Infinite CacheFlow in Software-Defined Networking

Naga Katta
Omid Alipourfard, Jennifer Rexford, David Walker

Princeton University
Echoes of Y2K: Engineers Buzz That Internet Is Outgrowing Its Gear

Routers That Send Data Online Could Become Overloaded as Number of Internet Routes Hits '512K'

By DREW FITZGERALD
Updated Aug. 13, 2014 7:38 p.m. ET

Network engineers are buzzing this week as the Internet outgrows some of its gear.

Internet providers, corporations and universities all rely on a common map of routes to send emails, videos and everything else on the Web where it's supposed to go.
SDN Promises Flexible Policies
SDN Promises Flexible Policies

Lot of fine-grained rules
SDN Promises Flexible Policies

Lot of fine-grained rules
SDN Promises Flexible Policies

Lot of fine-grained rules
SDN Promises Flexible Policies

Lot of fine-grained rules
SDN Promises Flexible Policies

Limited rule space!
## State of the Art

<table>
<thead>
<tr>
<th></th>
<th>Hardware Switch</th>
<th>Software Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rule Capacity</strong></td>
<td>Low (~2K-4K)</td>
<td>High</td>
</tr>
<tr>
<td><strong>Lookup Throughput</strong></td>
<td>High (&gt;400Gbps)</td>
<td>Low (~40Gbps)</td>
</tr>
<tr>
<td><strong>Port Density</strong></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>Expensive</td>
<td>Relatively cheap</td>
</tr>
</tbody>
</table>
TCAM as cache
TCAM as cache

The diagram illustrates the interaction between the Controller, CacheFlow, and TCAM, with <5% rules cached.
TCAM as cache

Controller

CacheFlow

S1

S2

low expected cache-misses
TCAM as cache

- High throughput + high rule space
TCAM as cache

- High throughput + high rule space

Flexible Deployment
A Correct, Efficient and Transparent Caching System

• Abstraction of an “infinite” switch
  ➢ Correct: realizes the policy
  ➢ Efficient: high throughput & large tables
  ➢ Transparent: unmodified applications/switches
1. Correct Caching
Caching under constraints

<table>
<thead>
<tr>
<th>Rule</th>
<th>Match</th>
<th>Action</th>
<th>Priority</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>110</td>
<td>Fwd 1</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>R2</td>
<td>100</td>
<td>Fwd 2</td>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>R3</td>
<td>101</td>
<td>Fwd 3</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

Easy: Cache rules greedily
Caching Ternary Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Match</th>
<th>Action</th>
<th>Priority</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>11*</td>
<td>Fwd 1</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>R2</td>
<td>1*0</td>
<td>Fwd 2</td>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>R3</td>
<td>10*</td>
<td>Fwd 3</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

- Greedy strategy breaks rule-table semantics
### Caching Ternary Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Match</th>
<th>Action</th>
<th>Priority</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>11*</td>
<td>Fwd 1</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>R2</td>
<td>1*0</td>
<td>Fwd 2</td>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>R3</td>
<td>10*</td>
<td>Fwd 3</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

- Greedy strategy breaks rule-table semantics

**Rules Overlap!**
## Caching Ternary Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Match</th>
<th>Action</th>
<th>Priority</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>11*</td>
<td>Fwd 1</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>R2</td>
<td>1*0</td>
<td>Fwd 2</td>
<td>2</td>
<td>60</td>
</tr>
<tr>
<td>R3</td>
<td>10*</td>
<td>Fwd 3</td>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

- Greedy strategy breaks rule-table semantics
- Beware of switches that claim large rule tables
## Dependency Graph

<table>
<thead>
<tr>
<th>Rule</th>
<th>Match</th>
<th>Action</th>
<th>Priority</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>0000</td>
<td>Fwd 1</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>R2</td>
<td>000*</td>
<td>Fwd 2</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>R3</td>
<td>00**</td>
<td>Fwd 3</td>
<td>4</td>
<td>90</td>
</tr>
<tr>
<td>R4</td>
<td>111*</td>
<td>Fwd 4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>R5</td>
<td>11**</td>
<td>Fwd 5</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>R6</td>
<td>1***</td>
<td>Fwd 6</td>
<td>1</td>
<td>120</td>
</tr>
</tbody>
</table>
Dependent-Set Caching

- All descendants in DAG are dependents
- Cache dependent rules for correctness
2. Efficient Caching
Dependent-Set Overhead

Too Costly?

R1
R2
R3

R4
R5
R6

(*)
Cover-Set

<table>
<thead>
<tr>
<th>Rule</th>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>000</td>
<td>Fwd 1</td>
</tr>
<tr>
<td>R2</td>
<td>00*</td>
<td>Fwd 2</td>
</tr>
<tr>
<td>R3</td>
<td>0**</td>
<td>Fwd 3</td>
</tr>
<tr>
<td>R4</td>
<td>11*</td>
<td>Fwd 4</td>
</tr>
<tr>
<td>R5</td>
<td>1*0</td>
<td>Fwd 5</td>
</tr>
<tr>
<td><strong>R5</strong></td>
<td>1*0</td>
<td><strong>To_SW</strong></td>
</tr>
<tr>
<td>R6</td>
<td>10*</td>
<td>Fwd 6</td>
</tr>
<tr>
<td>(*)</td>
<td>***</td>
<td><strong>To_SW</strong></td>
</tr>
</tbody>
</table>
Dependency **Splicing** reduces rule cost!

Dependent-Set

Cover-Set

Rule Space Cost
Deep Dependency Chains – Clear Gain

• ClassBench Generated ACL

![Graph showing the comparison between Cover-Set Algo and Dependent-Set Algo]
Shallow Dependency Chains – Marginal Gain

- Stanford Backbone Routing table
3. Transparent Caching
3. Transparent Design

Controller

CacheFlow

S1, S2, S3, S4

HW_Cache (TCAM)

OpenFlow

Datapath
3. Transparent Design

Virtual switch

Controller

CacheFlow

S1  S2  S3  S4

HW_Cache (TCAM)

OpenFlow

Datapath
3. Transparent Design

Emulates counters, barriers, timeouts etc.

Controller

Virtual switch

OpenFlow

Datapath

CacheFlow

S1, S2, S3, S4

HW_Cache (TCAM)
Conclusion

• Rule caching for OpenFlow rules
  ➢ Dependency analysis for correctness
  ➢ Splicing dependency chains for efficiency
  ➢ Transparent design
Infinite Ca$hFlow in SDN

Questions?