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

*Image copied from somewhere on the web.*
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
What are the limitations of a fixed function switch?
Need for 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 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).
Design Considerations

• Chip size
• High frequency
• Wiring and crossbars
• Amount of memory
The RMT Abstract Model

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

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

![Diagram of arbitrary fields: Ethernet, IPV4, TCP, IPV6, TCP, UDP]

13
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>

- 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

Table Graph
Match/Action Forwarding Model

Programmable Parser

In

Match Action Stage

Stage 1

Match Action Stage

Stage 2

... Match Action Stage

Stage N

Data

Queues

Deparser

Out
Feature 1: flexible compute close to memory

- Multiprocessor: memory bottleneck
- Change to pipeline
- Fixed function chips specialize processors
- Flexible switch needs general purpose CPUs
Feature 2: logical to physical mapping
Tiny 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).
Tiny 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/
Feature 2: logical to physical mapping
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>

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

Table Graph

Logical
Stage
Table
Match
Table
Action
Logical
Stage
Table
Match
Table
Action
Logical
Stage
Table
Match
Table
Action
Action Processing Model

Header In

Field

Field

Header Out

ALU

Data

Instruction

Match result
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
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
Summary

• Conventional switch chip are inflexible
• SDN demands flexibility…sounds expensive…
• How do they do it: The RMT switch model
• *Flexibility costs less than 15%*
### 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 to 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%
How is this paradigm different from active networking?
What are the limitations on flexibility?
Since 2013....

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

• Adoption of these switches?
On research....

• Disaggregated RMT
  • SIGCOMM’17

• Runtime programmability
  • HotNets’21

• Enabling stateful processing
  • HotNets’20

• And many others....