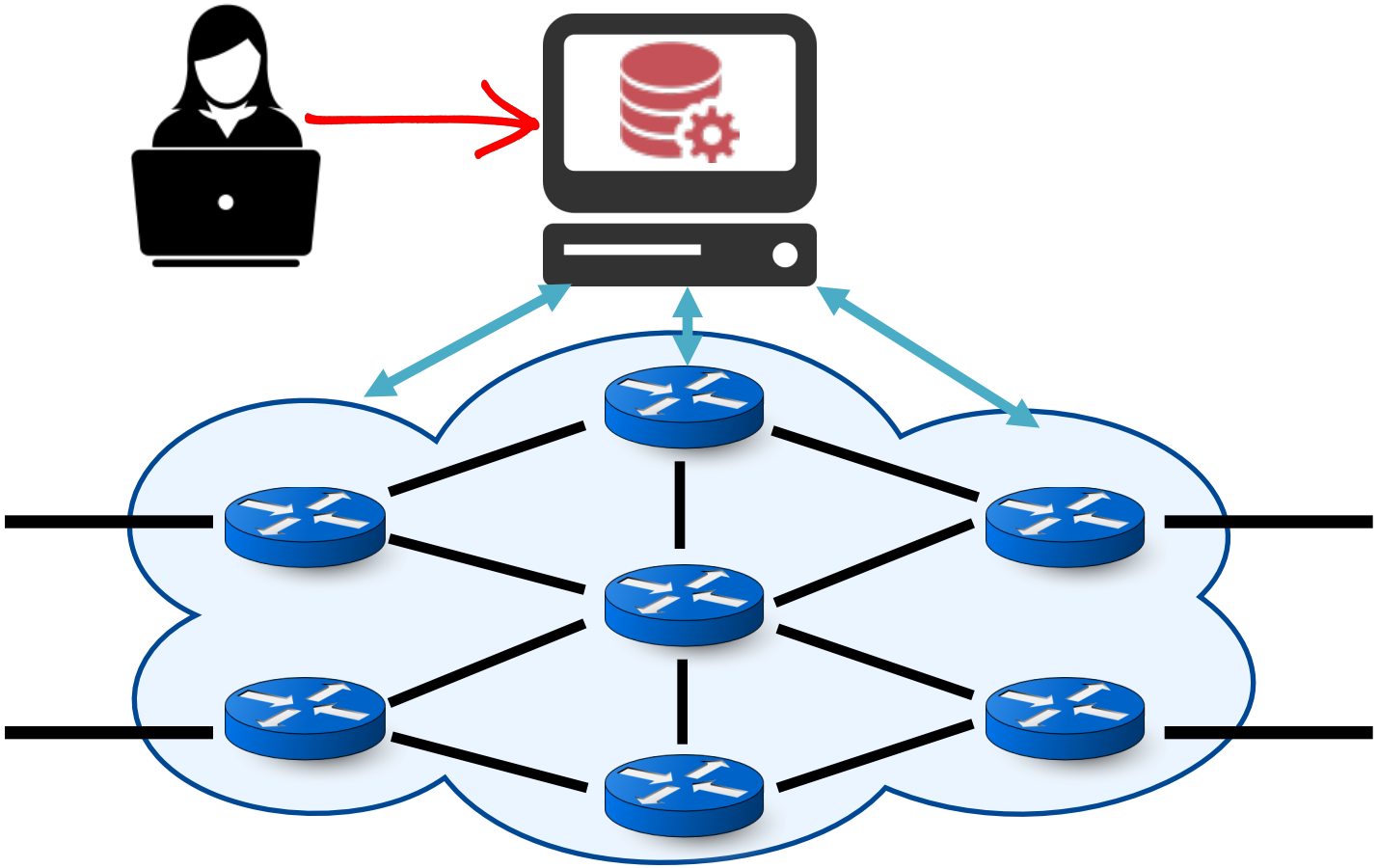
SNAP: Stateful Network-Wide Abstractions for Packet Processing

Mina Tahmasbi Arashloo[1], **Yaron Koral**[1], Michael Greenberg[2],
Jennifer Rexford[1], and David Walker[1]

[1] Princeton University, [2] Pomona College

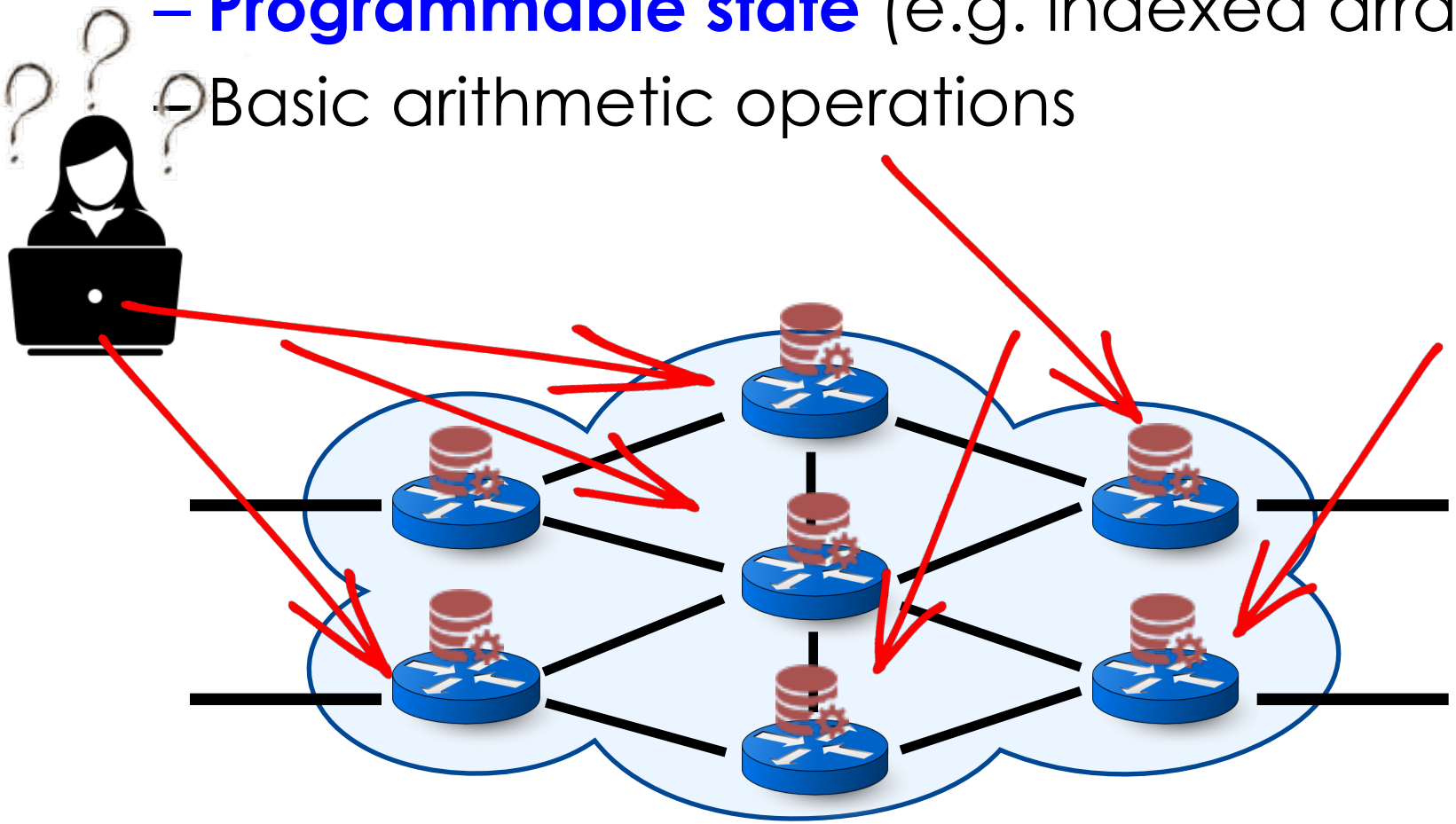
Early SDN Switch Interfaces

- Manipulate packet forwarding rules
- Read predefined set of counters

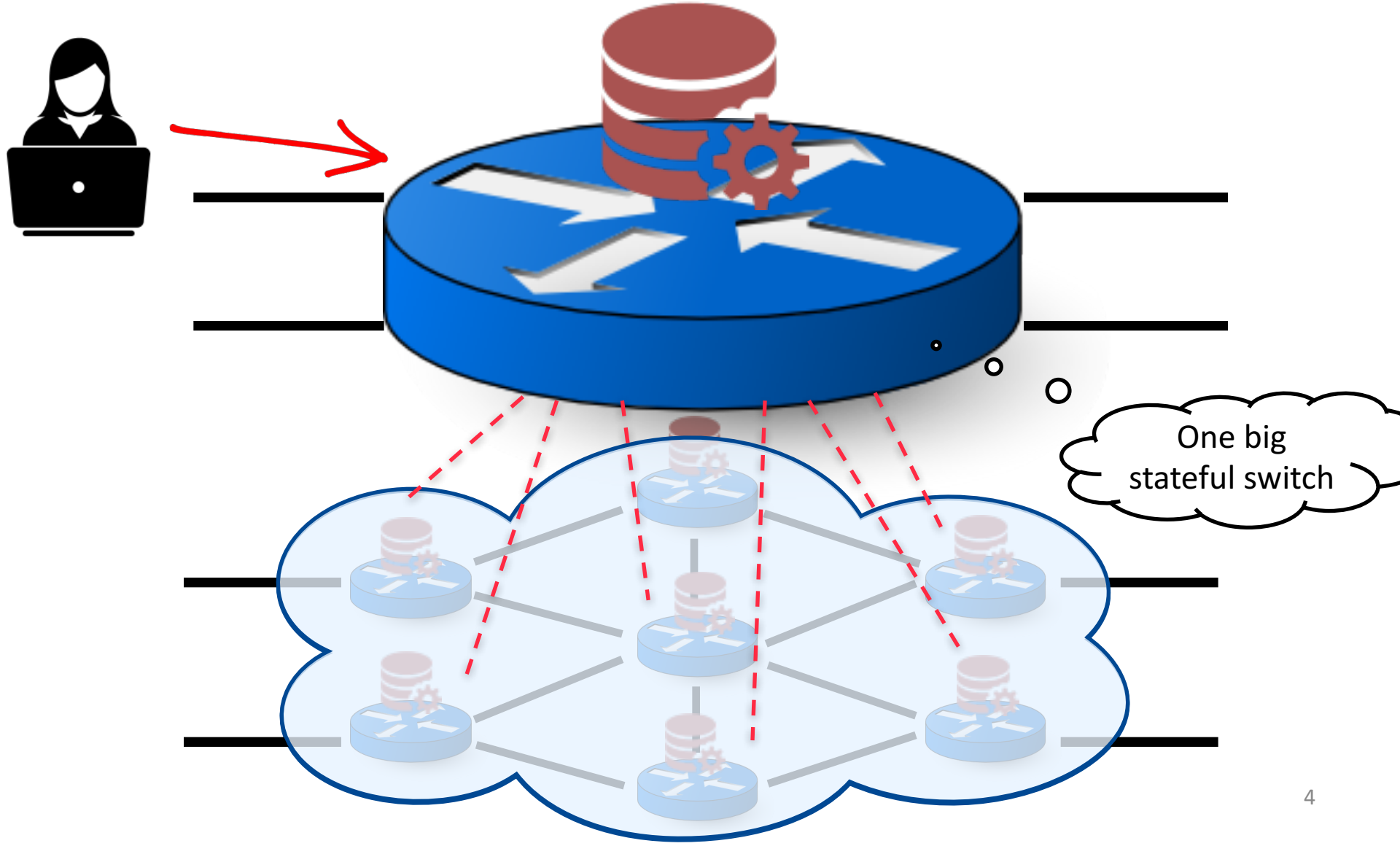


Programmable Switch Interfaces

- P4, OpenState, Open vSwitch, …
  - **Programmable state** (e.g. indexed arrays)
  - Basic arithmetic operations



