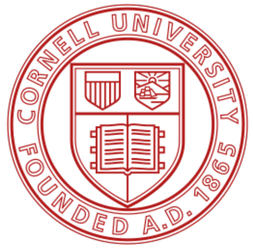
Weighted NetKAT

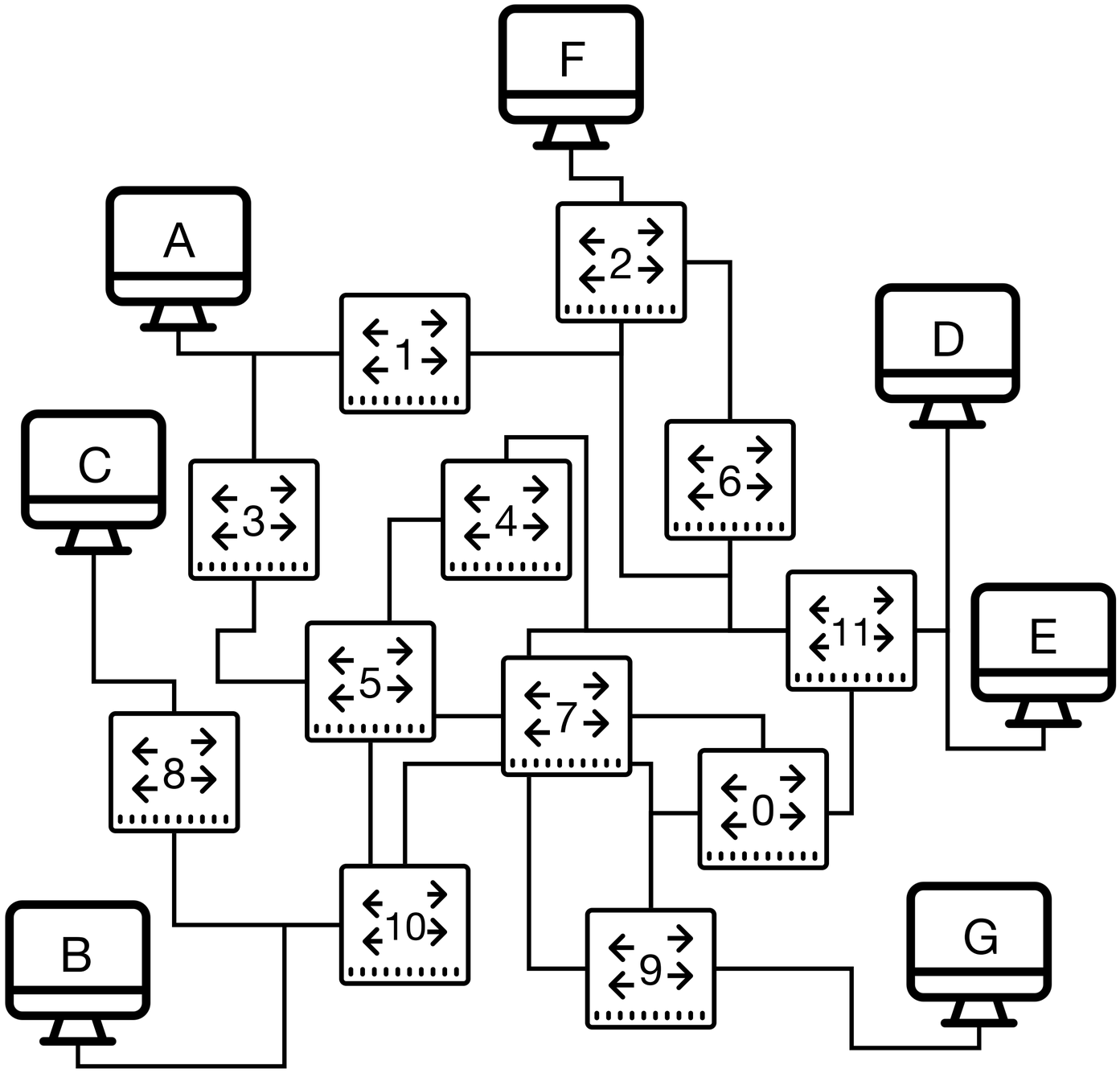
A Programming Language for Quantitative Network Verification

Emmanuel Suárez Acevedo
Tiago Ferreira
Kevin Batz
Oliver Bøving
Nate Foster
Alexandra Silva

