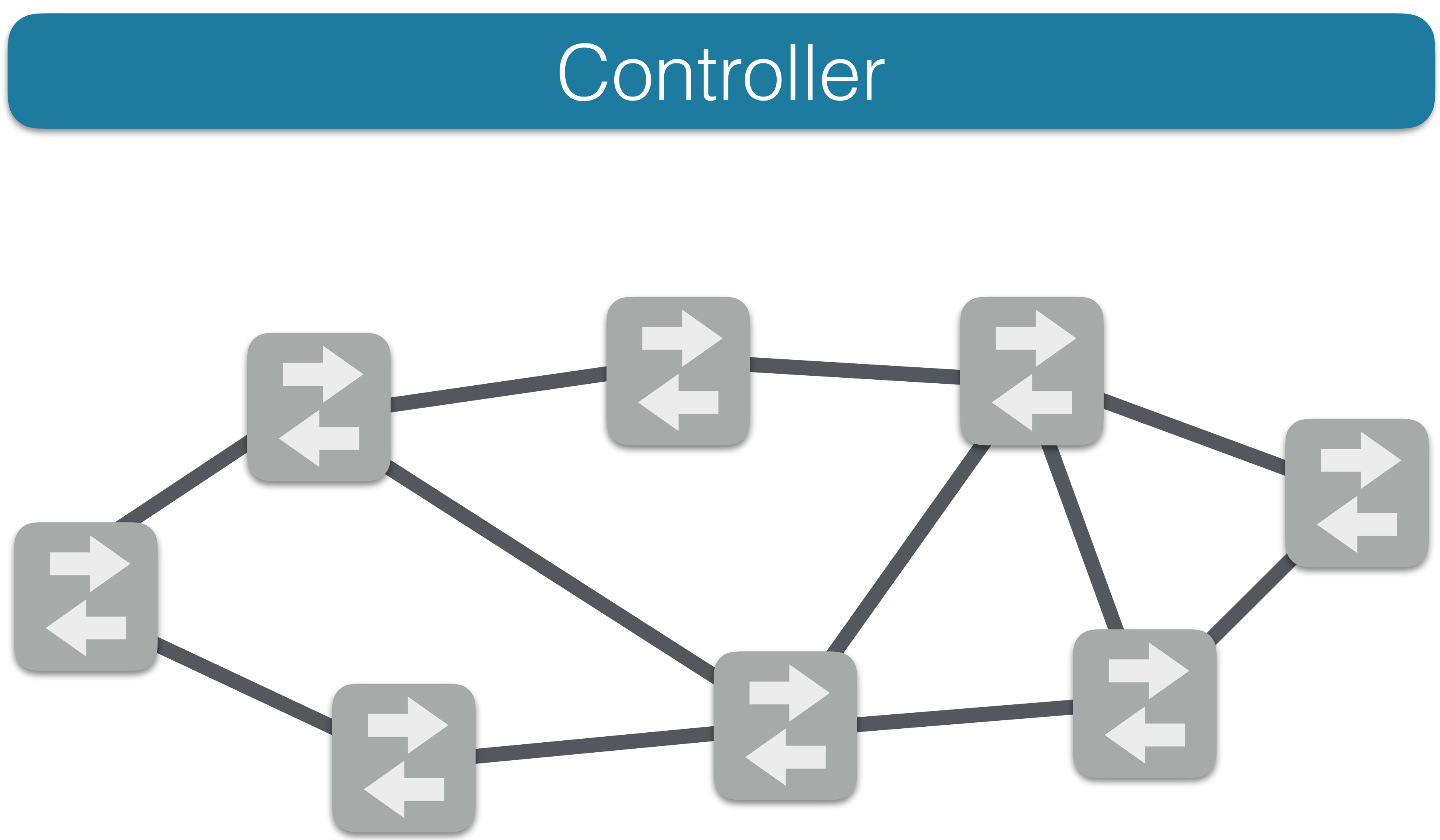
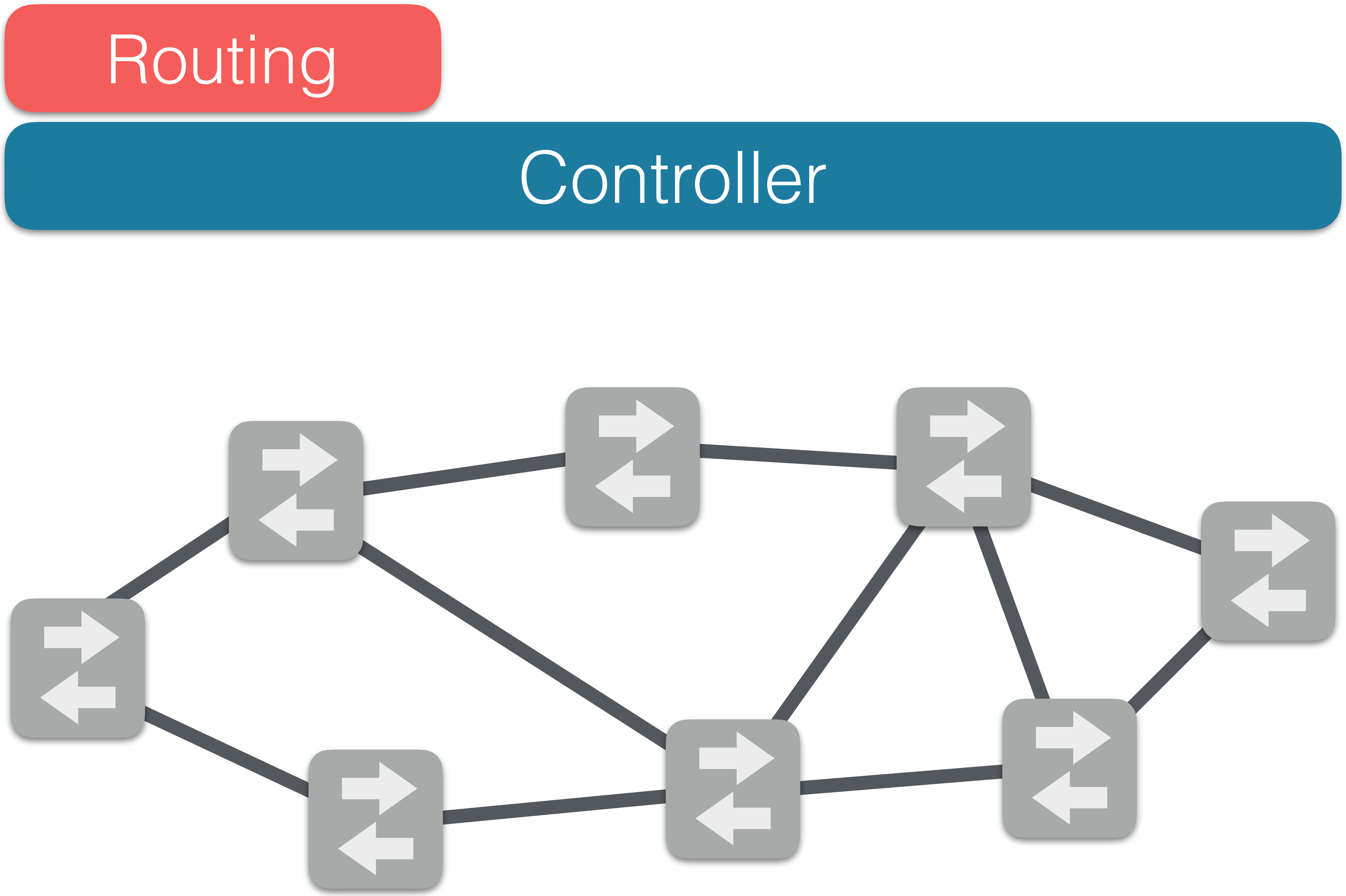
Temporal NetKAT

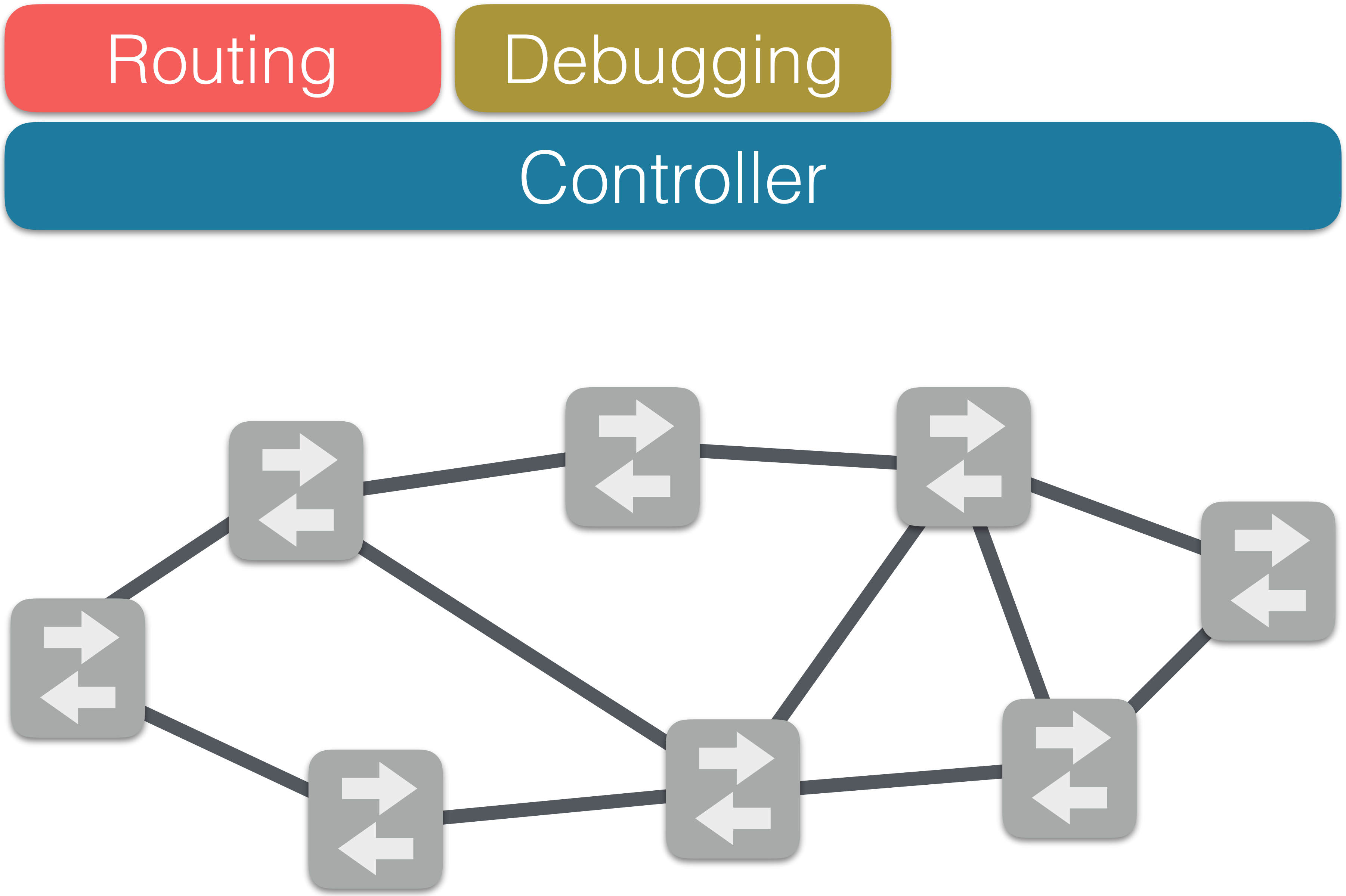
Ryan Beckett
Michael Greenberg*, David Walker

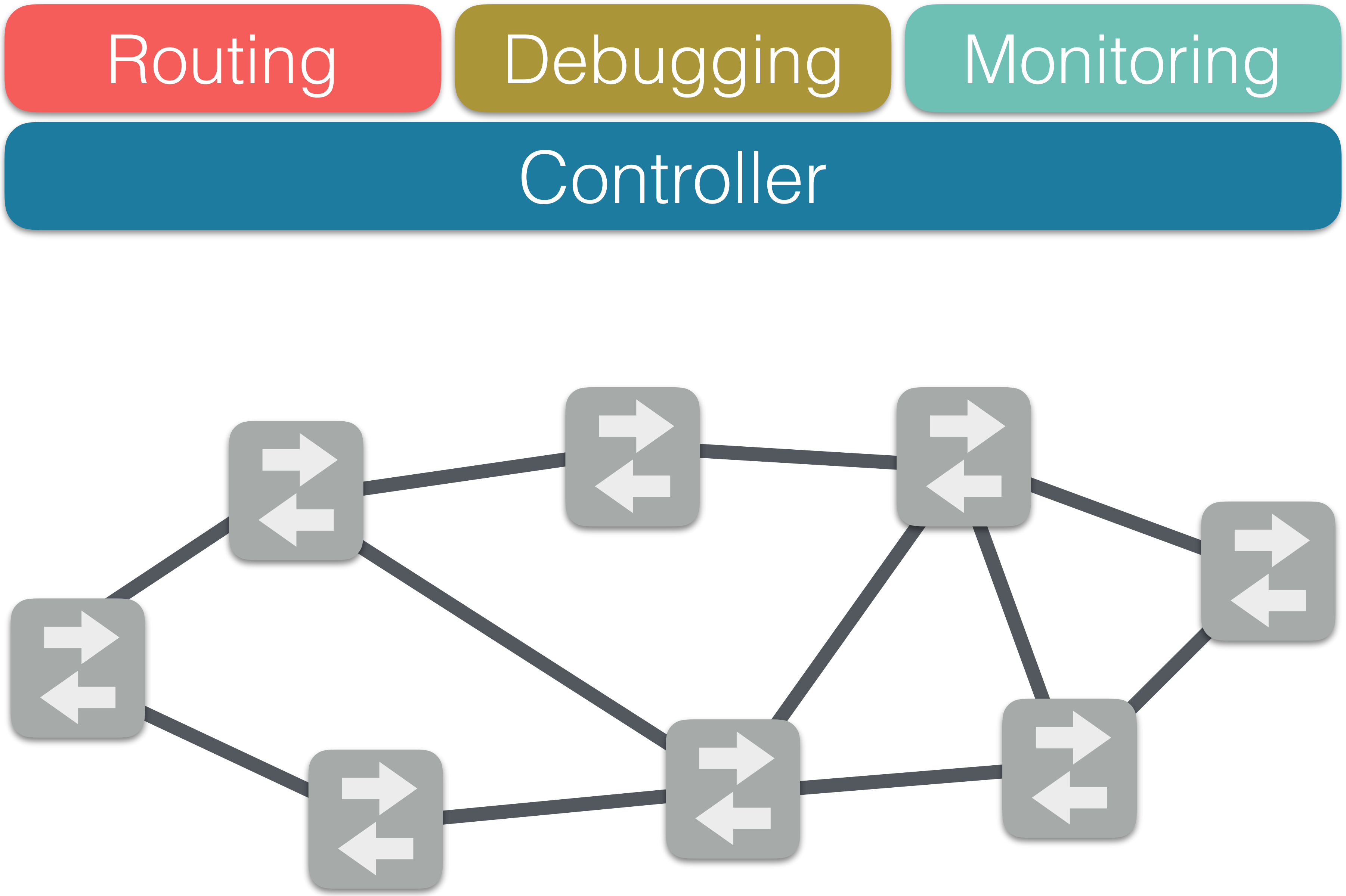
Princeton University

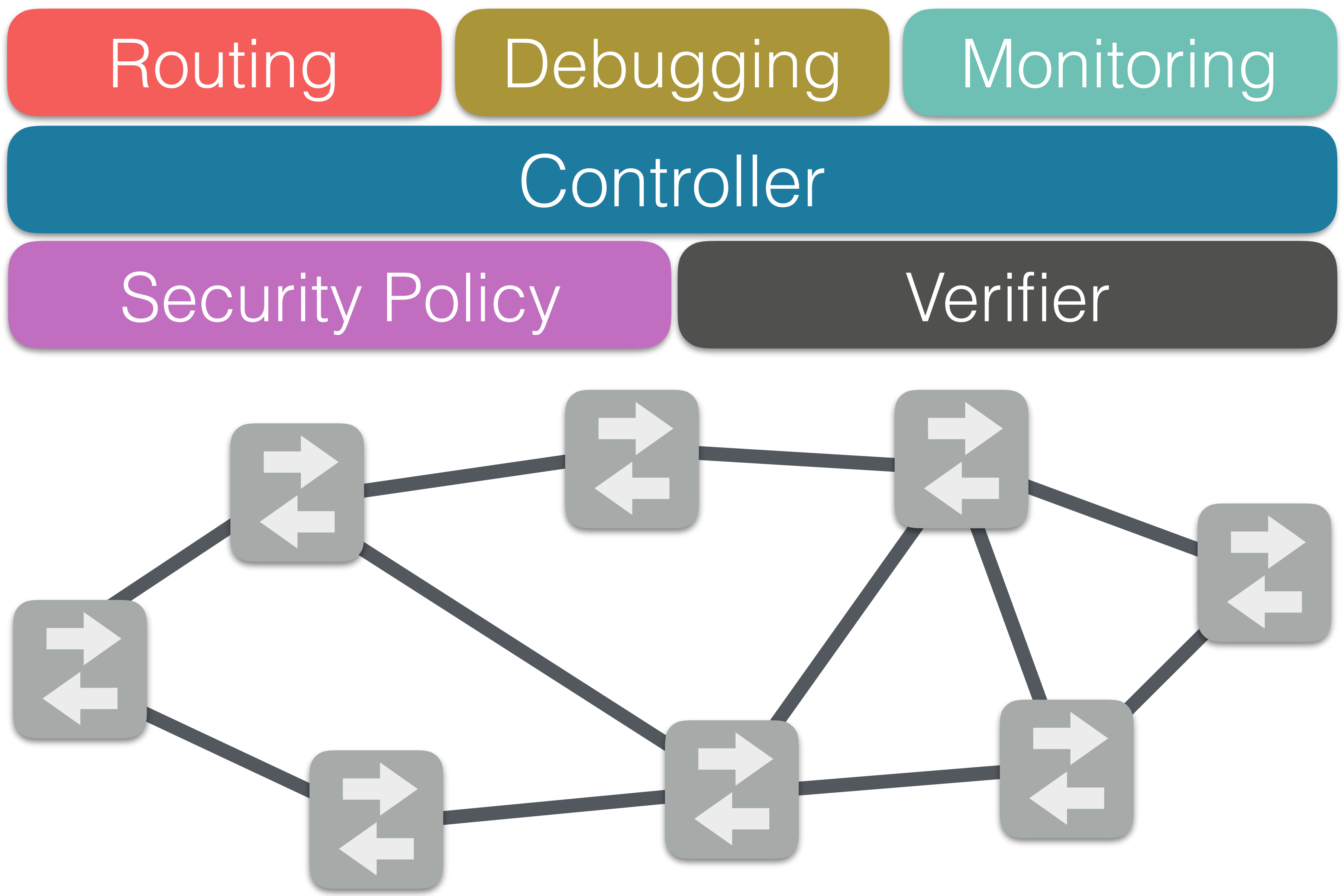
Pomona College*

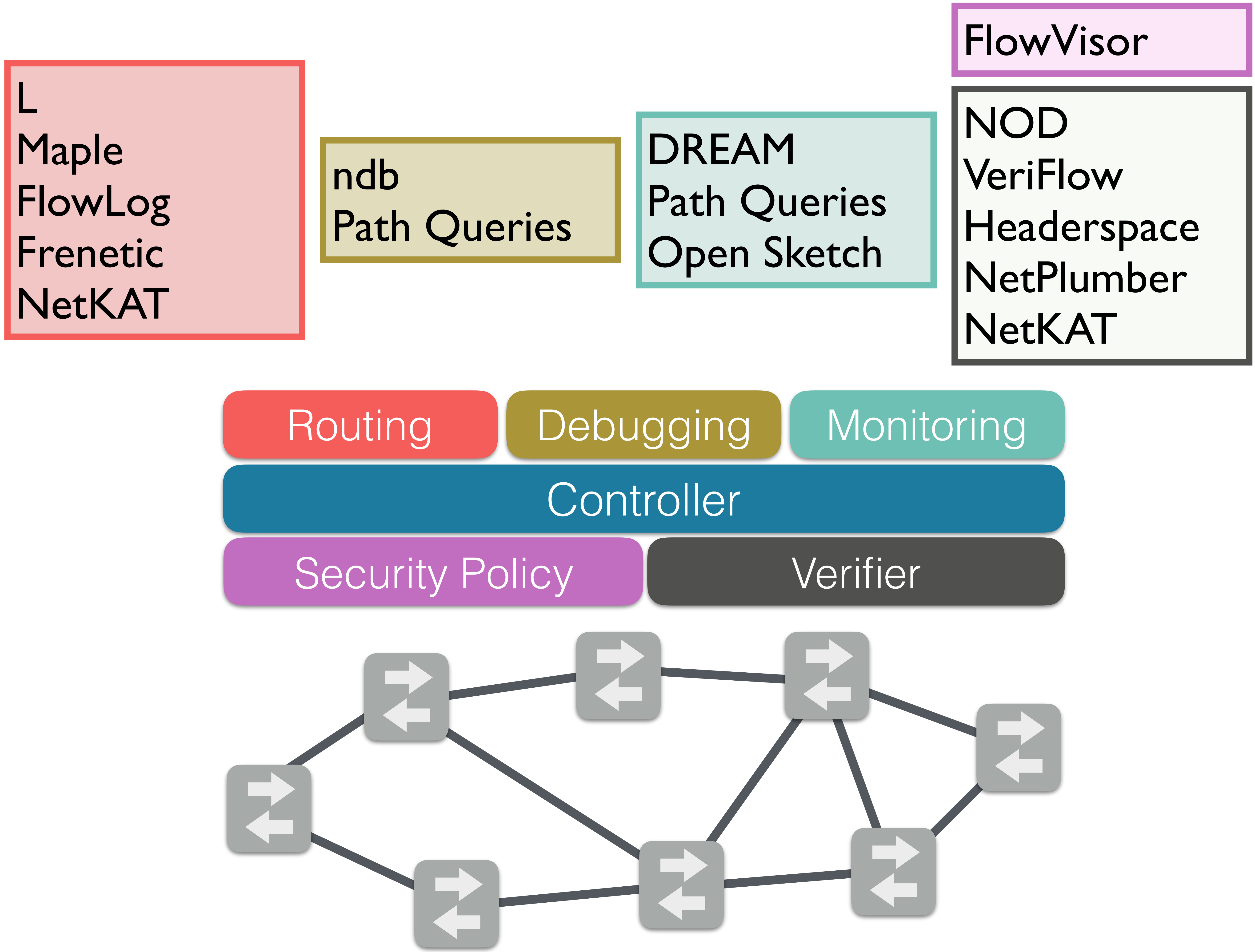

Controller


Routing
Controller

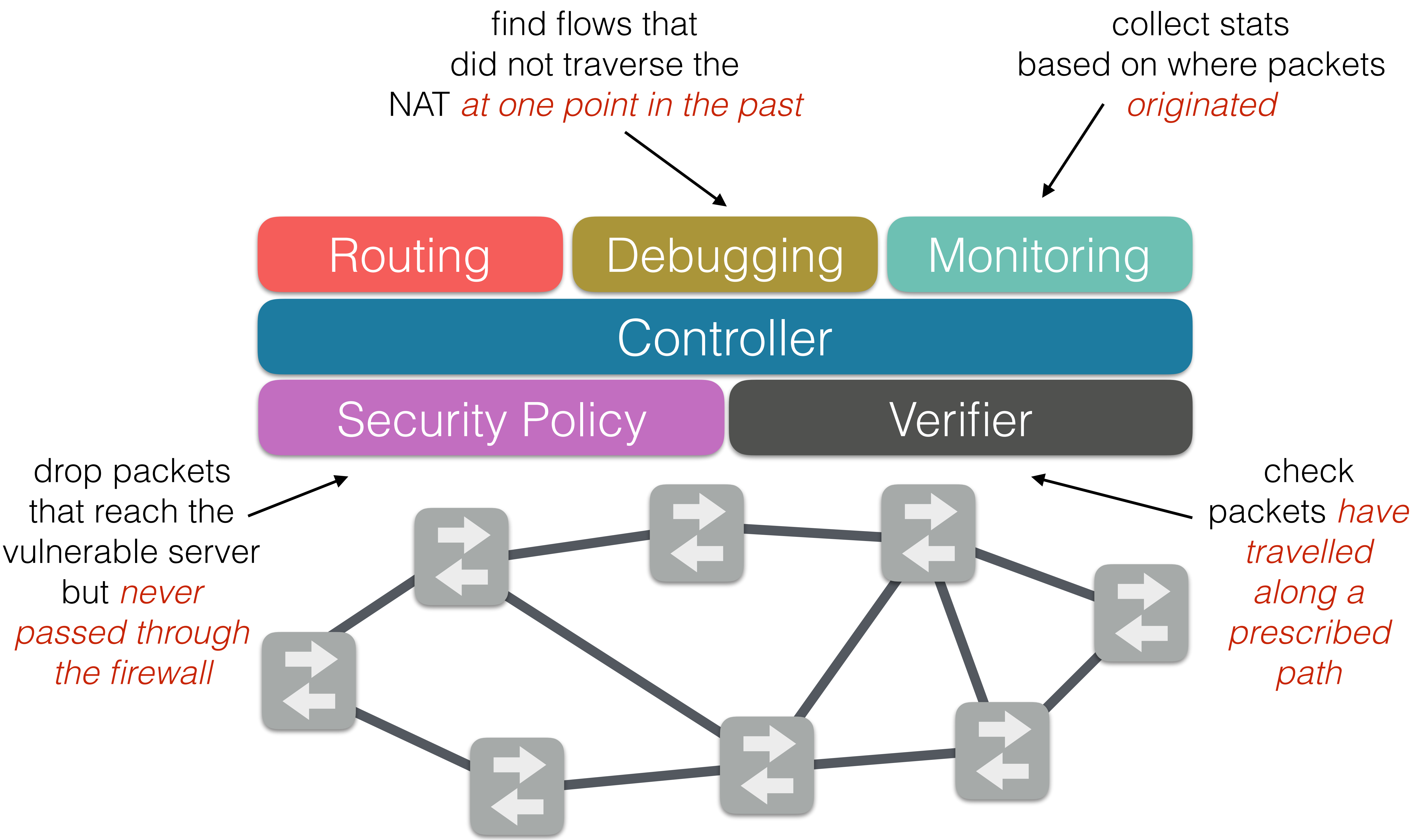

Routing
Debugging
Controller


Routing
Debugging
Monitoring
Controller


Routing
Debugging
Monitoring
Controller
Security Policy
Verifier


L
Maple
FlowLog
Frenetic
NetKAT
ndb
Path Queries
DREAM
Path Queries
Open Sketch
FlowVisor
NOD
VeriFlow
Headerspace
NetPlumber
NetKAT
Routing
Debugging
Monitoring
Controller
Security Policy
Verifier

Our work:
New network programming abstractions for acting on packet history



Overview

Temporal NetKAT [PLDI 2016]

- Extend NetKAT with **past time temporal logic**
- Study its use in several **applications**
- Define a **semantics** and **equational theory** for the language
- Prove **soundness** and network-wide **completeness**
- Define and implement a **compilation strategy**
- **Evaluate** compiler performance on several networks

Temporal NetKAT

NetKAT Review

Predicates

```
a,b ::= f = n      test
      | 1          true
      | 0          false
      | a + b      or
      | a · b      and
      | ¬a         negation
```

Policies

```
p,q ::= a          predicate
      | f ← v      assignment
      | p + q      union
      | p · q      sequence
      | p*         iteration
```

