Compiling Path Queries in Software-Defined Networks

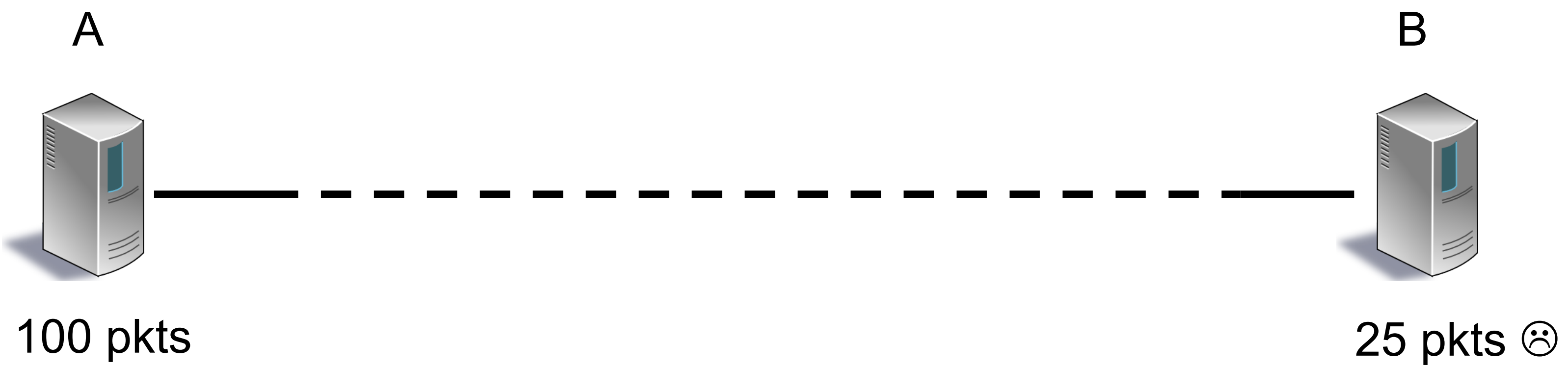
Srinivas Narayana

Jennifer Rexford and David Walker

Princeton University

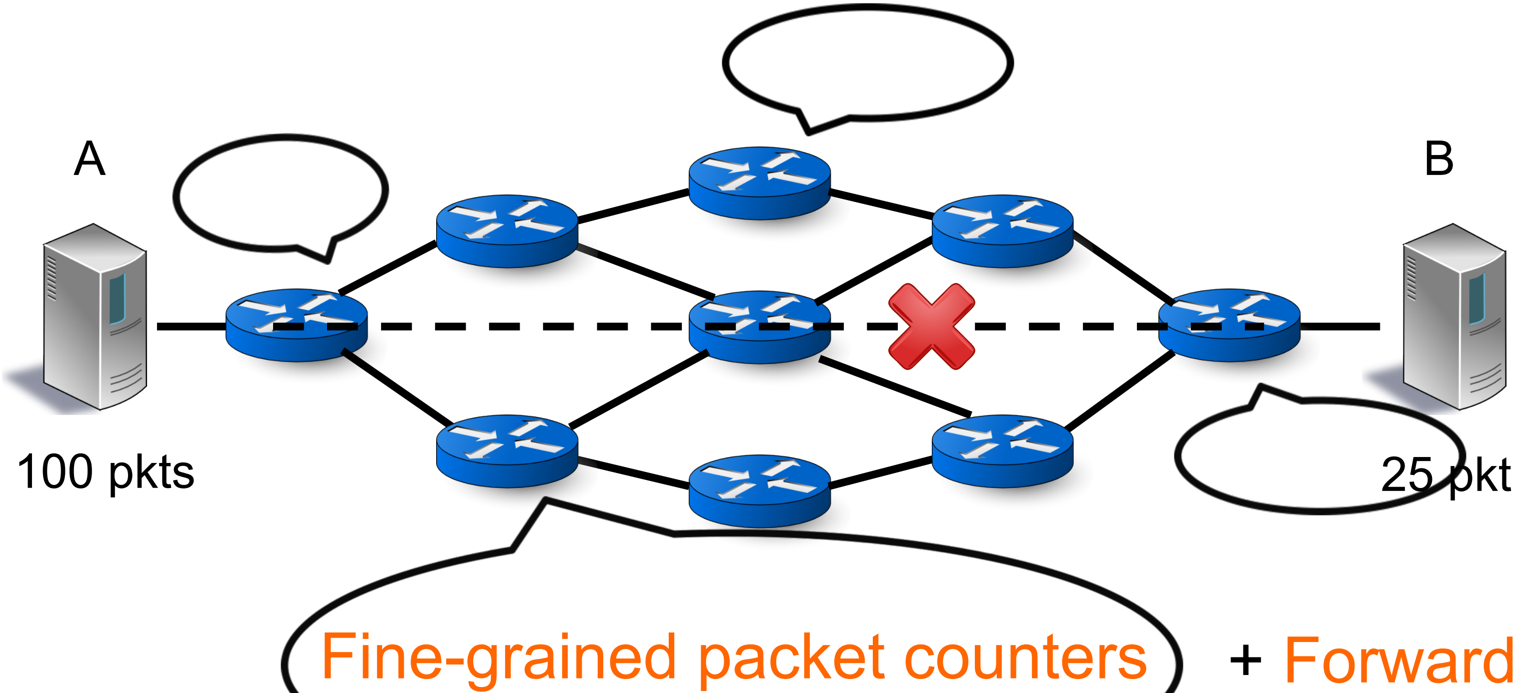
Where’s the packet loss?

