Programmable Switch Hardware

ECE/CS598HPN

Radhika Mittal
Conventional SDN

• Programmable *control plane*.

• Data plane can support high bandwidth.
  • But has limited flexibility.

• Restricted to conventional packet protocols.
Software Dataplane

• Very extensible and flexible.

• Extensive parallelization to meet performance requirements.
  • Might still be difficult to achieve 100’s of Gbps.

• Significant cost and power overhead.
Programmable Hardware

- More flexible than conventional switch hardware.
- Less flexible than software switches.

- Slightly higher power and cost requirements than conventional switch hardware.
- Significantly lower than software switches.
Other alternatives?

- CPU
- GPU
- FPGA
- ASIC

Flexibility
Efficiency
Cost per unit
Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN

Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, Mark Horowitz

Acknowledgements: Slides from Pat Bosshart’s SIGCOMM’13 talk
Fixed function switch

- **L2**: 128k x 48
  - Exact match
  - Action: set L2D
- **L3**: 16k x 32
  - Longest prefix match
  - Action: permit/deny
- **ACL**: 4k
  - Ternary match
  - Action: permit/deny

Diagram:
- **PBB Stage**
- **Parser**
- **Stage 1**: L2 Table
- **Stage 2**: L3 Table (TTL)
- **Stage 3**: ACL Table
- **Deparser**
- **Queues**
- **Data**
What if you need flexibility?

• Flexibility to:
  • Trade one memory size for another
  • Add a new table
  • Add a new header field
  • Add a different action

• SDN accentuates the need for flexibility
  • Gives programmatic control to control plane, expects to be able to use flexibility
  • OpenFlow designed to exploit flexibility.
What about Alternatives?
Aren’t there other ways to get flexibility?

- Software? 100x too slow, expensive
- NPUs? 10x too slow, expensive
- FPGAs? 10x too slow, expensive
What the Authors Set Out To Learn

• How to design a flexible switch chip?
• What does the flexibility cost?
RMT Switch Model

Enables flexibility through?

- Programmable parsing: support arbitrary header fields
- Ability to configure number, topology, width, and depths of match-tables.
- Programmable actions: allow a flexible set of actions (including arbitrary packet modifications).
What’s Hard about a Flexible Switch Chip?

• Big chip
• High frequency
• Wiring intensive
• Many crossbars
• Lots of TCAM
• Interaction between physical design and architecture
The RMT Abstract Model

- Parse graph
- Table graph
Arbitrary Fields: The Parse Graph

Packet:

<table>
<thead>
<tr>
<th>Ethernet</th>
<th>IPV4</th>
<th>TCP</th>
</tr>
</thead>
</table>

Diagram:

- Ethernet
  - IPV4
  - IPV6
  - TCP
  - UDP
Arbitrary Fields: The Parse Graph

Packet:

| Ethernet | IPV4 | TCP |

Diagram:

- **Ethernet**
  - **IPV4**
    - **TCP**
    - **UDP**
Arbitrary Fields: The Parse Graph

Packet:

<table>
<thead>
<tr>
<th>Ethernet</th>
<th>IPV4</th>
<th>RCP</th>
<th>TCP</th>
</tr>
</thead>
</table>

Diagram:

- Ethernet
  - IPV4
    - RCP
      - TCP
      - UDP
Arbitrary Fields: Programmable Parser

Figure 4: Programmable parser model.
Reconfigurable Match Tables: The Table Graph
Changes to Parse Graph and Table Graph

Parse Graph:
- Ethernet
- VLAN
- IPV4
- IPV6
- RCP
- TCP
- UDP
- Done

Table Graph:
- ETHERTYPE
- VLAN
- L2S
- IPV4-DA
- IPV6-DA
- L2D
- RCP
- ACL
- MY-TABLE
But the Parse Graph and Table Graph don’t show you how to build a switch
Match/Action Forwarding Model

Match/Action Stage 1

Match/Action Stage 2

...  

Match/Action Stage N

Queues

Data

Programmable Parser

In

Out

Deparser
Performance vs Flexibility

- Multiprocessor: memory bottleneck
- Change to pipeline
- Fixed function chips specialize processors
- Flexible switch needs general purpose CPUs
RMT Logical to Physical Table Mapping
Detour: CAMs and RAMs

• **RAM:**
  - Looks up the value associated with a memory address.

• **CAM**
  - Looks up memory address of a given value.
  - Two types:
    • Binary CAM: Exact match (matches on 0 or 1)
      • Can be implemented using SRAM.
    • Ternary CAM (TCAM): Allows wildcard (matches on 0, 1, or X).
Detour: CAMs

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Address (Binary)</th>
<th>Output Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101XX</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>0110X</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>011XX</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>10011</td>
<td>D</td>
</tr>
</tbody>
</table>

Source: https://www.pagiamtzis.com/cam/camintro/
Detour: CAMs

(b) Binary CAM cell.