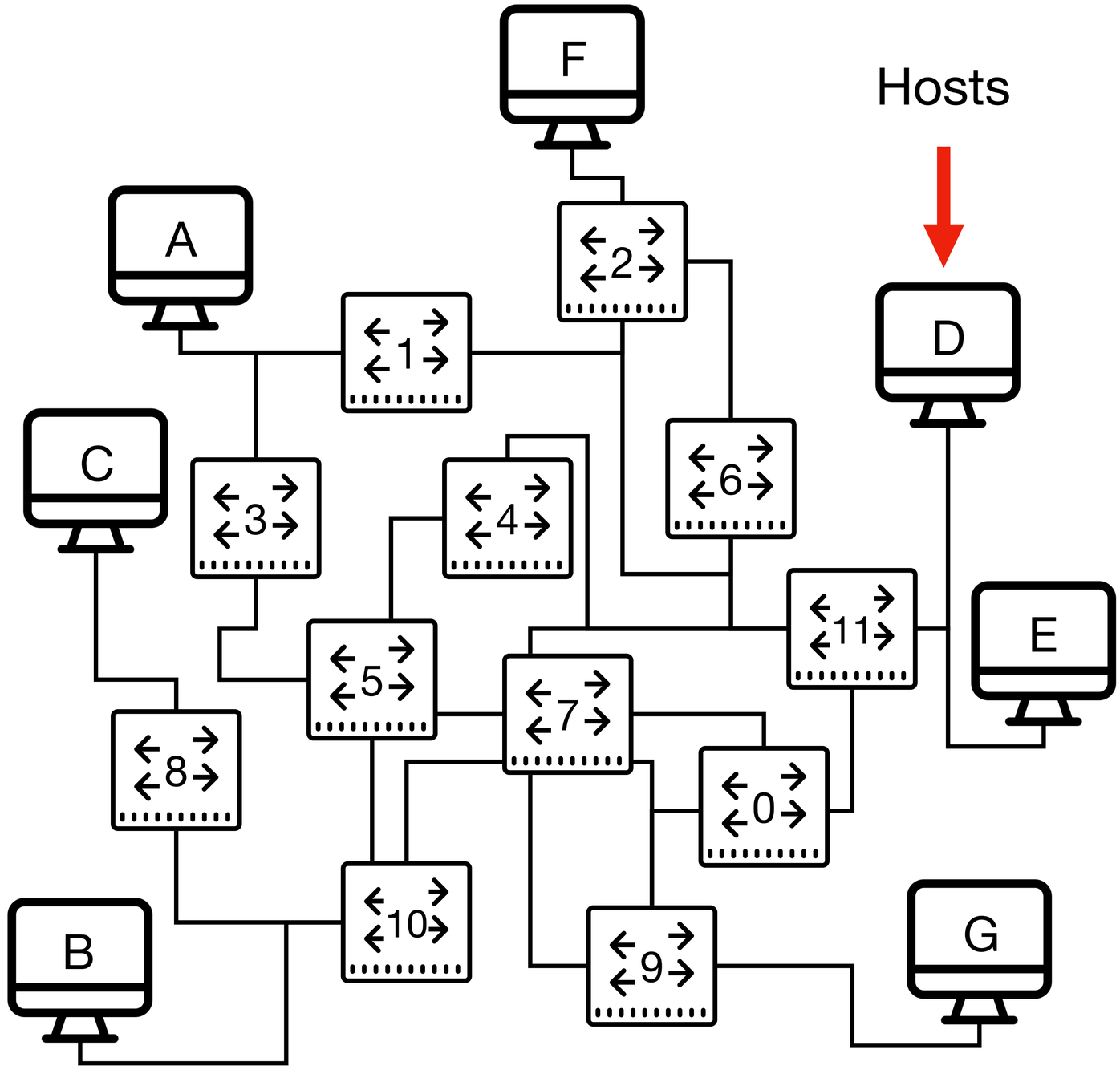


PLDI 2026

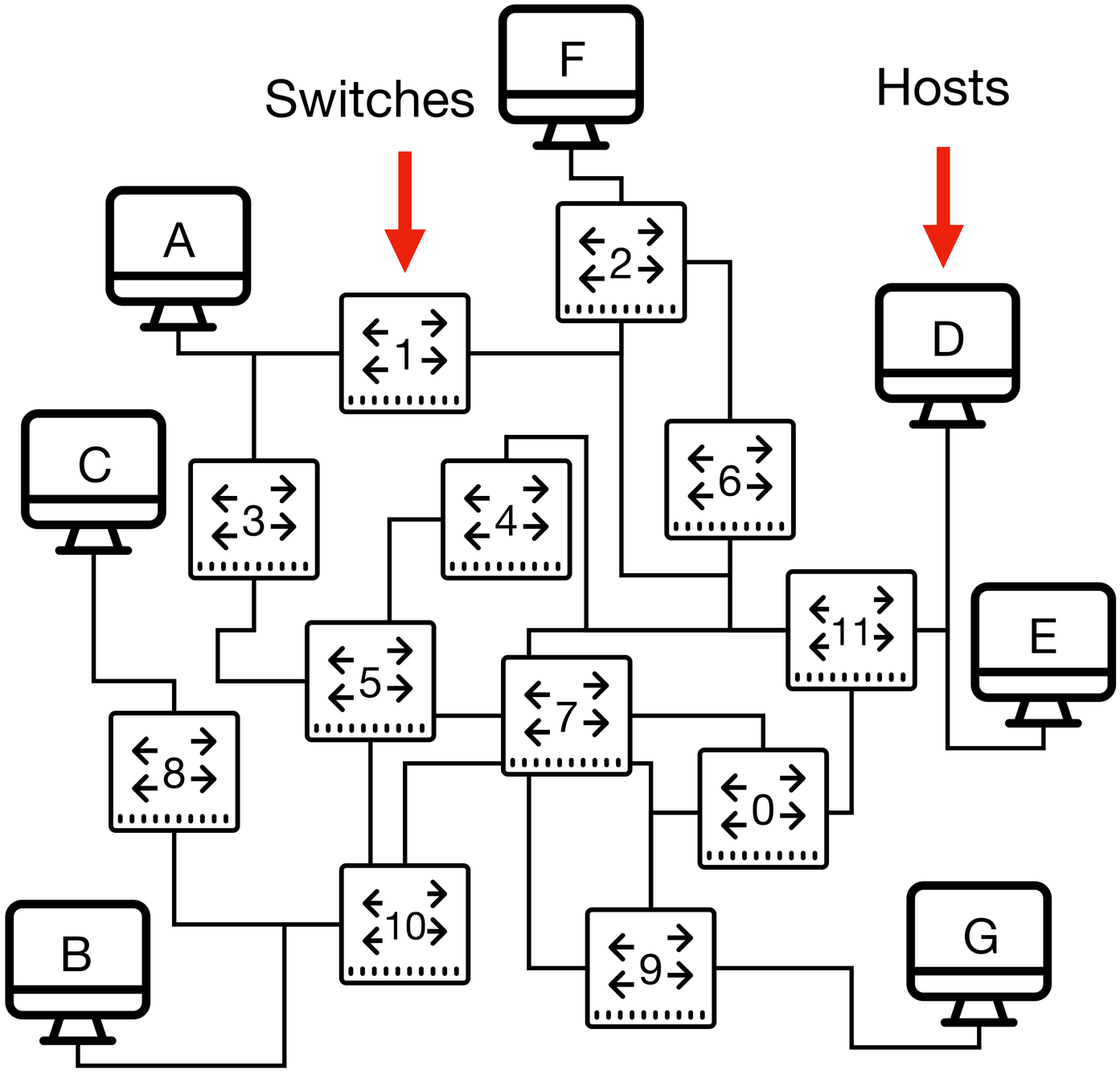