} Boolean Algebra

} Kleene Algebra

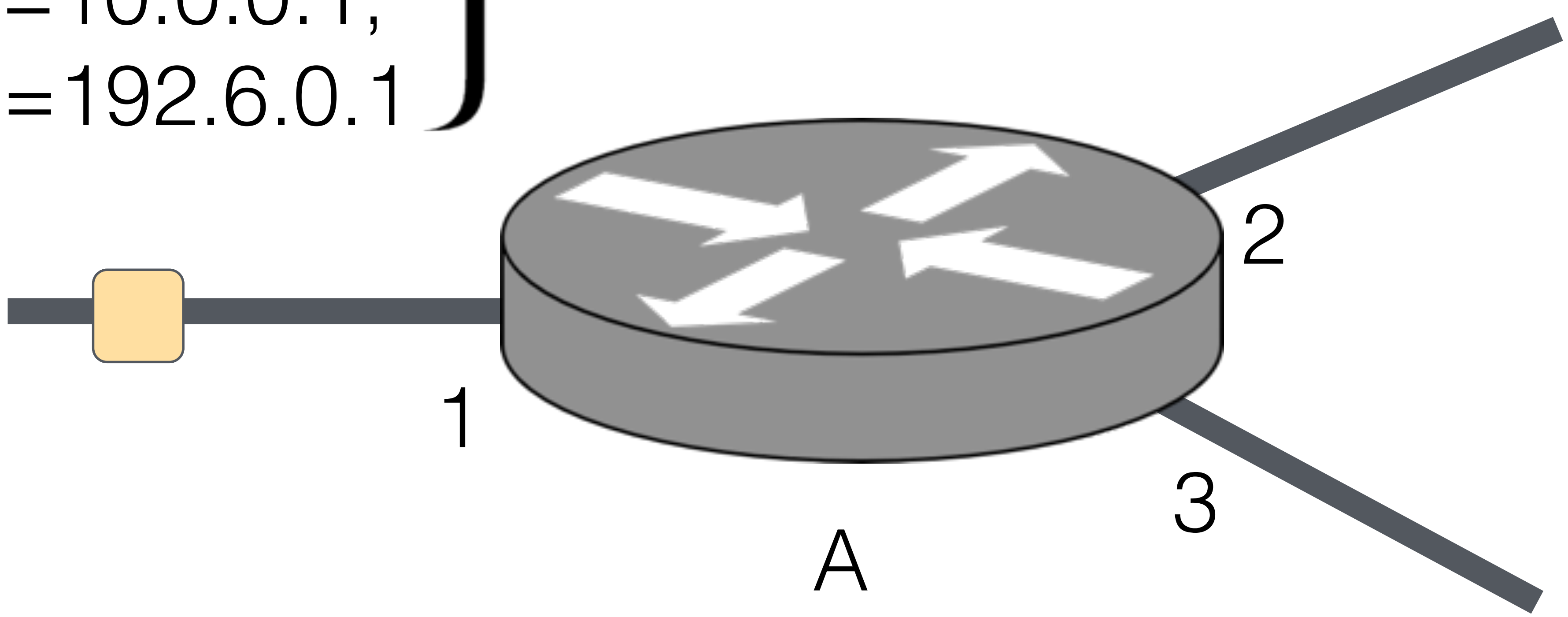
Based on KAT [Kozen & Smith '96]

Extended to networks [Anderson et al '14]

NetKAT Review

Packet: A record of fields and values

{
sw=A,
pt=1,
src=10.0.0.1,
dst=192.6.0.1
}

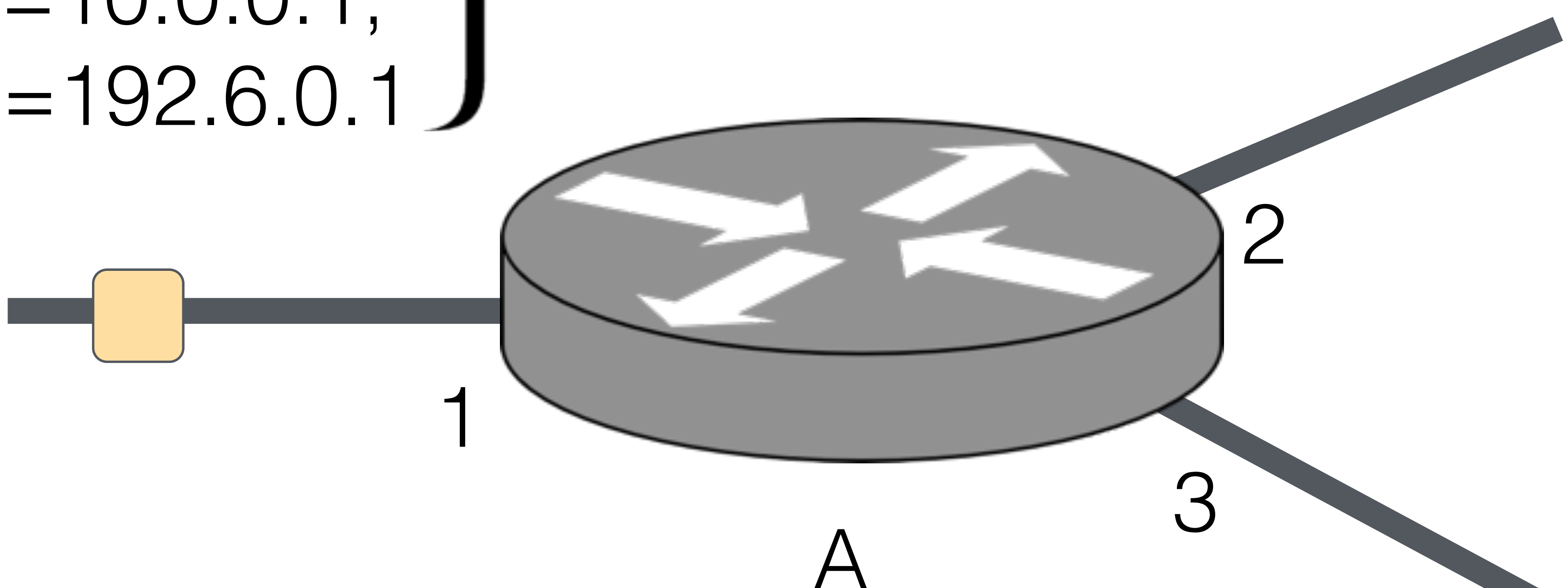


NetKAT Review

Language Features:

- Match packets
- Modify packets

{
sw=A,
pt=1,
src=10.0.0.1,
dst=192.6.0.1
}



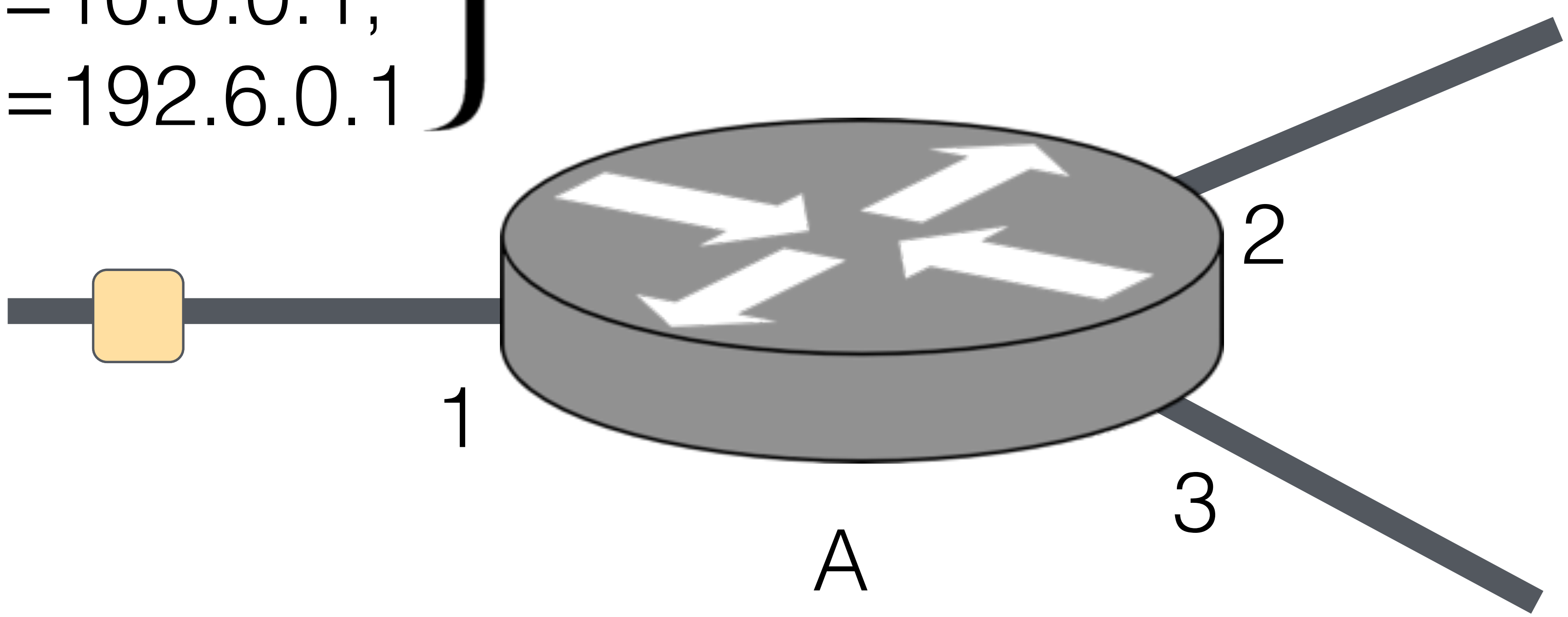
NetKAT Review

Language Features:

- **Match packets**
- Modify packets

{ sw=A,
pt=1,
src=10.0.0.1,
dst=192.6.0.1 }

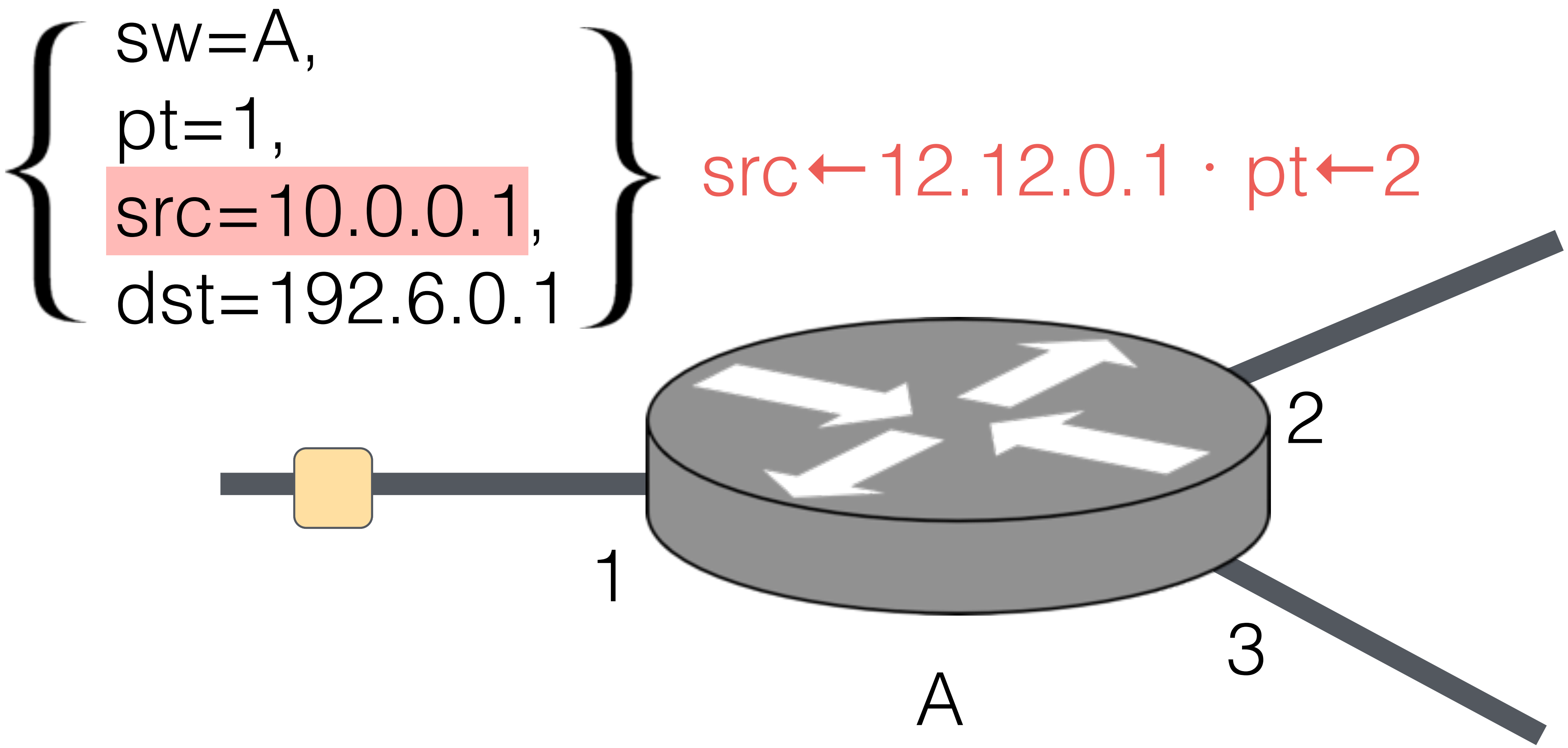
¬src=10.0.0.1 + sw=A



NetKAT Review

Language Features:

- Match packets
- **Modify packets**



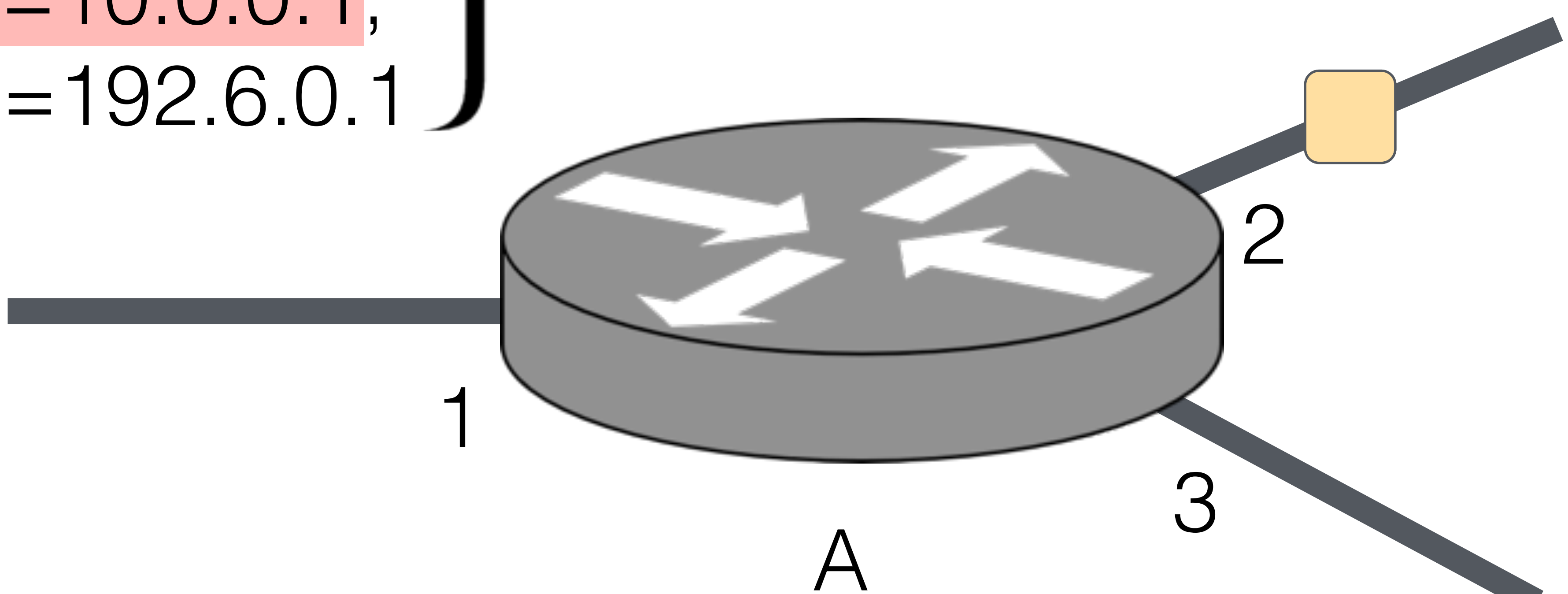
NetKAT Review

Language Features:

- Match packets
- **Modify packets**

{ sw=A,
pt=1,
src=10.0.0.1,
dst=192.6.0.1 }

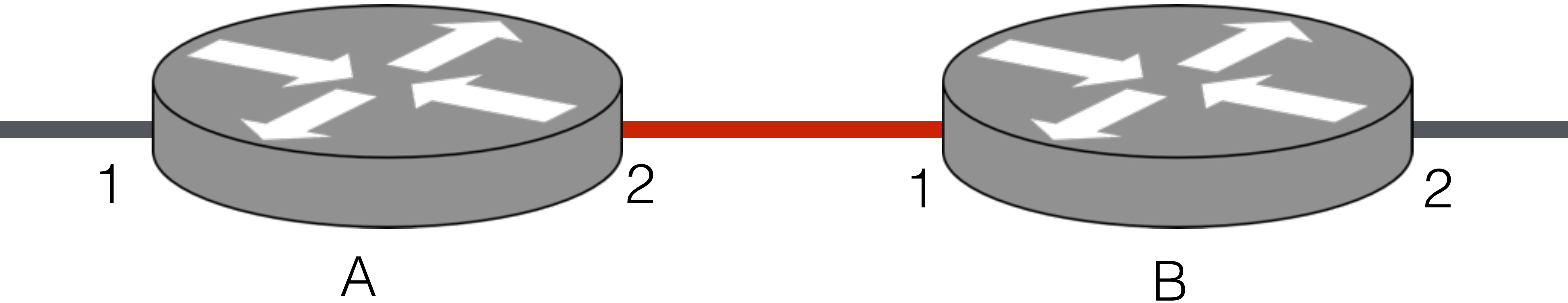
src←12.12.0.1 · pt←2



NetKAT Review

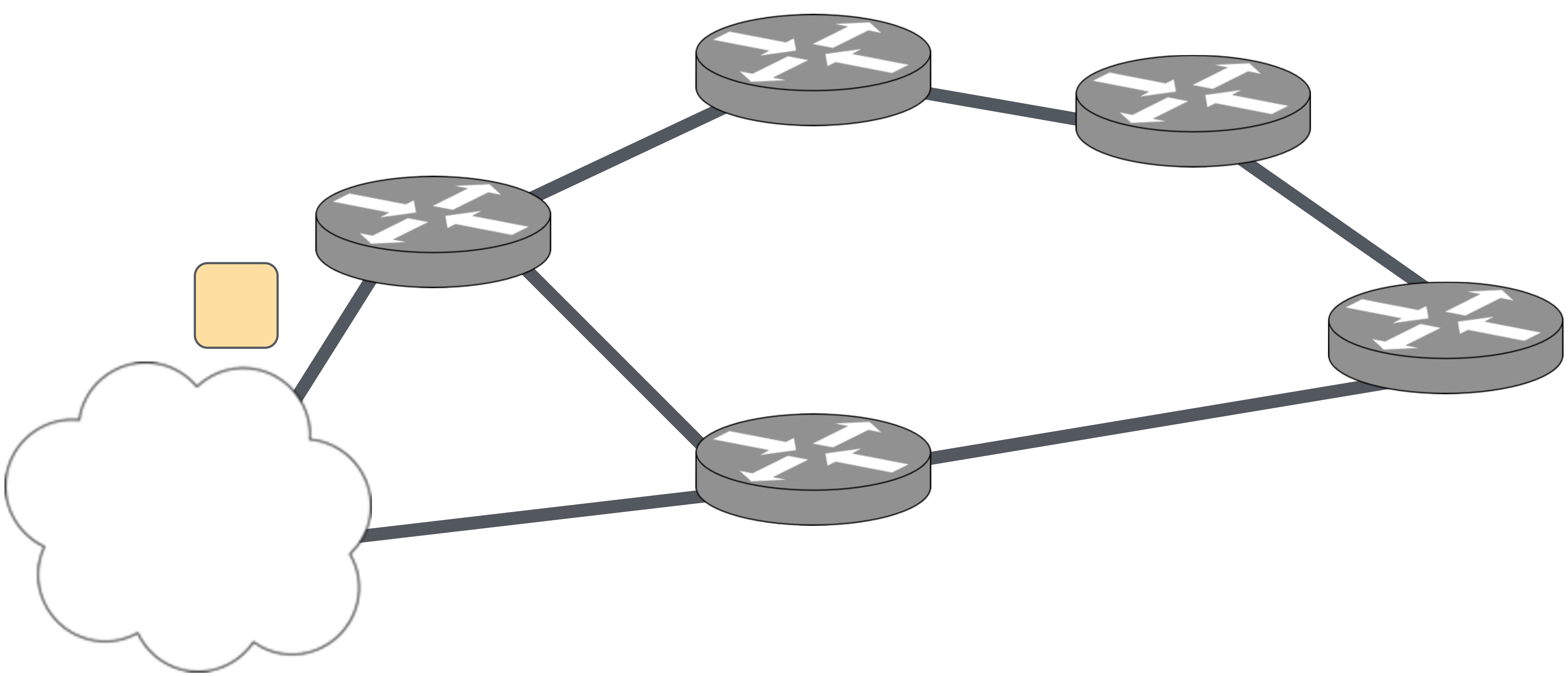
Modelling Network Topology:

$(sw=A \cdot pt=2) \cdot sw \leftarrow B \cdot pt \leftarrow 1$



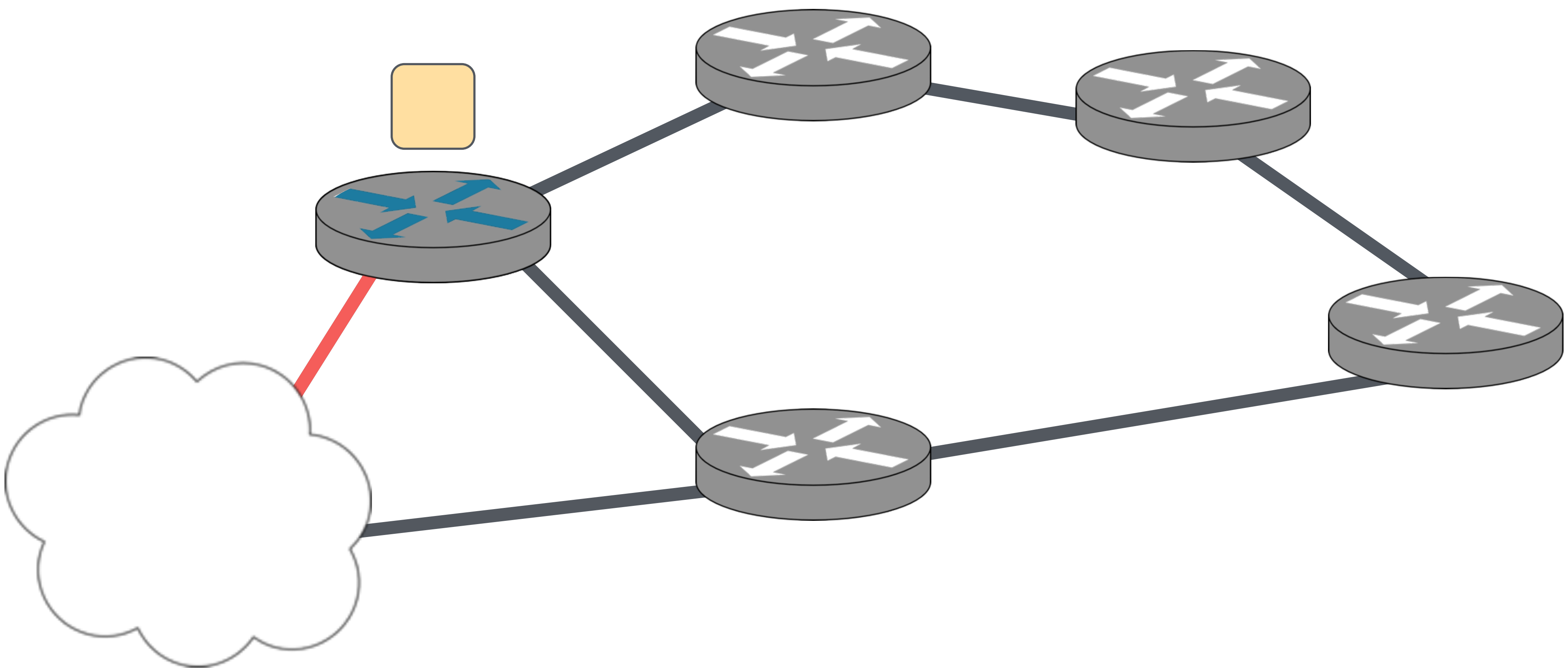
NetKAT — Network

Kleene Star: (topology · switch)*



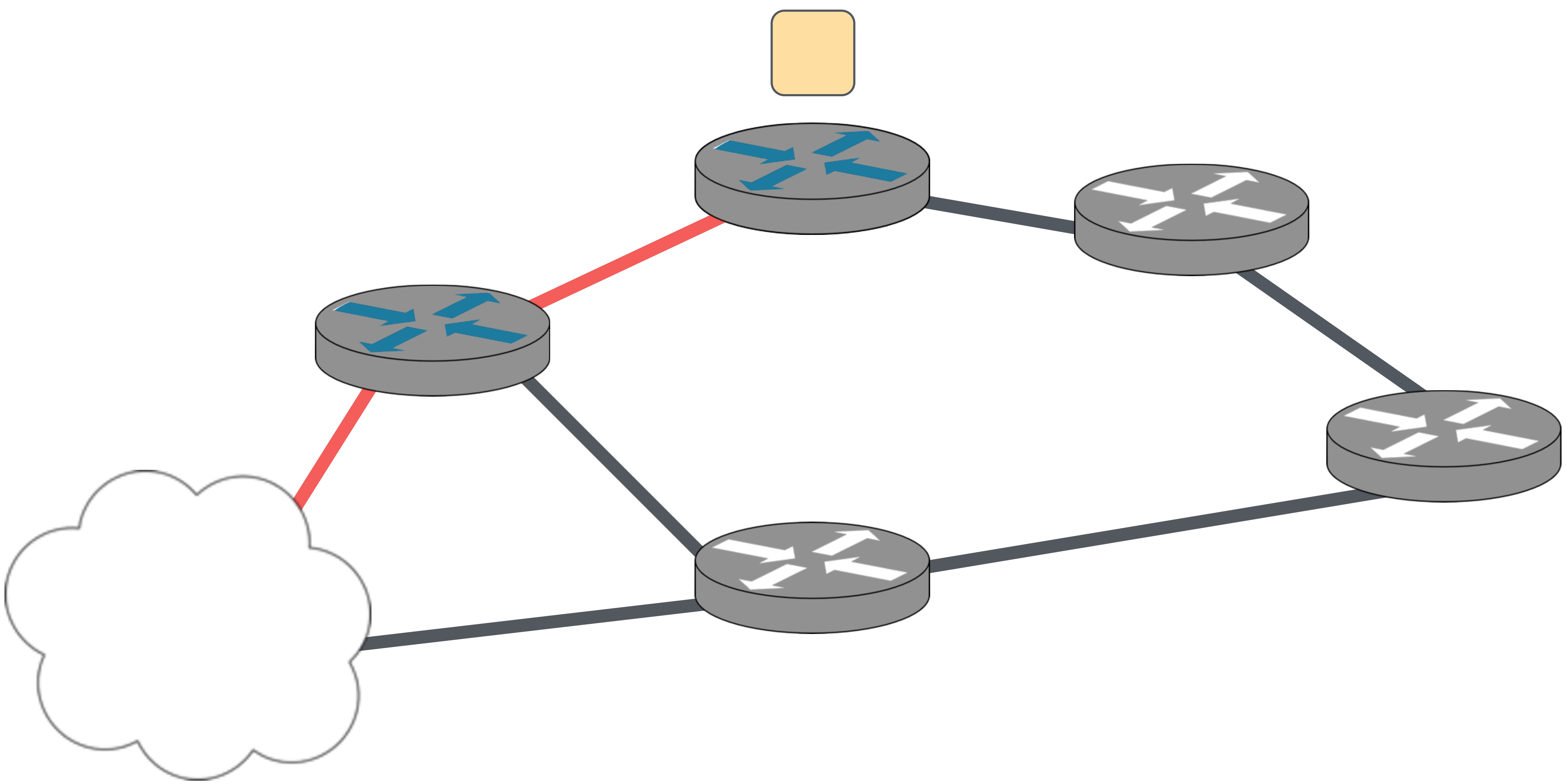
NetKAT — Network

Kleene Star: (topology · switch)*



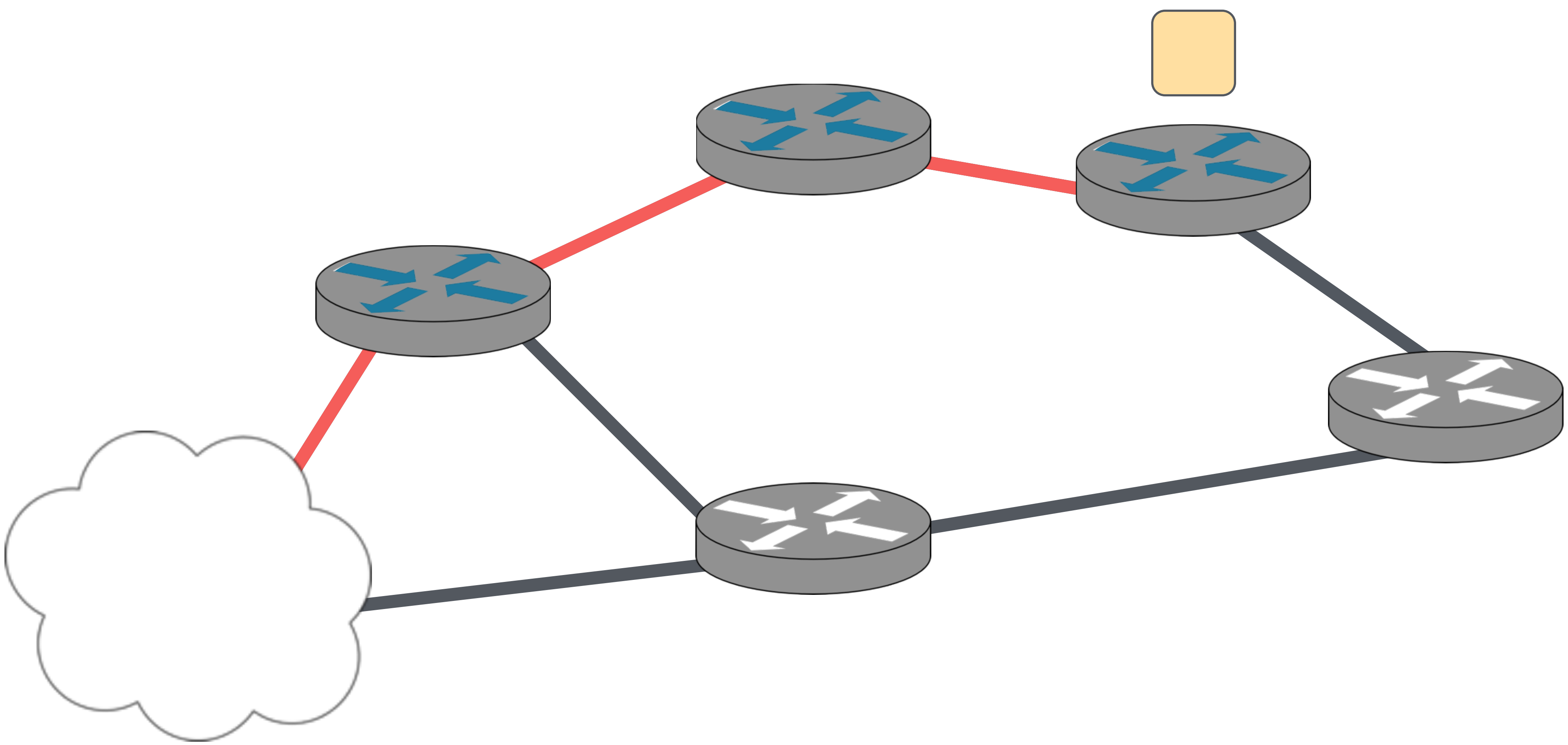
NetKAT — Network

Kleene Star: (topology · switch)*



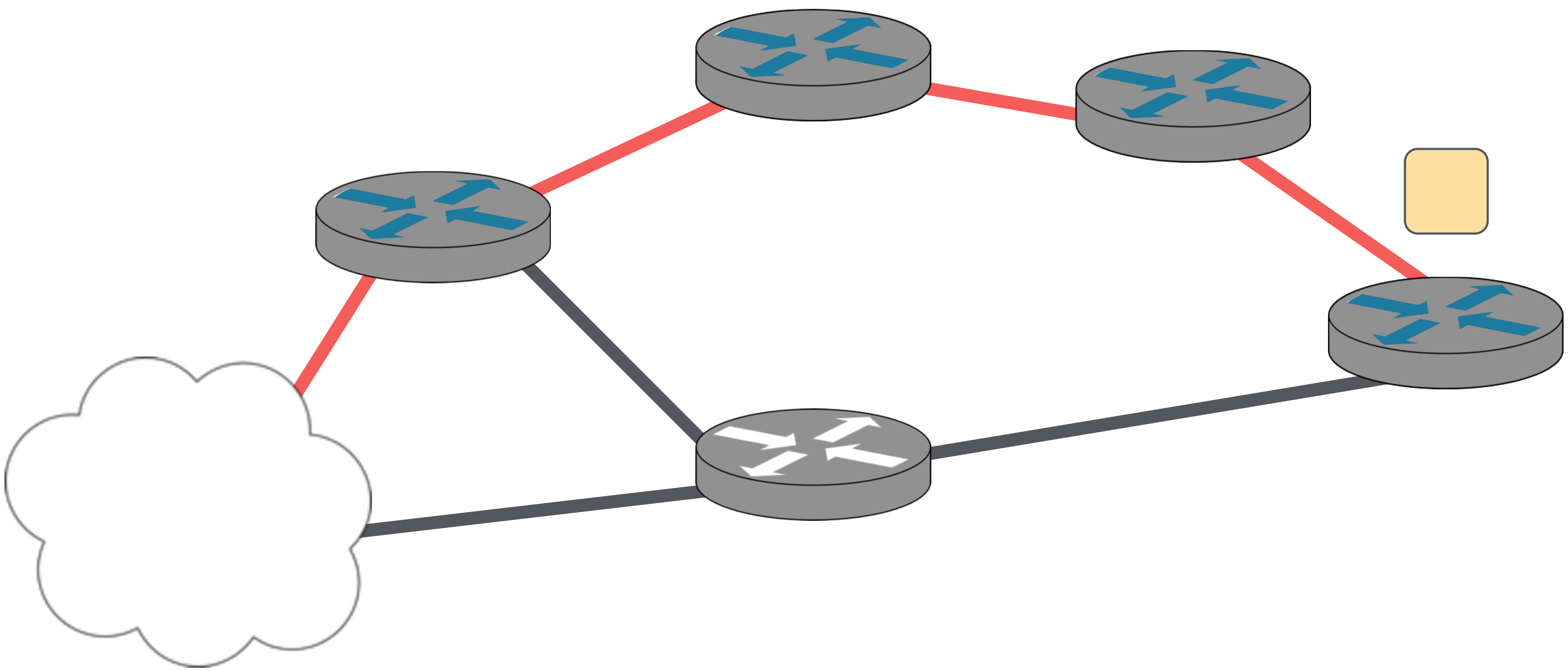
NetKAT — Network

Kleene Star: (topology · switch)*



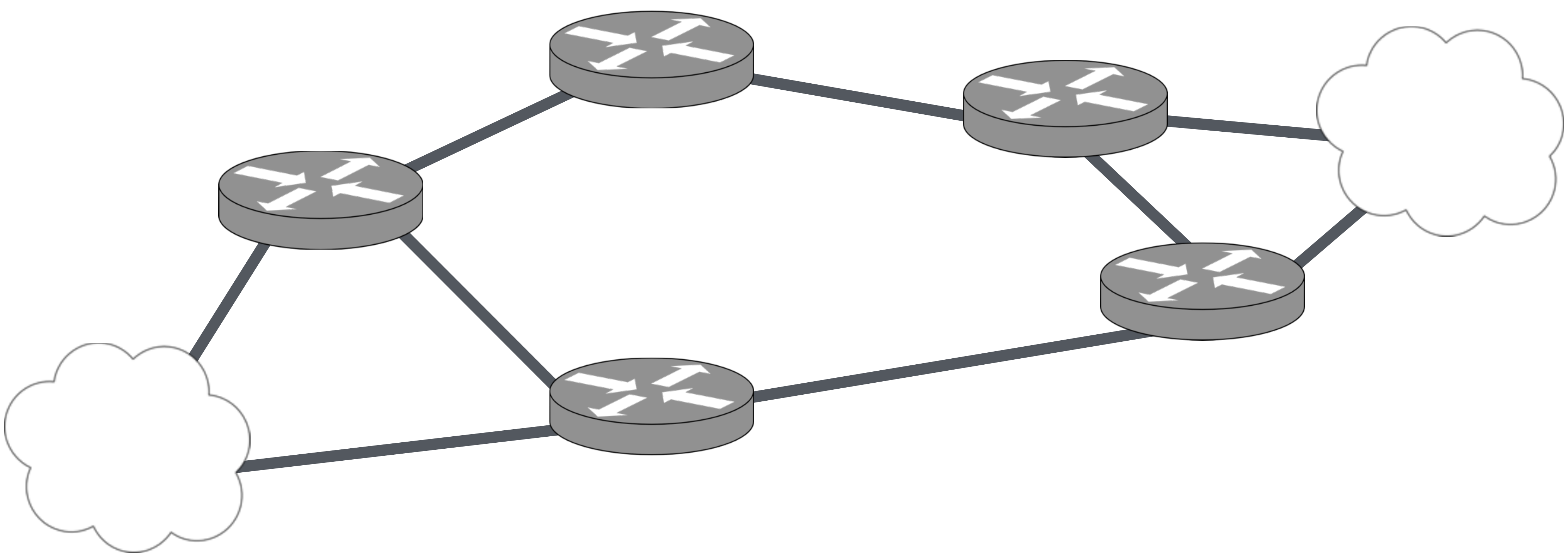
NetKAT — Network

Kleene Star: (topology · switch)*



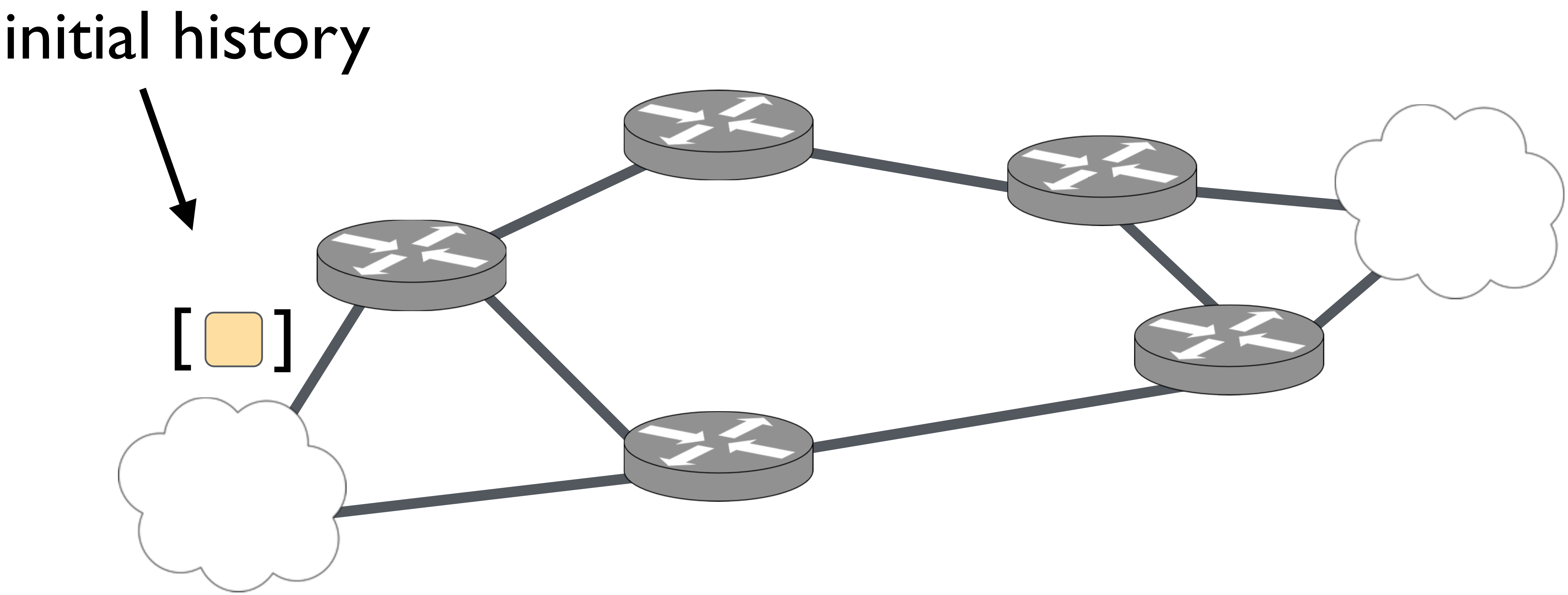
NetKAT: Packet History

A policy takes a packet history to a set of histories



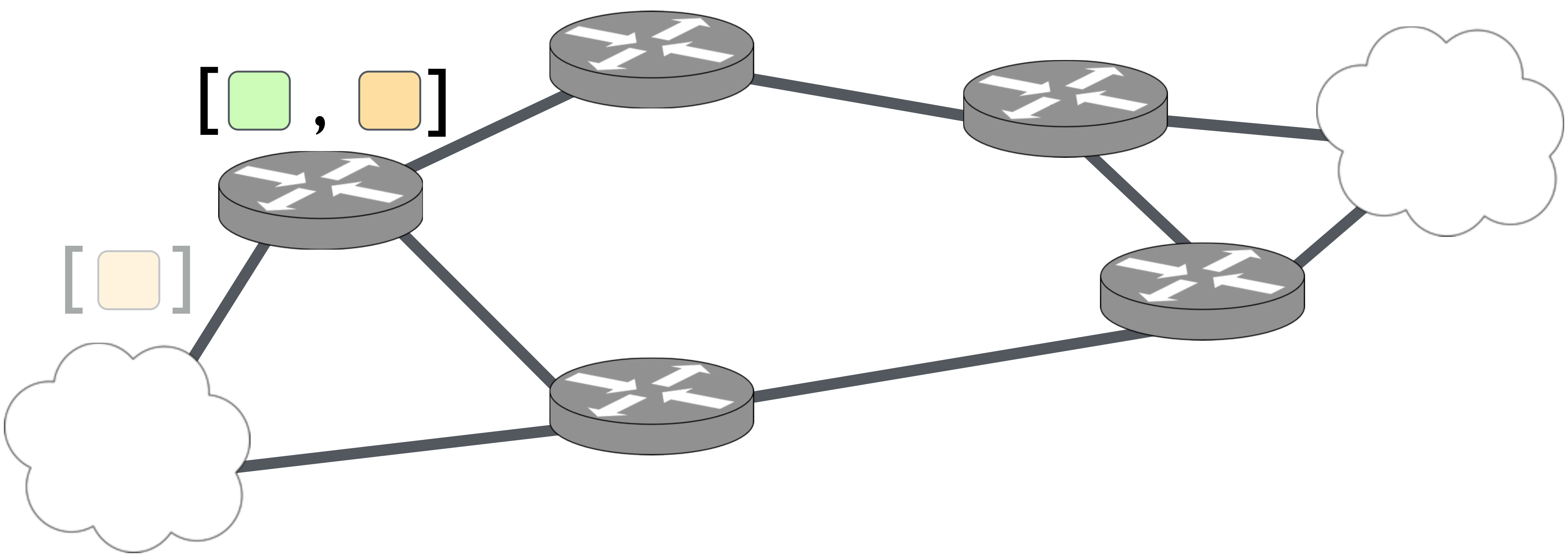
NetKAT: Packet History

A policy takes a packet history to a set of histories



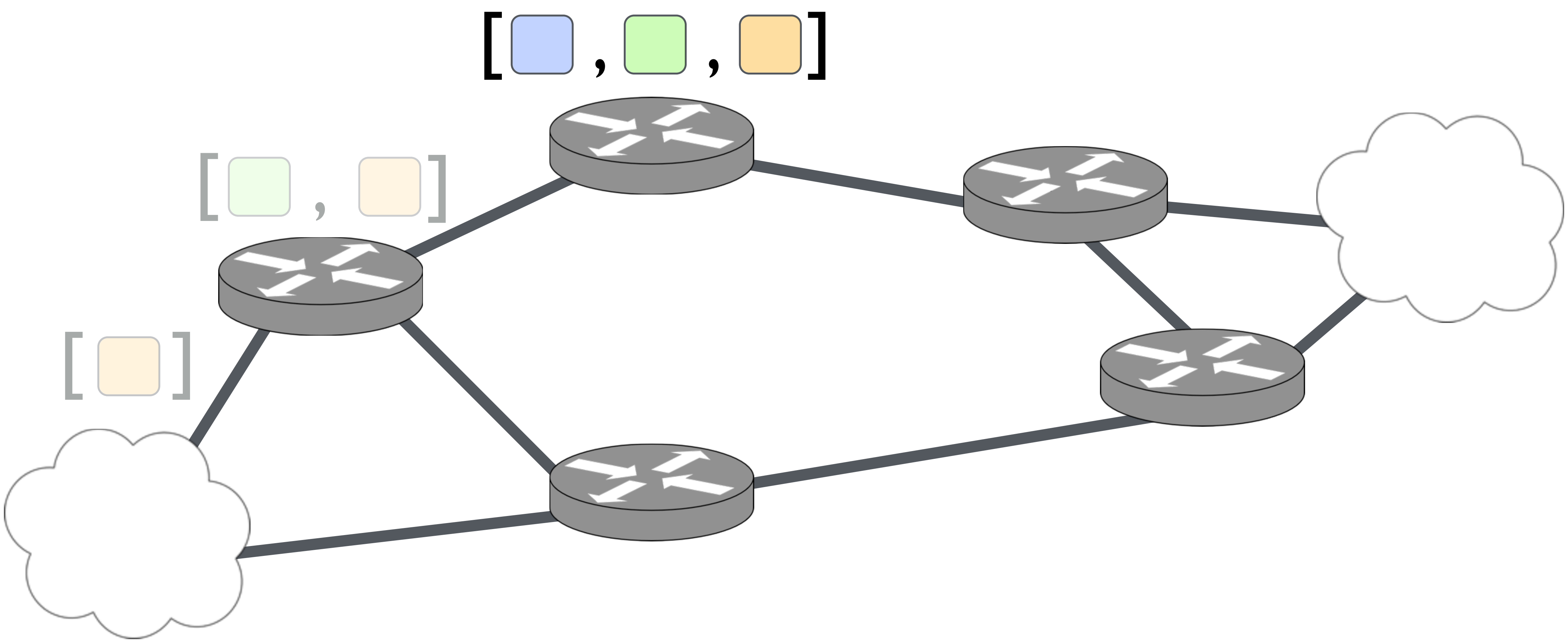
NetKAT: Packet History

A policy takes a packet history to a set of histories



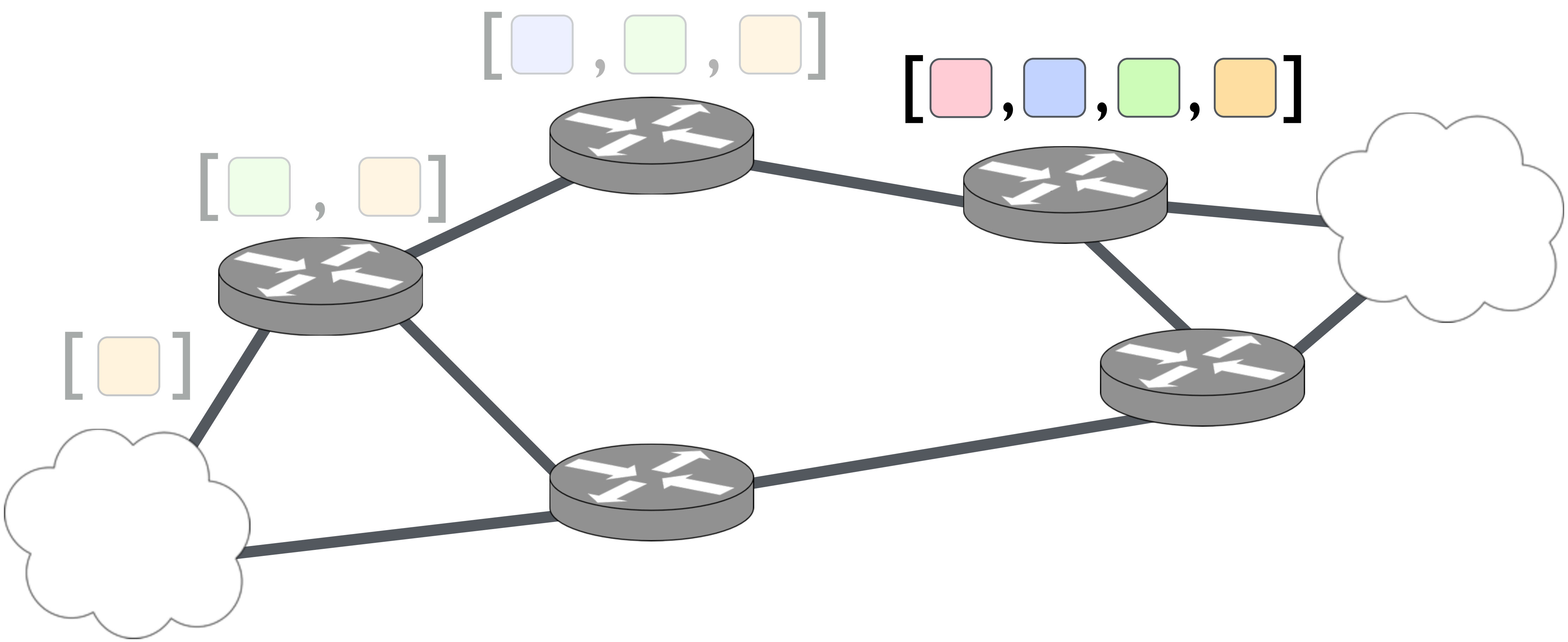
NetKAT: Packet History

A policy takes a packet history to a set of histories

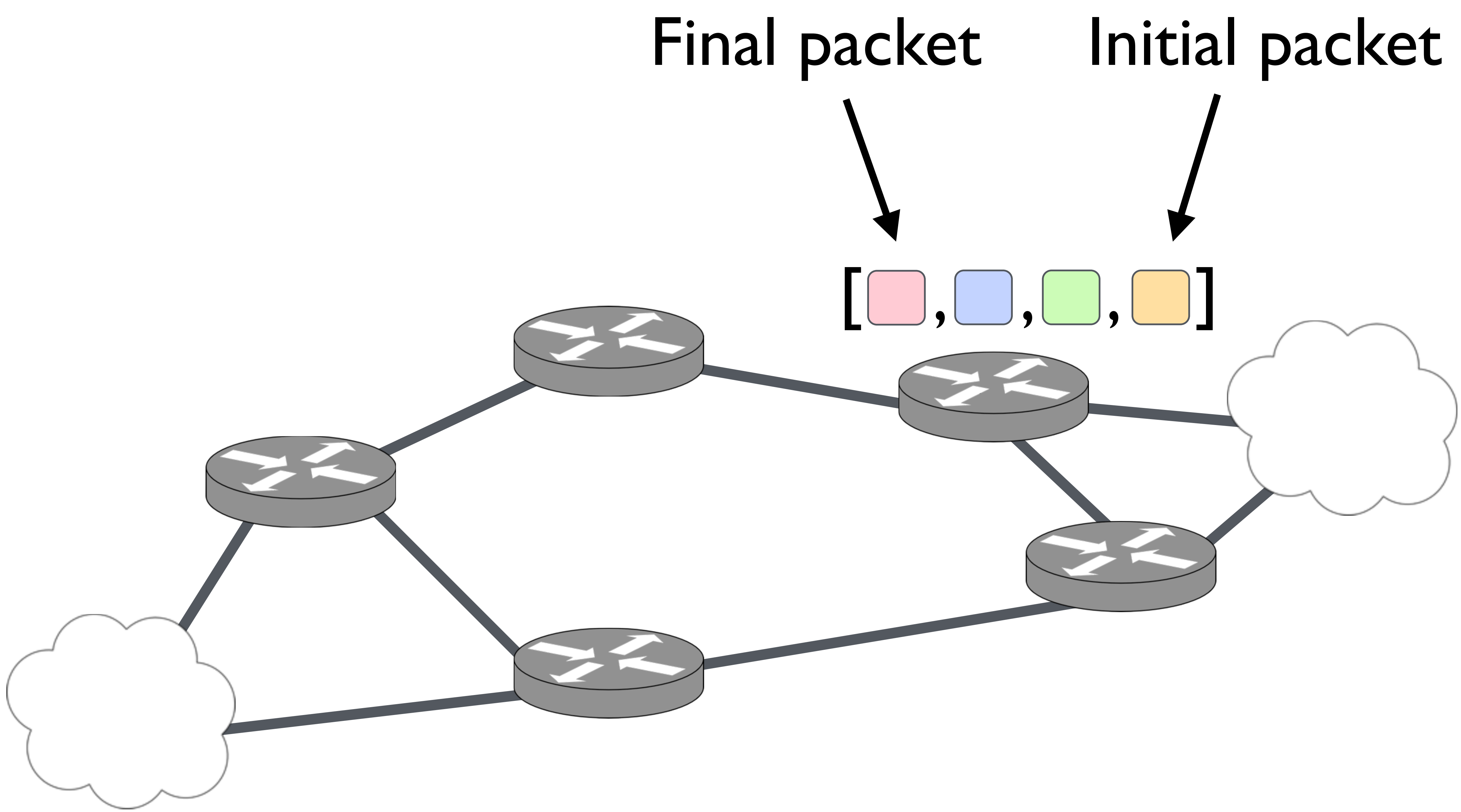


NetKAT: Packet History

A policy takes a packet history to a set of histories

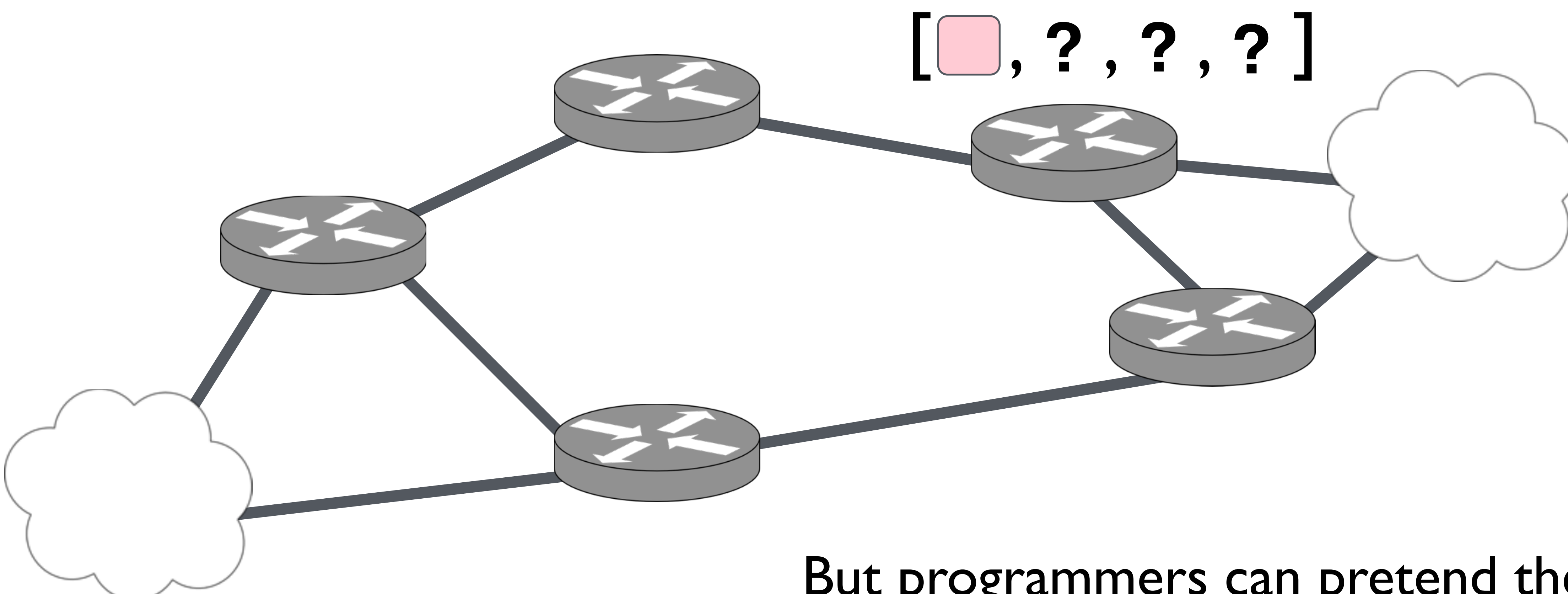


NetKAT: Packet History



NetKAT: Packet History

In practice, packets do not carry their history:



But programmers can pretend they do, leaving it to a compiler to implement this fiction faithfully.