SNAP: Stateful Network Wide Programming Language



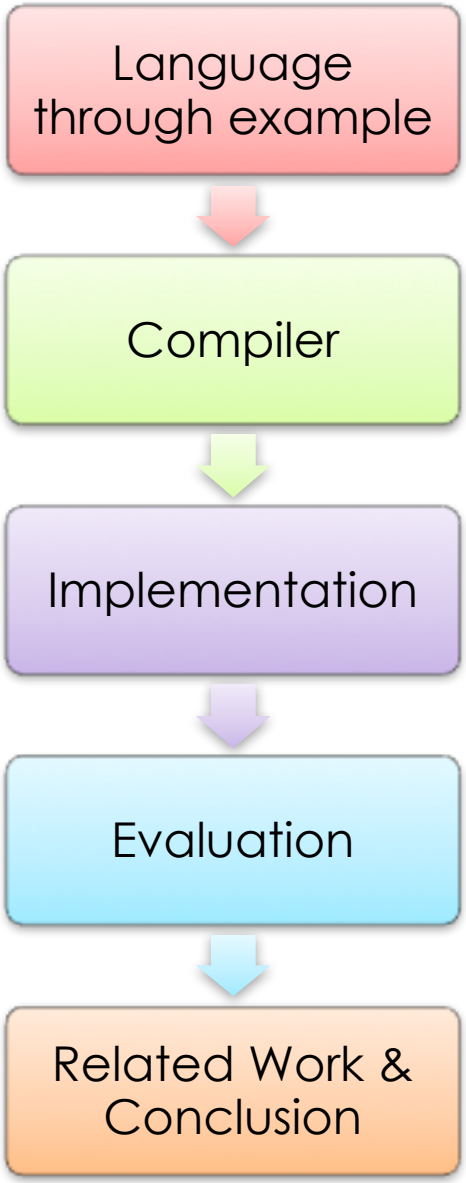
SNAP Contributions

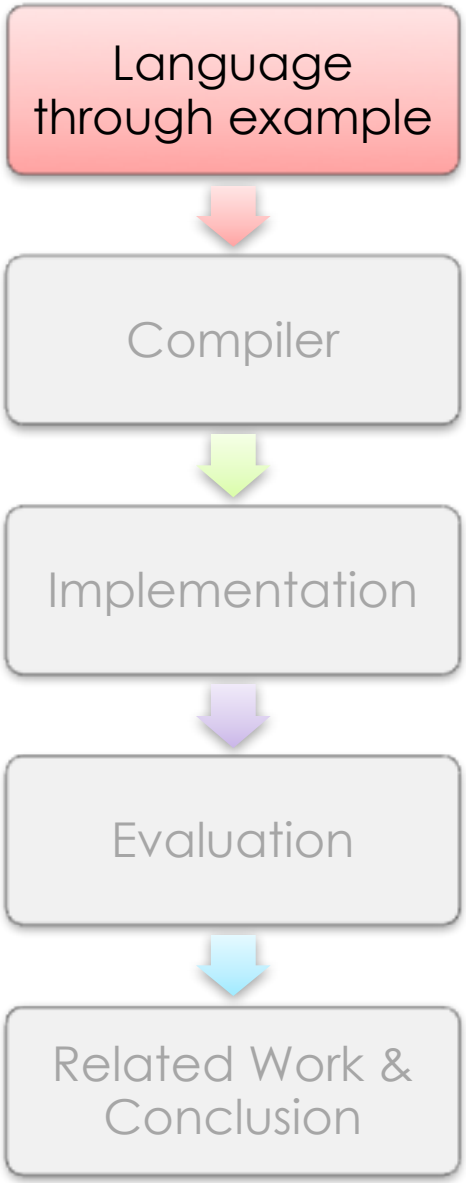
Modular Stateful Language

One Big Stateful Switch

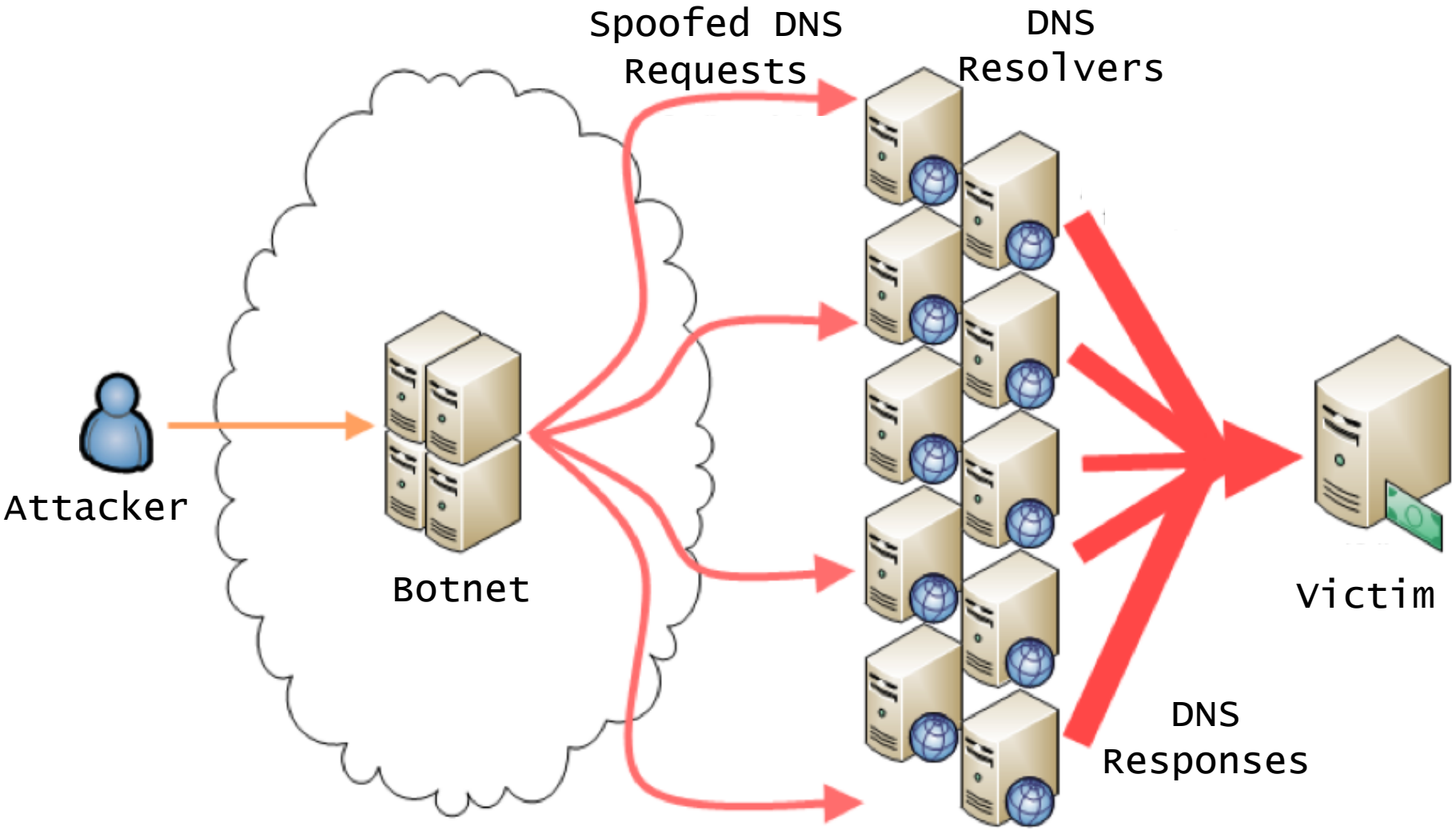
Placement + Routing

Talk Outline

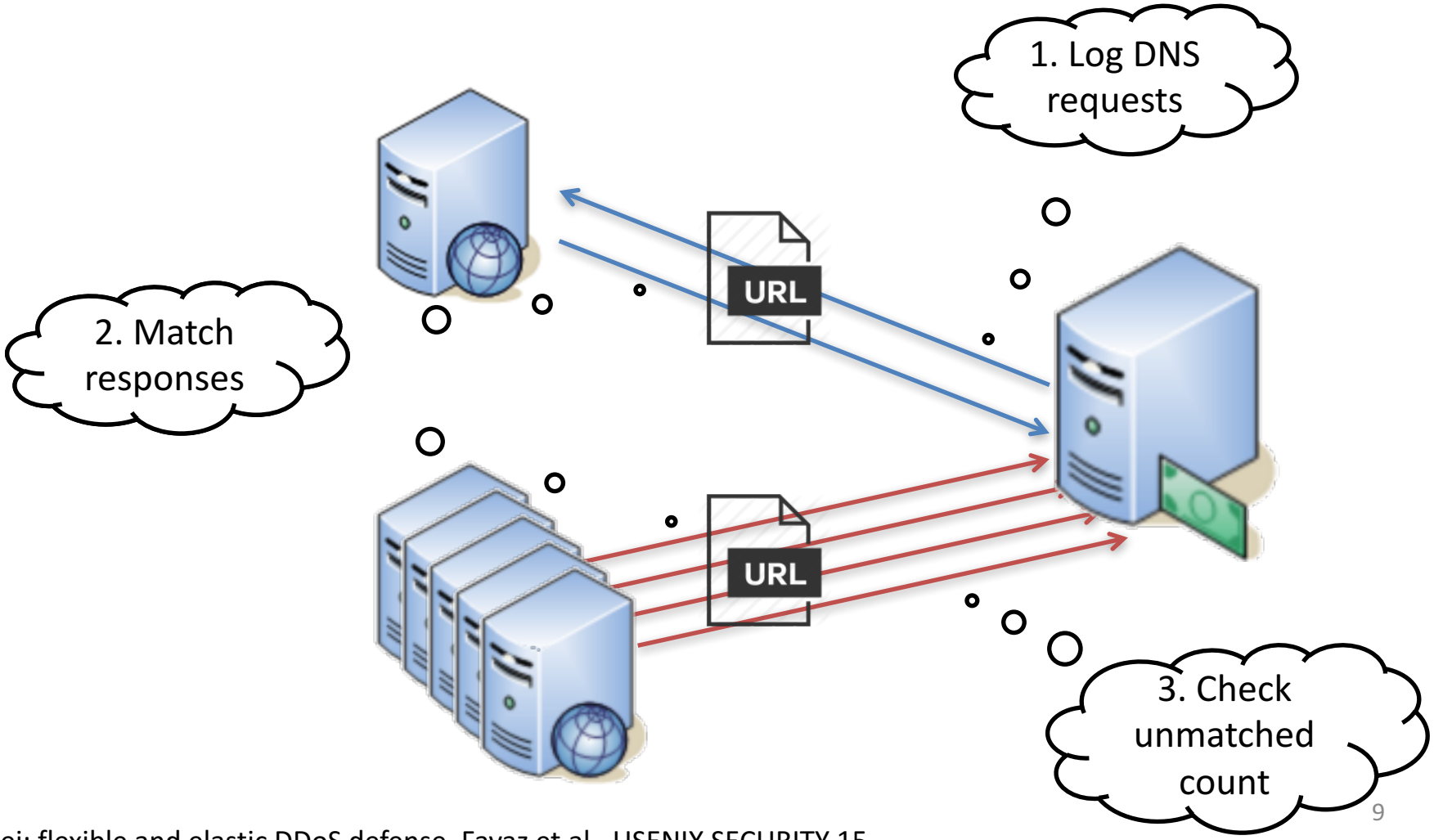



Language through example
Compiler
Implementation
Evaluation
Related Work & Conclusion

Example - DNS Reflection Attacks



Detecting DNS Reflection Attacks



Bohatei: flexible and elastic DDoS defense, Fayaz et.al., USENIX SECURITY 15