(c) Ternary CAM cell

Source: https://www.pagiamtzis.com/cam/camintro/
Detour: CAMs

<table>
<thead>
<tr>
<th>Line No.</th>
<th>Address (Binary)</th>
<th>Output Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>101XX</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>0110X</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>011XX</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>10011</td>
<td>D</td>
</tr>
</tbody>
</table>

![CAM and RAM diagram](https://www.pagiamtzis.com/cam/camintro/)

Source: https://www.pagiamtzis.com/cam/camintro/
RMT Logical to Physical Table Mapping

Table Graph

- ETH
- VLAN
- IPV4
- IPV6
- L2S
- L2D
- TCP
- UDP
- ACL

TCAM
640b

Physical Stage 1

Match Table

Action

Logical Table 1

SRAM

HASH

Logical Table 2

640b

Physical Stage 2

Match Table

Action

Logical Table 3

IPV4

VLAN

IPV6

Physical Stage n

Match Table

Action

Logical Table 6

L2D

IP

5

IPV6
Action Processing Model
Modeled as Multiple VLIW CPUs per Stage

Match result → VLIW Instructions
RMT Switch Design

- 64 x 10Gb ports
  - 960M packets/second
  - 1GHz pipeline
- Programmable parser
- 32 Match/action stages

- Huge TCAM: 10x current chips
  - 64K TCAM words x 640b
- SRAM hash tables for exact matches
  - 128K words x 640b
- 224 action processors per stage
- All OpenFlow statistics counters
Outline

• Conventional switch chip are inflexible
• SDN demands flexibility…sounds expensive…
• How do I do it: The RMT switch model
• Flexibility costs less than 15%
Cost of Configurability: Comparison with Conventional Switch

• Many functions identical: I/O, data buffer, queueing…
• Make extra functions optional: statistics
• Memory dominates area
  • Compare memory area/bit and bit count
• RMT must use memory bits efficiently to compete on cost
• Techniques for flexibility
  • Match stage unit RAM configurability
  • Ingress/egress resource sharing
  • Allows multiple tables per stage
  • Match memory overhead reduction and multi-word packing
### Chip Comparison with Fixed Function Switches

#### Area

<table>
<thead>
<tr>
<th>Section</th>
<th>Area % of chip</th>
<th>Extra Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO, buffer, queue, CPU, etc</td>
<td>37%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Match memory &amp; logic</td>
<td>54.3%</td>
<td>8.0%</td>
</tr>
<tr>
<td>VLIW action engine</td>
<td>7.4%</td>
<td>5.5%</td>
</tr>
<tr>
<td>Parser + deparser</td>
<td>1.3%</td>
<td>0.7%</td>
</tr>
<tr>
<td><strong>Total extra area cost</strong></td>
<td></td>
<td><strong>14.2%</strong></td>
</tr>
</tbody>
</table>

#### Power

<table>
<thead>
<tr>
<th>Section</th>
<th>Power % of chip</th>
<th>Extra Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O</td>
<td>26.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Memory leakage</td>
<td>43.7%</td>
<td>4.0%</td>
</tr>
<tr>
<td>Logic leakage</td>
<td>7.3%</td>
<td>2.5%</td>
</tr>
<tr>
<td>RAM active</td>
<td>2.7%</td>
<td>0.4%</td>
</tr>
<tr>
<td>TCAM active</td>
<td>3.5%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Logic active</td>
<td>16.8%</td>
<td>5.5%</td>
</tr>
<tr>
<td><strong>Total extra power cost</strong></td>
<td></td>
<td><strong>12.4%</strong></td>
</tr>
</tbody>
</table>
Conclusion

• How do we design a flexible chip?
  • The RMT switch model
  • Bring processing close to the memories:
    • pipeline of many stages
  • Bring the processing to the wires:
    • 224 action CPUs per stage

• How much does it cost?
  • 15%

• Lots of the details how we designed this in 28nm CMOS are in the paper
Limitations on Flexibility

• Your thoughts!
Since 2013….

- RMT switch has been commercialized
  - Barefoot Tofino
  - 6.5Tb/s

- Adoption of these switches?
Your opinions

• Pros
  • Proposes RMT as a more flexible alternative to SMT and MMT.
  • Shows viability of a flexible design.
  • Evaluates cost and power requirements, shows they are not significantly high.
    • (In contrast to RouteBricks)
  • Flexible memory allocation mechanism is innovative and efficient.
Your opinions

• Cons
  • Programmability limitations not discussed? Is it Turing-complete?
  • What are the scalability bottlenecks?
  • Why N=32?
  • Conflates memory allocation with match-action processing.
  • No programmability interface.
    • How are low-level configurations generated?
  • No actual hardware
  • Security?
Your opinions

• Ideas
  • A compiler for RMT
  • What can RMT’s programmability enable?
  • Extending the level of programmability / lifting restrictions.