Network Verification



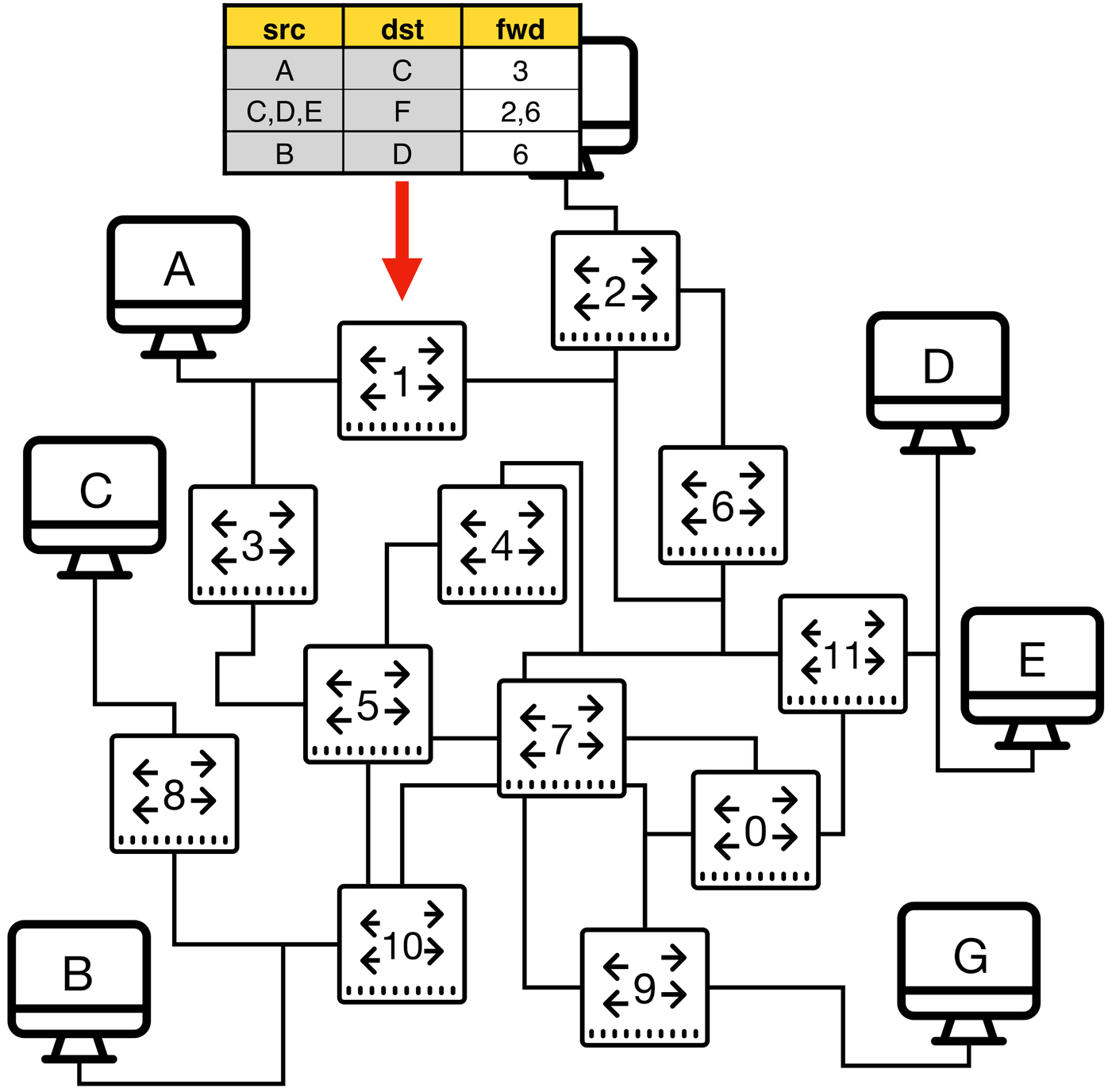
Network Verification



Network Verification