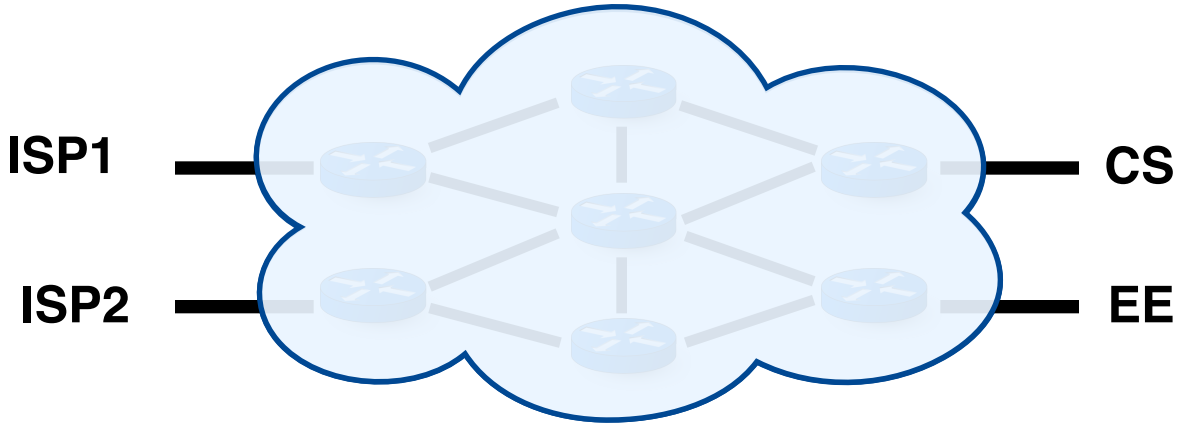
DNS Reflection Detection in SNAP

```
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

```
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
```



Single Network Policy

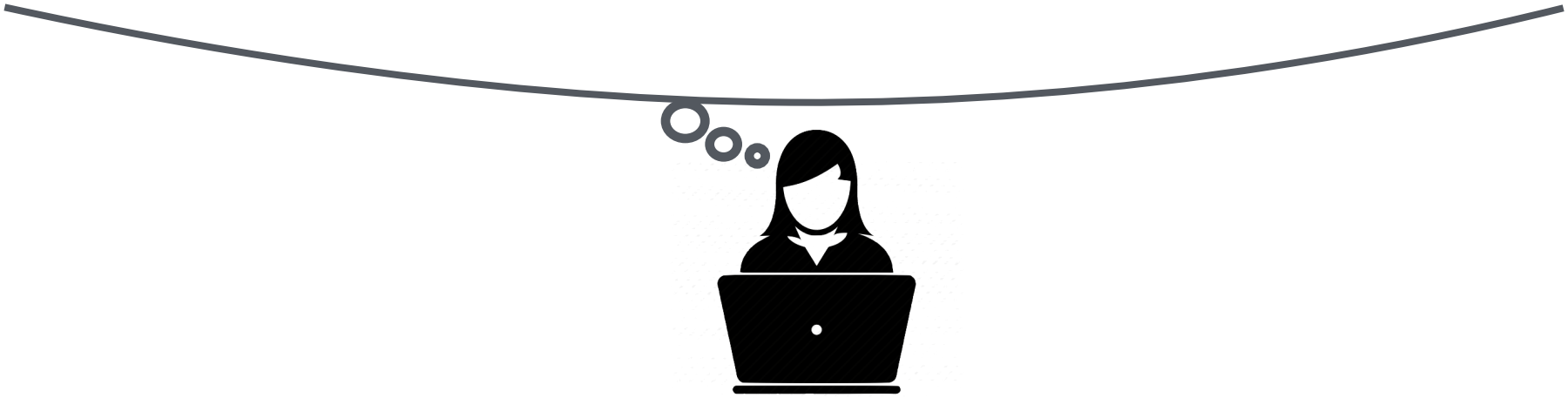
DNS Reflection Detection

```
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

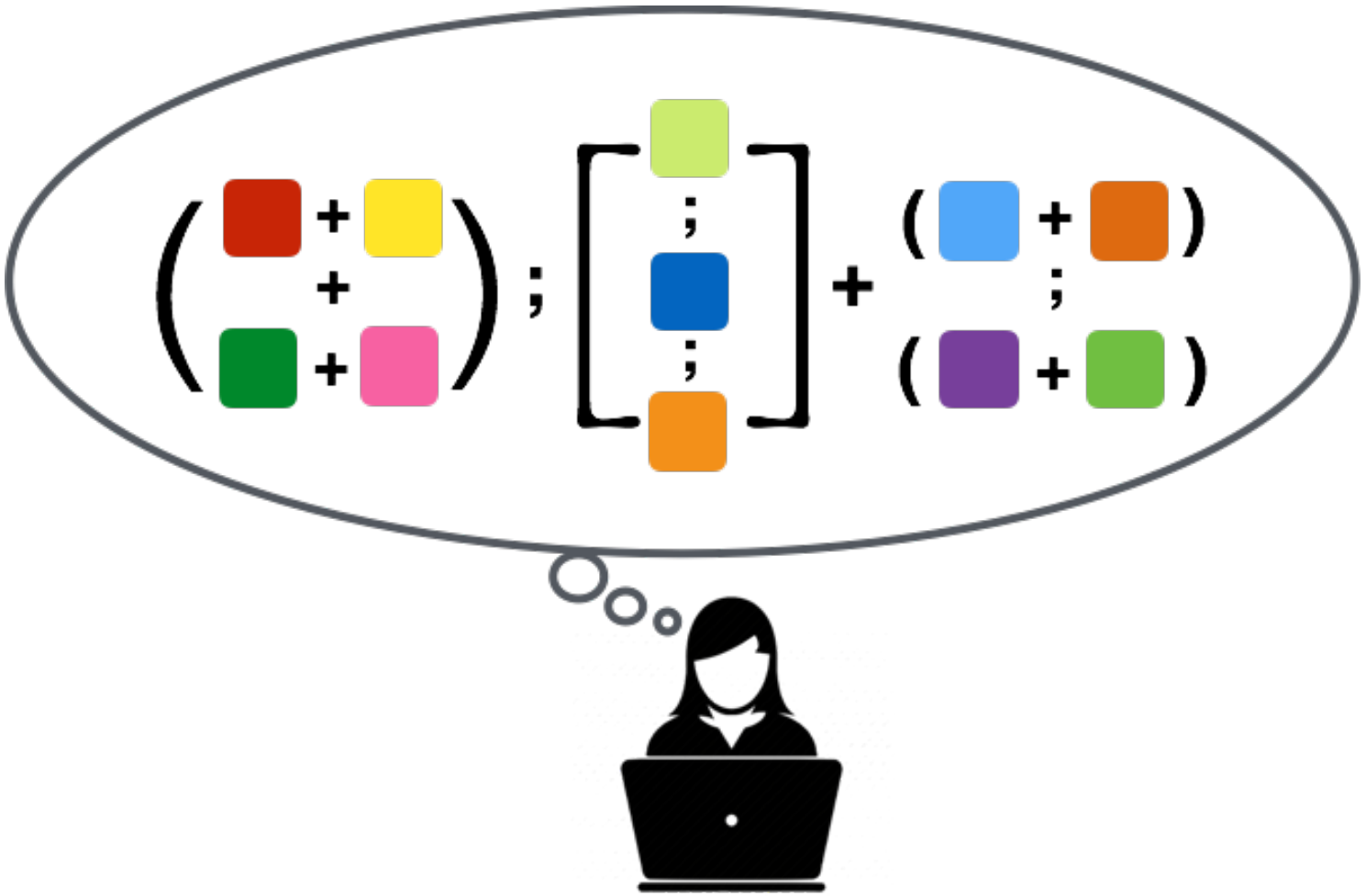
```
if dstip = CSNET then outport ← CS
else if dstip = EENET then outport ← EE
else if dstip = ISP1NET then outport ← ISP1
else if dstip = ISP2NET then outport ← ISP2
else drop
```

