Stochastic Pre-Classification for SDN Data Plane Matching

Luke McHale, C. Jasson Casey, Paul V. Gratz, Alex Sprintson

Presenter: Luke McHale
Ph.D. Student, Texas A&M University

Contact: luke.mchale@tamu.edu
Motivation

- SDNs increase stress on Packet Classification
  - Higher complexity compared to traditional networks
  - Generalizing increases timing variability
- Denial of Service (DoS) attacks on SDN switches are a potential issue
  - Wastes resources, crowding out legitimate packets
  - Inherent problem: traffic must be classified before it can be determined malicious
Motivation: Packet Classification

- **Exact Matching**
  - Hash Tables
- **Prefix Match**
  - Tries
- **Arbitrary**
  - TCAM (limited in size/availability)
Motivation: Flow Locality

- 35% of the flows contain 95% of the packets
  - The *active-flow window* is constantly changing

CAIDA Trace: equinix-sanjose.dirA20120119-125903
Outline

1. Motivation
   a. Packet Classification
   b. Flow Locality

2. Taking Advantage of Flow Locality
   a. Flow Caching
   b. Pre-Classification

3. Evaluation
   a. Experimental Setup
   b. Firewall Application

4. Results

5. Conclusions
Flow Caching

- Flow Locality -> Caching becomes fast-path
  - Keeps high-throughput flows
  - Lookup: exact-match using key (i.e. 5-tuple)
  - Cache: action set
  - Hits: bypass Table and Flow selection

Similar techniques: [I.L. Chvets et al., 2002], [K. Li et al., 2003]
Pre-Classification

- Attacks aim to stress slow-path (classification)
  - When stressed, prioritize established traffic
  - Lookup: exact-match using key (i.e. 5-tuple)
  - Cache: seen before
  - Hits: higher classification priority
Bloom Filter

- Hit: flow likely seen within epoch
- Miss: flow definitely not seen within epoch
- O[1] Lookups
- O[1] Inserts
- False positive rate is proportional to fill level
- Tradeoffs in design
Bloom Filter: XOR Hash Function

- Bit-level XOR helps preserve entropy
- Avoid mixing heavily correlated bits
Experimental Setup

- Cycle accurate simulator
- Frequency determined by array sizes using CACTI

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Plane Frequency</td>
<td>2 GHz</td>
</tr>
<tr>
<td>Data Plane Queue Depth</td>
<td>2 high, 2 low</td>
</tr>
<tr>
<td>Bloom Filter Size</td>
<td>320Kb (5 arrays, each 64Kb)</td>
</tr>
<tr>
<td>Bloom Filter Clearing Interval</td>
<td>60K insertions</td>
</tr>
<tr>
<td>Flow Cache Size</td>
<td>69Kb (512 138-bit entries)</td>
</tr>
<tr>
<td>Flow Cache Organization</td>
<td>2-way set associative, LRU</td>
</tr>
<tr>
<td>Flow Selection</td>
<td>8,000 entries</td>
</tr>
</tbody>
</table>

PCAP Trace → Time-Scaled Interface → Classification Simulator → Packet Collection & Statistics

- 10 Gbps
- {1, 10, 40, 100 Gbps}
Firewall Application

- Simulated Firewall Application
- Access Control List (ACL)
  - Protocol
  - IP source/destination
  - Port source/dest. ranges
- Test ACL Generation
  - 95%: nominal network conditions
  - 60%: network with significant malicious traffic
  - 20%: network under attack
Results: Throughput

![Diagram showing packet processing stages: Key Extraction, Table Selection, Flow Selection, Action Application, and packet output.]