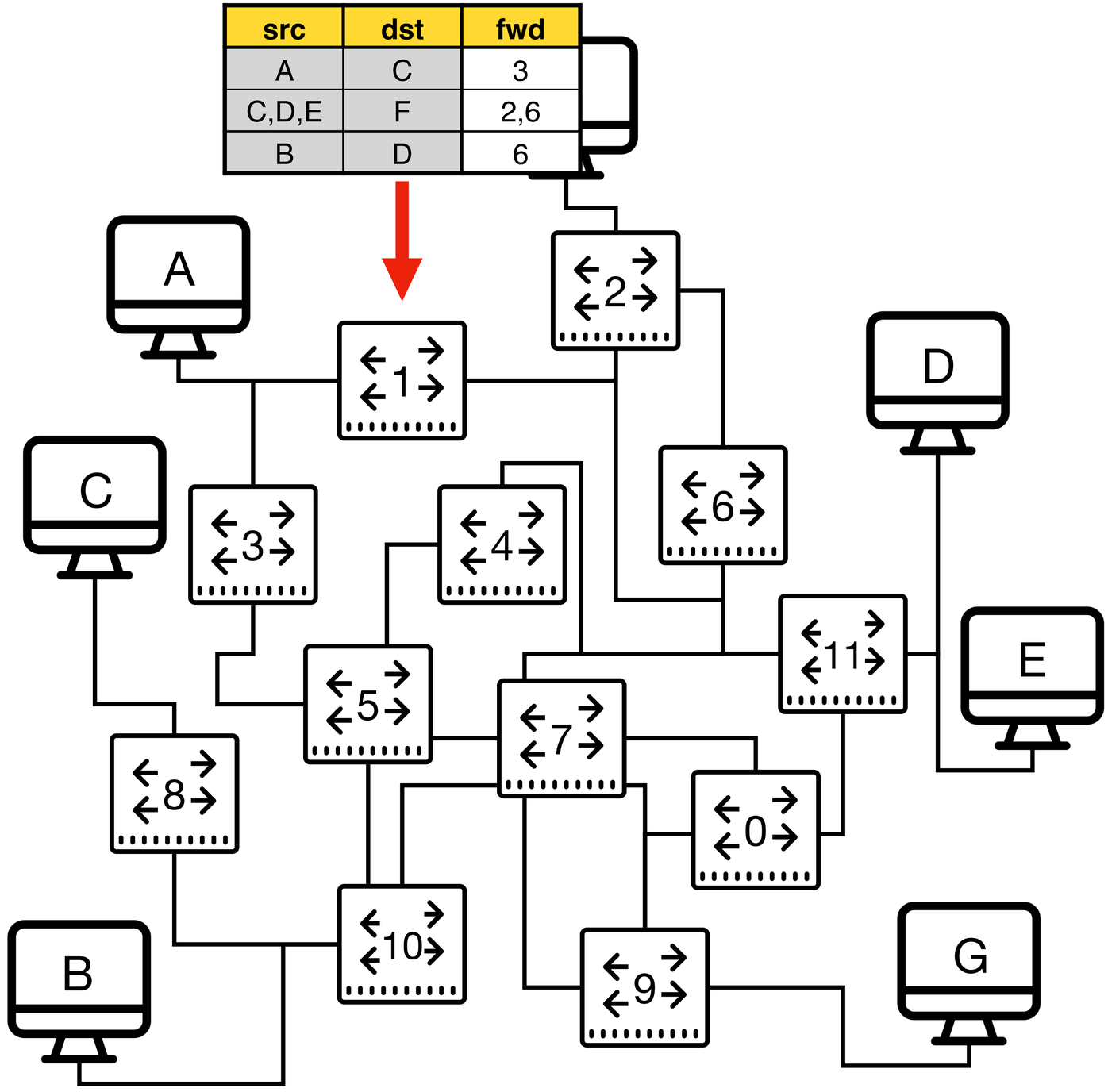


Network Verification



Network Verification

