables

Table Graph

Short Forward Branches
Dependencies and Pipelining

Different Stages
IPv6
IPv4
MAC Dest. Modification
Creates Match Data Dependency

STAGE
0
1
2
3

Concurrent
Sequential

Execution Pipelining

STAGE
0
1
2
3

ACL
How did we get the programmability without the penalty?

- Didn't use von Neuman processing model
- Compiler and compiler friendly design
- Regular architecture: more replication allows better optimization
- MAU area divided into
  - RAM arrays + support logic + standard switch features
  - Programmable portion
- Programmable portion area about 10% of chip area
- Power is less than the comparable fixed function switches
Aspects of switch performance

- CPU, Network Processor performance
  - Use run-to-completion model
  - Throughput dependent on program
- Hardware switch performance
  - Deterministic
  - Requires assured throughput at 100% traffic, all inputs
- Most important switch parameters
  - TM packet data ram size
  - Match table ram size
- Many specific behaviors required
  - Multicast, queue prioritization, WRED, PFC, ...
Applications

• Switching Applications:
  • Table and Feature scaling
  • Inband Network Telemetry: (INT P4.org)
    • Data added to packet on each hop - which switch, time spent there, ...
    • Data collected on exit
    • Answers question: if I'm delayed, who delayed me?
  • Bloom filters, heavy hitter detection

• Computational Networking Applications:
  • Layer 4 load balancer (SilkRoad: Sigcomm 2017)
    • Replaces ~500 CPUs
    • 500x lower latency
    • DDOS protection
  • DNS cache
  • Key-Value store cache (NetCache: SOSP 2017)
  • ML parameter server
Summary

• Switching is going programmable
• Tofino achieved no performance penalty vs fixed function switches
• P4 enables users to build their network their way
• Opens up wire-speed processing to the networking software industry
Thank You!