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

http://dl.acm.org/citation.cfm?id=2934892
Traditional Networks - Distributed Control

• The operator needs to decide
  • how to configure each individual device
  • such that the devices collectively realize the global network policy
Software Defined Networks (SDN) Centralized Control

Program your network from a logically central point!
Early SDN Switch Interfaces

- Manipulate packet forwarding rules
- Read predefined set of counters
Example - Detecting DNS Reflection Attacks

Attacker

Attacker-controlled botnet

Small spoofed DNS request

Amplified DNS response from open resolver

Victim

DNS Resolvers

http://ddosandbotnets.blogspot.com
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
Example - Detecting DNS Reflection Attacks

<table>
<thead>
<tr>
<th>IP</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>223</td>
</tr>
<tr>
<td>H1</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP</th>
<th>Unmatched</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>10</td>
</tr>
<tr>
<td>H1</td>
<td>0</td>
</tr>
</tbody>
</table>

DNS Request from: H1
ID: 100

CS
Example - Detecting DNS Reflection Attacks

<table>
<thead>
<tr>
<th>IP</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>223</td>
</tr>
<tr>
<td>H1</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP</th>
<th>Unmatched</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>10</td>
</tr>
<tr>
<td>H1</td>
<td>0</td>
</tr>
</tbody>
</table>

DNS Response to: H1
ID: 200

CS
Example - Detecting DNS Reflection Attacks

<table>
<thead>
<tr>
<th>IP</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>223</td>
</tr>
<tr>
<td>H1</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP</th>
<th>Unmatched</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>10</td>
</tr>
<tr>
<td>H1</td>
<td>1</td>
</tr>
</tbody>
</table>

Diagram showing network topology and IP addresses.
Practical Concerns

• Cannot send every packet to the controller

• Cannot have **per-packet stateful** processing

  • i.e. maintain state across packets and decide what to do with each packet based on the state!

  • e.g. security, monitoring, etc.
## Use Cases

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

- Recent switch interfaces expose
  - **Programmable state** (e.g. arrays of registers)
  - Basic arithmetic operations
  - e.g. P4, POF, OpenState
- Vendors starting to implement these interfaces
  - Barefoot, Netronome, Xilinx, Open vSwitch
Opportunity: Local State on the Data Plane
SNAP: Stateful Network Wide Programming Language
SNAP Language
**Stateless** Packet Processing Functions

- A function that specifies
  - how to process each packet
  - based on its **fields**
Forwarding

```plaintext
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
```
**Stateful** Packet Processing Functions

- A function that specifies
  - how to process each packet
  - based on its **fields** and the **program state**
DNS Reflection Detection in SNAP

```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
```
DNS Reflection Detection in SNAP

```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
```
DNS Reflection Detection in SNAP

```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
```
DNS Reflection Detection in SNAP

```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
```
DNS Reflection Detection in SNAP

```plaintext
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
```
Composition

```plaintext
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
```

```plaintext
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
```
Composition

\[(\text{red} + \text{yellow} + \text{green} + \text{pink}) + (\text{blue} + \text{orange}) + (\text{purple} + \text{green})]\]
# Applications in SNAP

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

```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

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
```
How to Forward Packets through State Variables?

```plaintext
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
```

```plaintext
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
```
How to Forward Packets through State Variables?

```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

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
```

HTTP Packet to: H10 in EE
How to Forward Packets through State Variables?

```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
```

```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
```

DNS Response to: H1 in CS ID: 200

CS

EE

unmatched

seen

susp

??
Program Analysis

• For each flow, find
  • all the state variables that it needs
  • the order in which the state variables should be visited
State Dependencies in 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
```
State Dependencies in DNS Reflection Detection