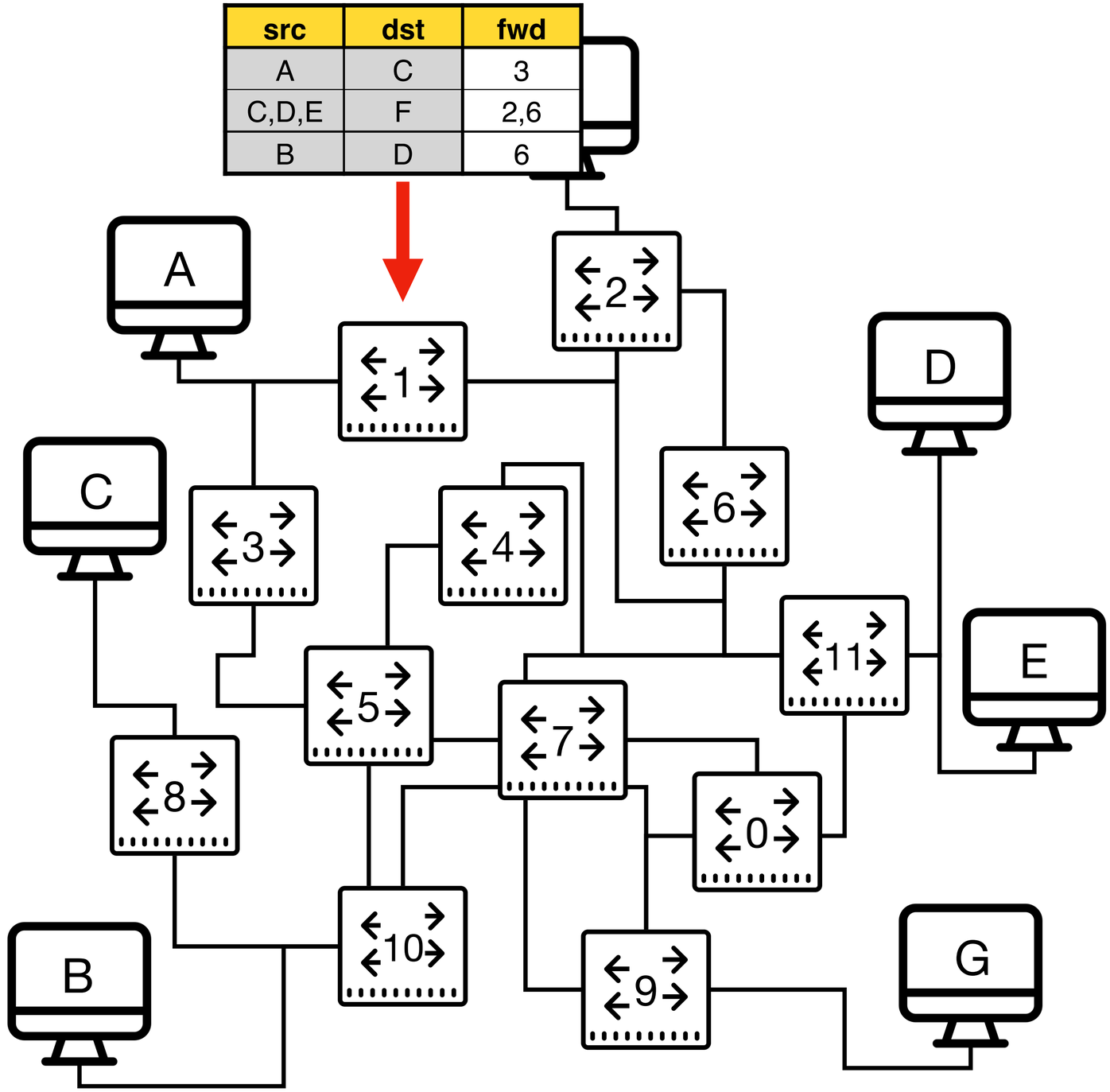
Automatically verify:



Network Verification

Automatically verify:

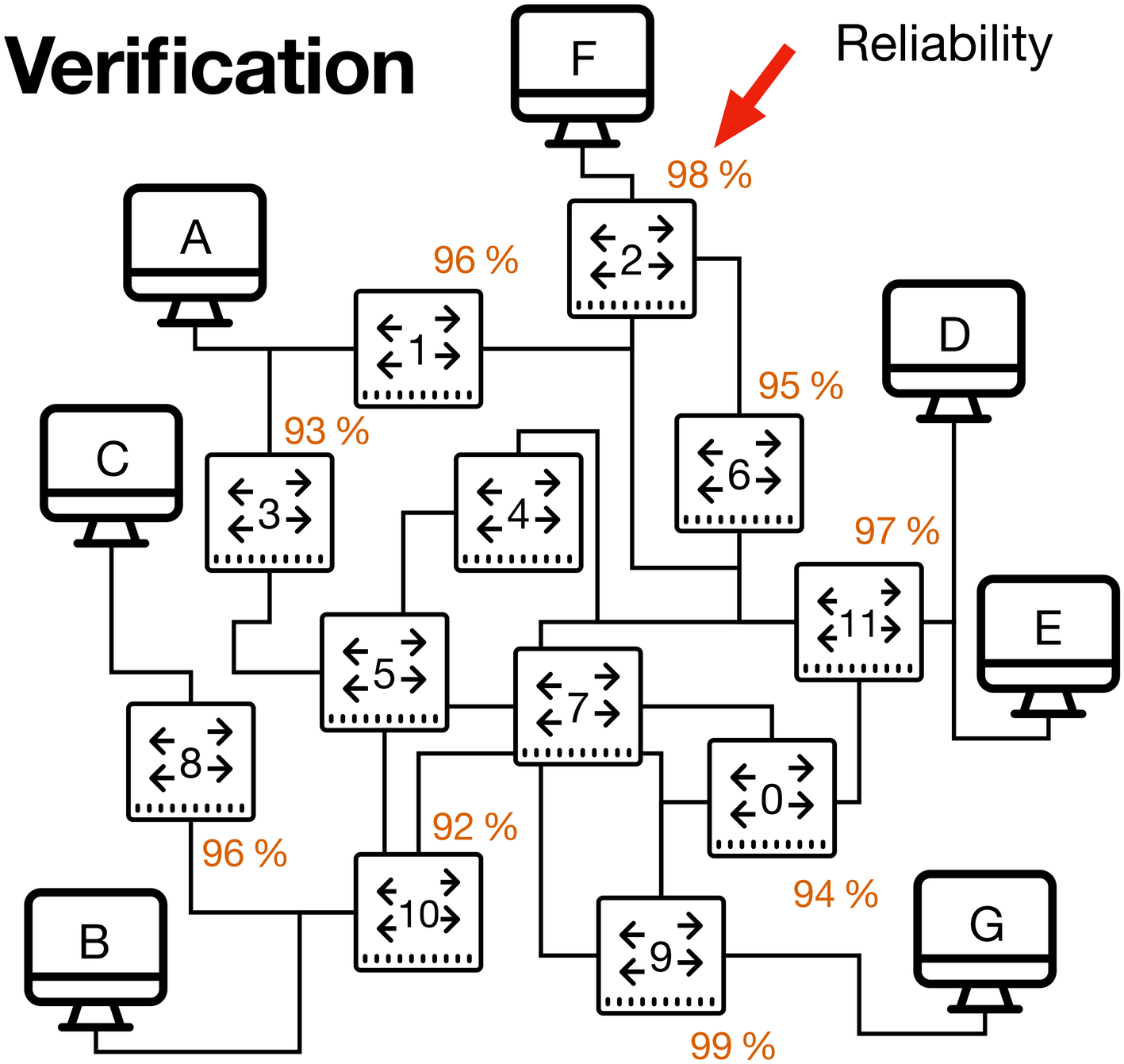
Can all hosts deliver packets to one another?



Quantitative Network Verification

Automatically verify:

Can all hosts deliver packets to one another?