Temporal NetKAT

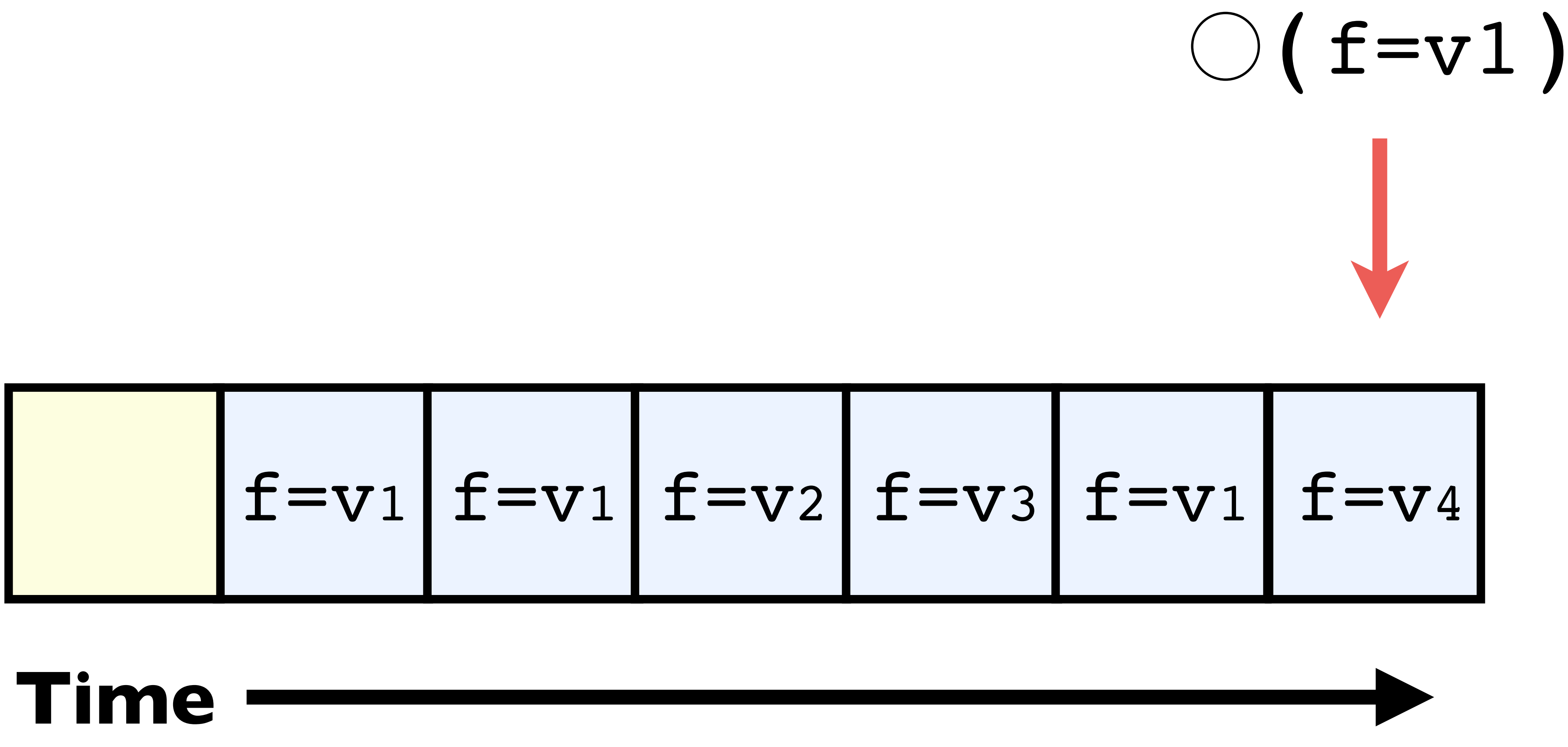
Predicates

a,b ::=	f = n	test	
|	1	identity	
|	0	drop	
|	a + b	or	
|	a · b	and	LTLf
|	¬a	negation	
|	○a	**last**	
|	(a *S* b)	**since**	

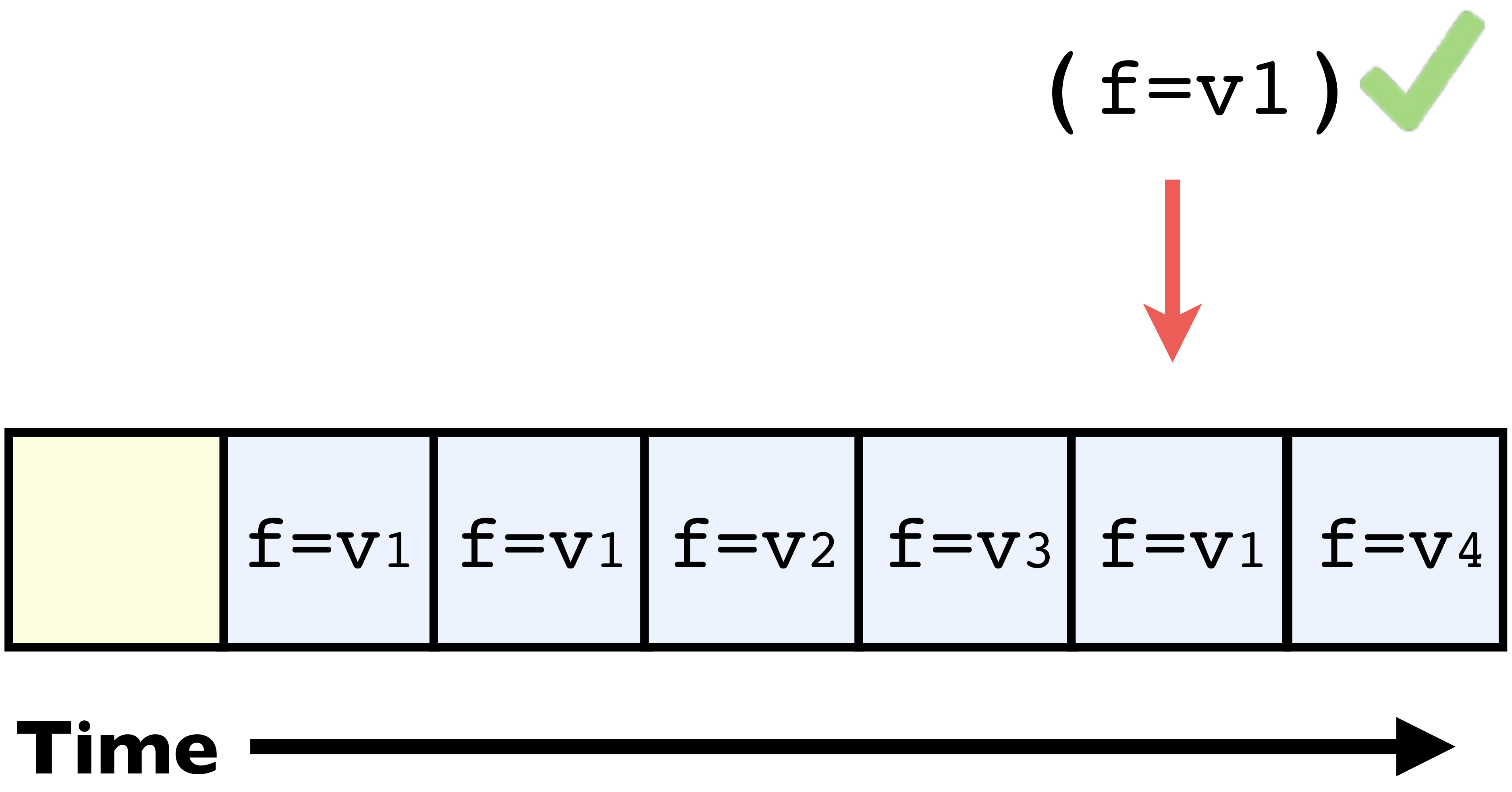
Policies

p,q ::=	a	predicate	
|	f ← v	assignment	
|	p + q	union	Kleene Algebra
|	p · q	sequence	
|	p*	iteration	

Temporal NetKAT

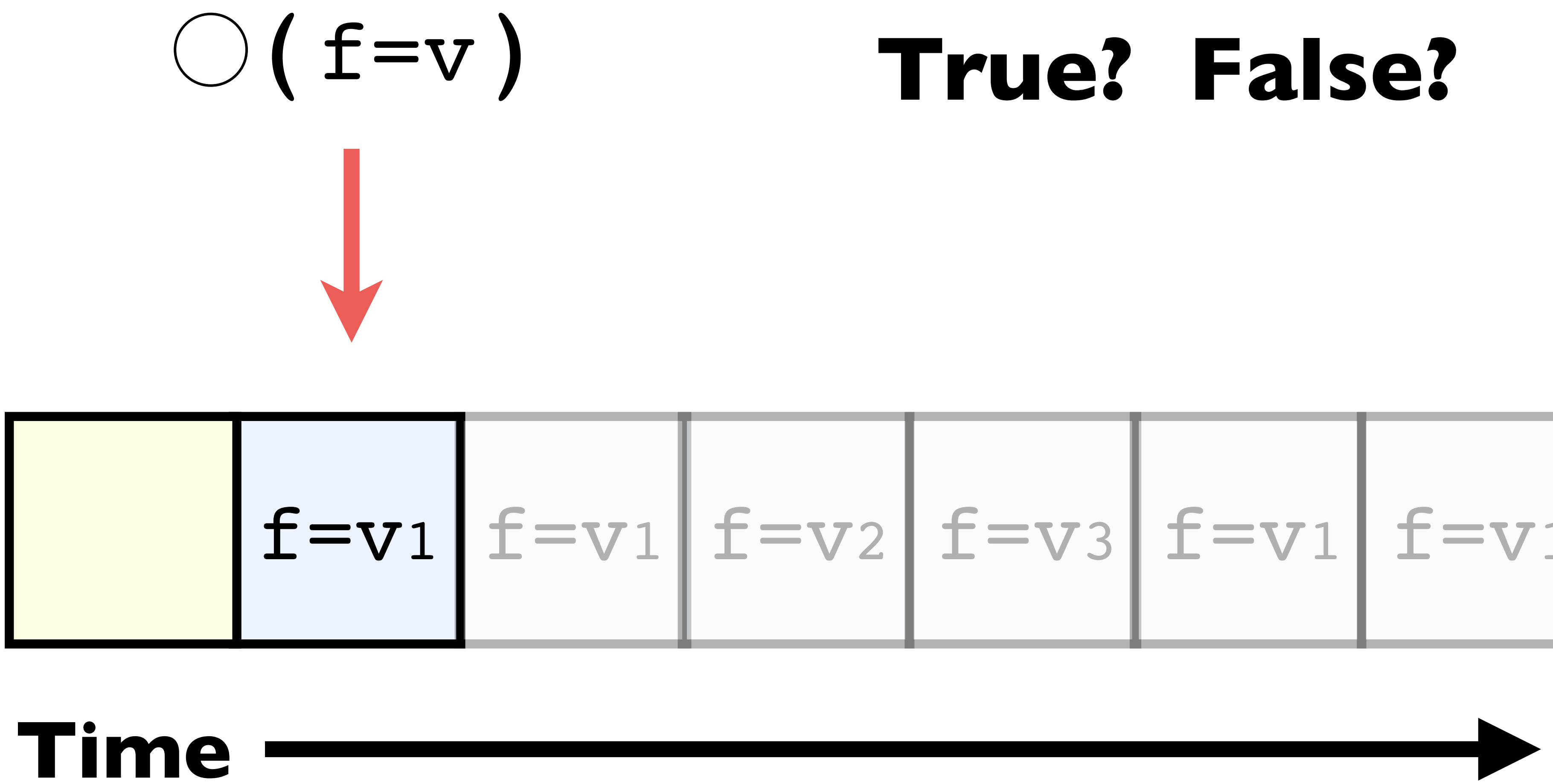


Temporal NetKAT



Temporal NetKAT

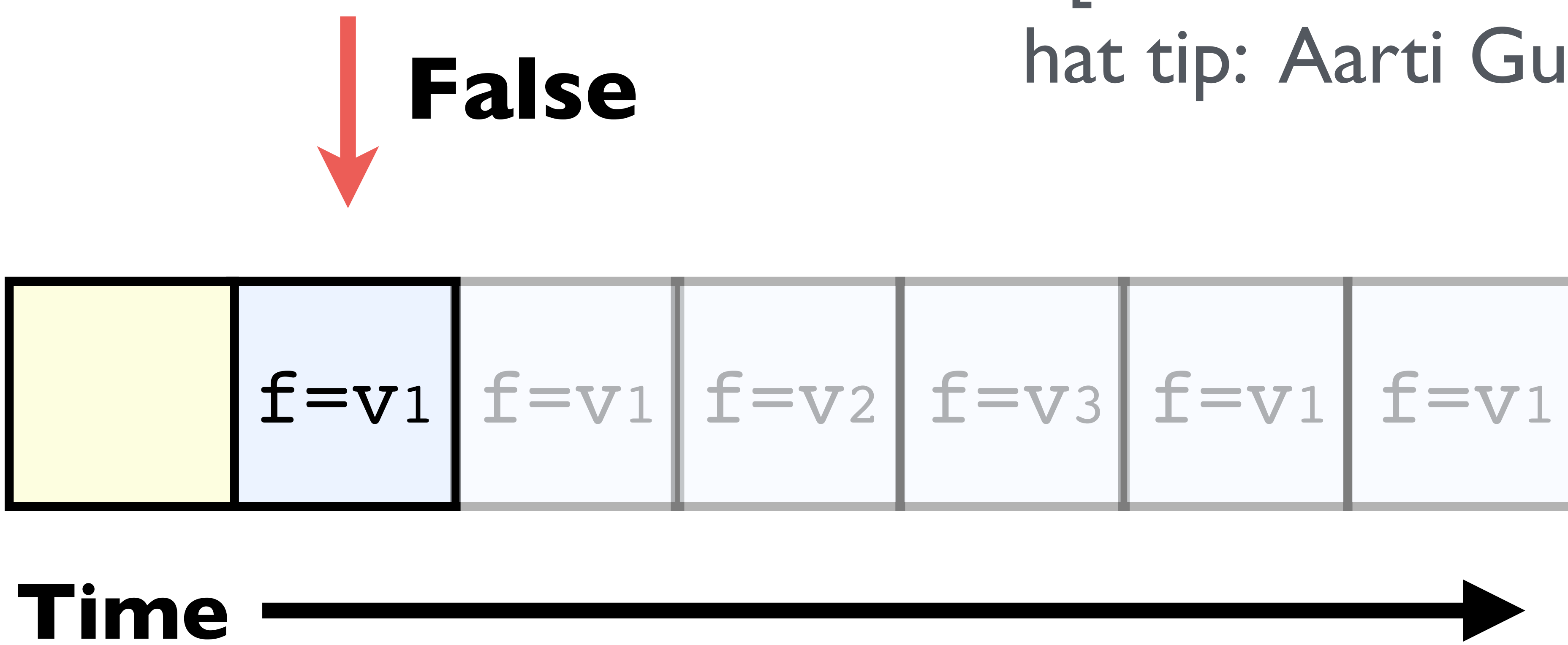
What to do when there is no history?



Temporal NetKAT

What to do when there is no history?

$\bigcirc$(f=v)



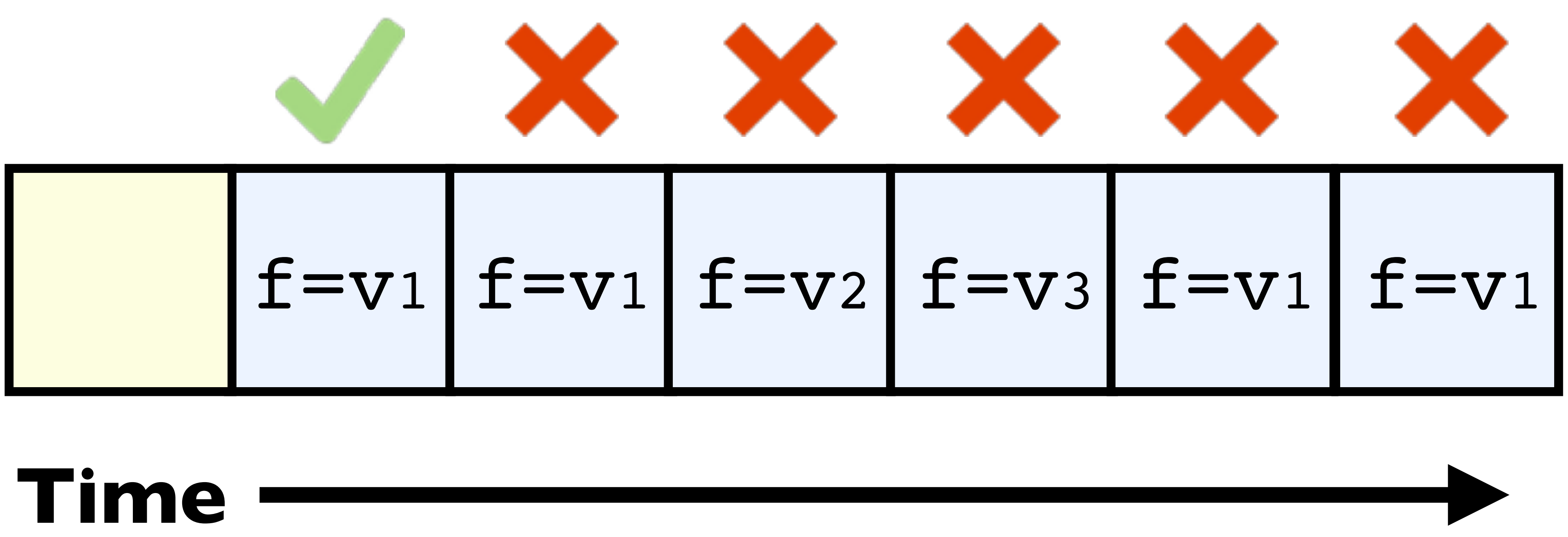
Finite trace semantics

LTLf [Giacomo & Vardi '13]

hat tip: Aarti Gupta

Temporal NetKAT

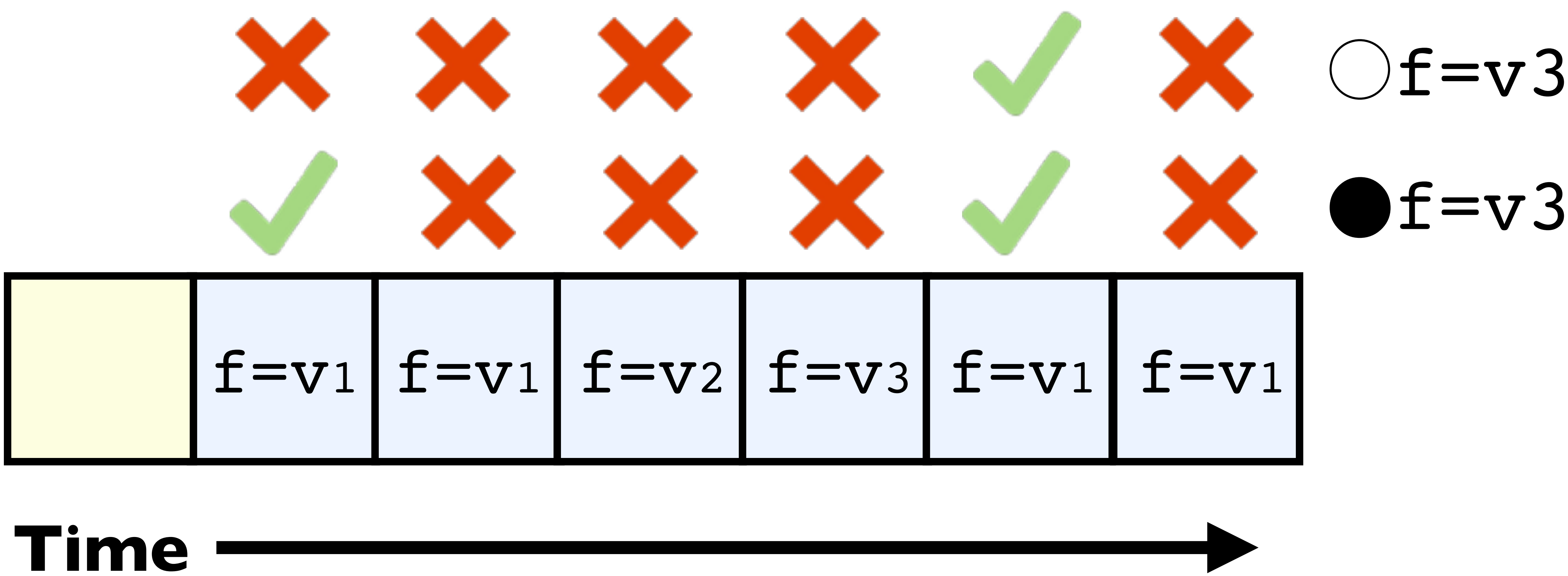
$$\texttt{start} = \neg \bigcirc 1$$



Temporal NetKAT

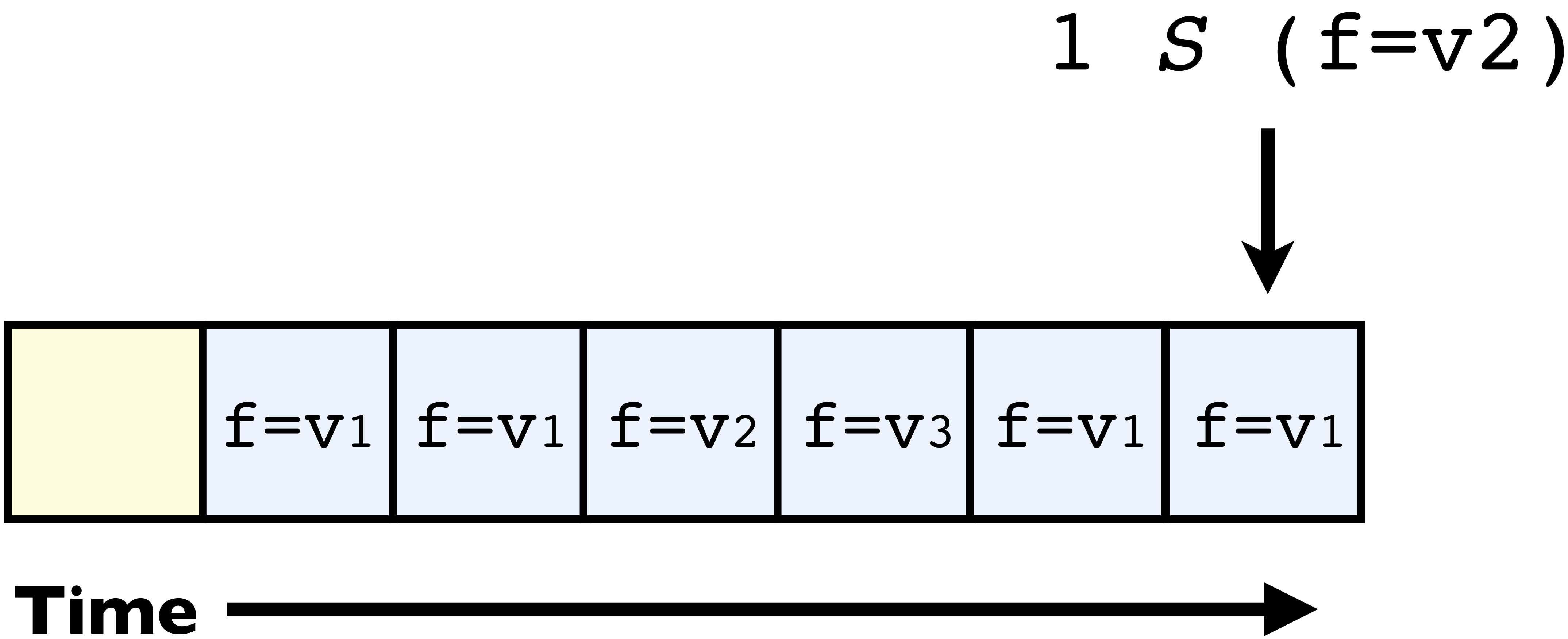
weak last — like last but it succeeds at network entry

