CutSplit: A Decision-Tree Combining Cutting and Splitting for Scalable Packet Classification

Wenjun Li†, Xianfeng Li†, Hui Li† and Gaogang Xie‡

†School of Electronic and Computer Engineering, Peking University
‡Institute of Computing Technology, Chinese Academy of Sciences

IEEE INFOCOM
Honolulu, HI, April 15-19, 2018
Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion
Multi-field Packet Classification

- Key for policy enforcement in packet forwarding
  - Firewall, QoS, OpenFlow, P4, etc.

![Diagram showing the process of packet classification and forwarding](attachment:image)

An example OpenFlow 1.0 classifier/flow table (12-tuple)

<table>
<thead>
<tr>
<th>#</th>
<th>Ingress Port</th>
<th>Ether src</th>
<th>Ether dst</th>
<th>Ether type</th>
<th>VLAN id</th>
<th>VLAN priority</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>r₁</td>
<td>3</td>
<td>*</td>
<td>*</td>
<td>2048</td>
<td>*</td>
<td>*</td>
<td>Action₁</td>
</tr>
<tr>
<td>IP src</td>
<td>IP dst</td>
<td>IP proto</td>
<td>ToS bits</td>
<td>TCP/UDP</td>
<td>TCP/UDP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP Address</th>
<th>CIDR</th>
<th>Protocol</th>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.25.70.8/30</td>
<td>18.15.125.3/28</td>
<td>0x11/0xff</td>
<td>1</td>
<td>1024 : 65535</td>
</tr>
</tbody>
</table>

School of Electronic and Computer Engineering, Peking University
Previous Works & Key Metrics

- Taxonomy of previous packet classifications
  - Algorithmic: Desired but speed/memory inefficient
  - Architectural: Fast but expensive, power hungry, poor scalability and suffer from range expansion

- Key metrics of scalable packet classification
  - Low memory consumption
  - Low memory accesses
  - Bounded worst-case performance
  - Low pre-processing time
  - Low incremental update time

- Our proposed algorithm: CutSplit
  - A decision-tree based algorithmic approach
**A Little Review on Decision-trees**

- **Decision-tree construction in packet classification**
  1. Rule table matching ↔ Point location in geometric space
  2. Partition the searching space into sub-spaces recursively
    - Root node: Whole searching space containing all rules
    - Internal node: #rule covered by sub-space > a predefined number of rules
    - Leaf node: #rule covered by sub-space ≤ a predefined number of rules

- **Two major threads of building decision-trees**
  - Equal-sized cutting & Equal-dense splitting

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Field X</th>
<th>Field Y</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>111*</td>
<td>*</td>
<td>A1</td>
</tr>
<tr>
<td>R2</td>
<td>110*</td>
<td>*</td>
<td>A2</td>
</tr>
<tr>
<td>R3</td>
<td>*</td>
<td>010*</td>
<td>A3</td>
</tr>
<tr>
<td>R4</td>
<td>*</td>
<td>011*</td>
<td>A4</td>
</tr>
<tr>
<td>R5</td>
<td>01**</td>
<td>10**</td>
<td>A5</td>
</tr>
<tr>
<td>R6</td>
<td>*</td>
<td>*</td>
<td>A6</td>
</tr>
</tbody>
</table>
Two Major Threads in Decision-trees

- **Equal-sized cutting based decision-trees**
  - Separate the searching space into many equal-sized sub-spaces
  - e.g., HiCuts, HyperCuts, EffiCuts, HybridCuts, etc.

- **Equal-dense splitting based decision-trees**
  - Unequal-sized sub-spaces containing nearly equal number of rules
  - e.g., HyperSplit, ParaSplit, SmartSplit, etc.
Why Yet Another Decision-tree?

A well established problem without

Well established solutions

<table>
<thead>
<tr>
<th>Scalability</th>
<th>HyperSplit</th>
<th>EffiCuts</th>
<th>HybridCuts</th>
<th>SmartSplit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory consumption</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Memory accesses</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Worst-case performance</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Pre-processing time</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Incremental update</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion
Rule replication: Main trouble-maker for decision-trees

- In case a rule spans multiple sub-spaces/nodes in decision-tree, rule replication happens, which is an undesirable case.

More insights on rule replication

- Rule replication factor: #stored rules / rule set size

Evaluation of rule replication factor for different rule sets
Recent efforts: Effectiveness & Influence

Optimization methods

- pushing-upwards
- rule overlap
- region compaction, etc.

