SNAP: Stateful Network-Wide Abstractions for Packet Processing

Mina Tahmasbi Arashloo¹, Yaron Koral¹, Michael Greenberg², Jennifer Rexford¹, and David Walker¹

¹ Princeton University, ² Pomona College
Early SDN Switch Interfaces

- Manipulate packet forwarding rules
- Read predefined set of counters
Programmable Switch Interfaces

• P4, OpenState, Open vSwitch, ...
  – Programmable state (e.g. indexed arrays)
  – Basic arithmetic operations
**SNAP: Stateful Network Wide Programming Language**

One big stateful switch
SNAP Contributions

Modular Stateful Language

One Big Stateful Switch

Placement + Routing
Talk Outline

1. Language through example
2. Compiler
3. Implementation
4. Evaluation
5. Related Work & Conclusion
Example - DNS Reflection Attacks

- Attacker
- Botnet
- Spoofed DNS Requests
- DNS Resolvers
- Victim
  - DNS Responses
Detecting DNS Reflection Attacks

1. Log DNS requests
2. Match responses
3. Check unmatched count

Bohatei: flexible and elastic DDoS defense, Fayaz et.al., USENIX SECURITY 15
DNS Reflection Detection in SNAP

if srcip in CSNET & dstport = 53 then
    seen[srcip][dns.id] ← True
else if dstip in CSNET & srcport = 53 then
    if ~seen[dstip][dns.id] then
        unmatched[dstip]++;
        if unmatched[dstip] = threshold then
            susp[dstip] ← True
    else
        id
else
    id

• Seen: Keep track of DNS requests by client and DNS identifier
• Unmatched: Count DNS responses that don’t match prior requests
• Susp: Suspected victims receive many unmatched responses
OBSS Forwarding in SNAP

\[
\begin{align*}
\text{if } \text{dstip} &= \text{CSNET then } \text{outport} \leftarrow \text{CS} \\
\text{else if } \text{dstip} &= \text{EENET then } \text{outport} \leftarrow \text{EE} \\
\text{else if } \text{dstip} &= \text{ISP1NET then } \text{outport} \leftarrow \text{ISP1} \\
\text{else if } \text{dstip} &= \text{ISP2NET then } \text{outport} \leftarrow \text{ISP2} \\
\text{else } &\text{drop}
\end{align*}
\]
Single Network Policy

DNS Reflection Detection

```python
if srcip in CSNET & dstport = 53 then
    seen[srcip][dns.id] ← True
else if dstip in CSNET & srcport = 53 then
    if ~seen[dstip][dns.id] then
        unmatched[dstip]++;
        if unmatched[dstip] = threshold then
            susp[dstip] ← True
    else id
else id
```

Forwarding

```python
if dstip = CSNET then outport ← CS
else if dstip = EENET then outport ← EE
else if dstip = ISP1NET then outport ← ISP1
else if dstip = ISP2NET then outport ← ISP2
else drop
```
## SNAP Applications

<table>
<thead>
<tr>
<th>Source</th>
<th>Application</th>
</tr>
</thead>
</table>
| Chimera (USENIX Security’12) | Number of domains sharing the same IP address  
                        | Number of distinct IP addresses under the same domain  
                        | DNS TTL change tracking  
                        | DNS tunnel detection  
                        | Sidejack detection  
                        | Phishing/spam detection |
| FAST (HotSDN’14)    | Stateful firewall  
                        | FTP monitoring  
                        | Heavy-hitter detection  
                        | Super-spreader detection  
                        | Sampling based on flow size  
                        | Selective packet dropping (MPEG frames)  
                        | Connection affinity |
| Bohatei (USENIX Security’15) | SYN flood detection  
                        | DNS reflection (and amplification) detection  
                        | UDP flood mitigation  
                        | Elephant flows detection |
| Others           | Bump-on-the-wire TCP state machine  
                        | Snort flowbits |
Single Network Policy
SNAP Compiler

Where to place state variables

How to forward packets through them
Routing + Placement Jointly

if ~seen[dstip][dns.id] then
    unmatched[dstip]++;
else if dstip in CSNET & srcport = 53
    if dstip = CSNET then outport ← CS

Minimize congestion
Intermediate Representation (IR)

Composable and easily partitioned IR

Maintain all programs in a single data structure

Distribute the program to switches
**xFDDs: Extended Forwarding Decision Diagrams**

- **Intermediate node**: test on header fields and state
- **Leaf**: set of action sequences

- Three kinds of tests
  - field = value
  - field₁ = field₂
  - state_var[idx] = val

```
dstip = 10.0.0.1
srcip = dstip
s[srcip] = 2
{s[dstip] ← 2} {drop}
```
Maintain all programs in a single data structure
xFDD for DNS Reflection Detection

Fixes the order in which programs access state.