```plaintext
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
```
Mixed-Integer Linear Program (MILP)

Program Analysis Results

Traffic Matrix

Traffic Matrix
## MILP Constraints

<table>
<thead>
<tr>
<th>Routing Constraints</th>
<th>State Placement Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have equal incoming and outgoing traffic for each switch</td>
<td>Place each state on the path of traffic that needs it</td>
</tr>
<tr>
<td>Stay within link capacities</td>
<td>Respect state dependencies</td>
</tr>
</tbody>
</table>
Mixed-Integer Linear Program (MILP)
How to distribute a SNAP program?

Intermediate Representation!
Extended Forwarding Decision Diagrams (xFDDs)

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

\[
\begin{align*}
\text{dstip} &= 10.0.0.1 \\
\text{srcip} &= \text{dstip} \\
s[\text{srcip}] &= 2 \\
\{s[\text{dstip}] \leftarrow 2\} &\quad \{\text{drop}\}
\end{align*}
\]

* A Fast Compiler For NetKAT, Steffen Smolka et.al, ICFP 2015
Extended Forwarding Decision Diagrams (xFDDs)

• Three types of tests
  • field = value
  • field\(_1\) = field\(_2\)
  • state\(_{\text{var}}[\text{e}_1]\) = e\(_2\)
Extended Forwarding Decision Diagrams (xFDDs)

- Three types of tests
  - field = value
  - field$_1$ = field$_2$
  - state_var[e$_1$] = e$_2$

```plaintext
\[
\text{dstip} = 10.0.0.1
\]
\[
\text{srcip} = \text{dstip}
\]
\[
\text{s[srcip]} = 2
\]
\[
\{s[dstip] \leftarrow 2\}
\]
\[
\{\text{drop}\}
\]
**xFDD for DNS Reflection Detection**

```plaintext
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
```
xFDD for the Running Example
Partitioning and Distribution of the xFDD
Partitioning and Distribution of the xFDD
Putting It All Together

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
Putting It All Together

ISP1

ISP2

CS

EE

srcip in CSNET

dstip in CSNET

seen[dstip] [dns.id]

outport ← CS

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

srcport = 53

dstip in EENET

unmatched[dstip] = threshold - 1

{unmatched[dstip]++; outport ← CS}

outport ← EE

1

2

3

4

5

6

7

8

9

10
Putting It All Together

ISP1  

ISP2  

CS  

EE  

srcip in CSNET  

dstip in CSNET  

dstip in EENET  

1  

2  

srcport = 53  

3  

4  

seen[dstip] 
[dns.id]  

5  

outport ← CS  

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

6  

unmatched[dstip] = threshold - 1  

7  

8  

9  

10  

outport ← EE
Putting It All Together

ISP1

ISP2

CS

EE

srcip in CSNET

srcport = 53

dstip in EENET

dstip in CSNET

seen[dstip] [dns.id]

outport ← CS

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

unmatched[dstip] = threshold - 1

outport ← EE
SNAP Prototype

- Compiler is written in **Python**
- **Gurobi Optimizer** is used as MILP solver
- Resulting configurations are in **NetASM** (language + software switch)

The case for an intermediate representation for programmable data planes, M. Shahbaz and N. Feamster, SOSR 2015.
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 - Scenarios

<table>
<thead>
<tr>
<th>Phase</th>
<th>Cold Start</th>
<th>Policy Change</th>
<th>Topology or TM Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MILP creation/update</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routing</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>MILP solving</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State placement</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Routing</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Configuration generation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Compiler Evaluation - Results

![Graph showing time in seconds for different scenarios and institutions.]

- Stanford
- Berkley
- Purdue
- ISP 1755
- ISP 1221
- ISP 6461
- ISP 3257

Scenarios:
- Topology/TM Change
- Policy Change
- Cold Start
Other Experiments

- How the compiler scales with
  - size of the network
  - number of state variables
- The curves are close to exponential
  - Joint state placement and routing is NP complete
- Reaction to topology and traffic matrix change stays within a minute
## Related work

<table>
<thead>
<tr>
<th>Stateful Languages</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>SNAP (SIGCOMM’16)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Stateful NetKAT (PLDI’16)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Kinetic (NSDI’15)</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Switch-level Mechanisms</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domino (SIGCOMM’16)</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>OpenState (SIGCOMM-CCR’14)</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>FAST (HotSDN’14)</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Optimizing Routing and Placement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merlin (CONEXT’14)</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Slick (SOSR’15)</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Stratos (TR’13)</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
</tbody>
</table>
Summary

- SNAP Language
  - One big stateful switch abstraction
  - Intuitive and flexible composition

- SNAP Compiler
  - Decides state placement and routing
  - Distributes an intermediate representation of the program over the network
Thank You!