Faulty network device(s) along the way. But where?



Where's the packet loss?

Solution idea: Check how far packets get from A to B before being dropped somewhere.



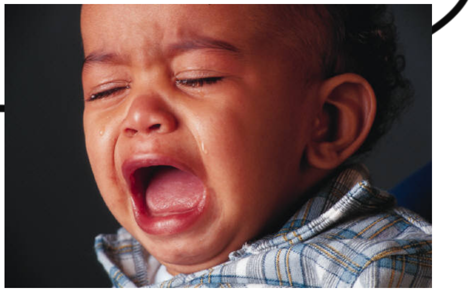
A

B

100 pkts

25 pkts ☹

Fine-grained packet counters

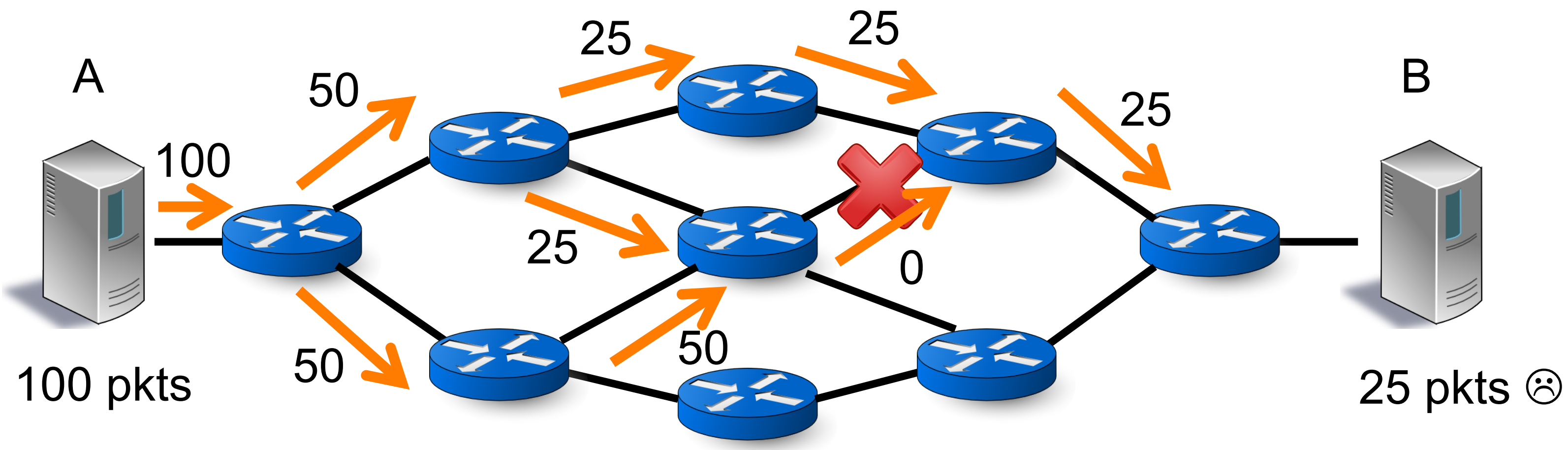


+ Forwarding

Complex policies

Where’s the packet loss?

Instead: nice to get A → B packet counts each step along paths where A → B traffic flows