$$●a = ¬○¬a$$



Temporal NetKAT

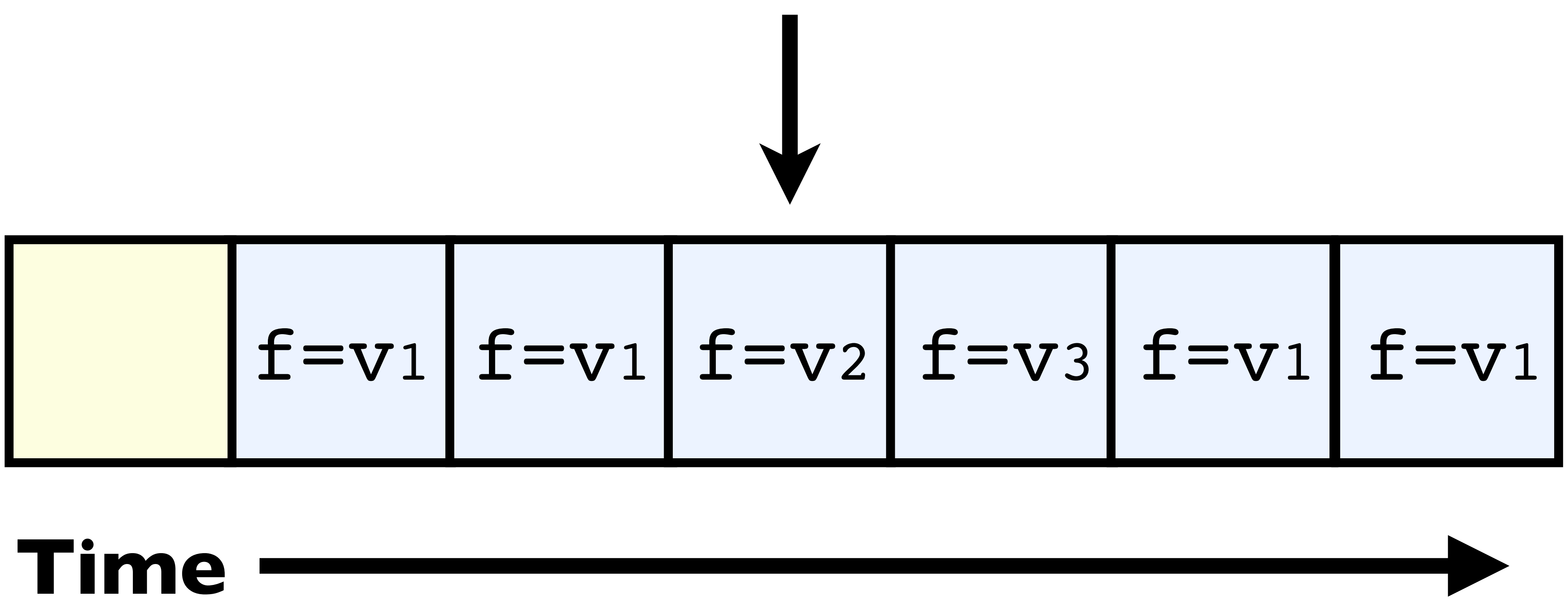
"a since b" a *S* b



Temporal NetKAT

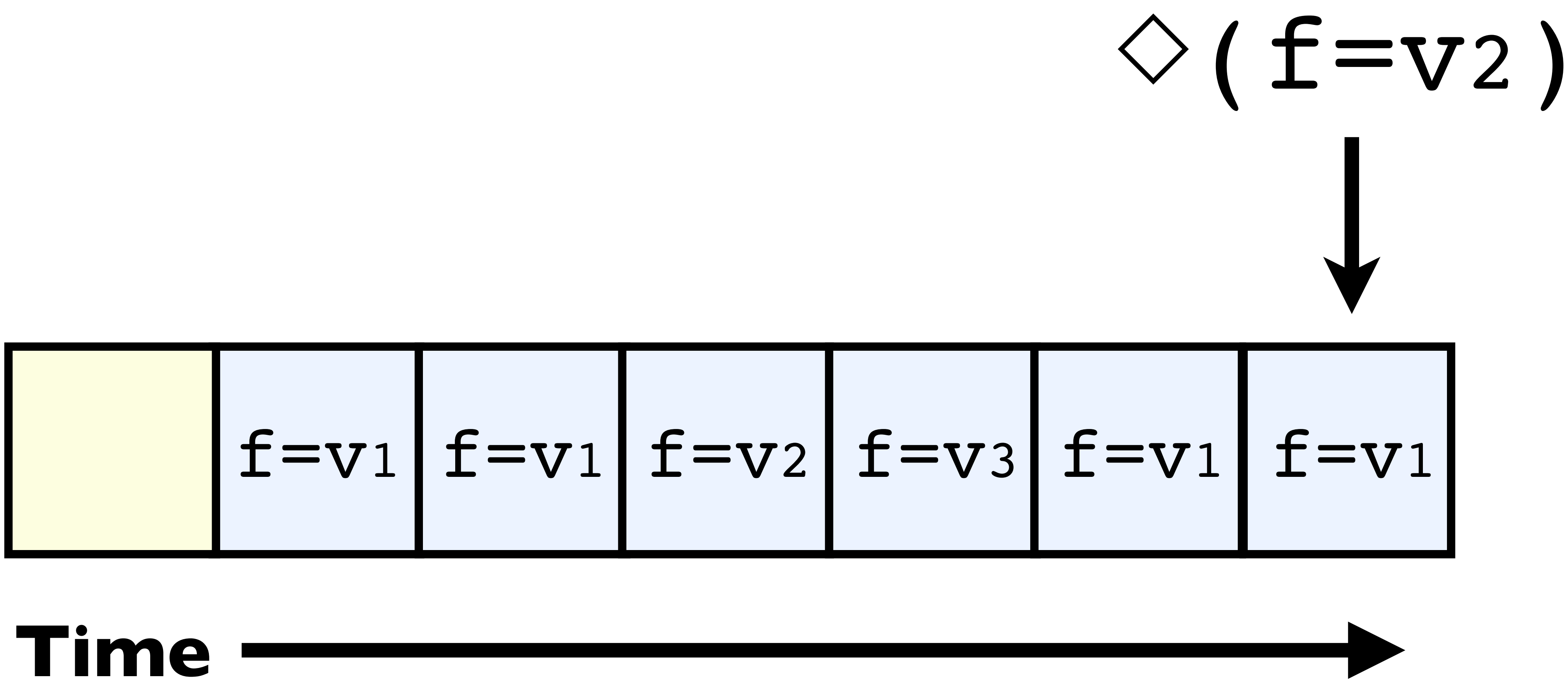
"a since b" a *S* b

1 *S* (f=v2) ✓



Temporal NetKAT

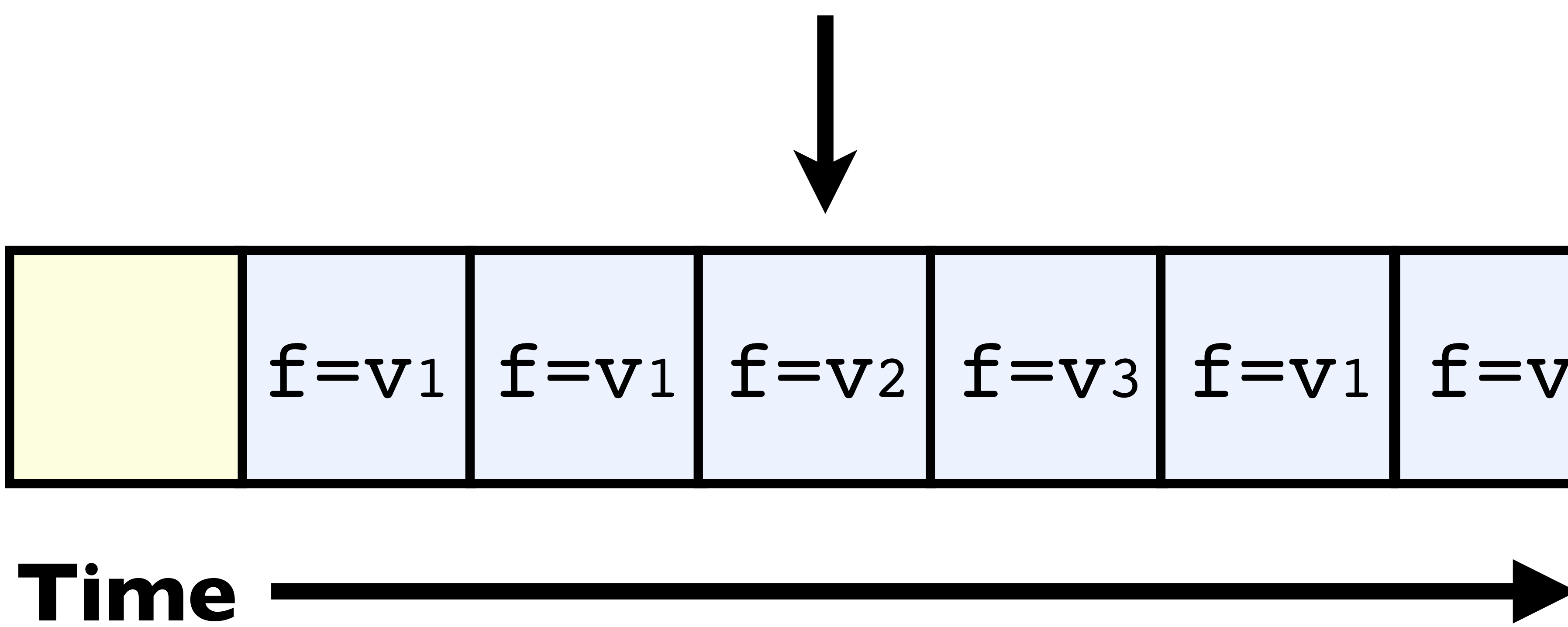
"ever a" $\Diamond \texttt{a} = (\texttt{1}\ \mathcal{S}\ \texttt{a})$



Temporal NetKAT

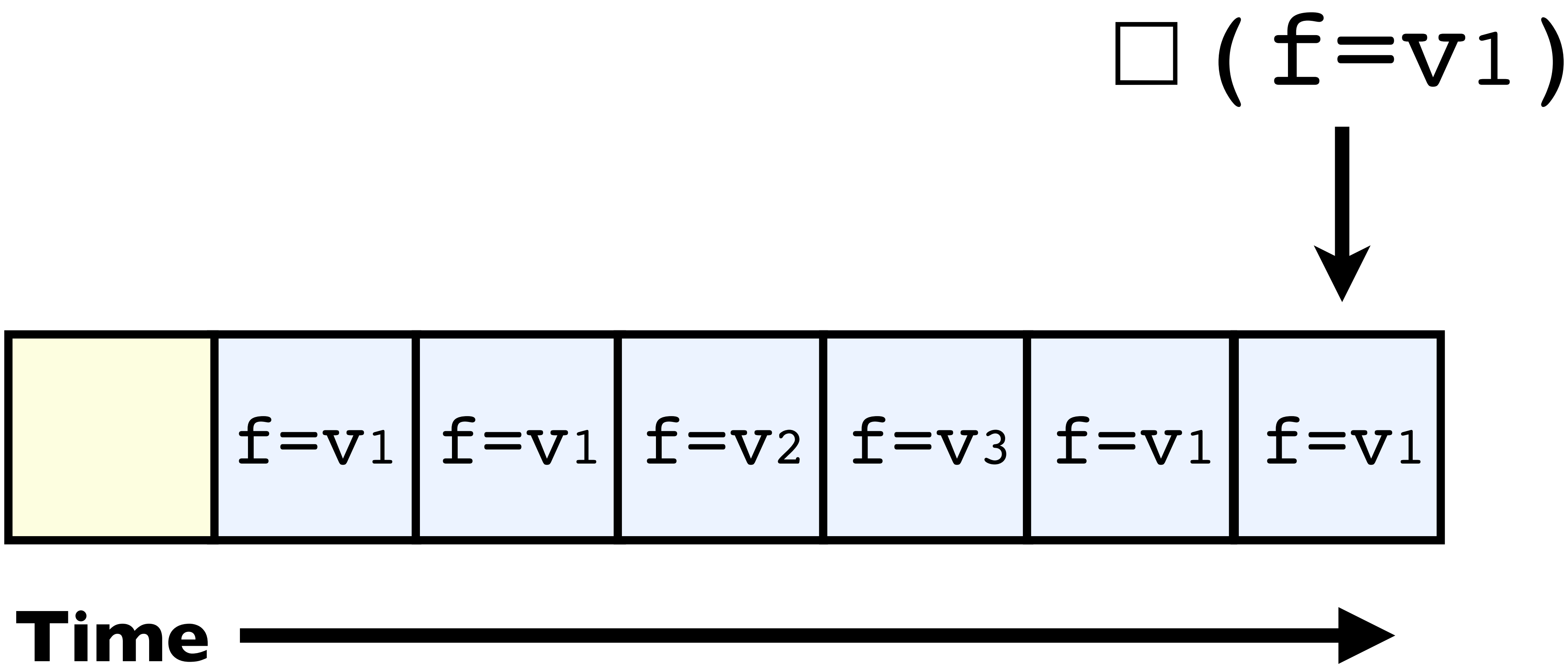
"ever a" $\Diamond a = (1\ S\ a)$

$\Diamond(f{=}v_2)$ ✓



Temporal NetKAT

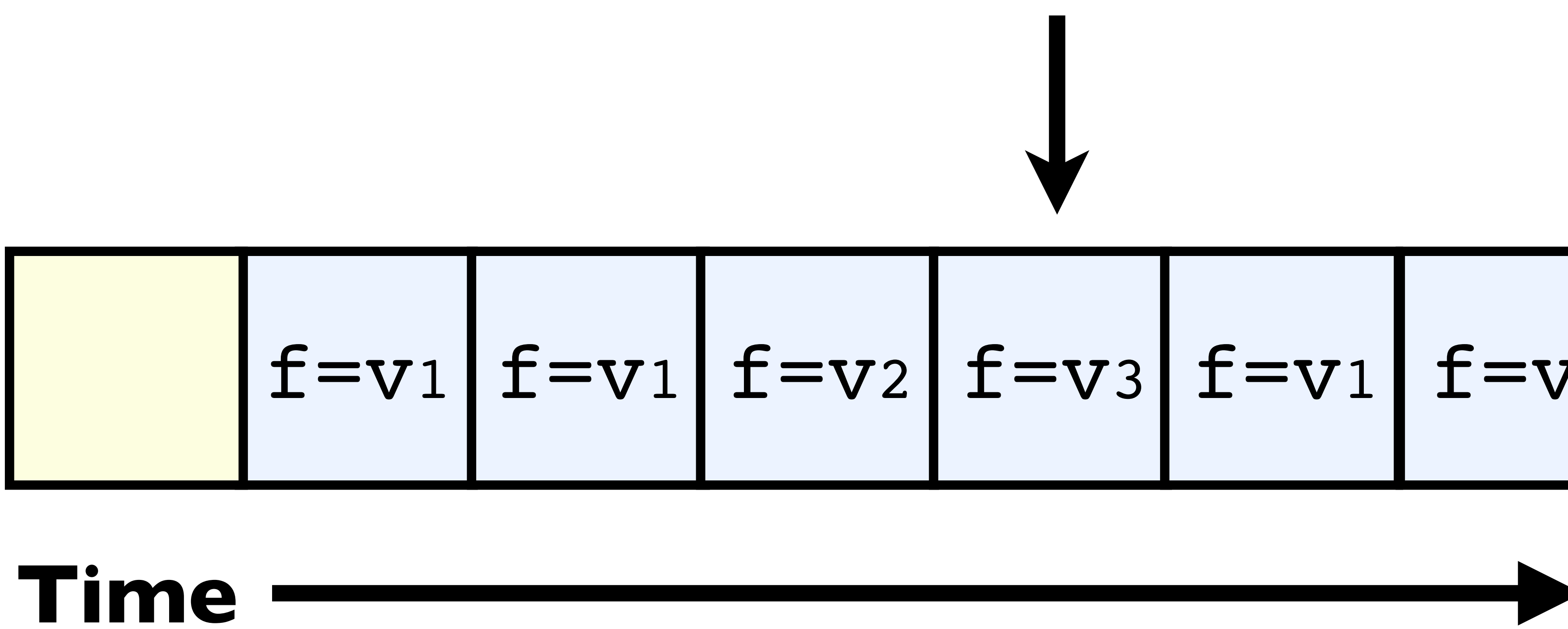
"always a" $\Box a \;=\; \neg\Diamond\neg a$



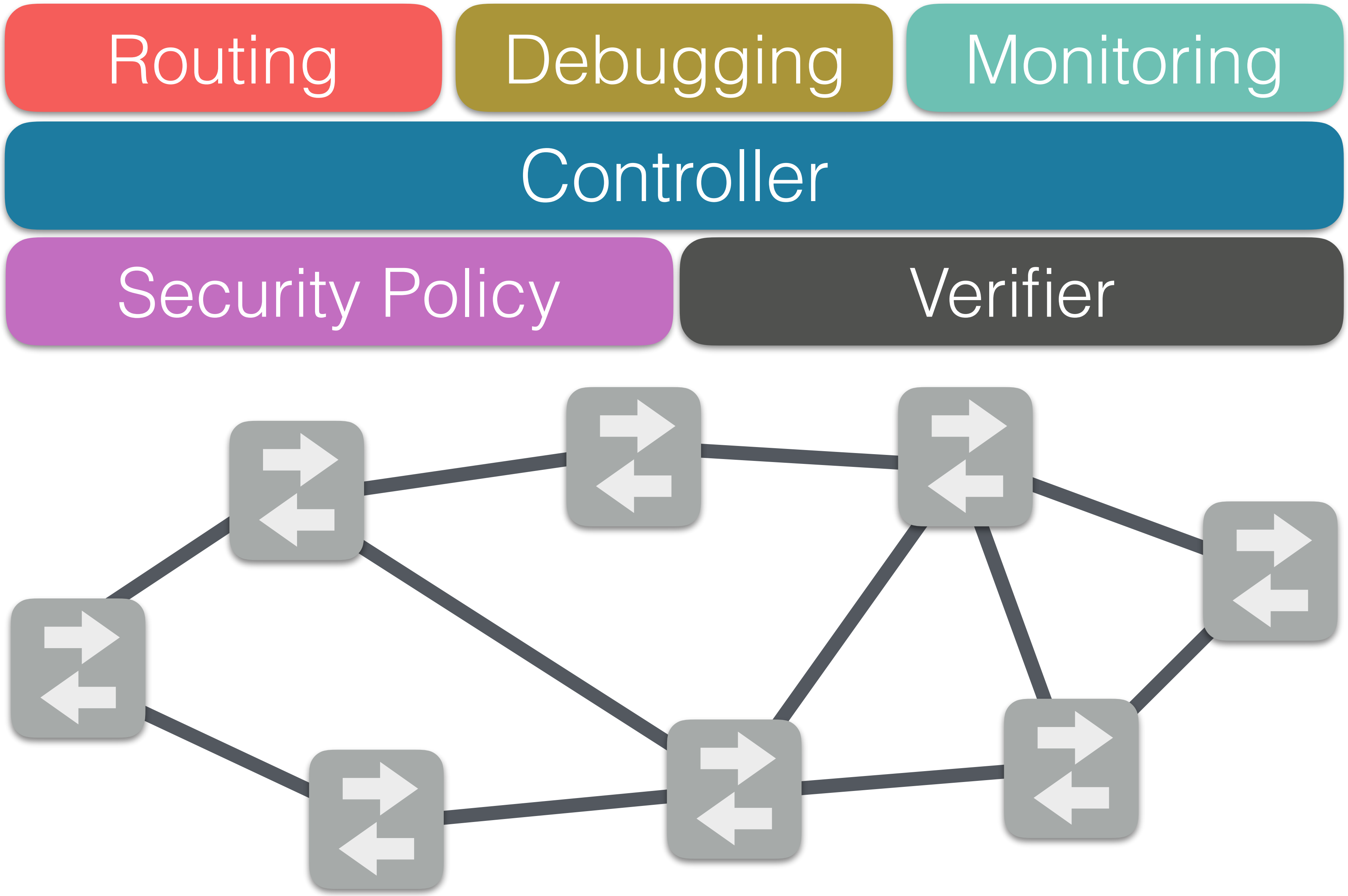
Temporal NetKAT

"always a" $\square a = \neg\Diamond\neg a$

$\square(f=v_1)$ ✗

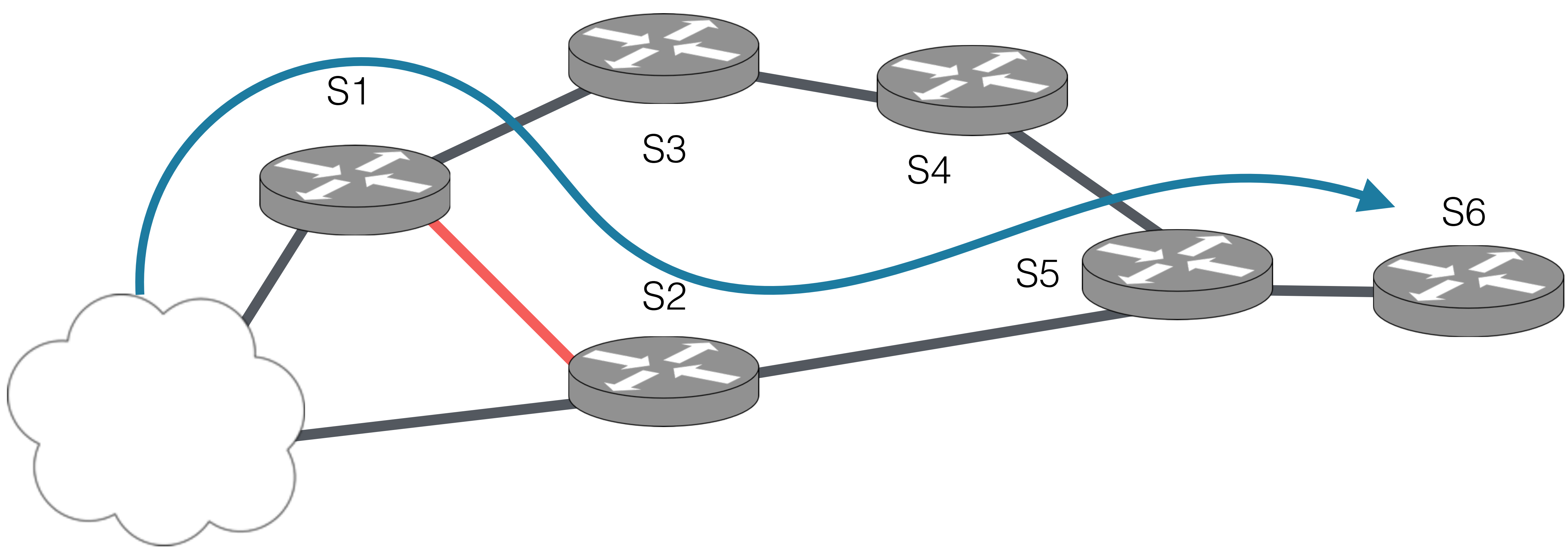


Examples


Routing
Debugging
Monitoring
Controller
Security Policy
Verifier

Example: Debugging/Monitoring

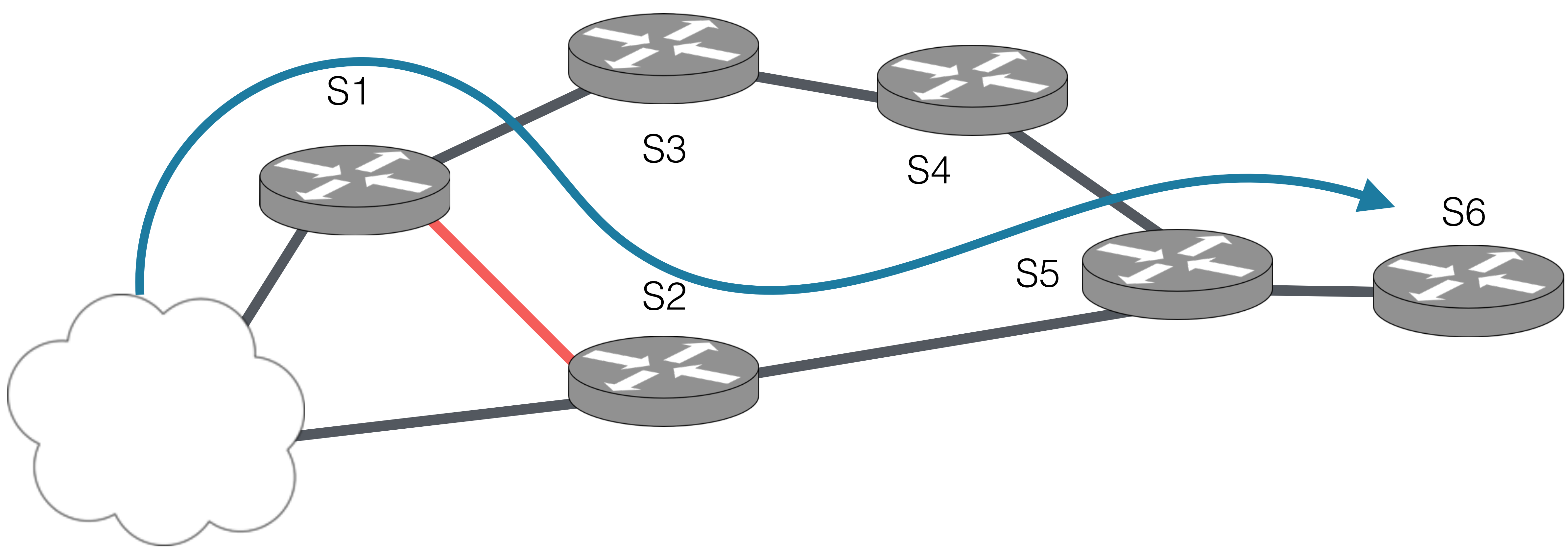
Determine flows utilizing a congested link



Example: Debugging/Monitoring

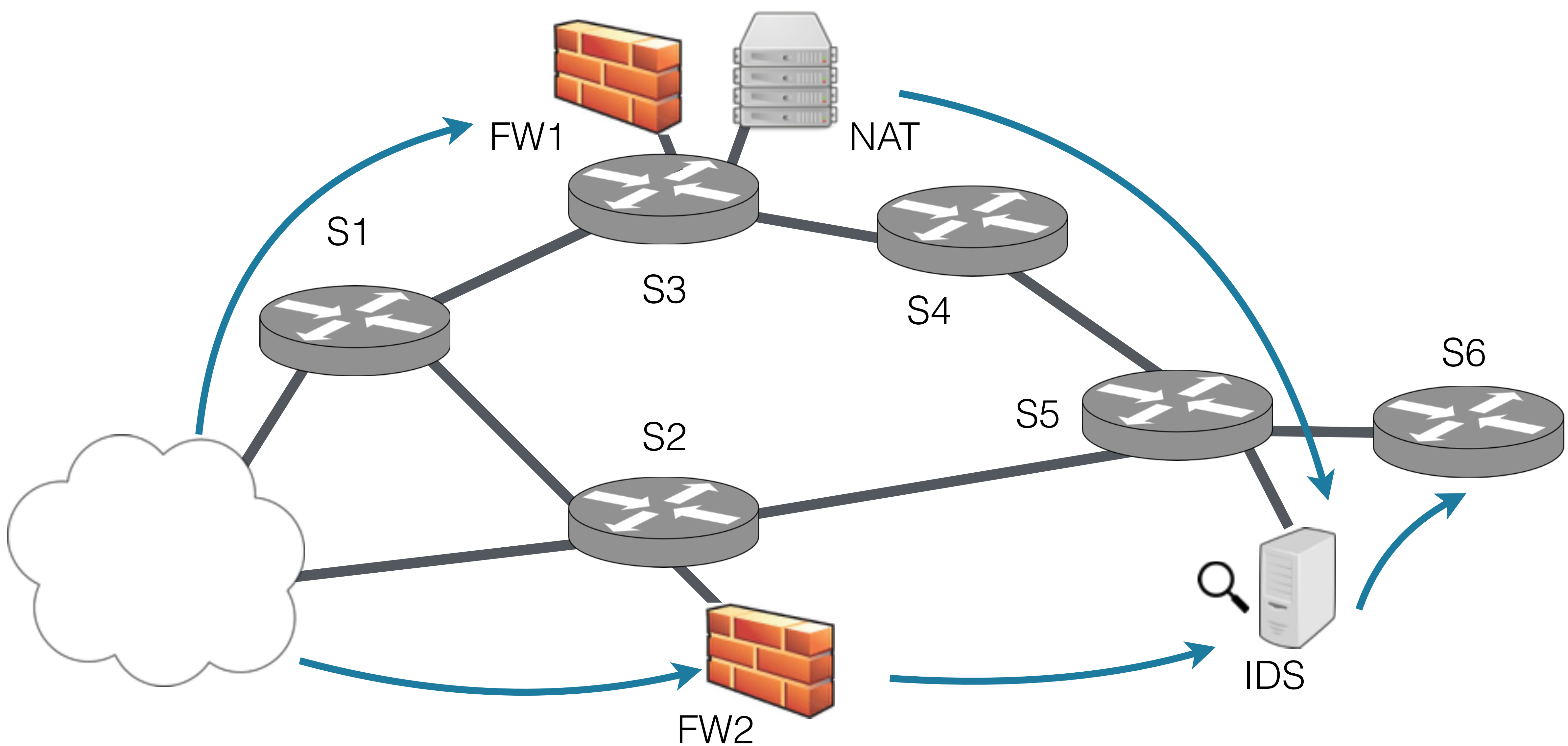
Determine flows utilizing a congested link

```
pol + sw=S6·
        ◇(   sw=S2·
                ○(sw=S1)   )·pt←controller
```



Example: Security

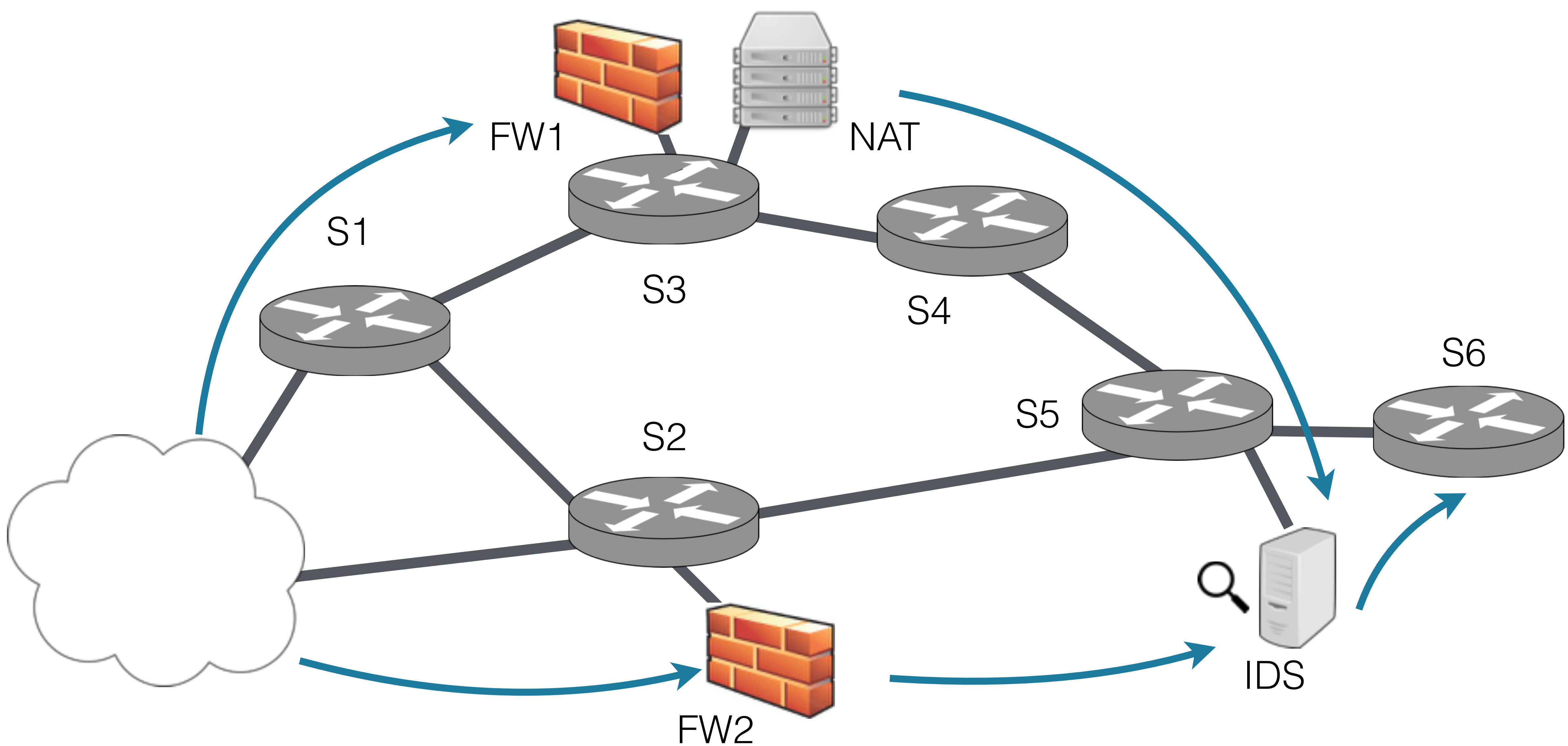
Ensure all traffic arriving at S6 went through a **FW** and **IDS**