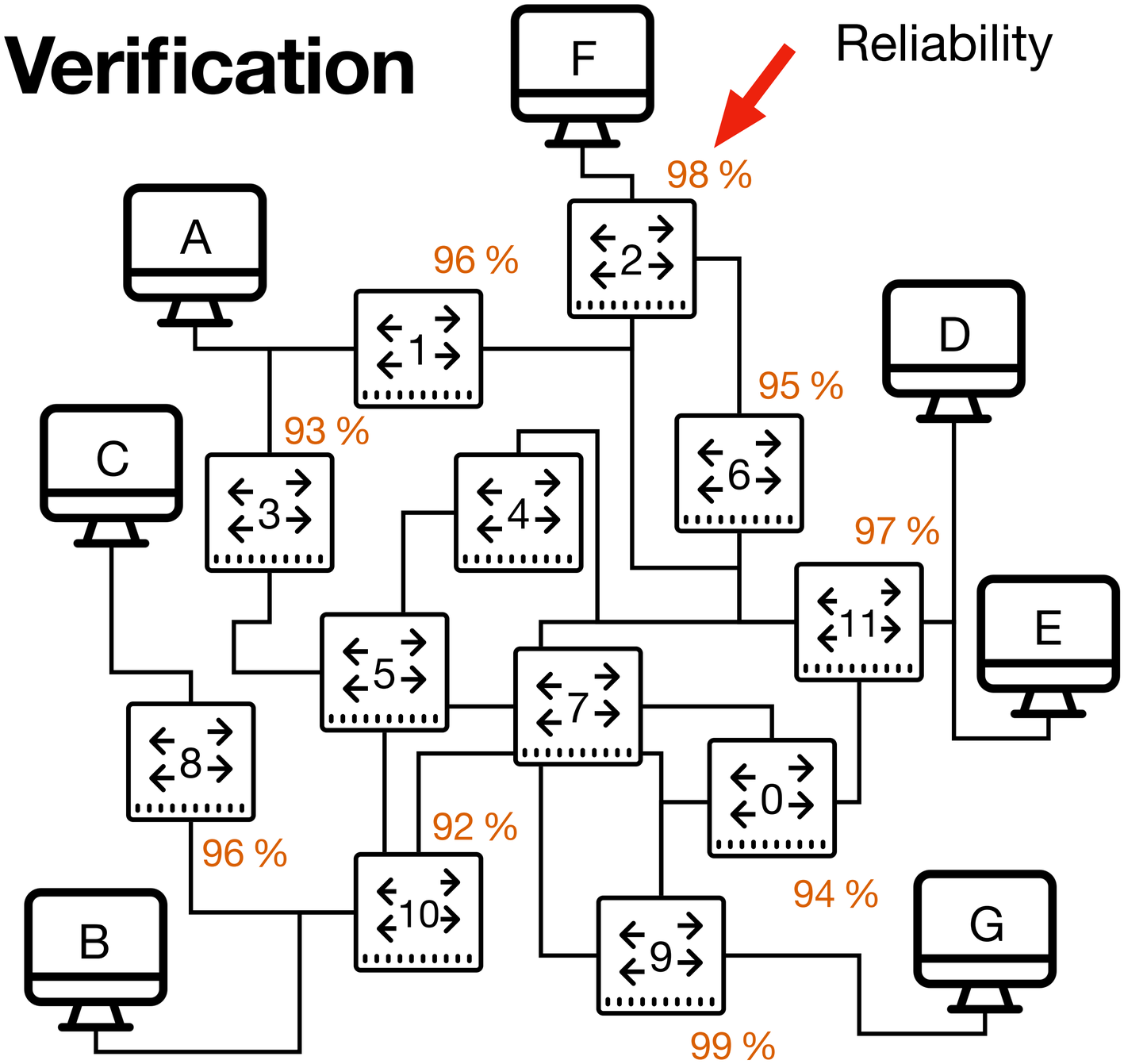


Quantitative Network Verification

Automatically verify:

Can all hosts deliver packets to one another?

Does all traffic get delivered with at least 90% reliability?