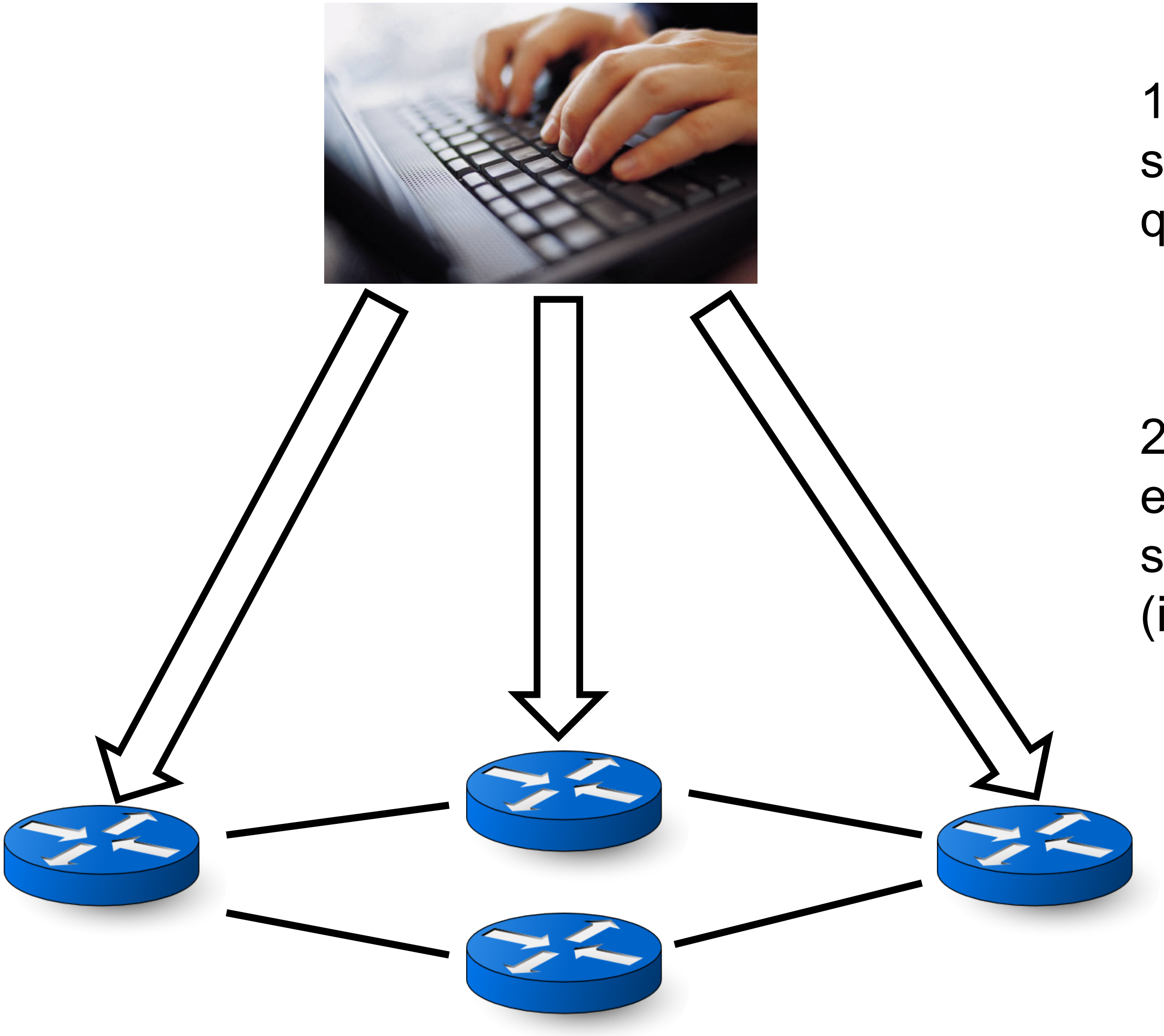


Wouldn’t it be nice to ask questions about packet paths in a network?

Problem: we only observe a given packet *independently* at different switches.

We've designed a path query system that analyzes packet paths directly in the data plane.