Example: Security

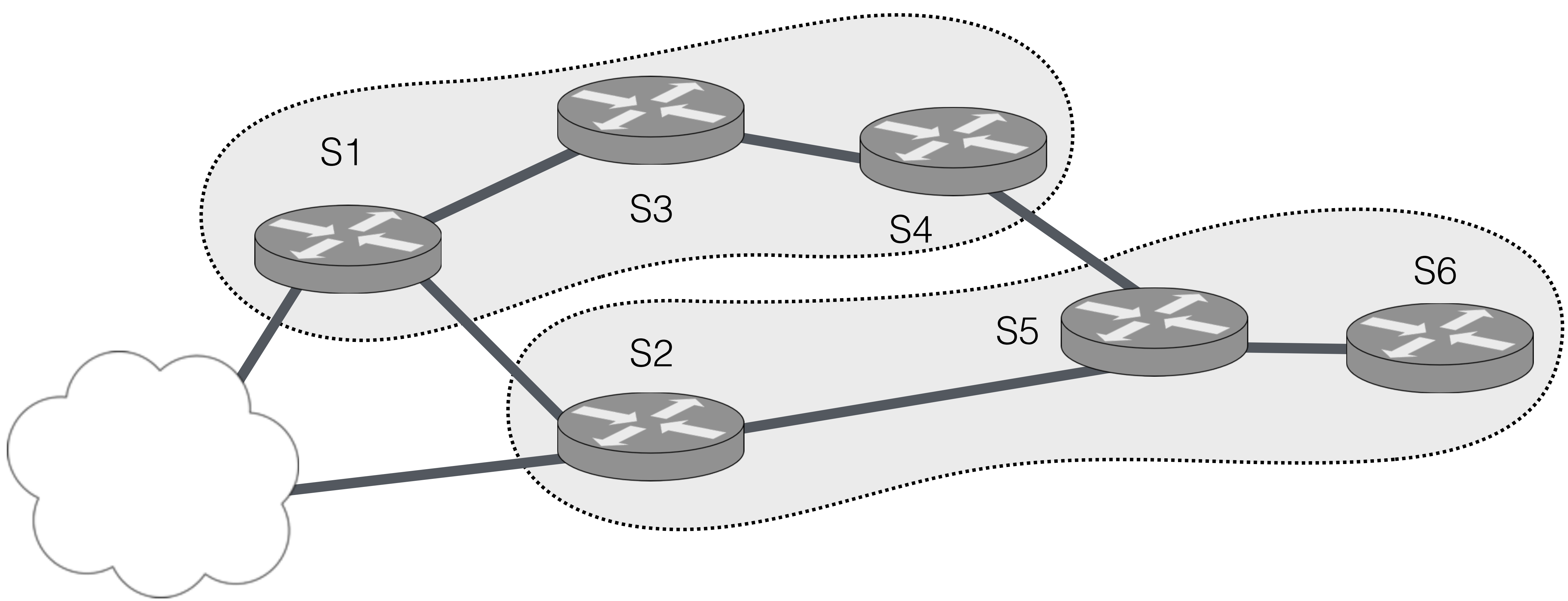
Ensure all traffic arriving at S6 went through a **FW** and **IDS**

`sw=S6 • ◇(sw=FW) • ◇(sw=IDS)`



Example: Isolation

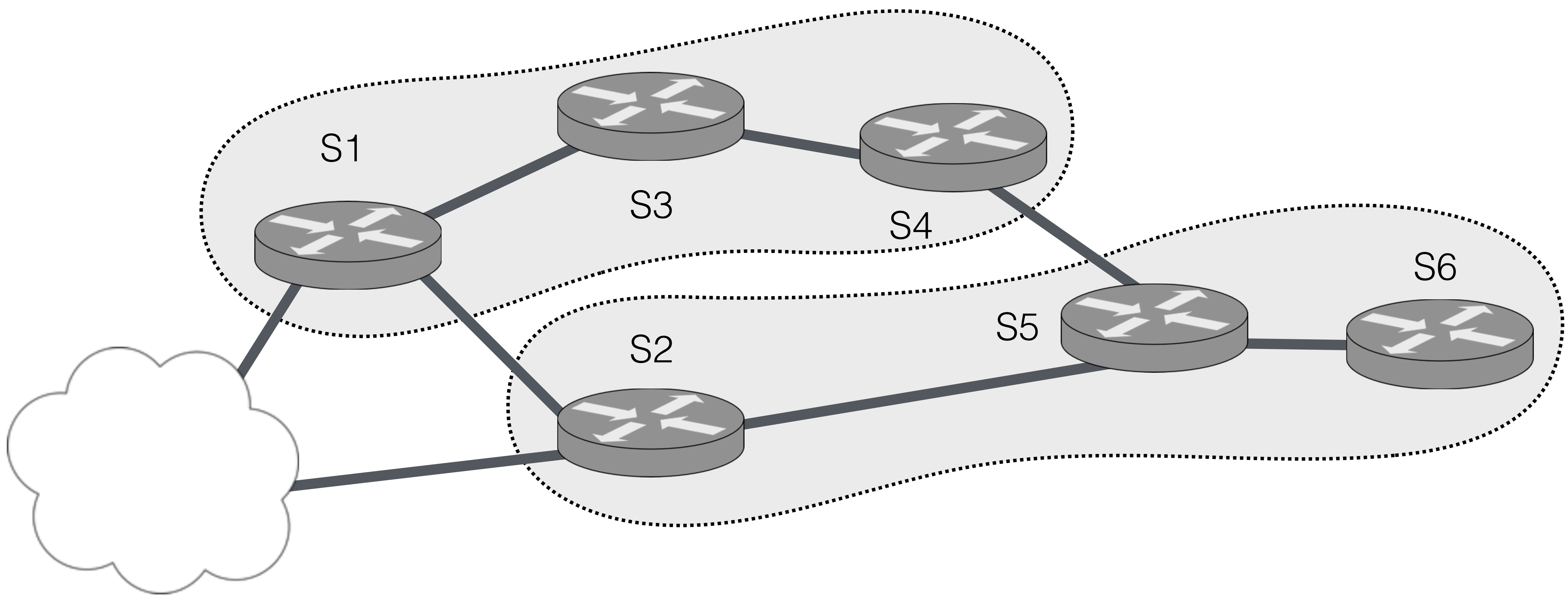
Enforce physical isolation of **S1**, **S3**, **S4** from **S2**, **S5**, **S6**



Example: Isolation

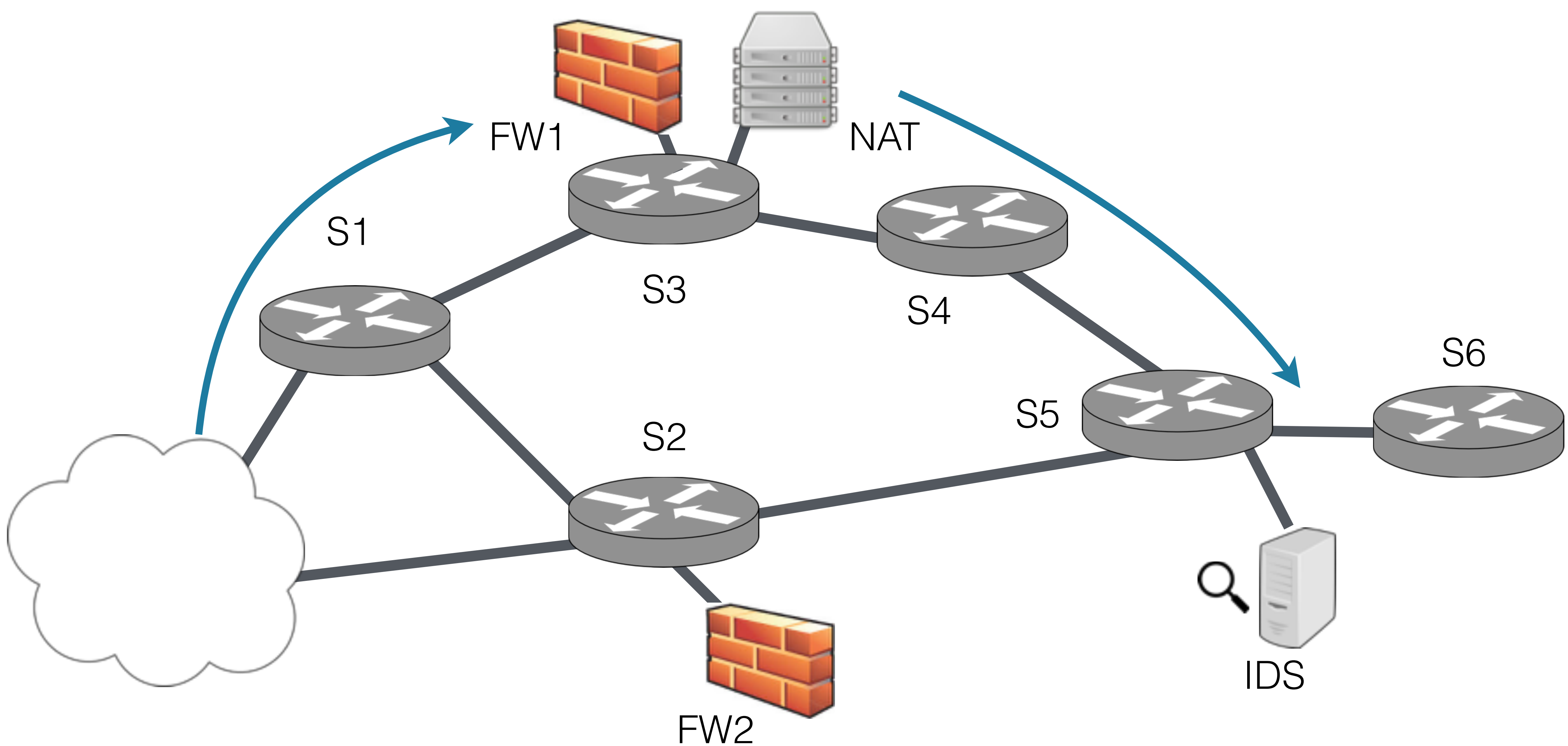
Enforce physical isolation of **S1**, **S3**, **S4** from **S2**, **S5**, **S6**

$$\texttt{pol}\cdot(\Box(sw{=}S_1+sw{=}S_3+sw{=}S_4)\ +\ \Box(sw{=}S_2+sw{=}S_5+sw{=}S_6))$$



Example: Verification

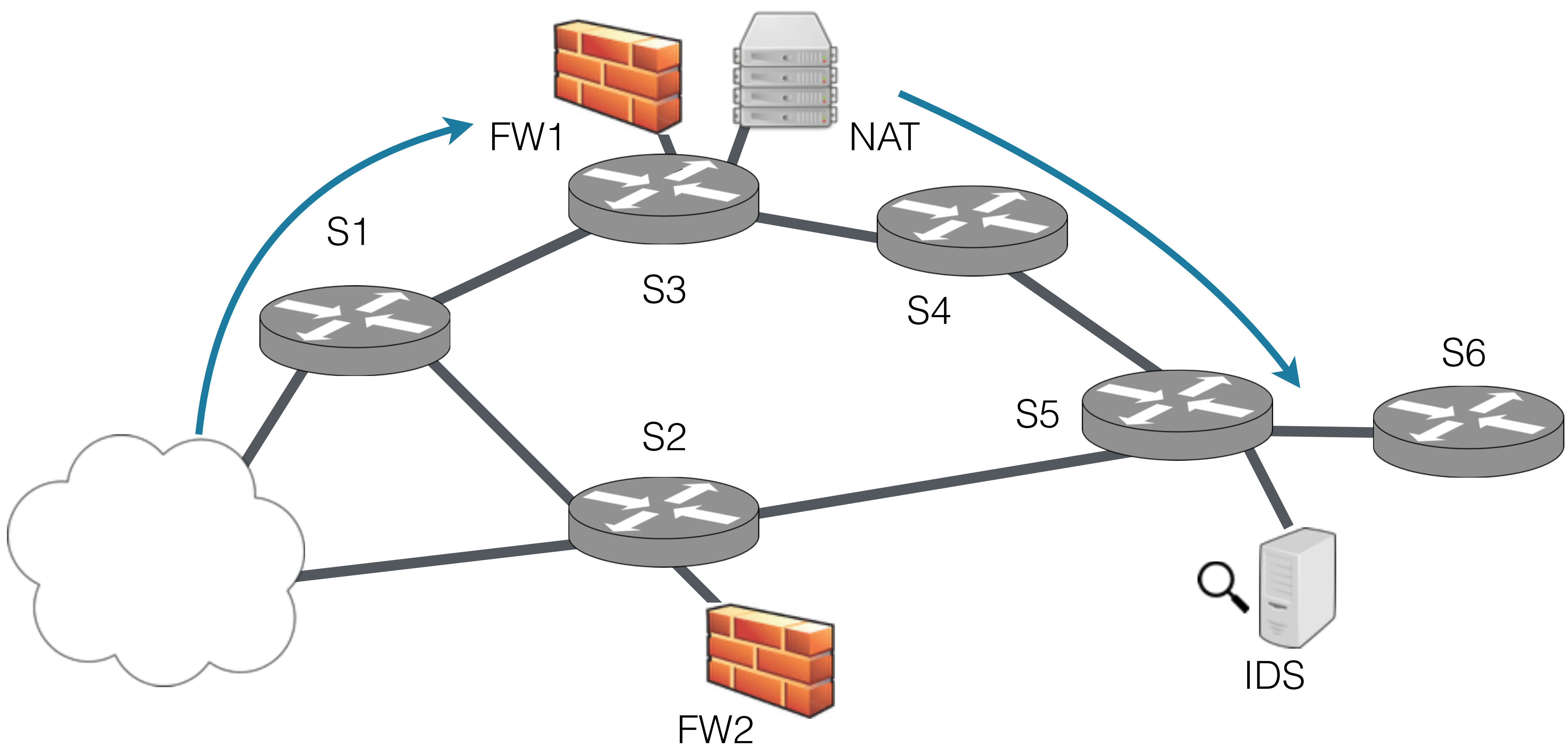
Does the NAT modify the dst IP address to 10.0.0.17?



Example: Verification

Does the NAT modify the dst IP address to 10.0.0.17?

```
pol ≡ pol·((dst=10.0.0.17) S (sw=NAT))
```



Questions

Reasoning

- When are two programs **equivalent**?
- What program transformations are valid?

Compilation

- How to **compile** Temporal NetKAT to switch rules?
- Can we **scale** compilation to realistic topologies/policies?

Reasoning

Equational Theory

Kleene Algebra Axioms

Idempotent Semiring Laws

$(p+q)r \equiv pr+qr$	$p+p \equiv p$
$p+q \equiv q+p$	$1p \equiv p1 \equiv p$
$p+0 \equiv p$	$p0 \equiv 0p \equiv 0$
$p(q+r) \equiv pq+pr$	$p(qr) \equiv (pq)r$
$p+(q+r) \equiv (p+q)+r$	

*Axioms for **

$p^* \equiv 1+pp^*$	$q+px \leq x \Rightarrow p^*q \leq x$
$p^* \equiv 1+p^*p$	$q+px \leq x \Rightarrow p^*q \leq x$

Boolean Algebra Axioms

$aa \equiv a$

$a \cdot \neg a \equiv 0$

$a + 1 \equiv a$

$a + \neg a \equiv 1$

$(p + q)r \equiv pr + qr$

$a + bc \equiv (a + b)(a + c)$

Packet Axioms

$$\sum (f = v) \equiv 1$$

$$(f = v)\cdot(f' = v') \equiv 0$$

$$(f \leftarrow v)\cdot(f = v) \equiv f \leftarrow n$$

$$(f \leftarrow v)\cdot(f' = v') \equiv (f' = v')\cdot(f \leftarrow v)$$

NetKAT

Equational Theory

Kleene Algebra Axioms

Idempotent Semiring Laws

$(p+q)r \equiv pr+qr$ $\qquad$ $p+p \equiv p$

$p+q \equiv q+p$ $\qquad$ $1p \equiv p1 \equiv p$

$p+0 \equiv p$ $\qquad$ $p0 \equiv 0p \equiv 0$

$p(q+r) \equiv pq+pr$ $\qquad$ $p(qr) \equiv (pq)r$

$p+(q+r) \equiv (p+q)+r$

*Axioms for **

$p^* \equiv 1+pp^*$ $\qquad$ $q+px \leq x \Rightarrow p^*q \leq x$

$p^* \equiv 1+p^*p$ $\qquad$ $q+px \leq x \Rightarrow p^*q \leq x$

Packet Axioms

$$\sum (f = v) \equiv 1$$

$$(f = v)\cdot(f' = v') \equiv 0$$

$$(f \leftarrow v)\cdot(f = v) \equiv f \leftarrow n$$

$$(f \leftarrow v)\cdot(f' = v') \equiv (f' = v')\cdot(f \leftarrow v)$$

Boolean Algebra Axioms

$aa \equiv a$

$a\cdot\neg a \equiv 0$

$a + 1 \equiv a$

$a + \neg a \equiv 1$

$(p + q)r \equiv pr + qr$

$a + bc \equiv (a + b)(a + c)$

LTL$_f$ Axioms

$\bullet 1 \equiv 1$

$\bigcirc(a+b) \equiv \bigcirc a + \bigcirc b$

$\bigcirc(a\cdot b) \equiv \bigcirc a \cdot \bigcirc b$

$(a \; S \; b) \equiv b + a\cdot\bigcirc(a \; S \; b)$

$\neg(a \; S \; b) \equiv (\neg b) \; B \; (\neg a\cdot\neg b)$

$\Box a \leq \Diamond(start\cdot a)$

$(a \leq \bullet a\cdot b) \Rightarrow (a \leq \Box a)$

Equational Theory

Kleene Algebra Axioms

Idempotent Semiring Laws

$(p+q)r \equiv pr+qr$ $\qquad$ $p+p \equiv p$

$p+q \equiv q+p$ $\qquad$ $1p \equiv p1 \equiv p$

$p+0 \equiv p$ $\qquad$ $p0 \equiv 0p \equiv 0$

$p(q+r) \equiv pq+pr$ $\qquad$ $p(qr) \equiv (pq)r$

$p+(q+r) \equiv (p+q)+r$

*Axioms for **

$p^* \equiv 1+pp^*$ $\qquad$ $q+px \leq x \Rightarrow p^*q \leq x$

$p^* \equiv 1+p^*p$ $\qquad$ $q+px \leq x \Rightarrow p^*q \leq x$

Boolean Algebra Axioms

$aa \equiv a$

$a \cdot \neg a \equiv 0$

$a + 1 \equiv a$

$a + \neg a \equiv 1$

$(p + q)r \equiv pr + qr$

$a + bc \equiv (a + b)(a + c)$

LTLf Axioms

$\bullet 1 \equiv 1$

$\bigcirc(a+b) \equiv \bigcirc a + \bigcirc b$

$\bigcirc(a \cdot b) \equiv \bigcirc a \cdot \bigcirc b$

$(a \; S \; b) \equiv b + a \cdot \bigcirc(a \; S \; b)$

$\neg(a \; S \; b) \equiv (\neg b) \; B \; (\neg a \cdot \neg b)$

$\Box a \leq \Diamond(start \cdot a)$

$(a \leq \bullet a \cdot b) \Rightarrow (a \leq \Box a)$

Packet Axioms

$\sum (f = v) \equiv 1$

$(f = v) \cdot (f' = v') \equiv 0$

$(f \leftarrow v) \cdot (f = v) \equiv f \leftarrow n$

$(f \leftarrow v) \cdot (f' = v') \equiv (f' = v') \cdot (f \leftarrow v)$

Step Axiom

$(f \leftarrow v) \cdot \bigcirc a \equiv a \cdot (f \leftarrow v)$

Metatheory

NetKAT:

Soundness: If $\vdash p \equiv q$, then $[\![p]\!] = [\![q]\!]$

Completeness: If $[\![p]\!] = [\![q]\!]$, then $\vdash p \equiv q$

Temporal NetKAT:

Soundness: If $\vdash p \equiv q$, then $[\![p]\!] = [\![q]\!]$

Completeness: If $[\![\text{start}\cdot p]\!] = [\![\text{start}\cdot q]\!]$, then $\vdash \text{start}\cdot p \equiv \text{start}\cdot q$

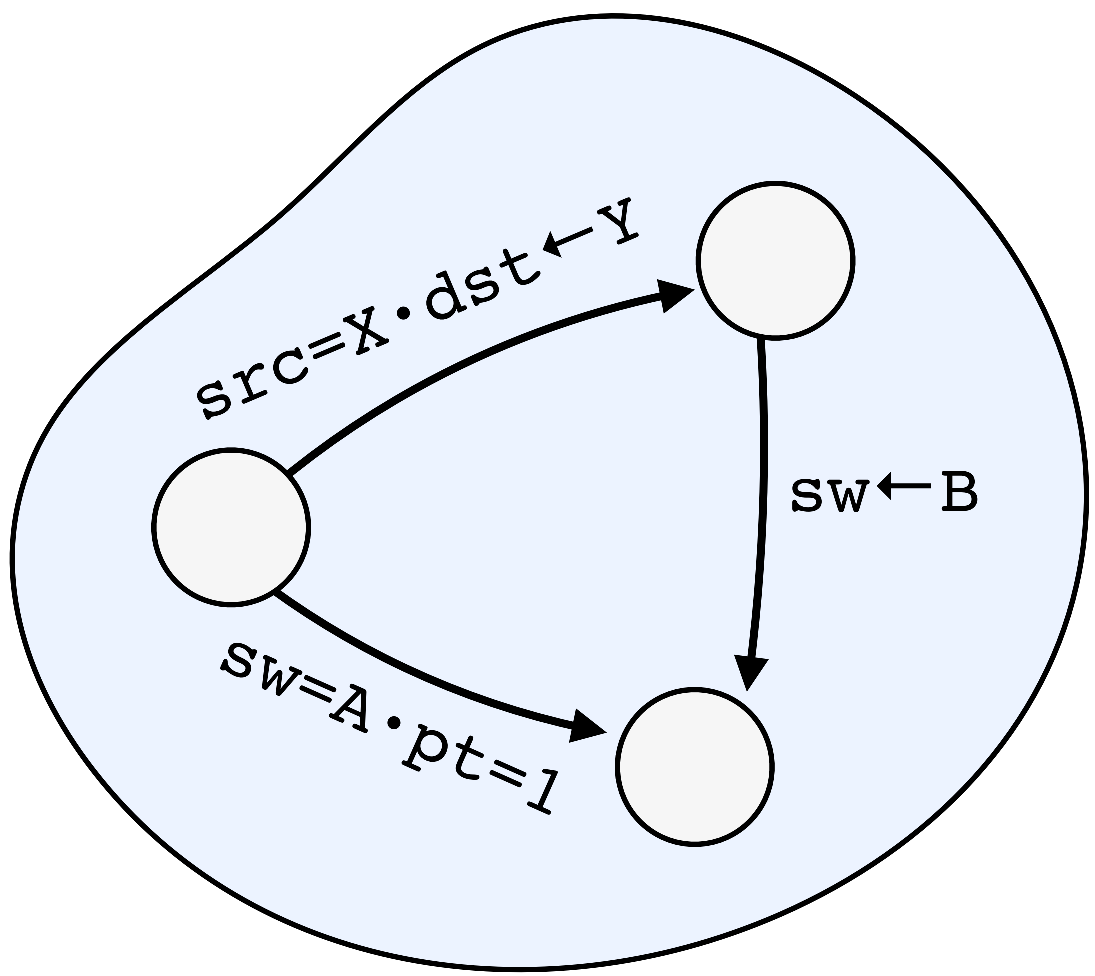
- Completeness for network-wide policies
- Normalization reduces Temporal NetKAT terms to NetKAT
- Interesting induction invariant — talk to Ryan!

Compilation

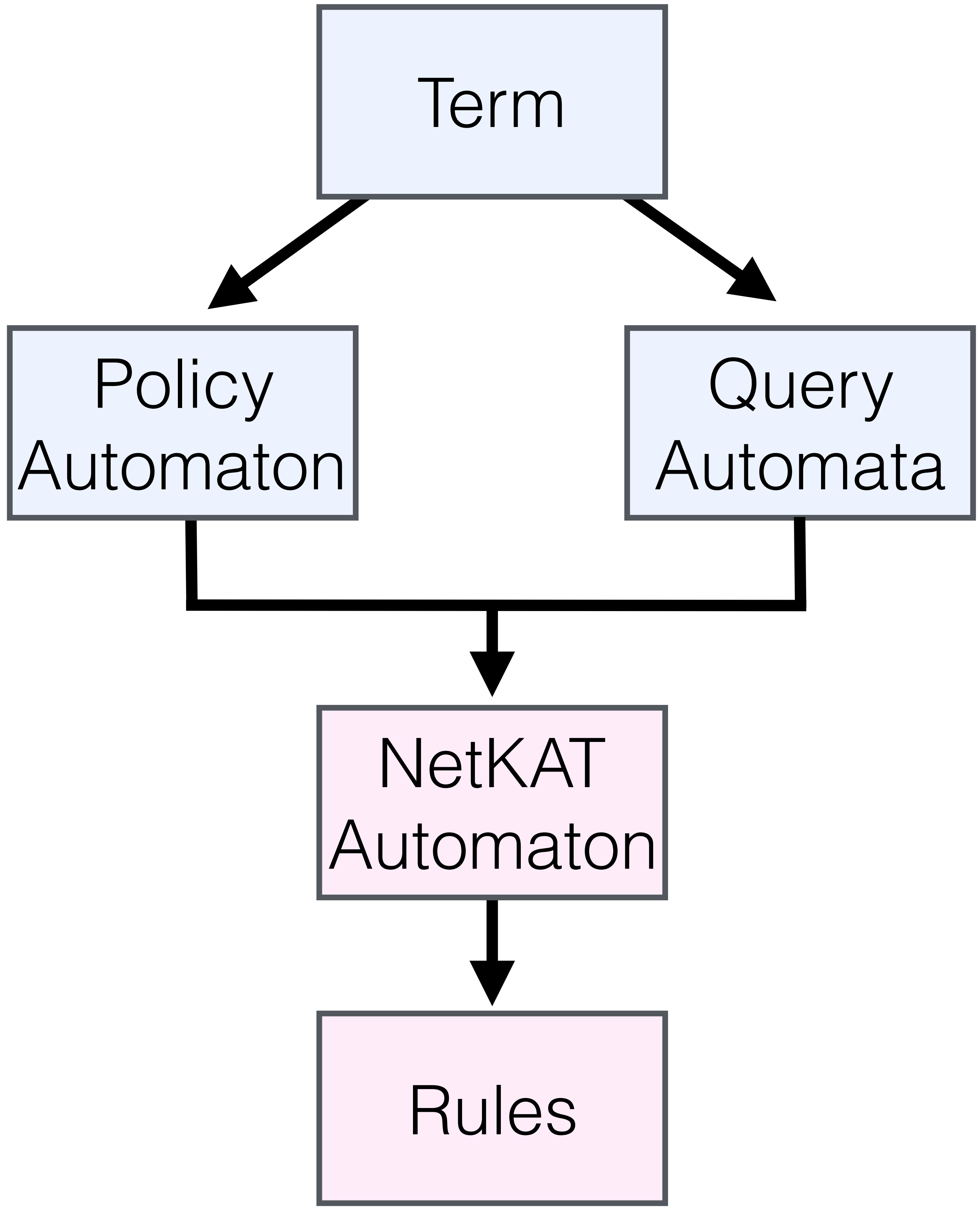
Compilation

A Fast Compiler for NetKAT [Smolka et al '15]

- Translates NetKAT policies into symbolic NetKAT automata
- Represents the transition function using FDDs,
  a variant of BDDs
- Generates packet-processing rules