We could distribute the programs by placing cuts.
Partitioning to Sub-Programs

Distribute the program to switches
Partitioning to Sub-Programs

Distribute the program to switches
Putting It All Together

ISP1

ISP2

CS

EE

srcip in CSNET

dstip in CSNET

seen[dstip] [dns.id]

srcport = 53

dstip in EENET

outport ← EE

outport ← CS

unmatched[dstip] = threshold - 1

{unmatched[dstip]++; susp[srcip][dstip] ← True; outport ← CS}

{unmatched[dstip]++; outport ← CS}
Putting It All Together

ISP1

ISP2

1. srcip in CSNET

2. srcport = 53

3. dstip in CSNET

4. seen[dstip][dns.id]

5. outport ← CS

6. unmatched[dstip] = threshold - 1

7. {unmatched[dstip]++, susp[srcip][dstip] ← True; outport ← CS}

8. {unmatched[dstip]++; outport ← CS}

9. dstip in EENET

10. outport ← EE
Putting It All Together

ISP1

ISP2

CS

EE

1

2

srcip in CSNET

srcport = 53

dstip in CSNET

dstip in EENET

4

5

6

7

8

seen[dstip] [dns.id]

unmatched[dstip] = threshold - 1

outport ← CS

{unmatched[dstip]++;
 susp[srcip][dstip] ← True;
 outport ← CS}

outport ← EE
Putting It All Together

ISP1

ISP2

CS

EE

1. srcip in CSNET

2. srcport = 53

3. dstip in CSNET

4. seen[dstip] [dns.id]

5. outport ← CS

6. unmatched[dstip] = threshold - 1

7. {unmatched[dstip]++; susp[srcip][dstip] ← True; outport ← CS}

8. {unmatched[dstip]++; outport ← CS}

9. dstip in EENET

10. outport ← EE
Language through example

Compiler

Implementation

Evaluation

Related Work & Conclusion
SNAP Implementation

• Compiler written in Python
• MILP solver: Gurobi Optimizer
• Resulting switch code NetASM (language + software switch)

M. Shahbaz and N. Feamster. The case for an intermediate representation for programmable data planes. SOSR 2015.
Talk Outline

Language through example

Compiler

Implementation

Evaluation

Related Work & Conclusion
Compiler Evaluation

• 7 campus and ISP topologies
• Order of 100s of switches and links

• Scenarios
  – Cold start (freq. weeks)
  – Policy change (freq. days)
  – Topology/TM change (freq. minutes)
Compiler Evaluation - Results

![Graph showing time (sec.) for Topology/TM Change, Policy Change, and Cold Start for different institutions and time intervals: Planned in advance.]
## Related Work

<table>
<thead>
<tr>
<th></th>
<th>Programmable State</th>
<th>Network-Wide</th>
<th>Data Plane State</th>
<th>Joint Placement and Routing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SNAP</strong> (SIGCOMM’16)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Stateful NetKAT</strong> (PLDI’16)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td><strong>Kinetic</strong> (NSDI’15)</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Domino</strong> (SIGCOMM’16)</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td><strong>OpenState</strong> (SIGCOMM-COR’14)</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td><strong>FAST</strong> (HotSDN’14)</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td><strong>Merlin</strong> (CONEXT’14)</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td><strong>Slick</strong> (SOSR’15)</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Stratos</strong> (TR’13)</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
</tbody>
</table>
Conclusion - SNAP

• A new modular stateful SDN programming language with:
  – One-big switch programming model
  – Persistent global arrays

• Compiler implements algorithms that:
  – Jointly optimize routing and state placement
  – Use efficient IR based on FDDs

• Evaluated about 20 applications