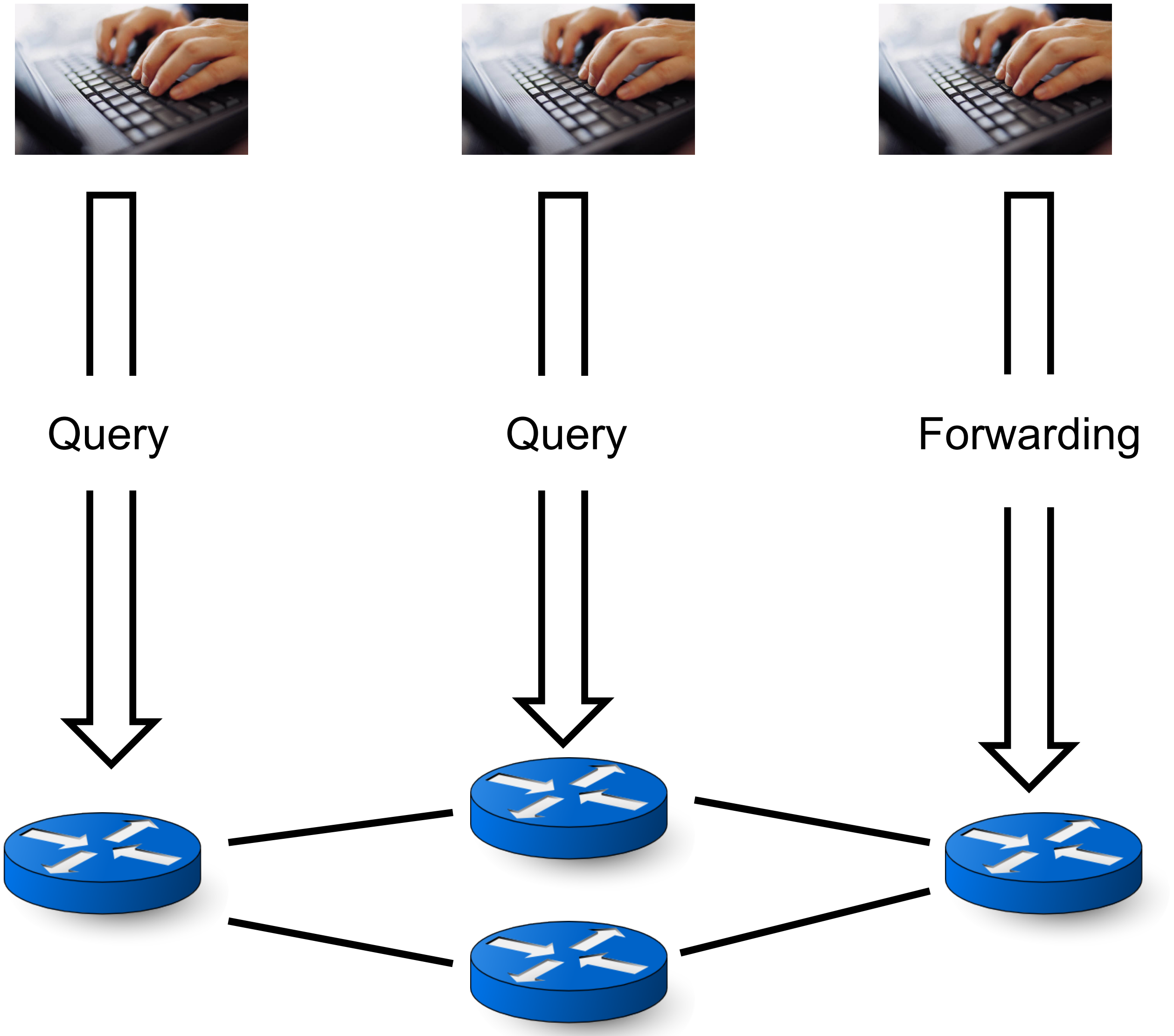
Problem statement



1. Operator/application specifies network path queries

2. Translate into efficient and direct switch measurements (i.e., data plane rules)

Problem statement



Query

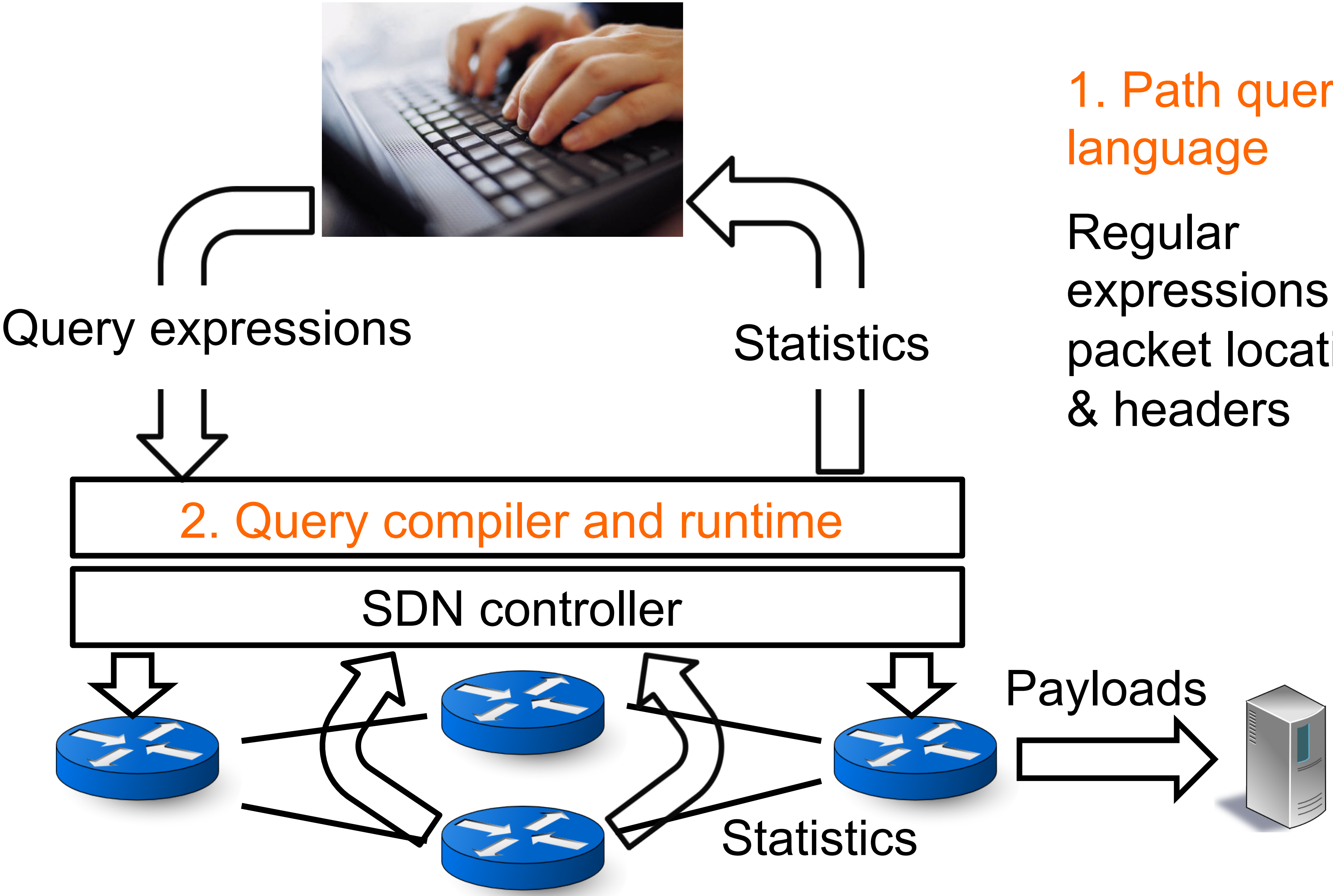
Query

Forwarding

Independent specifications

Compiled into data plane rules

Solution architecture



Query expressions

Statistics

2. Query compiler and runtime

SDN controller

Payloads

Statistics

1. Path query language

Regular expressions of packet location & headers