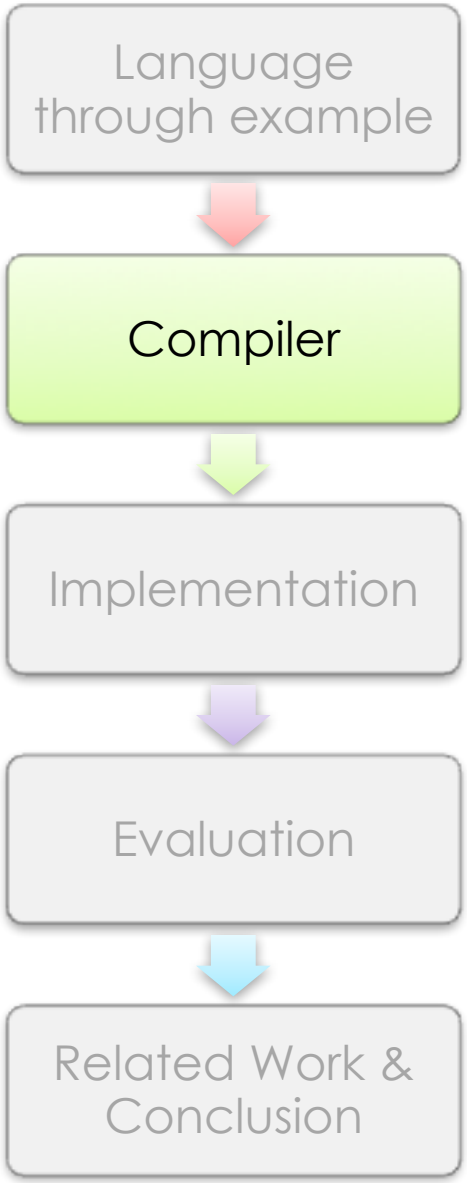


SNAP Applications

Source	Application
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




Language through example
Compiler
Implementation
Evaluation
Related Work & Conclusion

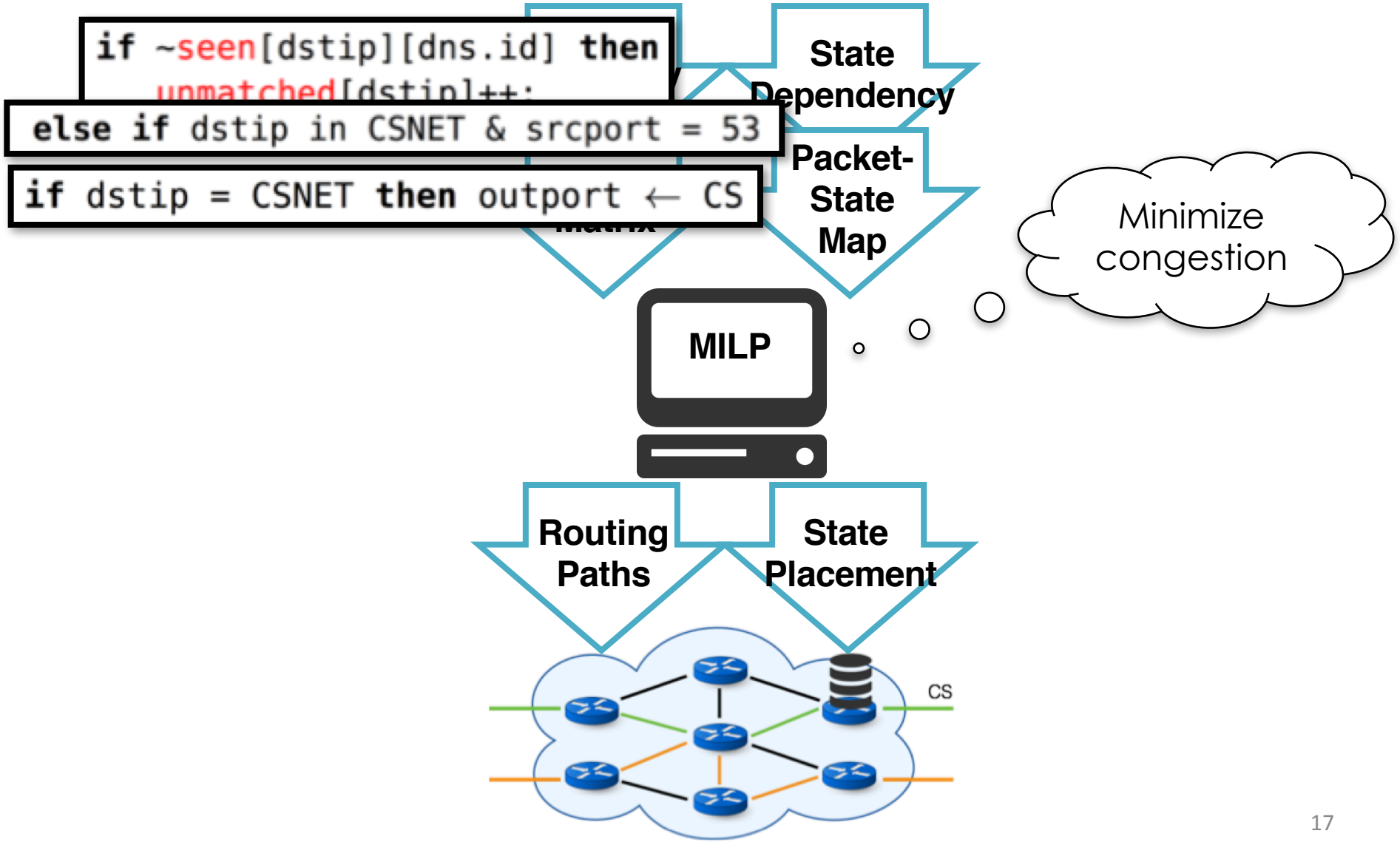
SNAP Compiler

Where to place state variables

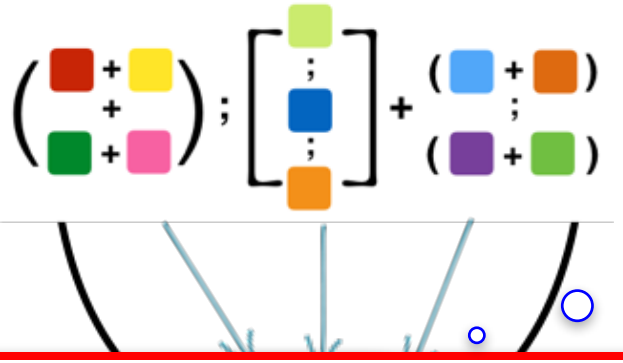
How to forward packets through them

Routing + Placement Jointly

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

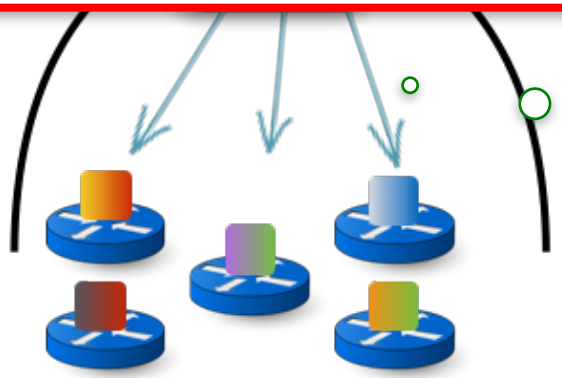


Intermediate Representation (IR)