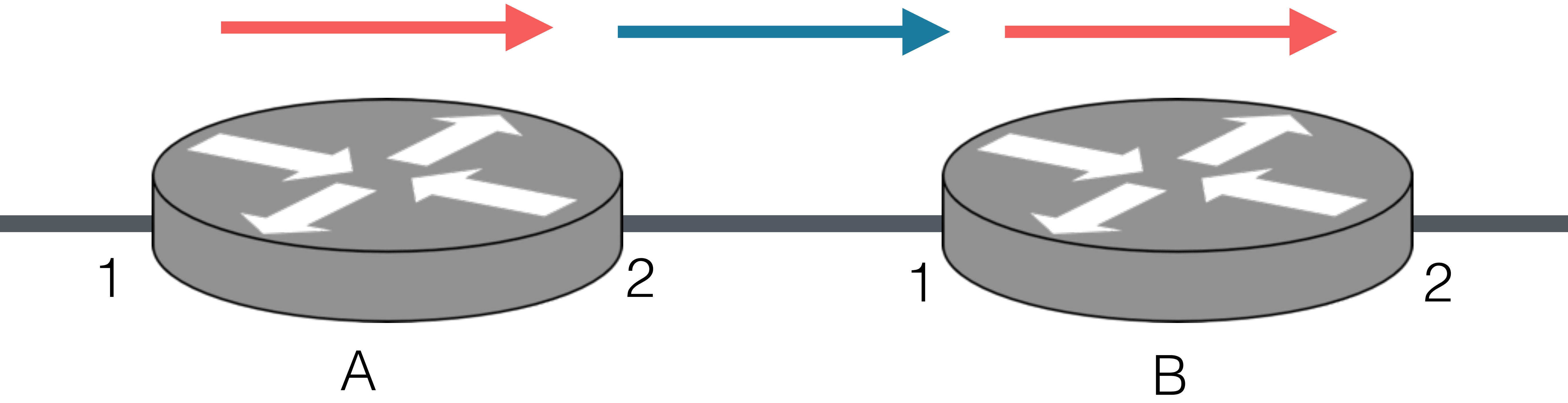


Compilation



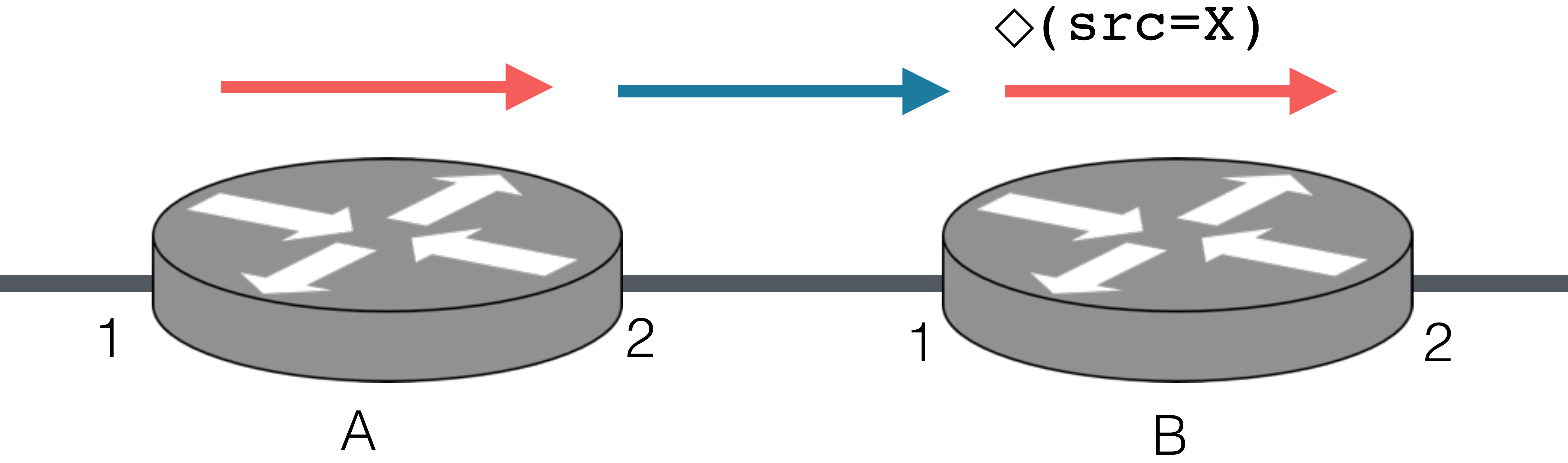
Compilation: Example

```
polA = (sw=A·pt=1)·(pt←2)
link = (sw←B)·(pt←1)
polB = (sw=B·pt=1)·(pt←2)
```



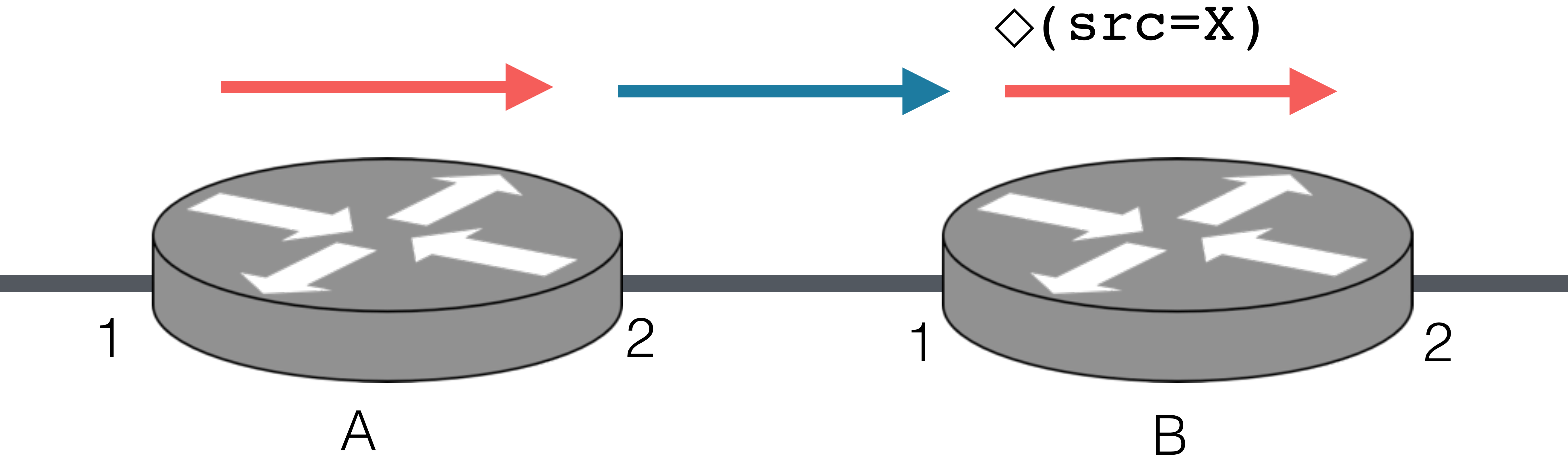
Compilation: Example

```
polA  = (sw=A·pt=1)·(pt←2)
link  = (sw←B)·(pt←1)
polB  = (sw=B·pt=1)·(pt←2)
pol   = polA·link·◇(src=X)·polB
```



Compilation: Example

$$\mathtt{pol_A} \cdot \mathtt{link} \cdot \Diamond(\mathtt{src=X}) \cdot \mathtt{pol_B}$$



Compilation: Example

$\texttt{pol}_{\texttt{A}} \cdot \texttt{link} \cdot \Diamond(\texttt{src=X}) \cdot \texttt{pol}_{\texttt{B}}$

Compilation: Example

$\texttt{pol}_{A} \cdot \texttt{link} \cdot \Diamond(\texttt{src=X}) \cdot \texttt{pol}_{B}$

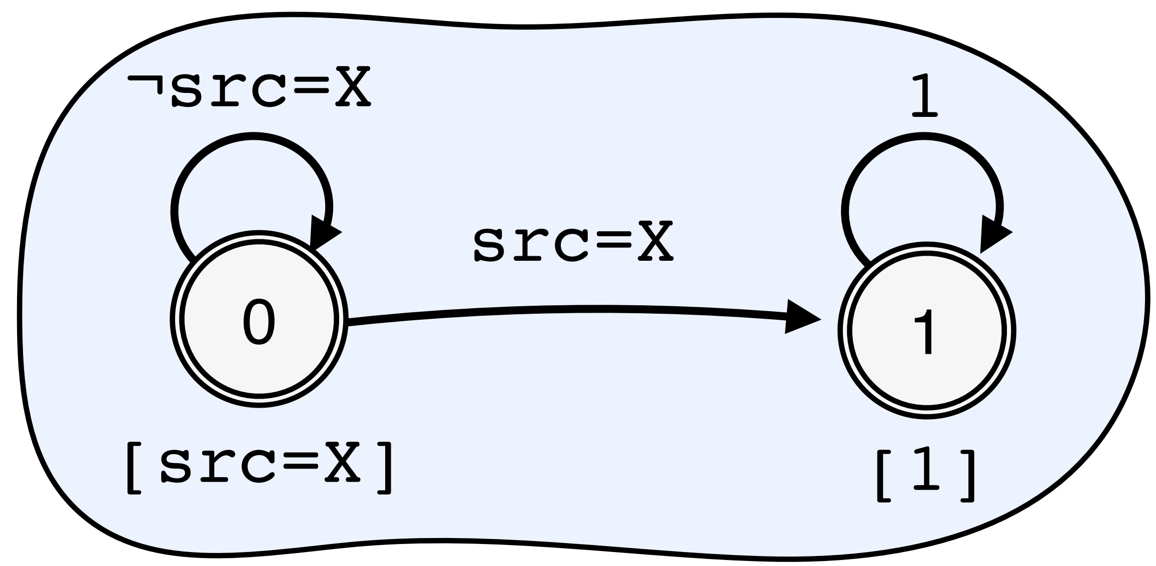
↓ *abstract predicate*

$\texttt{pol}_{A} \cdot \texttt{link} \cdot \alpha \cdot \texttt{pol}_{B}$

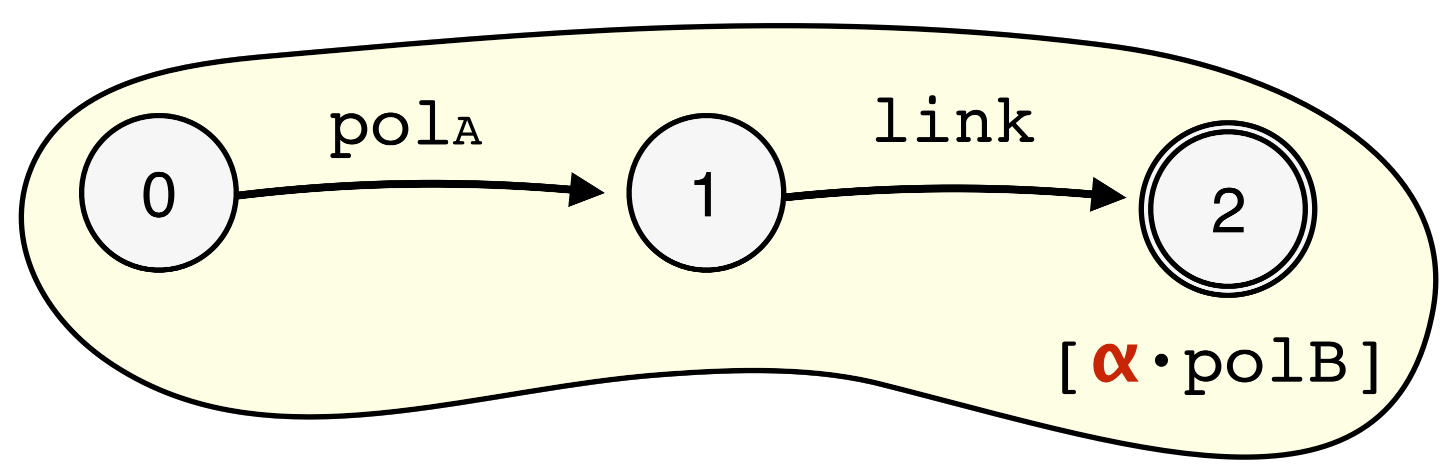
Compilation: Example

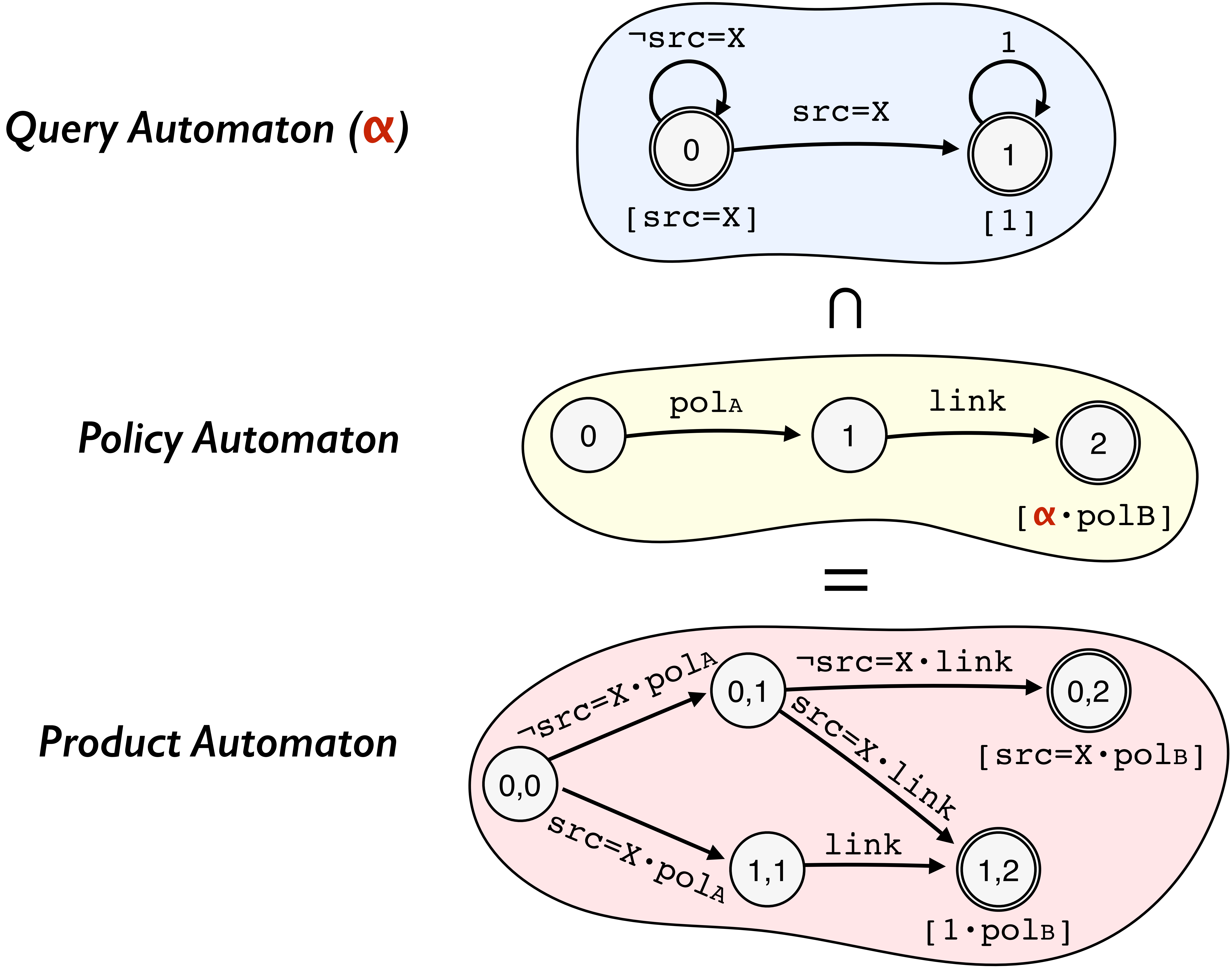
pol$_A$ · link · α · pol$_B$

Query Automaton (α)



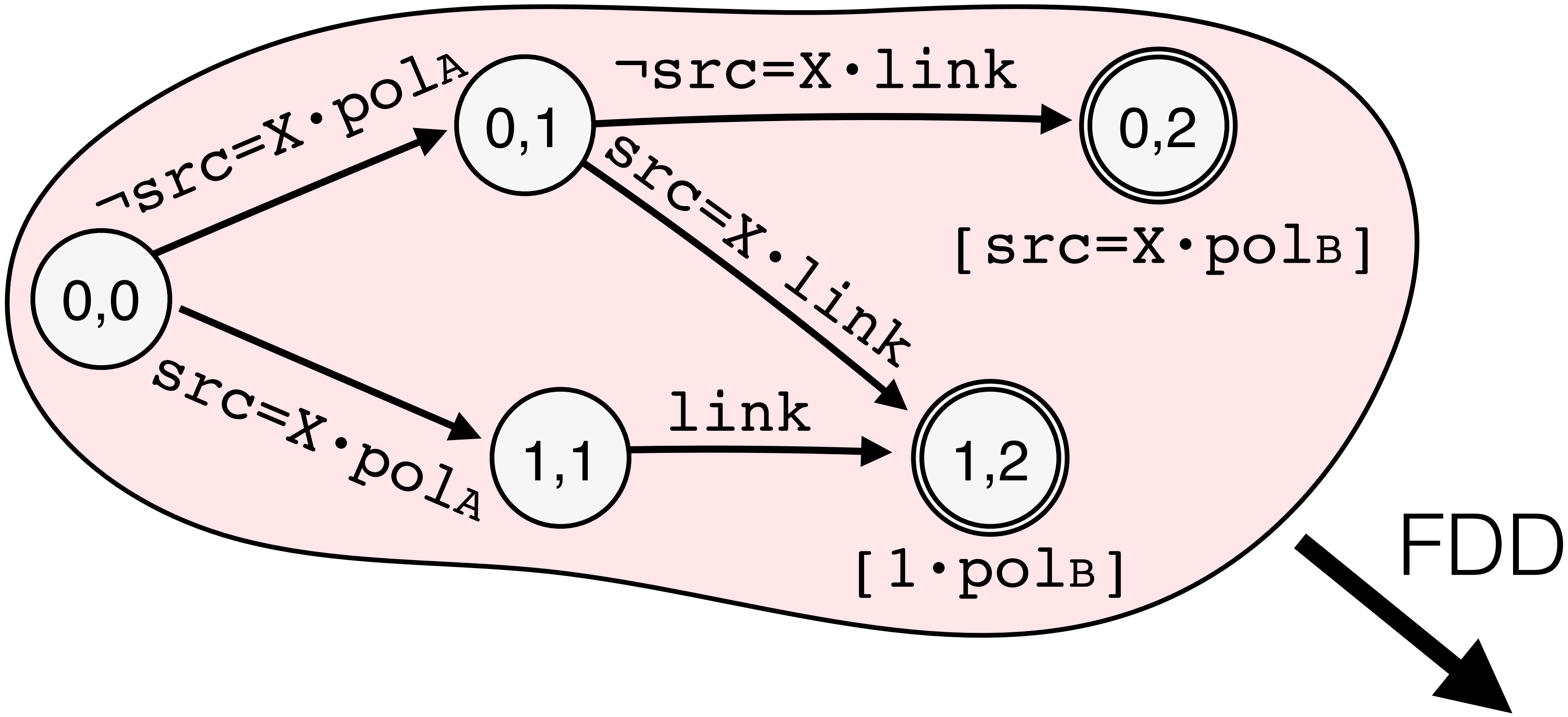
Policy Automaton

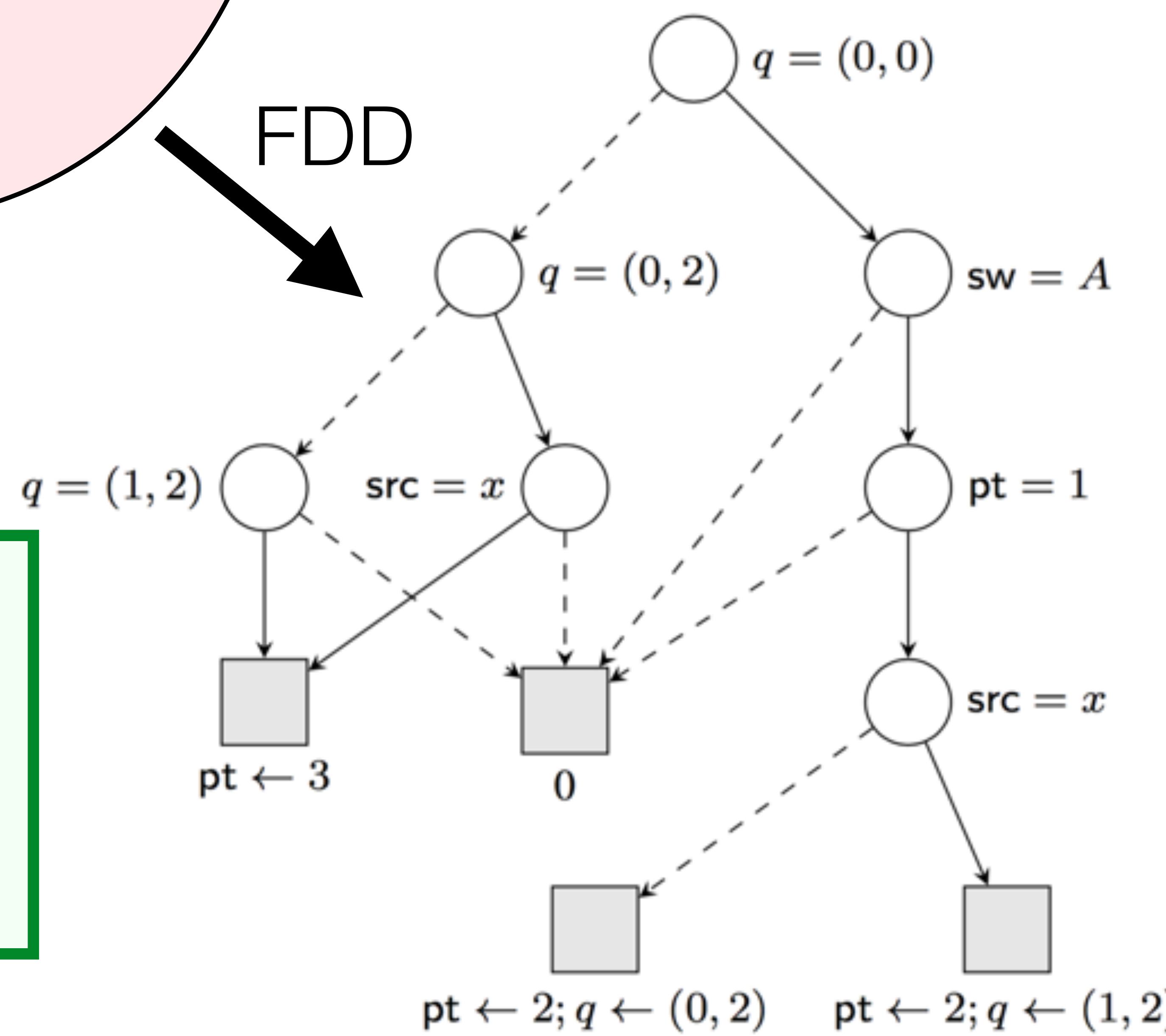




Query Automaton (α)
¬src=X
1
src=X
0
1
[src=X]
[1]
∩
Policy Automaton
polA
link
0
1
2
[α·polB]
=
Product Automaton
¬src=X·polA
¬src=X·link
0,1
0,2
src=X·link
[src=X·polB]
0,0
src=X·polA
link
1,1
1,2
[1·polB]

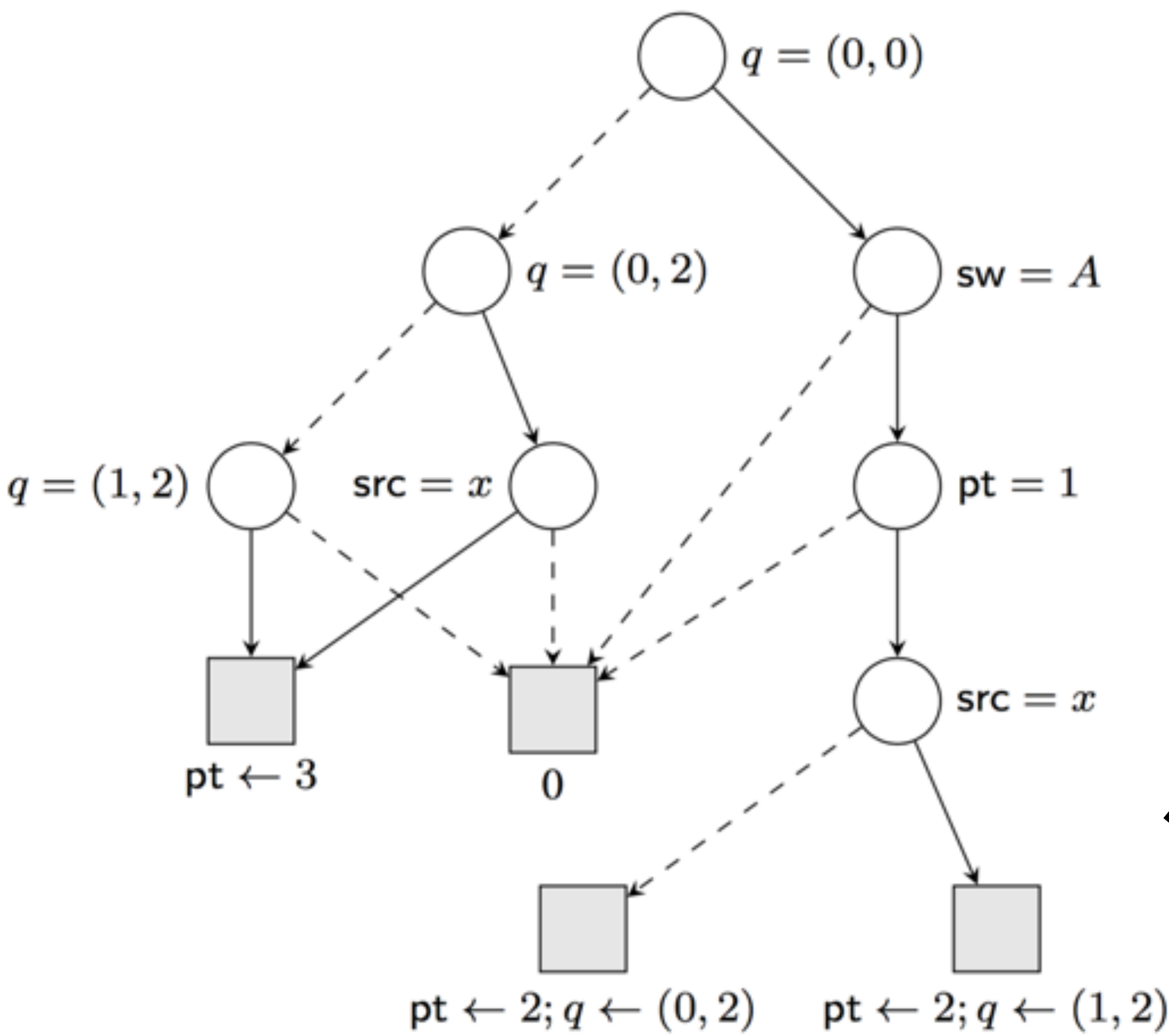
Compilation: Example





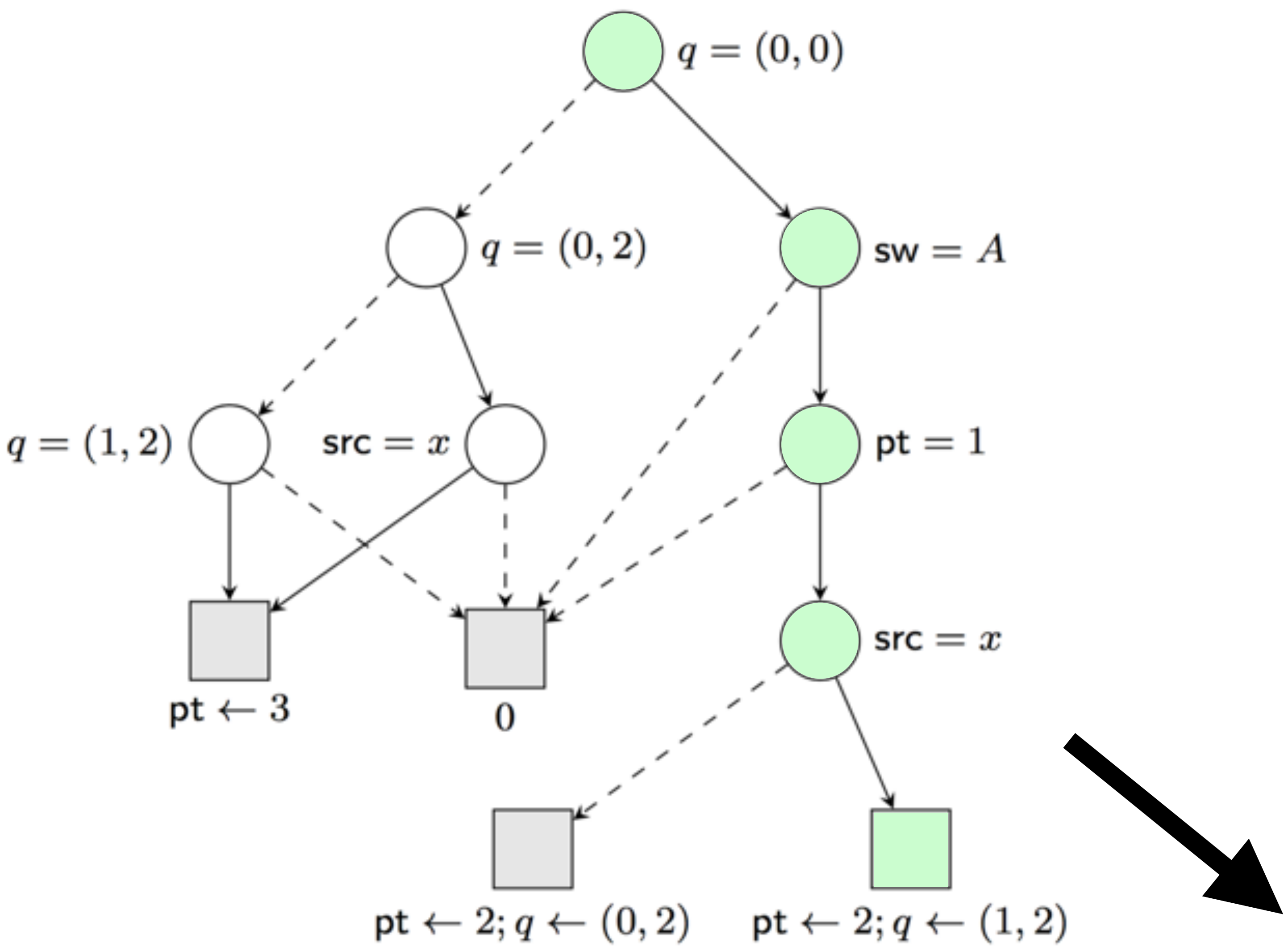
Compilation:
Automaton state encoded as a packet field in the FDD [See Smolka et al]