Effectiveness of pushing-upwards for HyperCuts-8

Influence of pushing-upwards for HyperCuts-8
Recent efforts: Effectiveness & Influence

- Optimization methods
- Rule set partitioning
  - all field based: EffiCuts
  - single field based: HybridCuts
  - IP address based: SmartSplit

Effectiveness of rule set partitioning for EffiCuts-8

Influence of rule set partitioning for EffiCuts-8
Recent efforts: Effectiveness & Influence

- Optimization methods
- Rule set partitioning
- Cutting or Splitting?
  - EffiCuts: HyperCuts + Equi-dense cutting option (i.e., splitting)
  - HybridCuts: One- + multi-dimensional cuttings (i.e., HyperCuts)
  - SmartSplit: \{HyperCuts, HyperSplit\} based on memory estimator

However, the performance of these algorithms drop quickly with the size of rule sets increases: Poor scalability of HyperCuts & HyperSplit

Thus, these efforts reduce *rule replications* while sacrificing search or update performance!
Better Solutions?

No optimization method (with better search & update performance)

+ 

More scalable rule set partitioning (with less rule groups)

+ 

Better combination of Cutting & Splitting (by exploiting characteristics)
Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion
Cutting can separate searching space into smaller sub-spaces quickly for faster classification

Splitting can significantly reduce rule replications and offer a bounded worst-case search performance for small rule sets

To foster the strengths and circumvent the weaknesses of cutting and splitting, the idea directly perceived is to combine the following two strategies: Faster Pre-Cutting & Explicit Post-Splitting
More details & Challenges

- Preprocessing & Constructing search structure

1\textsuperscript{st} Stage: Partitioning
- Few big rules
- 2\textsuperscript{nd} Sub-set
- i\textsuperscript{th} Sub-set
- K\textsuperscript{th} Sub-set

2\textsuperscript{nd} Stage: Decision-Tree Construction
- Few big rules
- HyperSplit
- Pre-Cutting
- Post-Splitting
- Pre-Cutting
- Post-Splitting
- Pre-Cutting
- Post-Splitting

- Fewer sub-sets
  - (Not only for 5-tuple)
- No/Fewer rule replication
- No optimization in cuttings
Observations (1)

At Least One Small Field

The ratio of big rules for seed-acl rule set

School of Electronic and Computer Engineering, Peking University
Observations (1)

- At Least One Small Field

The ratio of big rules for seed-fw rule set

School of Electronic and Computer Engineering, Peking University
Observations (1)

- At Least One Small Field

The ratio of big rules for seed-ipc rule set

School of Electronic and Computer Engineering, Peking University
Very Few **Small Fields**

Table. Statistical results for 5-tuple & OpenFlow-like rules (Assuming the value of $T_i$ is half of range length in field $F_i$)

<table>
<thead>
<tr>
<th>Rule set(#rules)</th>
<th>Number of big rules</th>
<th>Number of small-k rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>seed-acl(752)</td>
<td>3</td>
<td>k=1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>749</td>
</tr>
<tr>
<td>seed-fw(269)</td>
<td>4</td>
<td>265</td>
</tr>
<tr>
<td>seed-ipc(1550)</td>
<td>2</td>
<td>1548</td>
</tr>
<tr>
<td>openflow-1(716)</td>
<td>0</td>
<td>716</td>
</tr>
<tr>
<td>openflow-2(864)</td>
<td>0</td>
<td>864</td>
</tr>
</tbody>
</table>

*The two OpenFlow-like rule tables are generated by Tsinghua University, which are based on 216 real-life rules from enterprise customers. We are very grateful to Pro. Jun Li for his selflessness help in this evaluation.*
Scalable Partitioning

- **Step 1**: Remove very few *big rules*
  - HyperSplit for these rules

- **Step 2**: Select a few distinct fields
  - Top-k significant *small fields* (e.g., >95% rules included)
  - Remove remaining rules to *big rules* in Step 1

- **Step 3**: Fields-wise partitioning
  - $M$ fields selected for $F$-tuple rule sets $\rightarrow 2^M - 1$ sub-sets

- **Step 4**: Selective subset merging
  - Sub-set with very few rules $\rightarrow$ Sub-set with fewer *small fields*
CutSplit: **Pre-Cutting + Post-Splitting**

- **Pre-Cutting** on *small fields*
  - Simpler & More efficiently → No optimization (e.g., FiCuts)
- **Post-Splitting** on small sub-sets after cuttings

**When to switch to Post-Splitting?**

- Achieving threshold value → No rule replication in cutting stage

---

**CutSplit**

- **Pre-Cutting**
  - on *small fields*
    - Simpler & More efficiently
- **Post-Splitting** on small sub-sets after cuttings

---

**Rule Replications**

- No rule replication in cutting stage
Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion
Experimental Setup

Tested with
- Publicly available rule sets from Washington University
  - Used the ACL & FW & IPC 100, 1K, 5K, 10K
- ClassBench
  - Generate ACL & FW & IPC 100K

Compared with
- Cutting based: HyperCuts, EffiCuts and HybridCuts
- Splitting based: HyperSplit and SmartSplit

Primary metrics
- Memory consumption (Decision-tree data structure)
- Memory accesses
- Pre-processing time: decision-tree & sub-trees