Maintain all programs in a single data structure

Composable and easily partitioned IR

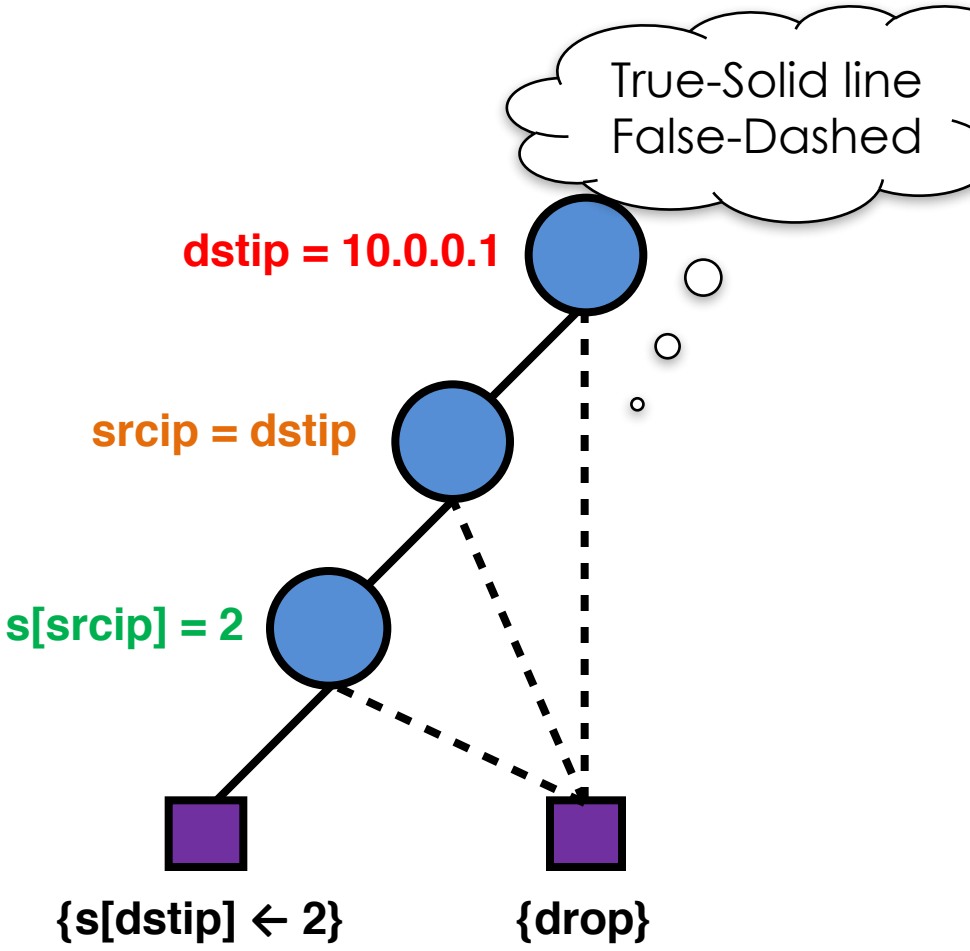


Distribute the program to switches

xFDDs: Extended Forwarding Decision Diagrams

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

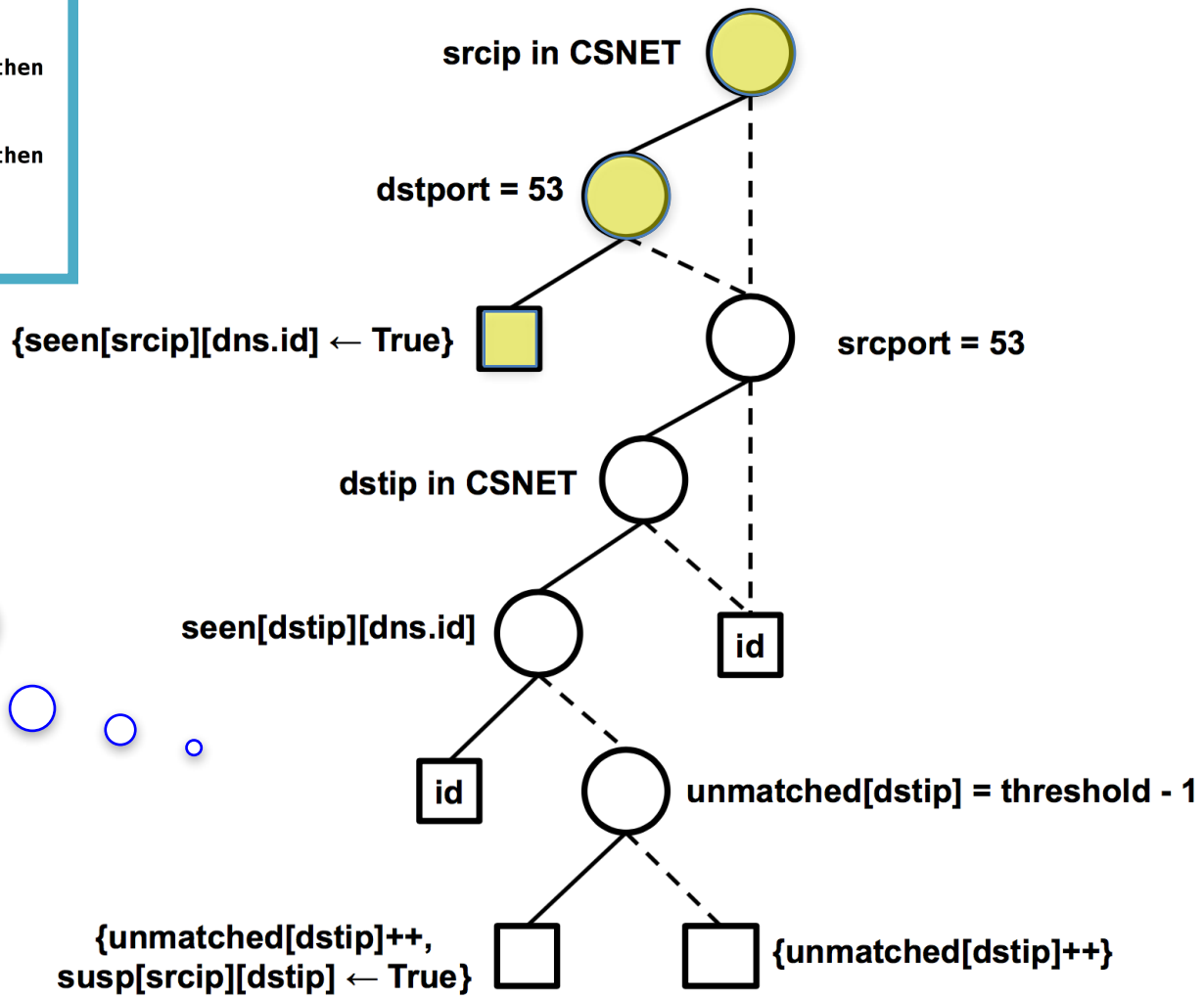
- Three kinds of tests
  - field = value
  - $field_1 = field_2$
  - state_var[idx] = val



xFDD for DNS Reflection Detection

```
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
```

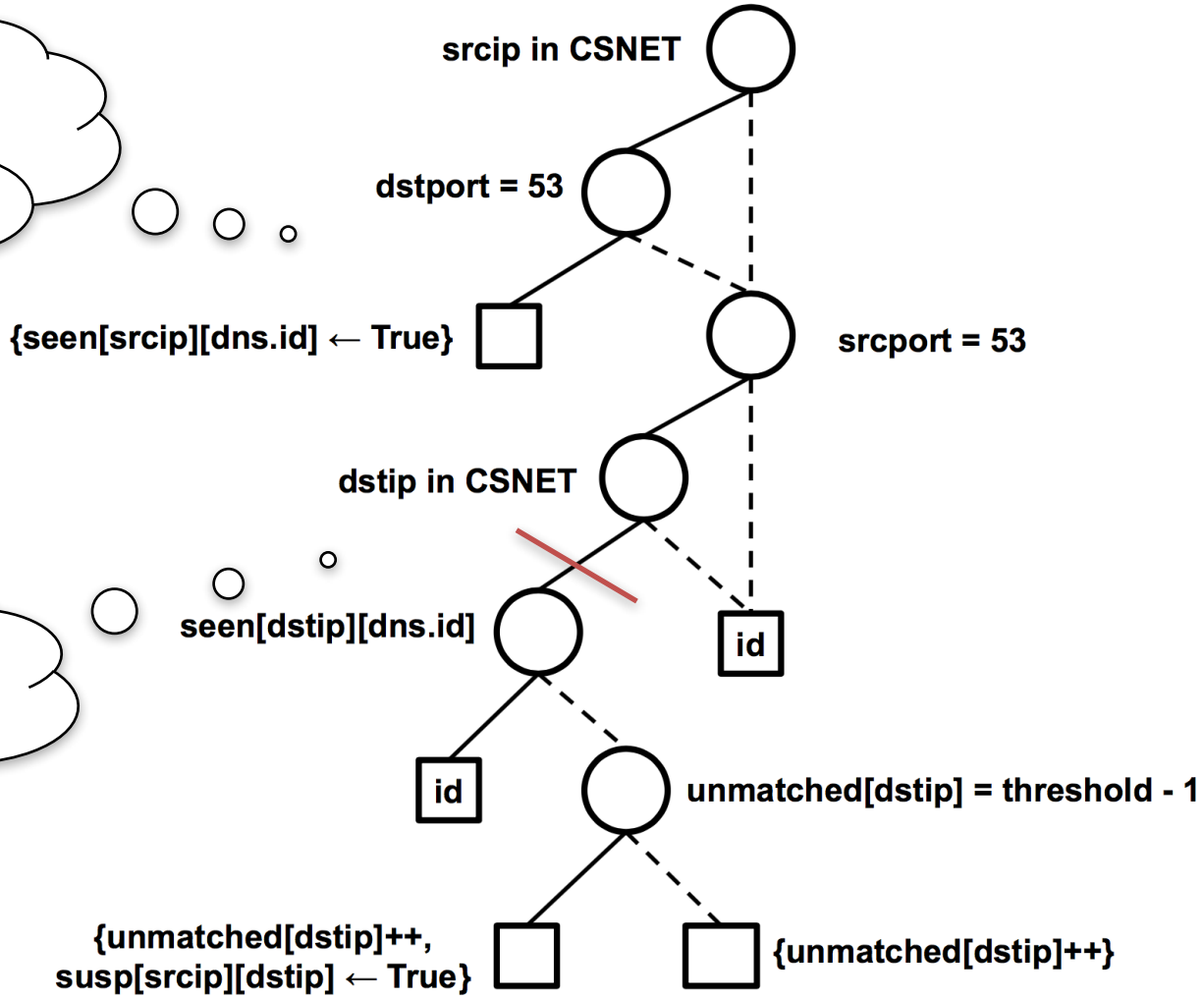
Maintain all programs in a single data structure



xFDD for DNS Reflection Detection

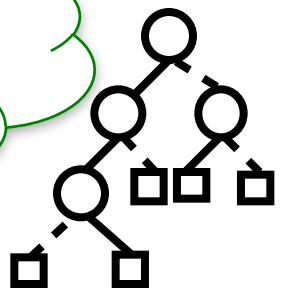
Fixes the order in which programs access state.

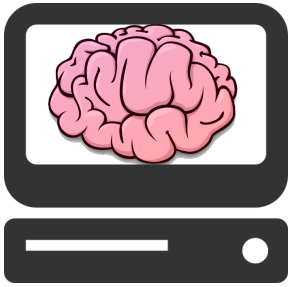
We could distribute the programs by placing cuts