Compilation: Example



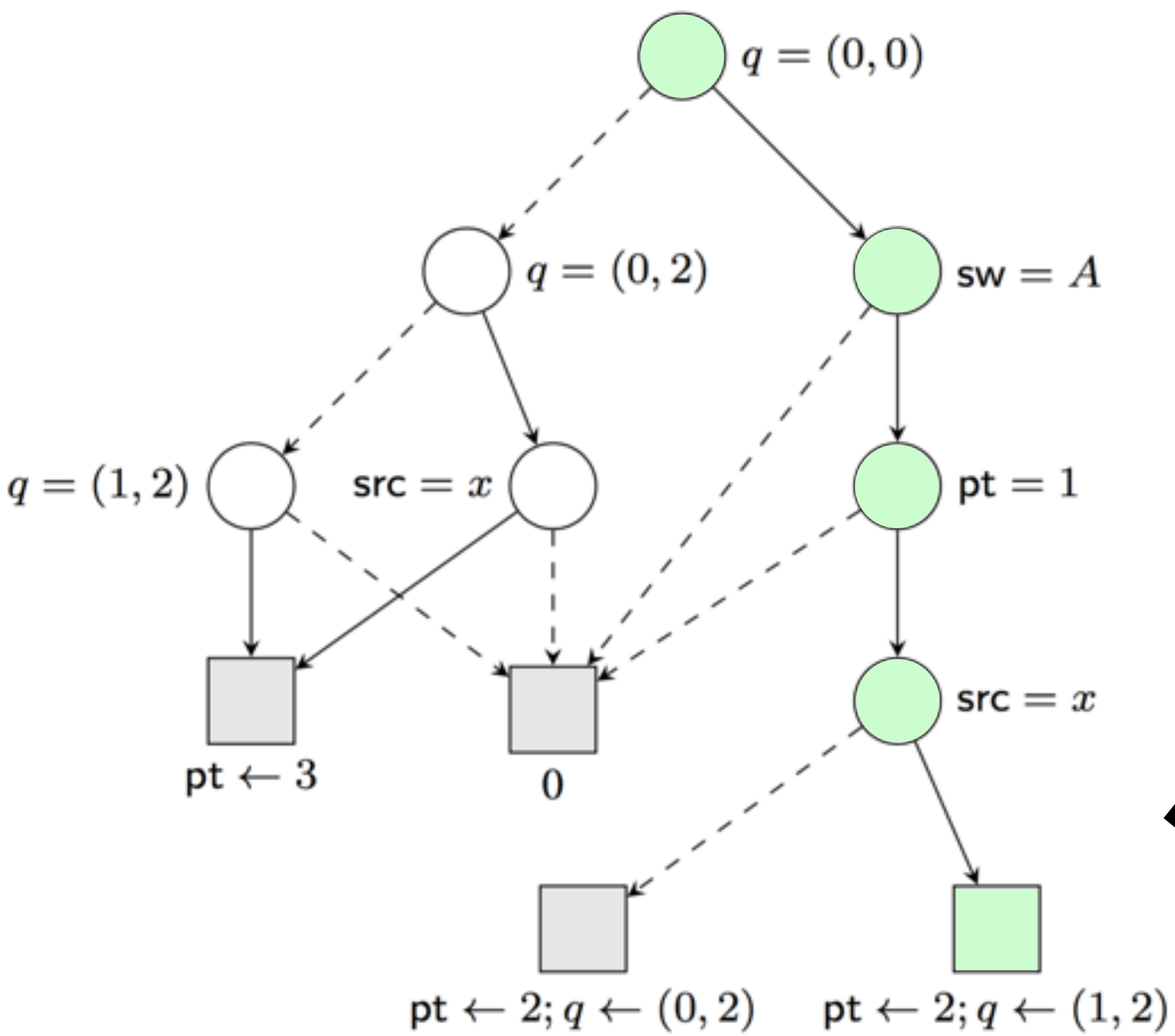
Match	Action
1. $q = (0,0)$; sw $= A$; pt $= 1$; src $= x$	pt $\leftarrow 2; q \leftarrow (1,2)$
2. $q = (0,0)$; sw $= A$; pt $= 1$	pt $\leftarrow 2; q \leftarrow (0,2)$
3. $q = (0,0)$	drop
4. $q = (0,2)$; src $= x$	pt $\leftarrow 3$
5. $q = (0,2)$	drop
6. $q = (1,2)$	pt $\leftarrow 3$
7. true	drop

Compilation: Example



Match	Action
1. $q = (0,0)$; sw $= A$; pt $= 1$; src $= x$	pt $\leftarrow 2; q \leftarrow (1,2)$
2. $q = (0,0)$; sw $= A$; pt $= 1$	pt $\leftarrow 2; q \leftarrow (0,2)$
3. $q = (0,0)$	drop
4. $q = (0,2)$; src $= x$	pt $\leftarrow 3$
5. $q = (0,2)$	drop
6. $q = (1,2)$	pt $\leftarrow 3$
7. true	drop

Compilation: Example



See the paper for additional optimizations!

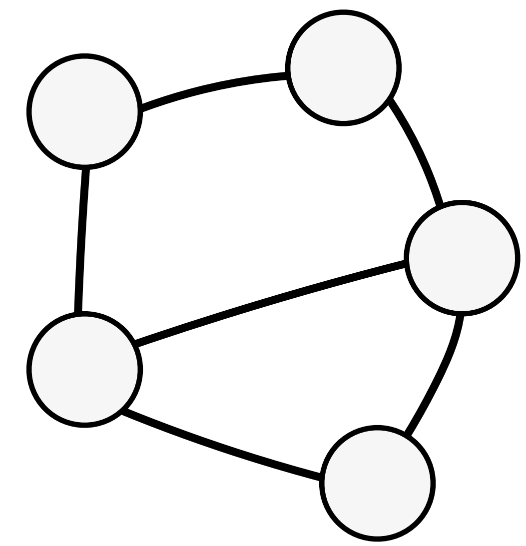
Match	Action
1. $q = (0,0)$; sw $= A$; pt $= 1$; src $= x$	pt $\leftarrow 2; q \leftarrow (1,2)$
2. $q = (0,0)$; sw $= A$; pt $= 1$	pt $\leftarrow 2; q \leftarrow (0,2)$
3. $q = (0,0)$	drop
4. $q = (0,2)$; src $= x$	pt $\leftarrow 3$
5. $q = (0,2)$	drop
6. $q = (1,2)$	pt $\leftarrow 3$
7. true	drop

Evaluation

Compiler Evaluation

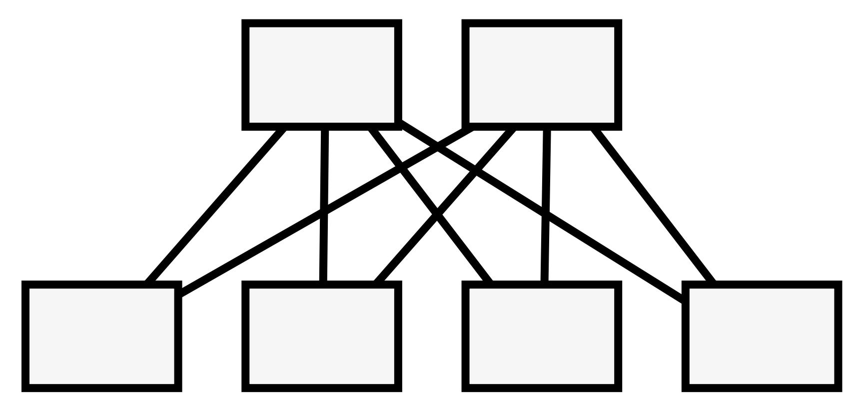
Topology Zoo

- Over 250 real topologies
- Shortest path routing



Stanford Campus Network

- Mid-sized campus network
- 16 core backbone routers
- Rich, non-uniform routing policy



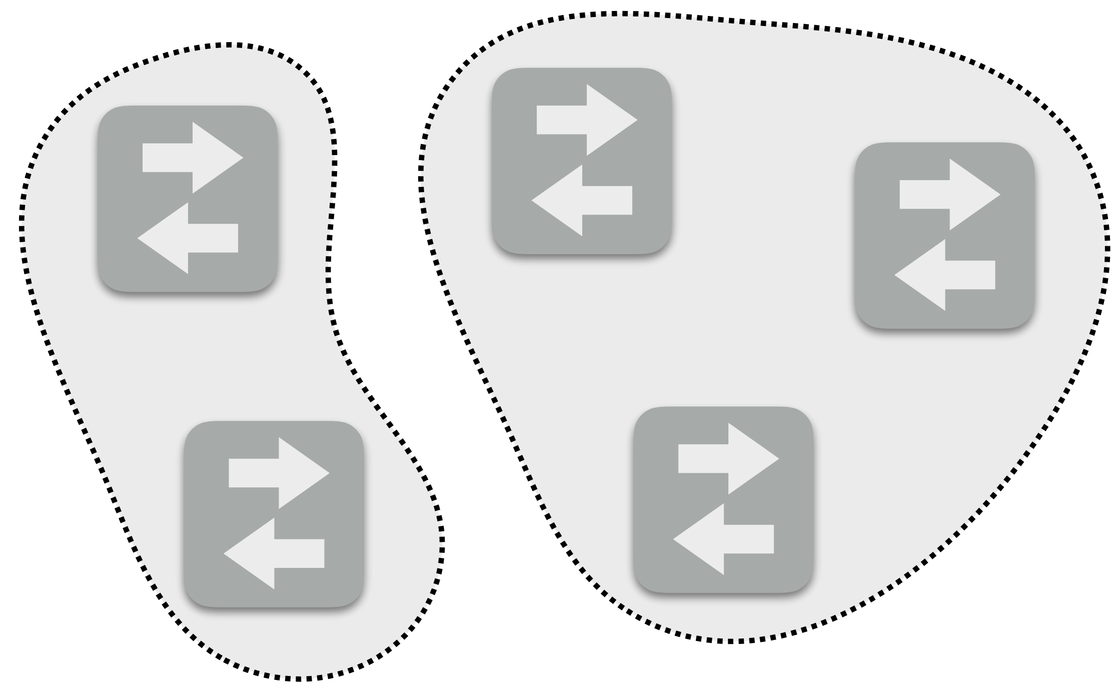
Compiler Evaluation

Baseline:

- Routing only

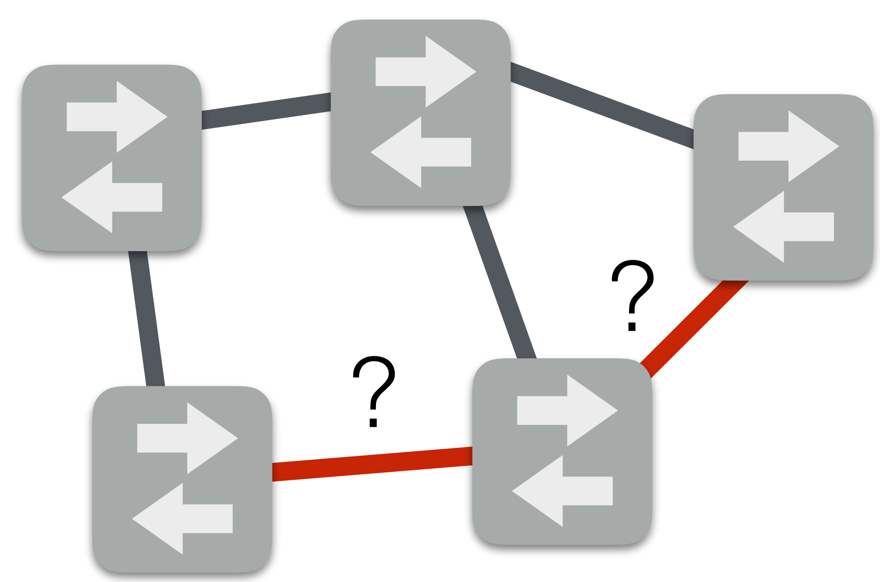
Security:

- Enforce physical isolation
- Enforce logical isolation



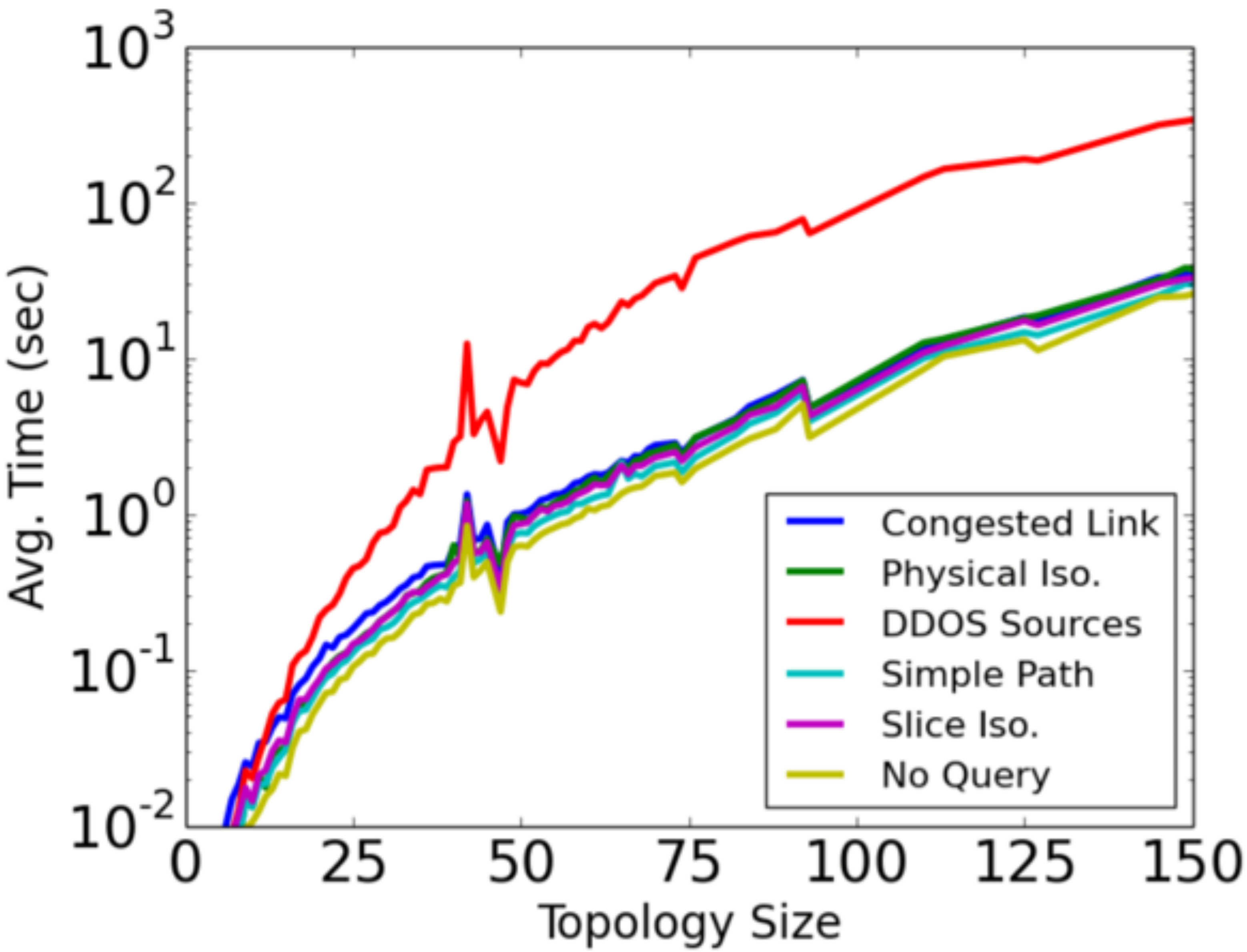
Debugging/Monitoring:

- Congested Link
- Simple path
- Port Matrix
- DDOS sources



Topology Zoo

Compilation Time



Most policies have very little overhead

~12 min worst case

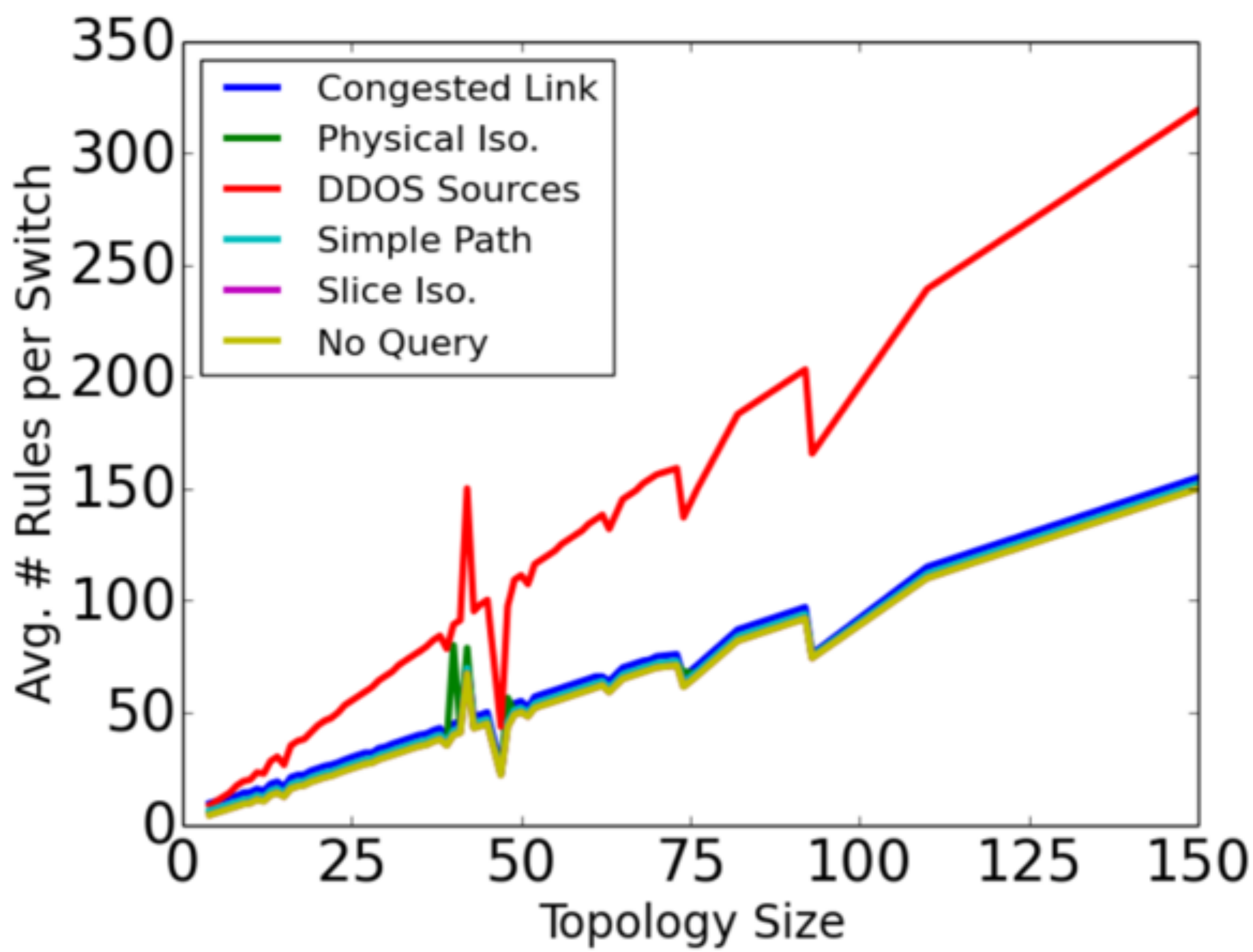
Main limiting factor: number of queries

Topology Zoo

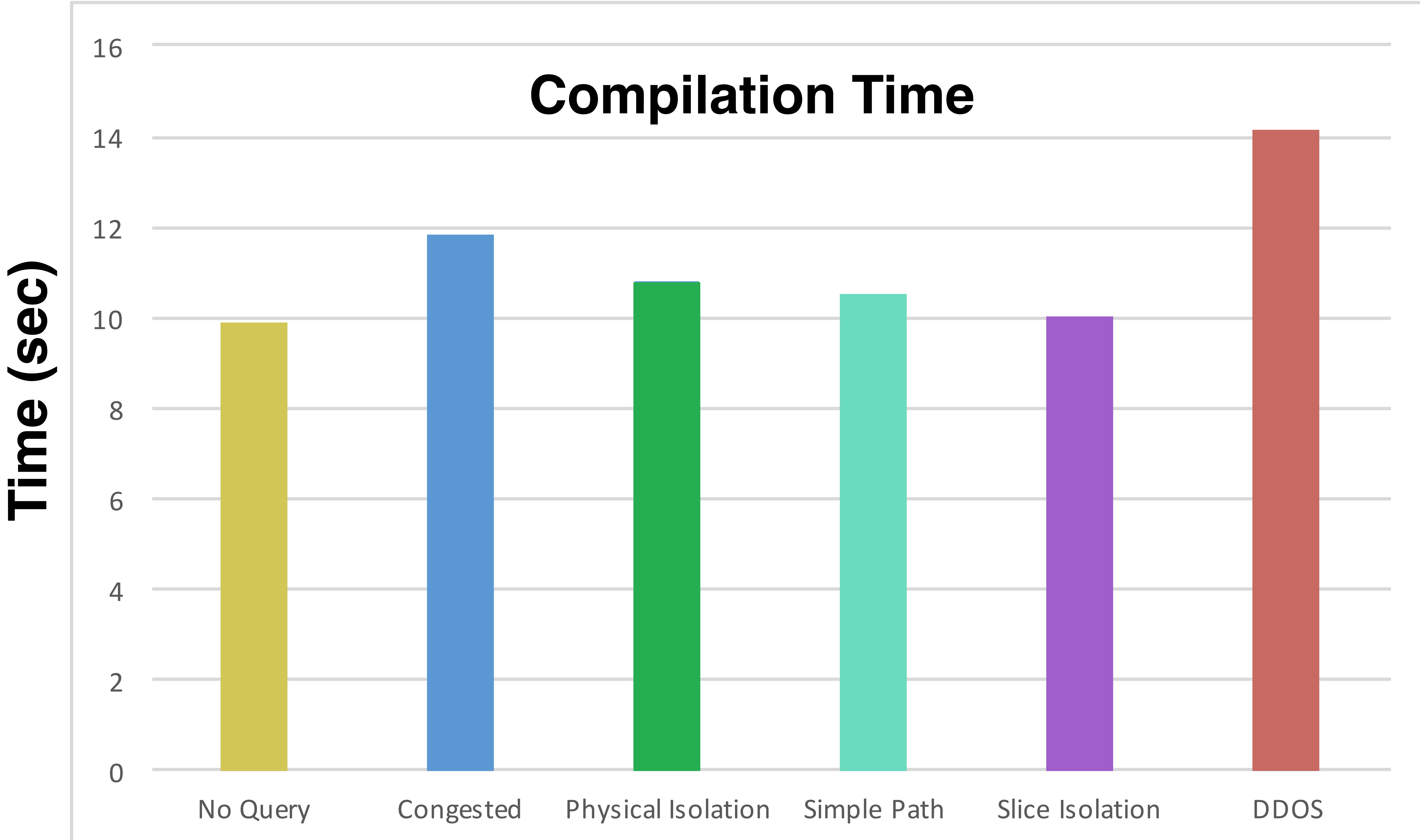
Anecdotally, manual inspection often near minimal rule overhead

~2x increase with DDOS query

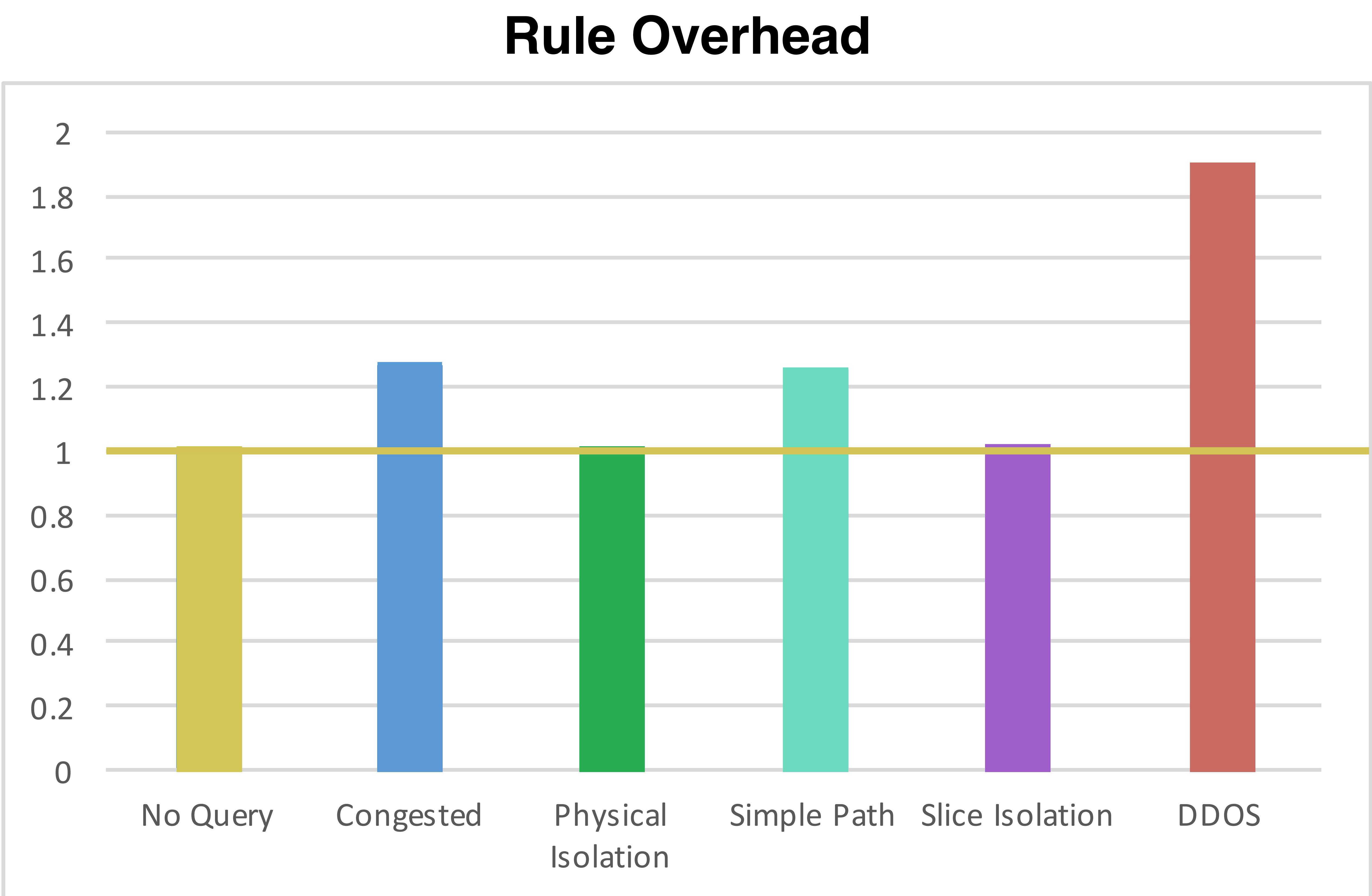
Number of Rules



Stanford Network



Stanford Network



Conclusions

Language

- Extension of NetKAT with **queries over packet history**
- Useful in a variety of network **applications**

Theory

- **Soundness** and **completeness** for network-wide programs
- New general strategy for completeness

Compiler

- Inspired by structure of the completeness proof
- **Scales** to many real network topologies/policies