Many thanks to authors of HyperCuts & HyperSplit & EffiCuts for their selflessness help (source codes) in evaluations. As a response, our implementation of CutSplit is publicly available in http://wenjunli.com/CutSplit/
Memory Consumption

Byte/rule

MB

School of Electronic and Computer Engineering, Peking University
Pre-processing time: Decision-Tree

Table IV. Pre-processing time for decision-tree construction (s)

<table>
<thead>
<tr>
<th>Rule set</th>
<th>EffiCuts</th>
<th>HybridCuts</th>
<th>SmartSplit</th>
<th>CutSplit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL1_100K</td>
<td>4784.4</td>
<td>183.1</td>
<td>632.5</td>
<td>11.7</td>
</tr>
<tr>
<td>ACL2_100K</td>
<td>8338.4</td>
<td>91.0</td>
<td>427.4</td>
<td>4.1</td>
</tr>
<tr>
<td>ACL3_100K</td>
<td>8453.6</td>
<td>148.6</td>
<td>6403.7</td>
<td>2.6</td>
</tr>
<tr>
<td>ACL4_100K</td>
<td>8232.6</td>
<td>161.8</td>
<td>3336.1</td>
<td>3.4</td>
</tr>
<tr>
<td>ACL5_100K</td>
<td>8905.3</td>
<td>138.5</td>
<td>2695.9</td>
<td>3.0</td>
</tr>
<tr>
<td>FW1_100K</td>
<td>4250.7</td>
<td>165.1</td>
<td>1392.1</td>
<td>3.0</td>
</tr>
<tr>
<td>FW2_100K</td>
<td>2842.2</td>
<td>161.9</td>
<td>1652.9</td>
<td>2.5</td>
</tr>
<tr>
<td>FW3_100K</td>
<td>4281.2</td>
<td>187.8</td>
<td>3855.4</td>
<td>3.0</td>
</tr>
<tr>
<td>FW4_100K</td>
<td>1662.1</td>
<td>280.3</td>
<td>4553.6</td>
<td>3.5</td>
</tr>
<tr>
<td>FW5_100K</td>
<td>3778.4</td>
<td>179.2</td>
<td>3212.7</td>
<td>2.7</td>
</tr>
<tr>
<td>IPC1_100K</td>
<td>8615.0</td>
<td>151.5</td>
<td>3133.4</td>
<td>2.6</td>
</tr>
<tr>
<td>IPC2_100K</td>
<td>6070.4</td>
<td>229.6</td>
<td>3187.9</td>
<td>2.6</td>
</tr>
<tr>
<td>MEAN</td>
<td>5851</td>
<td>173</td>
<td>2874</td>
<td>3.7</td>
</tr>
</tbody>
</table>
### Pre-processing time: Sub-trees

Table V. More details about splitting based sub-trees in CutSplit

<table>
<thead>
<tr>
<th>Rule set</th>
<th>Number of rules</th>
<th>Pre-processing time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Worst-case</td>
<td>Average</td>
</tr>
<tr>
<td>ACL1_100K</td>
<td>344</td>
<td>17.1</td>
</tr>
<tr>
<td>ACL2_100K</td>
<td>473</td>
<td>25.3</td>
</tr>
<tr>
<td>ACL3_100K</td>
<td>31</td>
<td>10.4</td>
</tr>
<tr>
<td>ACL4_100K</td>
<td>320</td>
<td>18.7</td>
</tr>
<tr>
<td>ACL5_100K</td>
<td>93</td>
<td>12.8</td>
</tr>
<tr>
<td>FW1_100K</td>
<td>193</td>
<td>16.4</td>
</tr>
<tr>
<td>FW2_100K</td>
<td>10</td>
<td>9.4</td>
</tr>
<tr>
<td>FW3_100K</td>
<td>118</td>
<td>14.2</td>
</tr>
<tr>
<td>FW4_100K</td>
<td>10</td>
<td>9.0</td>
</tr>
<tr>
<td>FW5_100K</td>
<td>111</td>
<td>14.4</td>
</tr>
<tr>
<td>IPC1_100K</td>
<td>14</td>
<td>9.7</td>
</tr>
<tr>
<td>IPC2_100K</td>
<td>10</td>
<td>9.6</td>
</tr>
<tr>
<td><strong>MEAN</strong></td>
<td>144</td>
<td>14</td>
</tr>
</tbody>
</table>
Outline

- Background
- Challenge Review
- Proposed CutSplit
- Evaluation
- Conclusion
Conclusion

- **CutSplit:**
  - In-depth challenge review
  - Novel observations
  - Scalable partitioning
  - Pre-Cutting & Post-Splitting

- **Future Works**
  - Determinacy on performance
  - Software-hardware combined, e.g., FPGA
  - Combine with TSS, TCAM, etc.
Thank you!

Web: http://www.wenjunli.com
E-mail: wenjunli@pku.edu.cn