Path Query Language

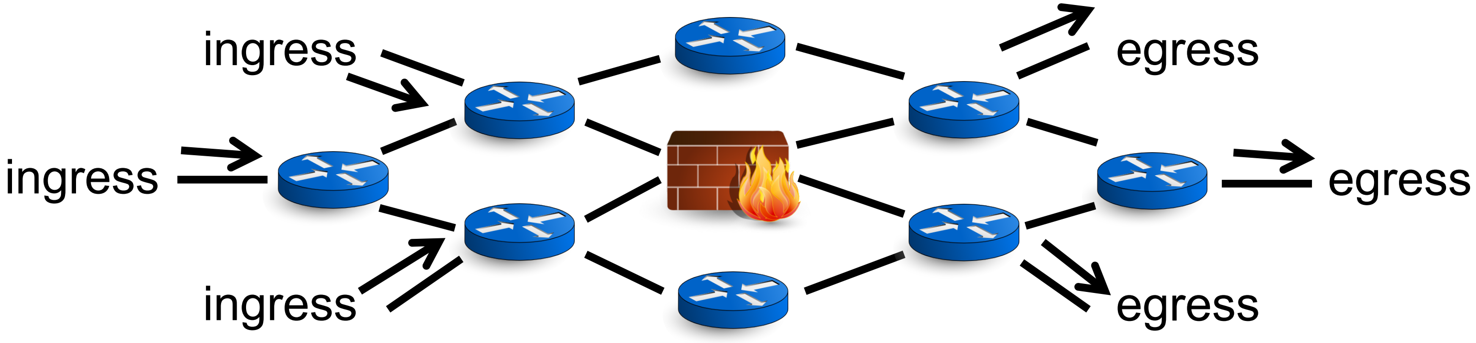
Let’s write some queries! (1/3)

- Count packets reaching switch S1, then S2 with an internal source IP address (10.0/16)

```
switch=S1

∧   ⇦   A hop on the wire

switch=S2, srcip=10.0/16
```

Let’s write some queries! (2/3)

- Capture packets evading a firewall in the network



```
ingress()
∧
(switch != FW)*
∧
egress()
```

0 or more repetitions

Let’s write some queries! (3/3)

- Switch-level traffic matrix:

	E1	E2	...
I1	250	100	...
I2	120	95	...
...	...	...	...