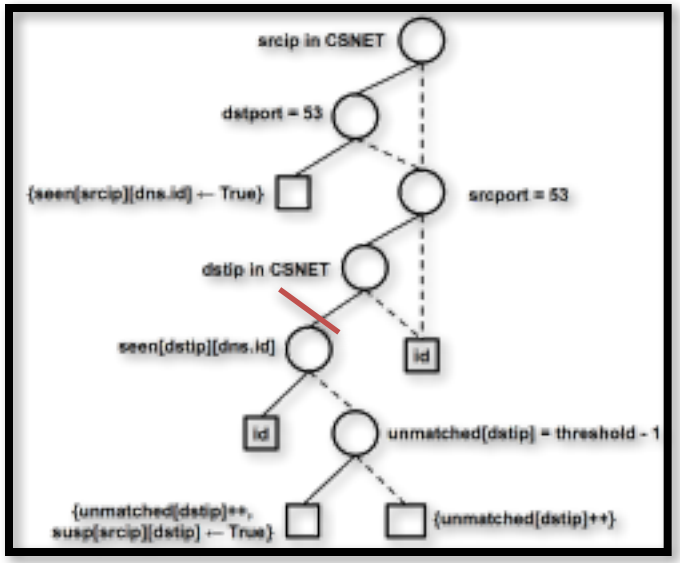


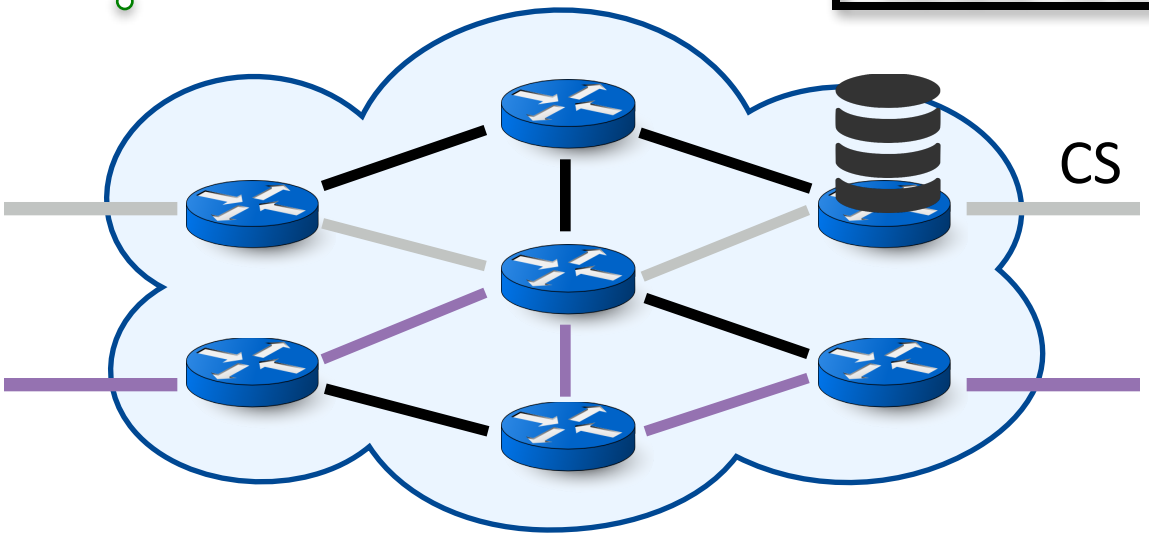
Partitioning to Sub-Programs

Distribute the program to switches



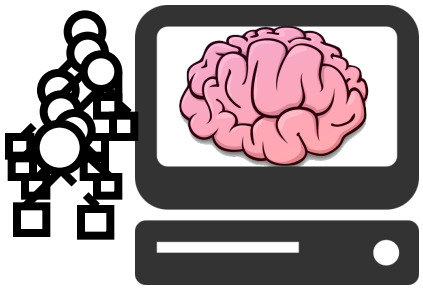


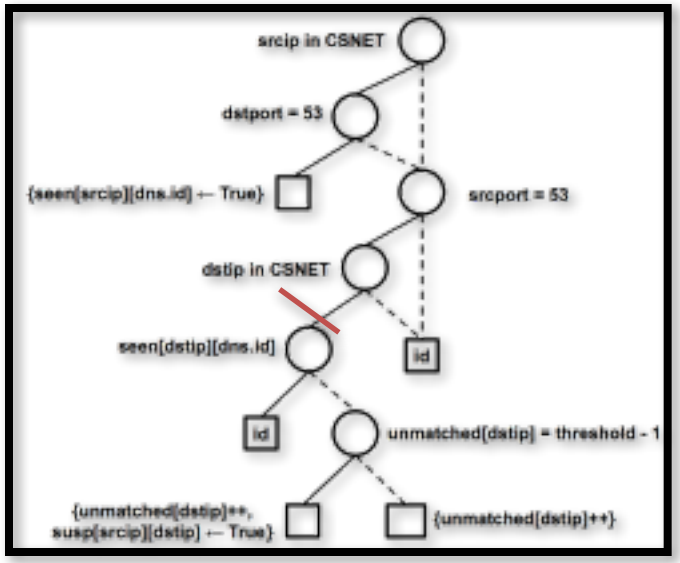


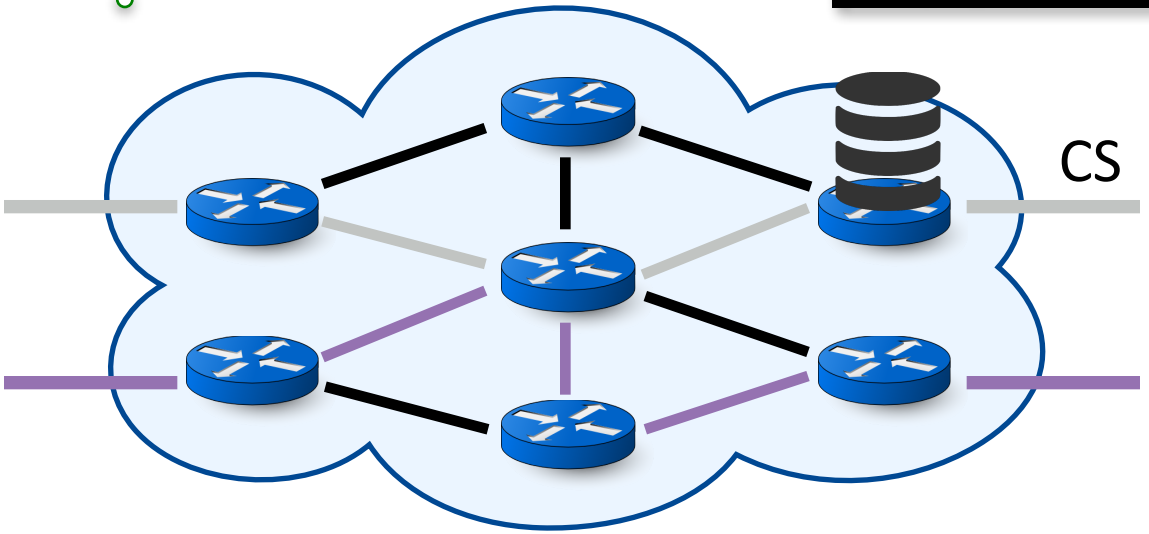


Partitioning to Sub-Programs

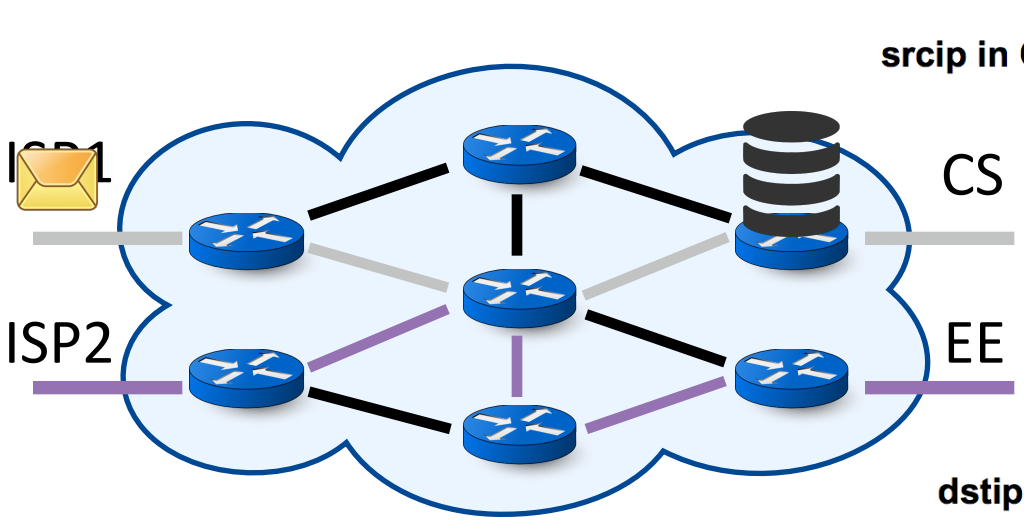
Distribute the program to switches

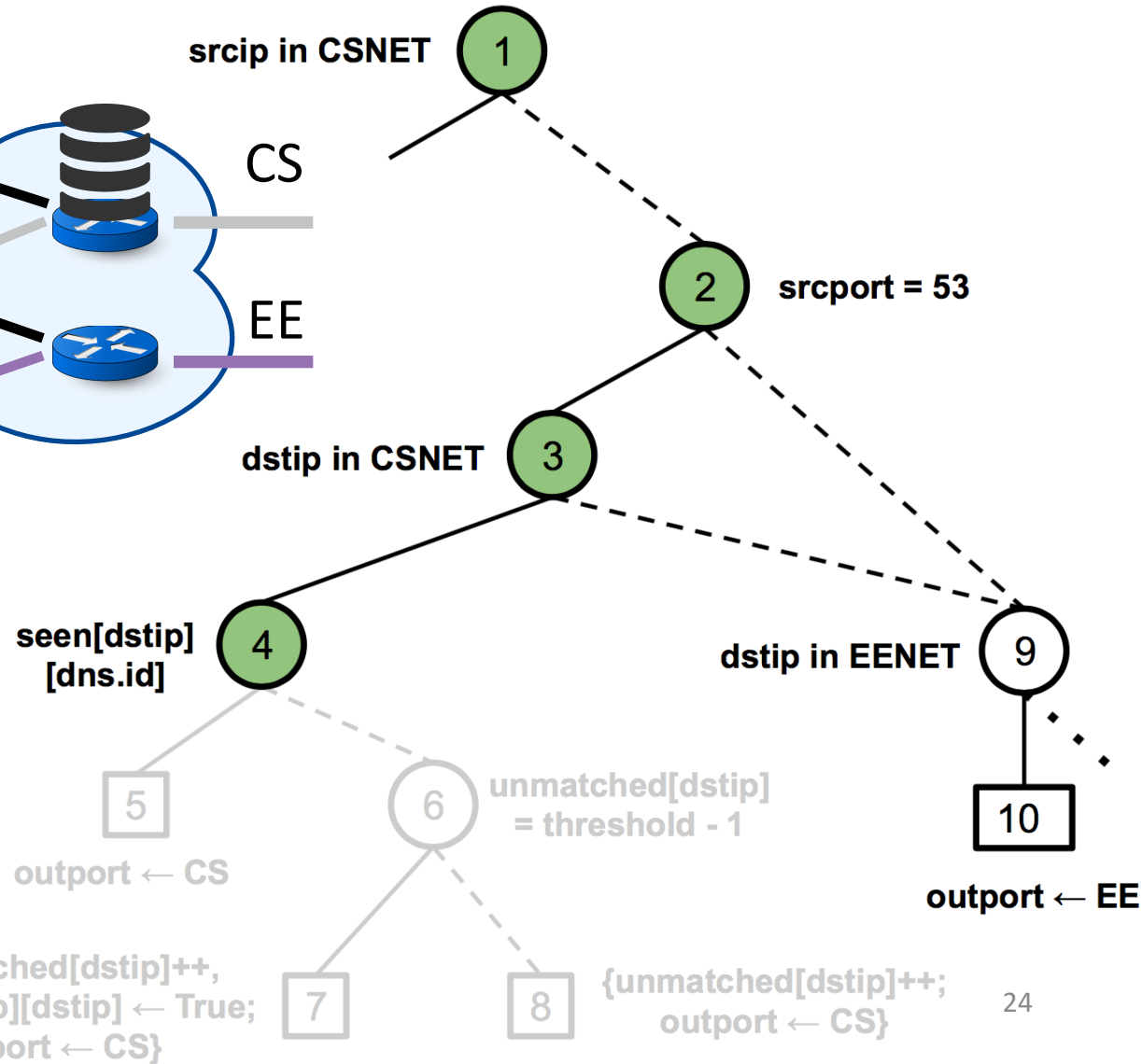




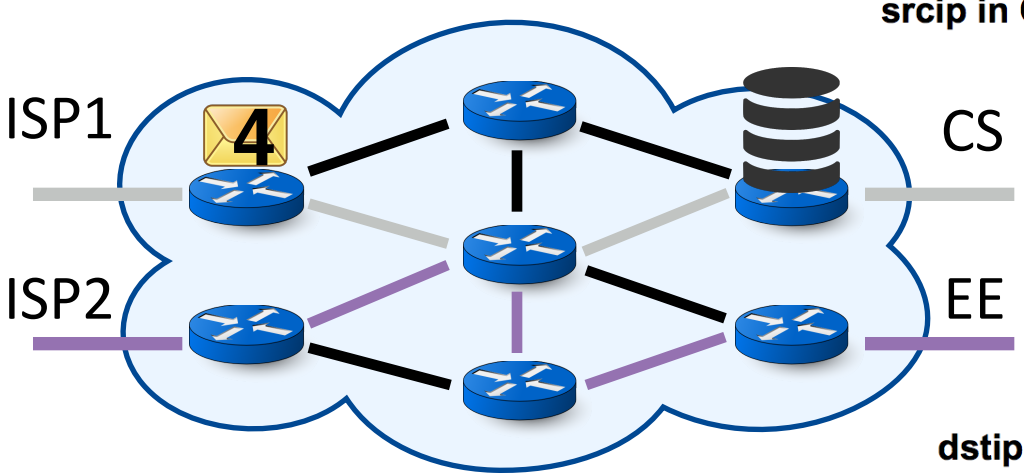


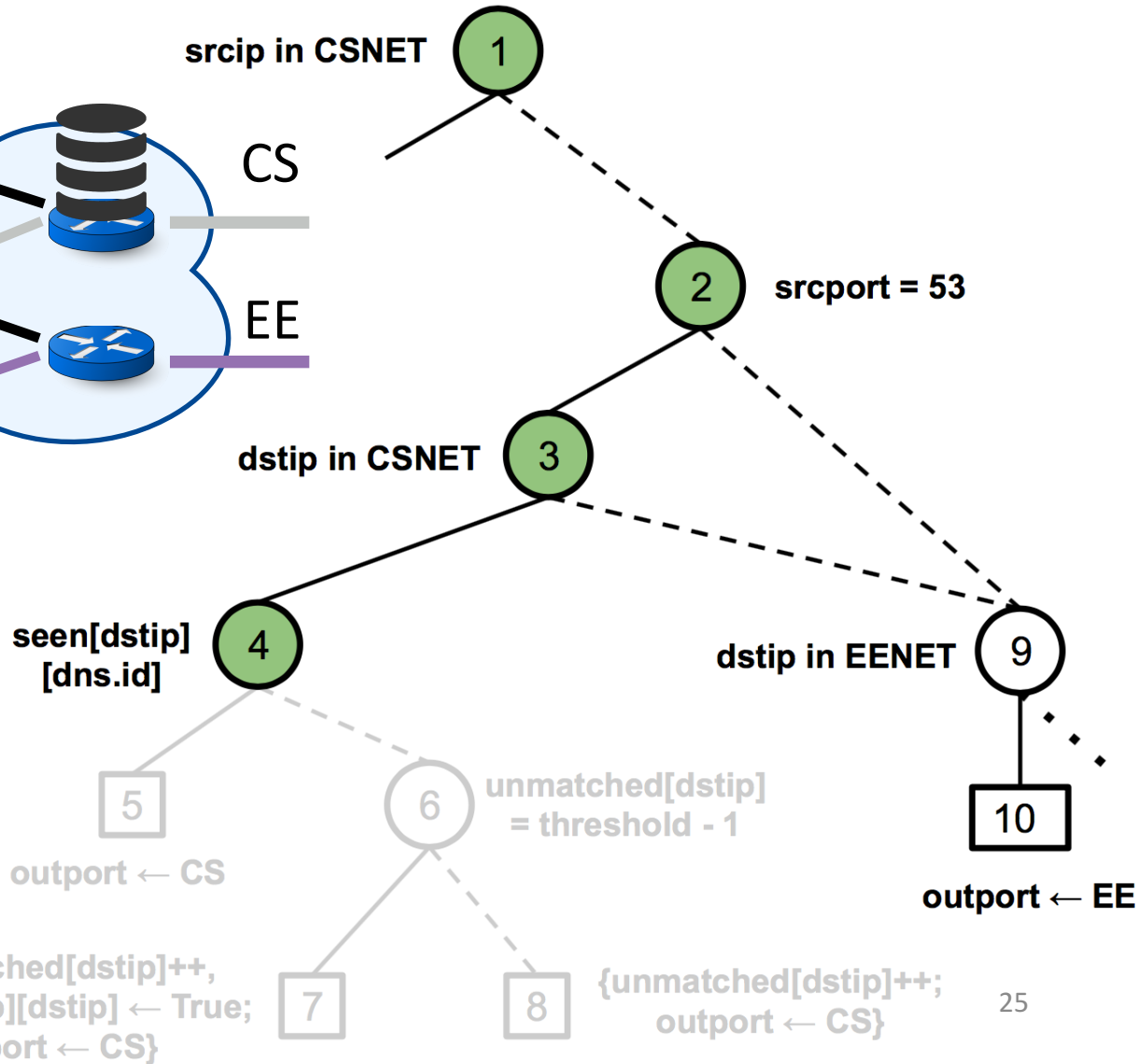
Putting It All Together



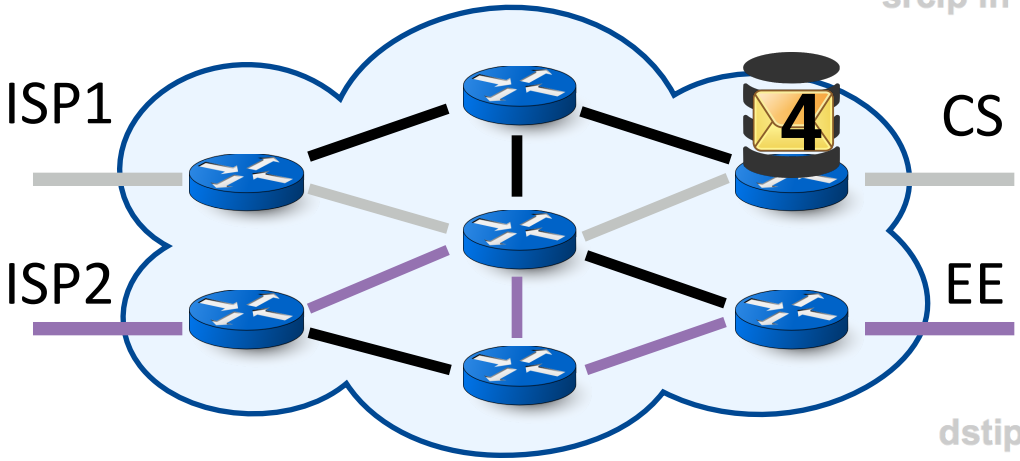