- Baseline - 95%
- Baseline - 60%
- Baseline - 20%

- Stressed >1 Gbps
Results: Throughput

- Stressed >1 Gbps
- Throughput proportional to unauthorized traffic
Results: Throughput

- Stressed >1 Gbps
- Throughput proportional to unauthorized traffic
- Unauthorized traffic has less impact on throughput
Results: Throughput

- Stressed >1 Gbps
- Throughput proportional to unauthorized traffic
- Unauthorized traffic has less impact on throughput
- More consistent throughput
Results: Latency

Queue saturation causes high latency >1 Gbps
Results: Latency

- Queue saturation causes high latency >1 Gbps
- Hits improve average latency
Results: Latency

Queue saturation causes high latency >1 Gbps

- Hits improve average latency
- Authorized traffic not yet seen incurs higher latency
  - Once flow is learned, latency consistent with Baseline
Results: Latency

- Queue saturation causes high latency >1 Gbps
- Hits improve average latency
- Authorized traffic not yet seen incurs higher latency
  - Once flow is learned, latency consistent with Cache
- Higher latency at start of flow
  - Latency is constant with cache thereafter
Results: Jitter

* Peaks at saturation point
Results: Jitter

- Peaks at saturation point
- Difference in fast vs. slow path increases variance
Results: Jitter

- Peaks at saturation point
- Difference in fast vs. slow path increases variance
- Learning path incurs higher latency -> jitter
  - Once flow is learned, jitter consistent with Caching
Results: Jitter

- Peaks at saturation point
- Difference in fast vs. slow path increases variance
- Learning path incurs higher latency -> jitter
  - Once flow is learned, jitter consistent with Caching
- Improves jitter incurred by priority mechanism
Conclusions

- SDN complexity increases stress on Classification
- Flow Cache minimizes the effect of repeatedly classifying high-throughput flows
  - Increases effective throughput
- Pre-Classification prioritizes known traffic
  - Reduces effect of malicious traffic
- Combined architecture provides orthogonal benefit
  - Helps decouple legitimate and malicious traffic
Results: Throughput

![Diagram of Throughput Results]

- **Known Flows**
- **Unknown Flows**

- **Key Extraction**
- **Bloom Filter**
- **Priority Scheduler**
- **Flow Cache**
- **Table Selection**
- **Flow Selection**
- **Action Application**

- **Baseline - 95%**
- **Baseline - 60%**
- **Baseline - 20%**
- **Caching - 95%**
- **Caching - 60%**
- **Caching - 20%**
- **Partition - 95%**
- **Partition - 60%**
- **Partition - 20%**
- **Partition+Caching - 95%**
- **Partition+Caching - 60%**
- **Partition+Caching - 20%**

**Graph:**
- **Normalized Throughput** on the y-axis.
- **Interface Speed (Gbps)** on the x-axis.

Legend:
- Solid lines: Baseline
- Dashed lines: Caching
- Dotted lines: Partition
- Dotted-dashed lines: Partition+Caching

**Axes:**
- **Y-axis:** 0 to 1
- **X-axis:** 1 to 100
Results: Latency

Mean Latency (µs) vs Interface Speed (Gbps)

- Baseline - 95%
- Baseline - 60%
- Baseline - 20%
- Caching - 95%
- Caching - 60%
- Caching - 20%
- Partition - 95%
- Partition - 60%
- Partition - 20%
- Partition+Caching - 95%
- Partition+Caching - 60%
- Partition+Caching - 20%
Results: Jitter

![Diagram of packet processing with Known and Unknown Flows]

- **Baseline** - 95%
- **Baseline** - 60%
- **Baseline** - 20%
- **Caching** - 95%
- **Caching** - 60%
- **Caching** - 20%
- **Partition** - 95%
- **Partition** - 60%
- **Partition** - 20%
- **Partition + Caching** - 95%
- **Partition + Caching** - 60%
- **Partition + Caching** - 20%

![Graph showing Jitter vs. Interface Speed]

- Jitter (µs)
- Interface Speed (Gbps)

- **Flow Selection**
- **Application**
- **Packet**
- **Update**

**Known Flows**
- Key Extraction
- Bloom Filter
- Priority Scheduler

**Unknown Flows**
- Packets
- Bloom Filter
- Priority Scheduler