Let’s write some queries! (3/3)

- Switch-level traffic matrix:

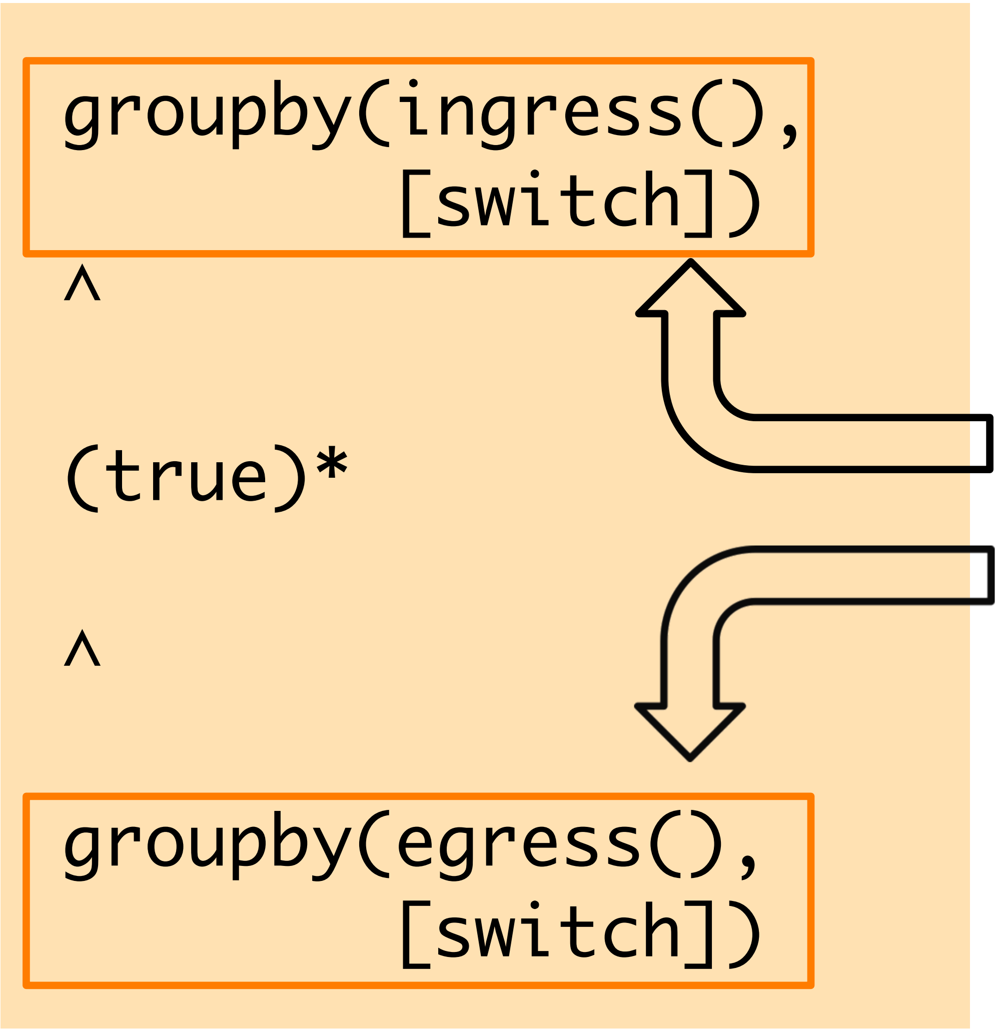
```
ingress()

^

(true)*

^

egress()
```

Flow	#pkts
*	1000

Count all packets, going from any ingress to any egress.

Let’s write some queries! (3/3)

- Switch-level traffic matrix:



```
groupby(ingress(),
          [switch])
∧

(true)*

∧

groupby(egress(),
          [switch])
```

Flow	#pkts
sw=I1, sw=E1	250
sw=I1, sw=E2	100
...	...

Group counts by packet’s ingress and egress switch!

➔ Traffic matrix!

Let’s write some queries!

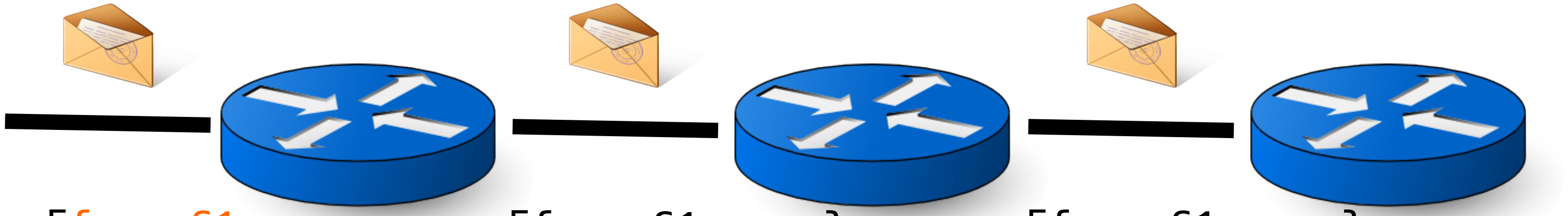
- More example queries in the paper

The Runtime System

How to analyze packet paths in the data plane?