Questions?
Backup Slides
Common Solution Nowadays: Middleboxes

- Monolithic hardware/software designed and optimized for specific types of packet processing

- Crucial to networks, but:
  - hard to customize to your own network
  - reduce visibility and control
  - hard to reason across network functions
Atomic Blocks

• We assume that state reads/writes in a single switch happen **atomically**.

• If the programmer puts a part of the program in the **atomic** block, all the state variables in the block end up on the same switch.
SNAP Expression to FDD

- Sequential Composition is tricky:

\[ s[\text{srcip}] \leftarrow 2 ; s[\text{dstip}] = 2 \]

- How do we know whether \( s[\text{srcip}] \) is the same location as \( s[\text{dstip}] \)?
  - field-field tests to the rescue!
  - \( \text{srcip} = \text{dstip} \) test node can separate the cases

- That is why they come earlier in the tree
Distributing a State Variable

- We can partition a state variable into disjoint parts and place the partitions on different switches.

- State variable $s$ from IP addresses to $X$ can be partitioned to state variables $s_i$ from a subset of IP addresses $IP_i$.
  - $IP_i$'s are disjoint.
  - Each $s_i$ can be placed on a separate switch.
## Compiler Evaluation - Benchmarks

<table>
<thead>
<tr>
<th>Topology</th>
<th>Number</th>
<th># Switches</th>
<th># Edges</th>
<th># Ingress-Egress Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campus</td>
<td>3</td>
<td>25-100</td>
<td>100-200</td>
<td>20k-30k</td>
</tr>
<tr>
<td>ISP</td>
<td>4</td>
<td>90-160</td>
<td>300-600</td>
<td>3.5k-12k</td>
</tr>
<tr>
<td>Synthesized</td>
<td>17</td>
<td>10-180</td>
<td>40-700</td>
<td>50-15k</td>
</tr>
</tbody>
</table>
Policy 19

```plaintext
if tcp.flags=SYN & tcp-state[srcip][dstip][srcport][dstport][proto] =CLOSED then
    tcp-state[srcip][dstip][srcport][dstport][proto] ← SYN-SENT
else
    if tcp.flags=SYN-ACK & tcp-state[dstip][srcip][srcport][dstport][proto]=SYN-SENT then
        tcp-state[dstip][srcip][dstport][srcport][proto] ← SYN-RECEIVED
    else
        if tcp.flags=ACK & tcp-state[dstip][srcip][srcport][dstport][proto]=SYN-RECEIVED then
            tcp-state[dstip][srcip][dstport][srcport][proto] ← ESTABLISHED
        else
            if tcp.flags=FIN & tcp-state[dstip][srcip][srcport][dstport][proto]=ESTABLISHED then
                tcp-state[dstip][srcip][dstport][srcport][proto] ← FIN-WAIT
            else
                if tcp.flags=FIN-ACK & tcp-state[dstip][srcip][srcport][dstport][proto]=FIN-WAIT then
                    tcp-state[dstip][srcip][dstport][srcport][proto] ← FIN-WAIT2
                else
                    if tcp.flags=ACK & tcp-state[dstip][srcip][srcport][dstport][proto]=FIN-WAIT2 then
                        tcp-state[dstip][srcip][dstport][srcport][proto] ← CLOSED
                    else
                        if tcp.flags=RST & tcp-state[dstip][srcip][srcport][dstport][proto]=ESTABLISHED then
                            tcp-state[dstip][srcip][dstport][srcport][proto] ← CLOSED
                        else
                            tcp-state[dstip][srcip][dstport][srcport][proto] ← ESTABLISHED +
                            tcp-state[dstip][srcip][dstport][srcport][proto] ← ESTABLISHED
```