Putting It All Together

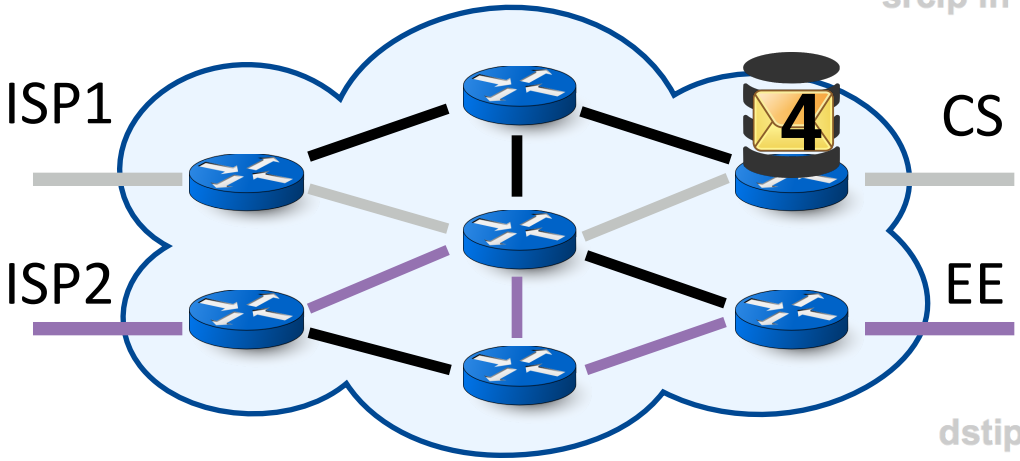


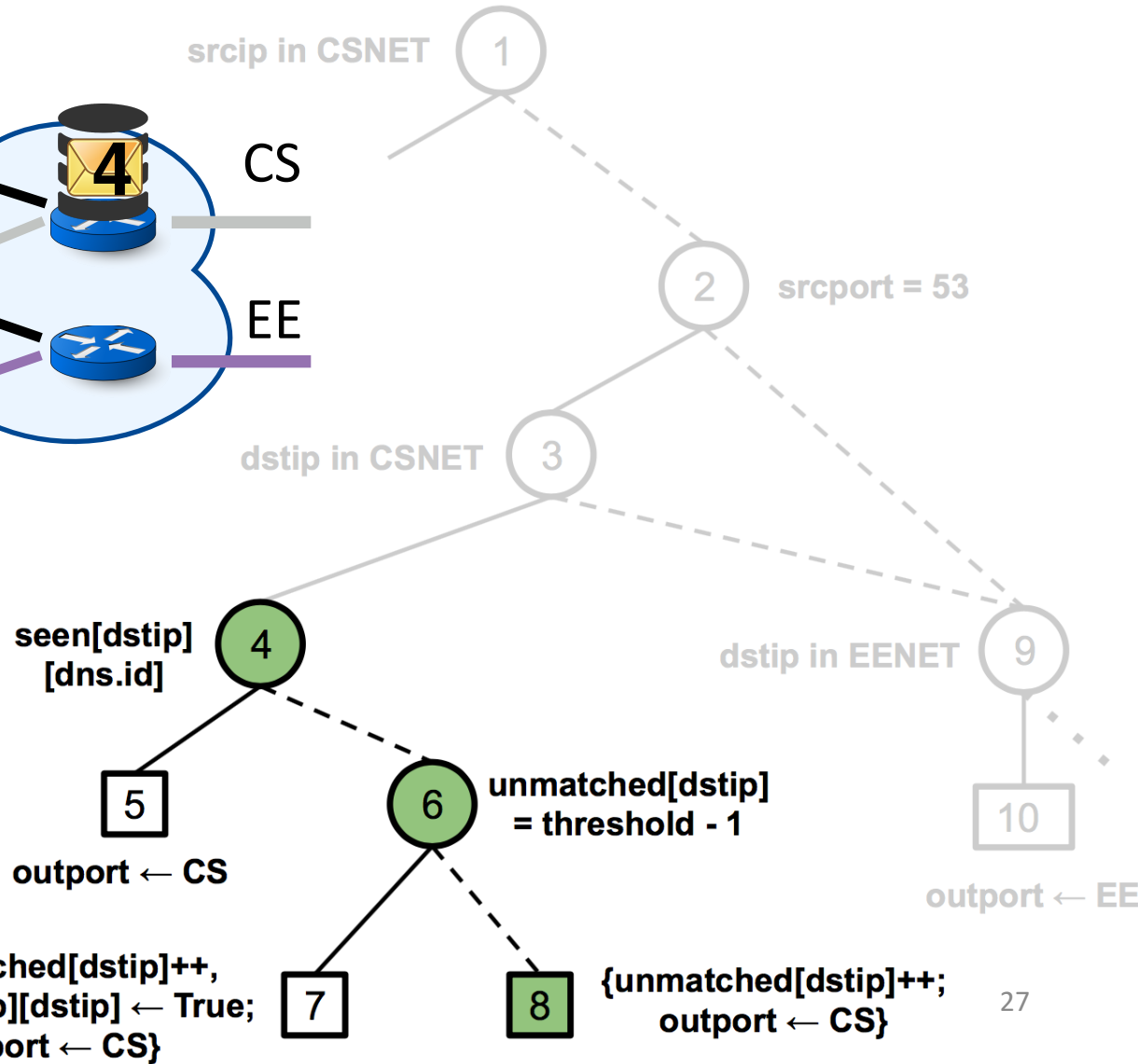


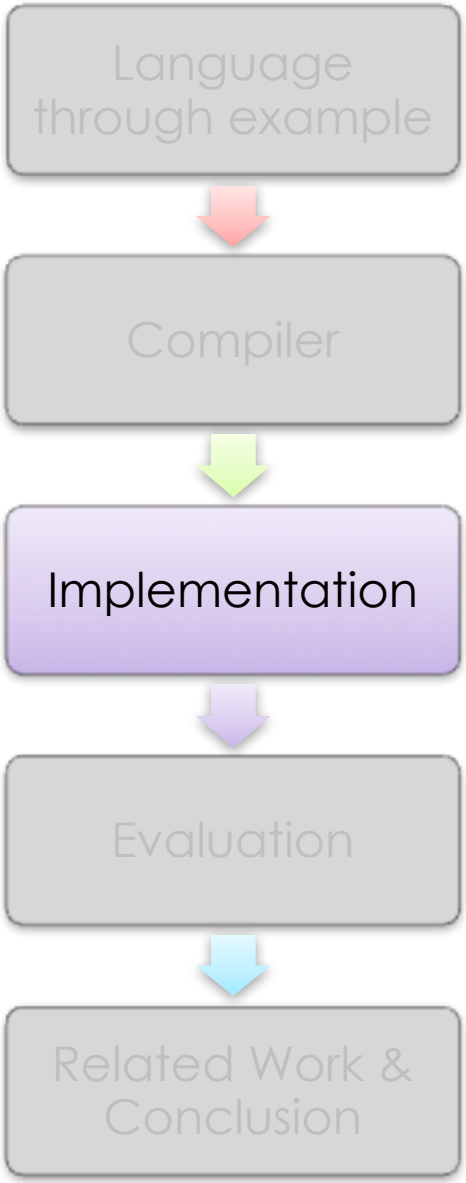
Putting It All Together



Putting It All Together





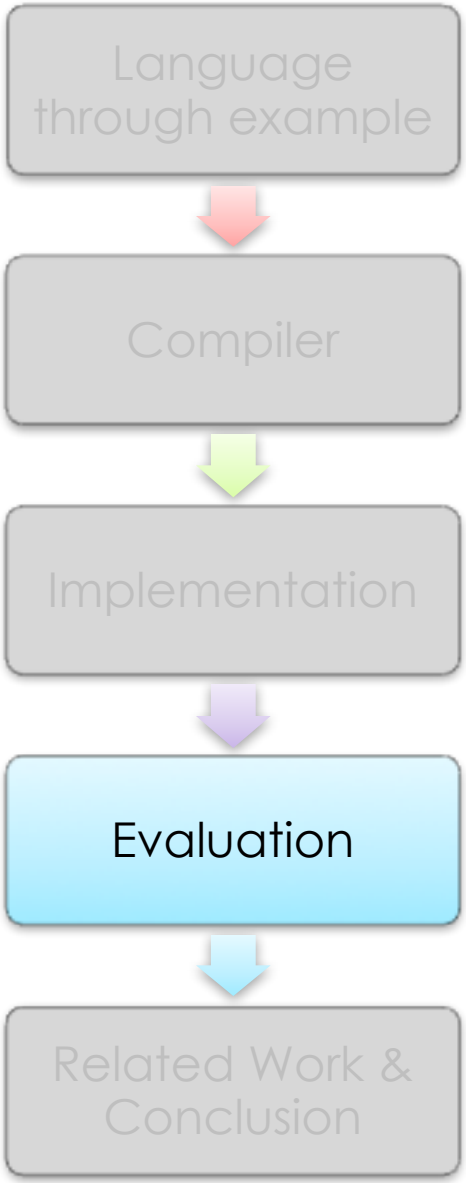

Language through example
Compiler
Implementation
Evaluation
Related Work & Conclusion

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

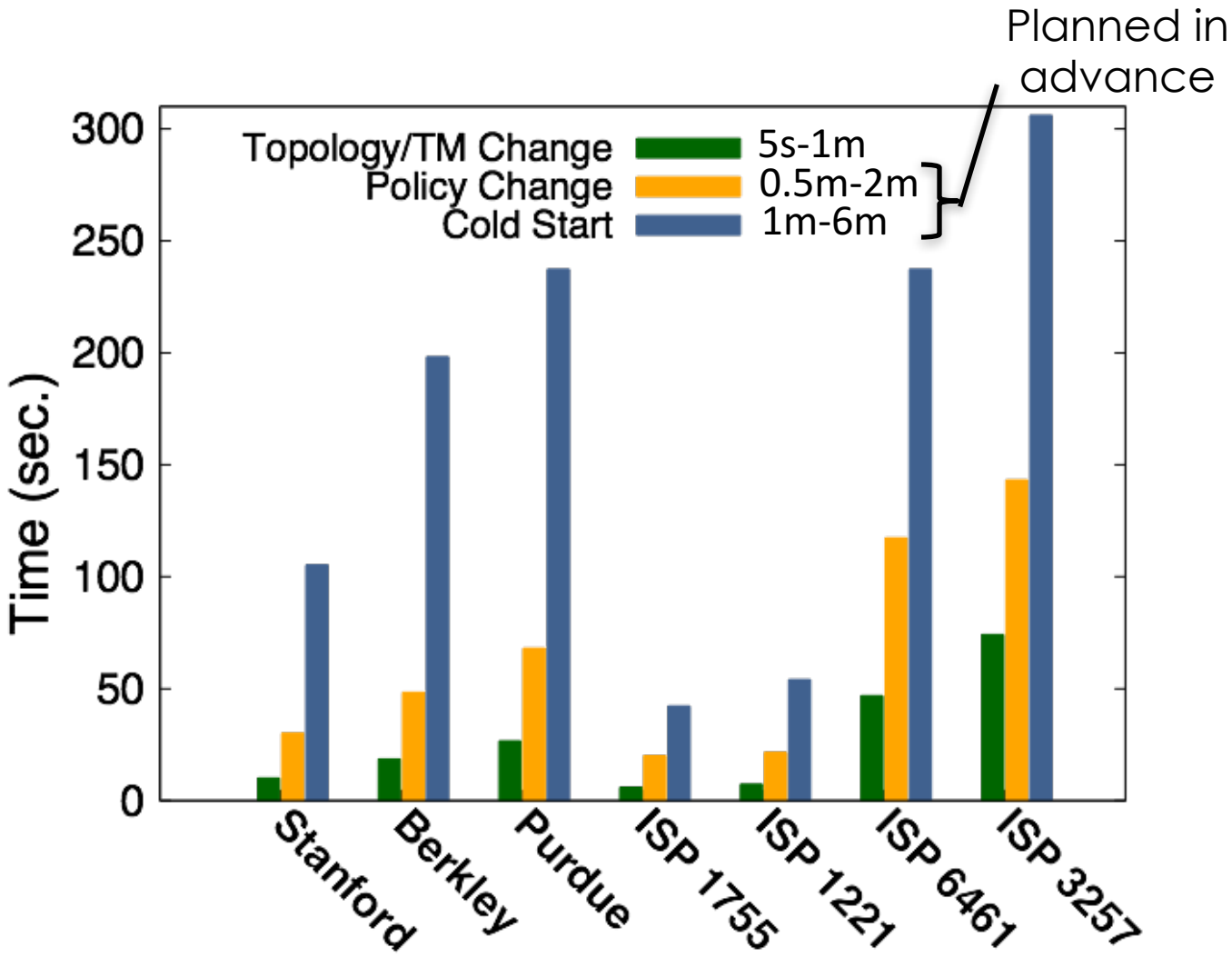


Compiler Evaluation

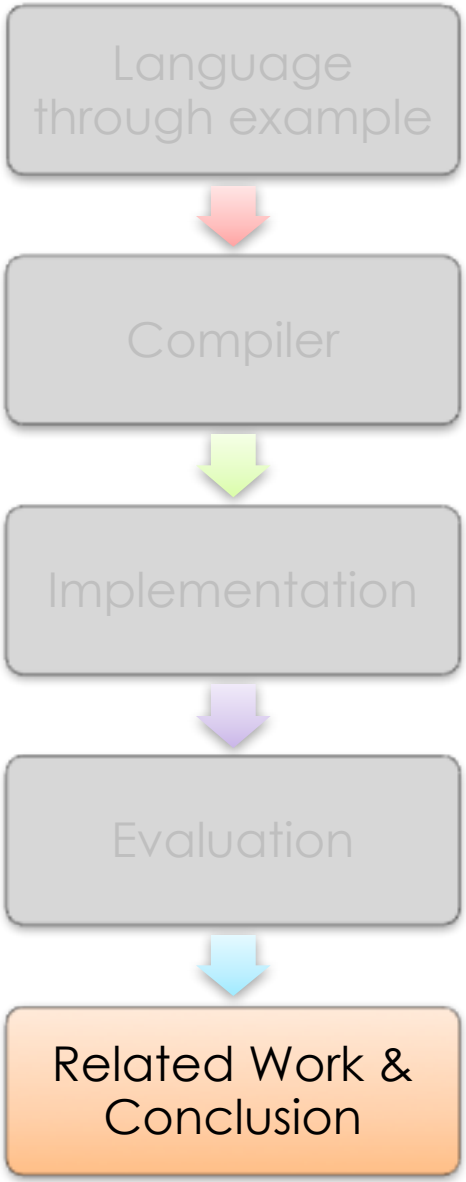
- 7 campus and ISP topologies
- Order of 100s of switches and links

- Scenarios
  - Cold start (freq. weeks)
  - Policy change (freq. days)
  - Topology/TM change (freq. minutes)

Compiler Evaluation - Results



Talk Outline



Related Work

	Programmable State	Network-Wide	Data Plane State	Joint Placement and Routing	
SNAP (SIGCOMM'16)	✓	✓	✓	✓	Stateful languages
Stateful NetKAT (PLDI'16)	✓	✓	✓	-	
Kinetic (NSDI'15)	✓	✓	-	-	
Domino (SIGCOMM'16)	✓	-	✓	-	Switch level mechanisms
OpenState (SIGCOMM-CCR'14)	✓	-	✓	-	
FAST (HotSDN'14)	✓	-	✓	-	
Merlin (CONEXT'14)	-	✓	✓	-	Optimizing placement & routing
Slick (SOSR'15)	✓	✓	-	-	
Stratos (TR'13)	-	✓	✓	-	

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