Packet paths on data plane

- Main idea: Record path information in packets



```
[{sw: S1
  port: 1
  srcmac: ...
  srcip: ...
  ...}]
```

```
[{sw: S1, ...},
 {sw: S2
 port: 3
 srcmac: ...
 ...}]
```

```
[{sw: S1, ...},
 {sw: S2, ...},
 {sw: S3
  port: 2
  ...}]
```

- As such, too much state!

Reducing path state on packets

- Observation 1: Queries already tell us what’s needed!
  - Only record path state needed by queries

- Observation 2: Queries are regular expressions
  - Regular expressions ➔ Finite automaton (DFA)
  - Distinguish only paths corresponding to DFA states

Reducing path state on packets

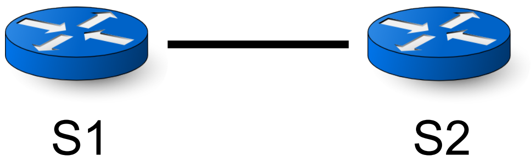
- Observation 1: Queries already tell us what's needed!
  - Only record path state needed by queries

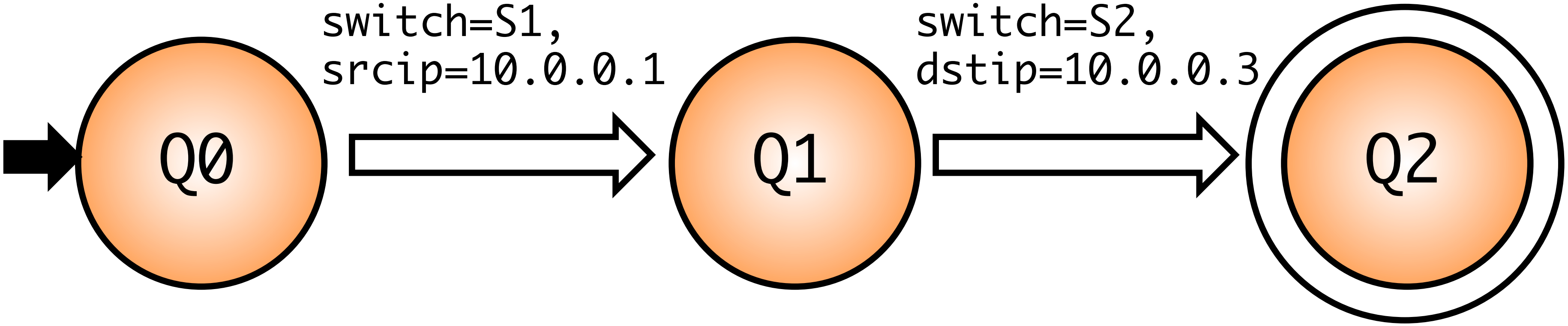
Record only DFA state on packets (1-2 bytes)

Use existing "tag" fields (e.g., VLAN)

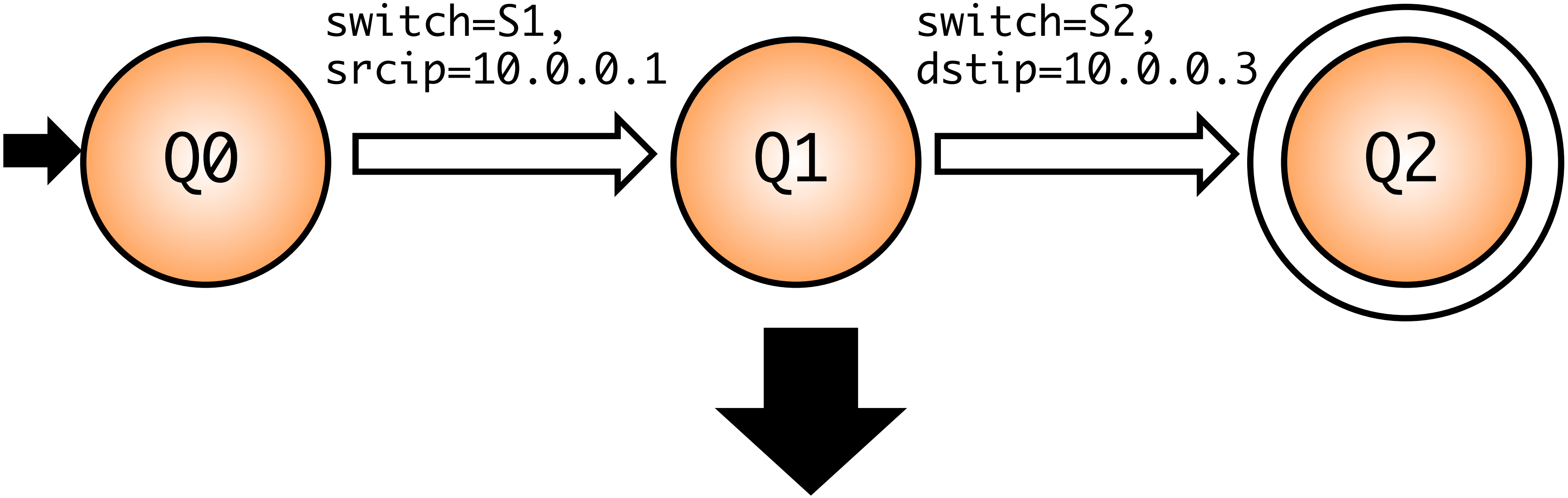
Example: Query Compilation (1/3)

```
Query:
  (switch=S1, srcip=10.0.0.1)
∧ (switch=S2, dstip=10.0.0.3)
```





Example: Query Compilation (2/3)

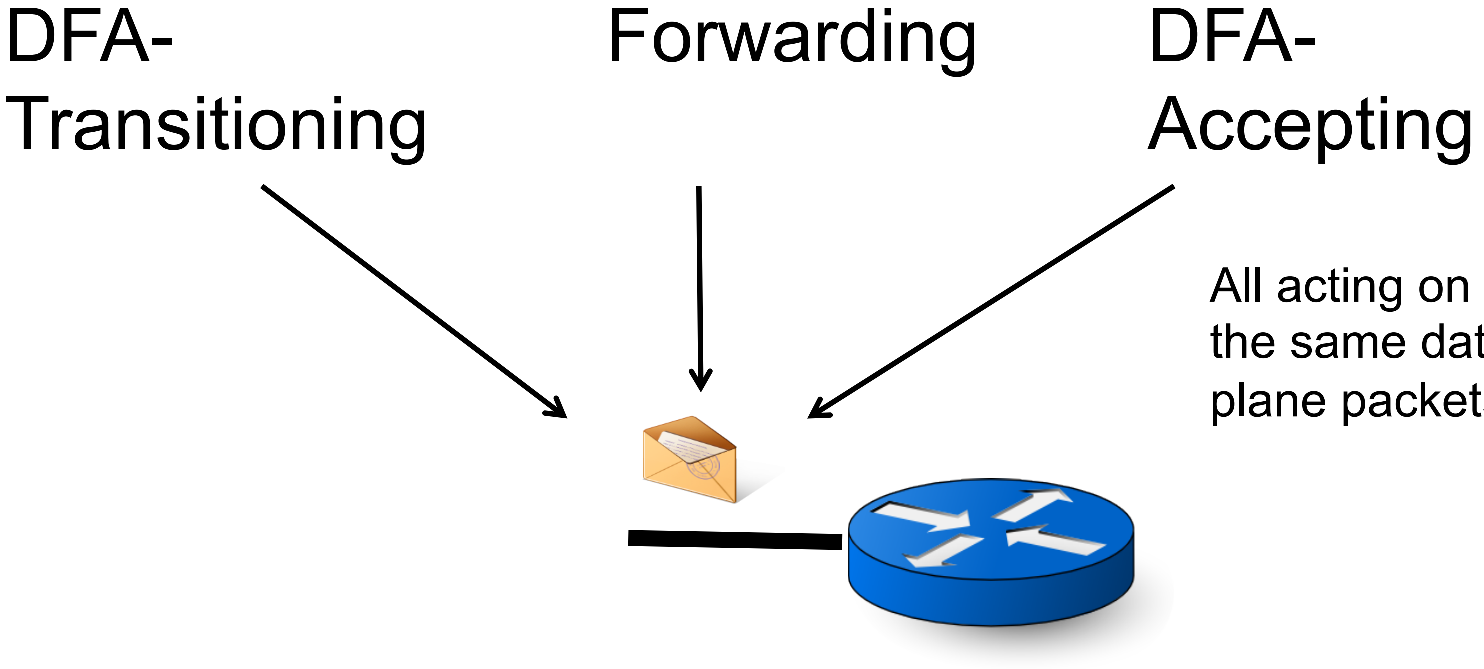


Switch	Match	Action
S1	state=Q0, srcip=10.0.0.1	state=Q1
S2	state=Q1, dstip=10.0.0.3	state=Q2
S2	state=Q1, dstip=10.0.0.3	count

DFA transition

DFA accept

Example: Query Compilation (3/3)



DFA-Transitioning

Forwarding

DFA-Accepting

All acting on the same data plane packets!

Frenetic composition operators (details in paper)

Implementation

- Prototype on the Pyretic (NSDI'13) SDN controller

- Implementation publicly available online
    - http://frenetic-lang.org/pyretic/

- Evaluation:
    - Payload collection bandwidth
    - Rule space
    - See paper.

Summary

DFA state can be used to track packet paths directly on the data plane.

Measurement and forwarding can be specified independently.

Happy to answer queries ;)

narayana@cs.princeton.edu