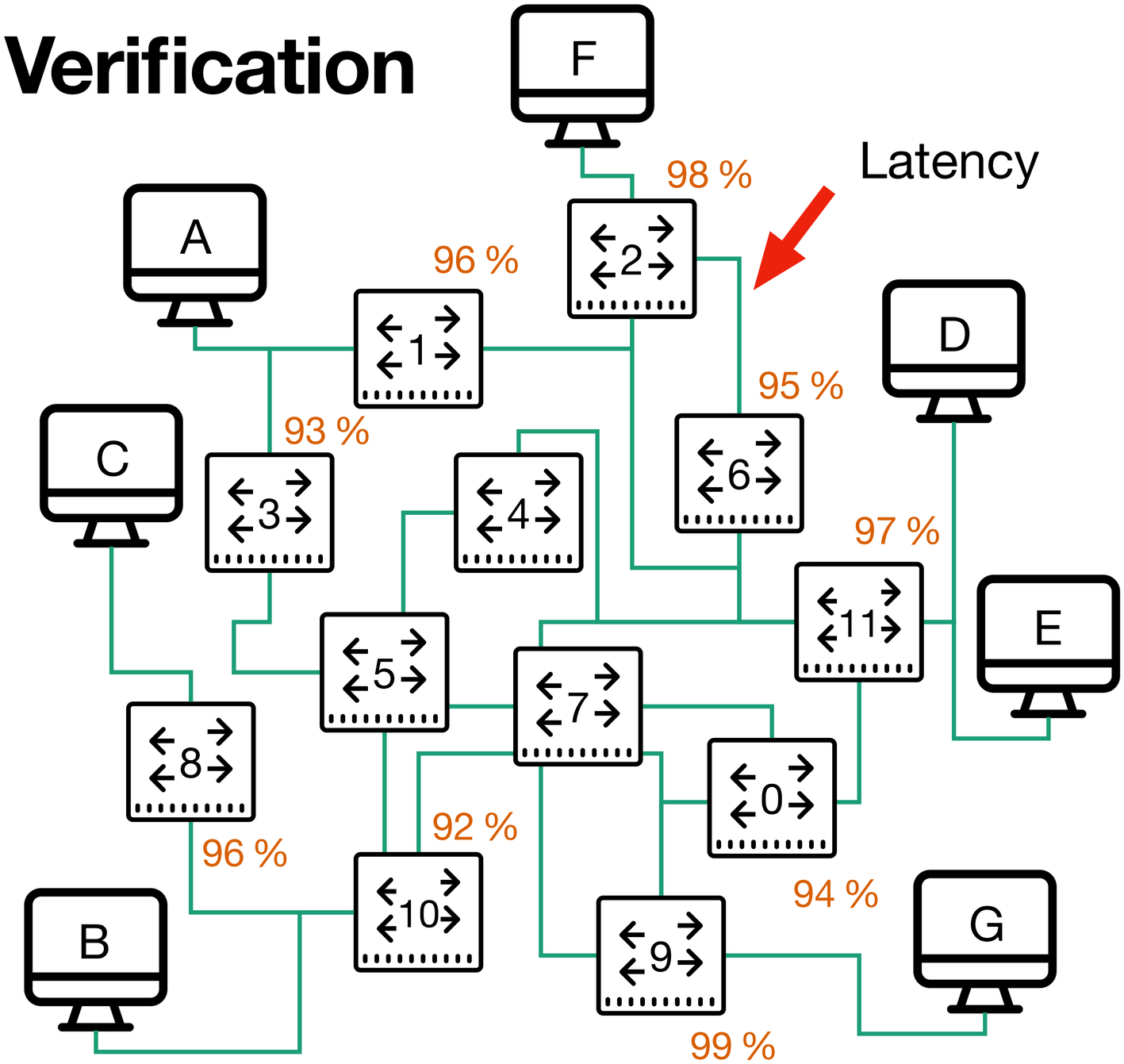


Quantitative Network Verification

Automatically verify:

Can all hosts deliver packets to one another?

Does all traffic get delivered with at least 90% reliability?



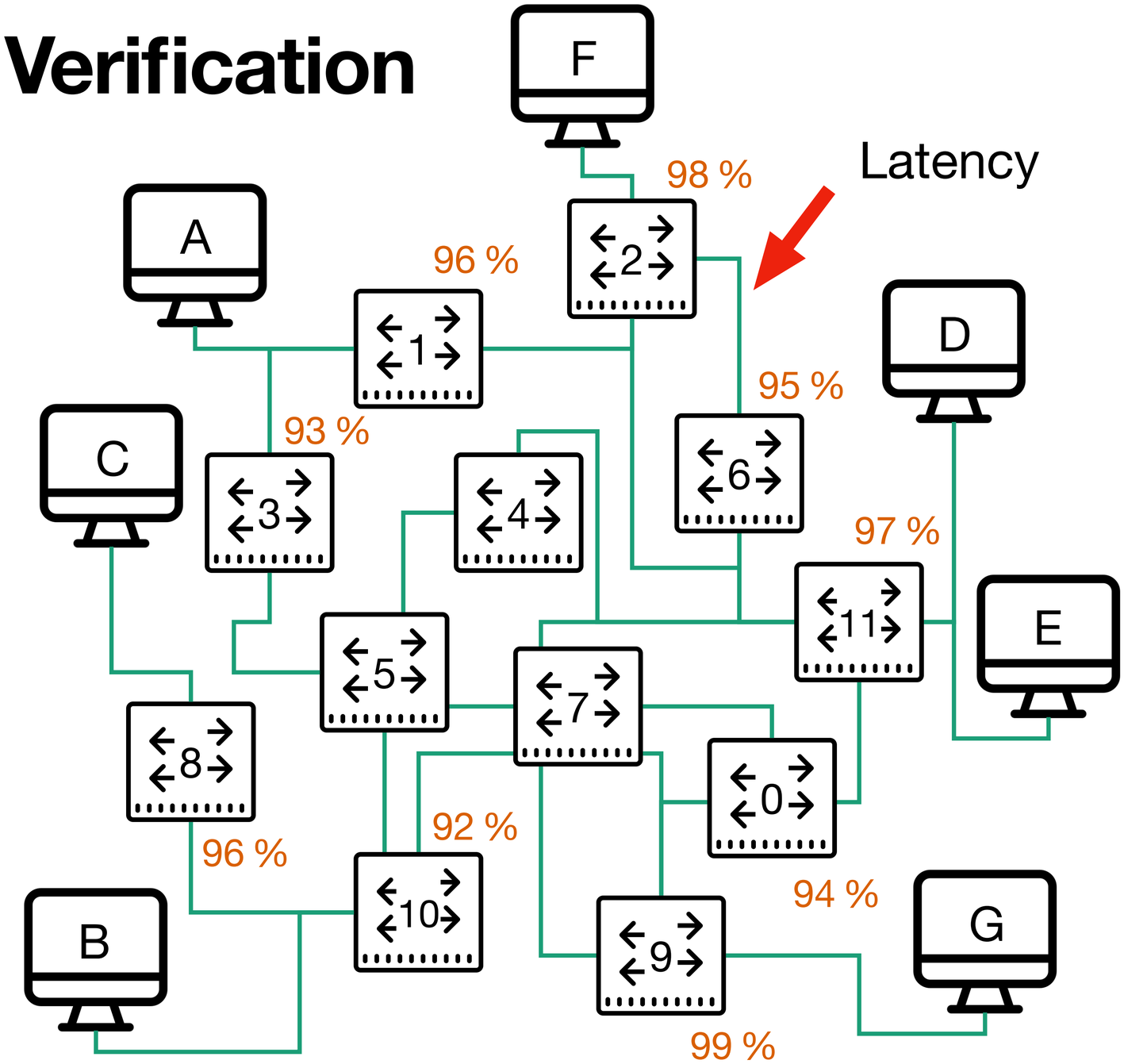
Quantitative Network Verification

Automatically verify:

Can all hosts deliver packets to one another?

Does all traffic get delivered with at least 90% reliability?

Can host A deliver packets to host B within 5ms?



Quantitative Network Verification

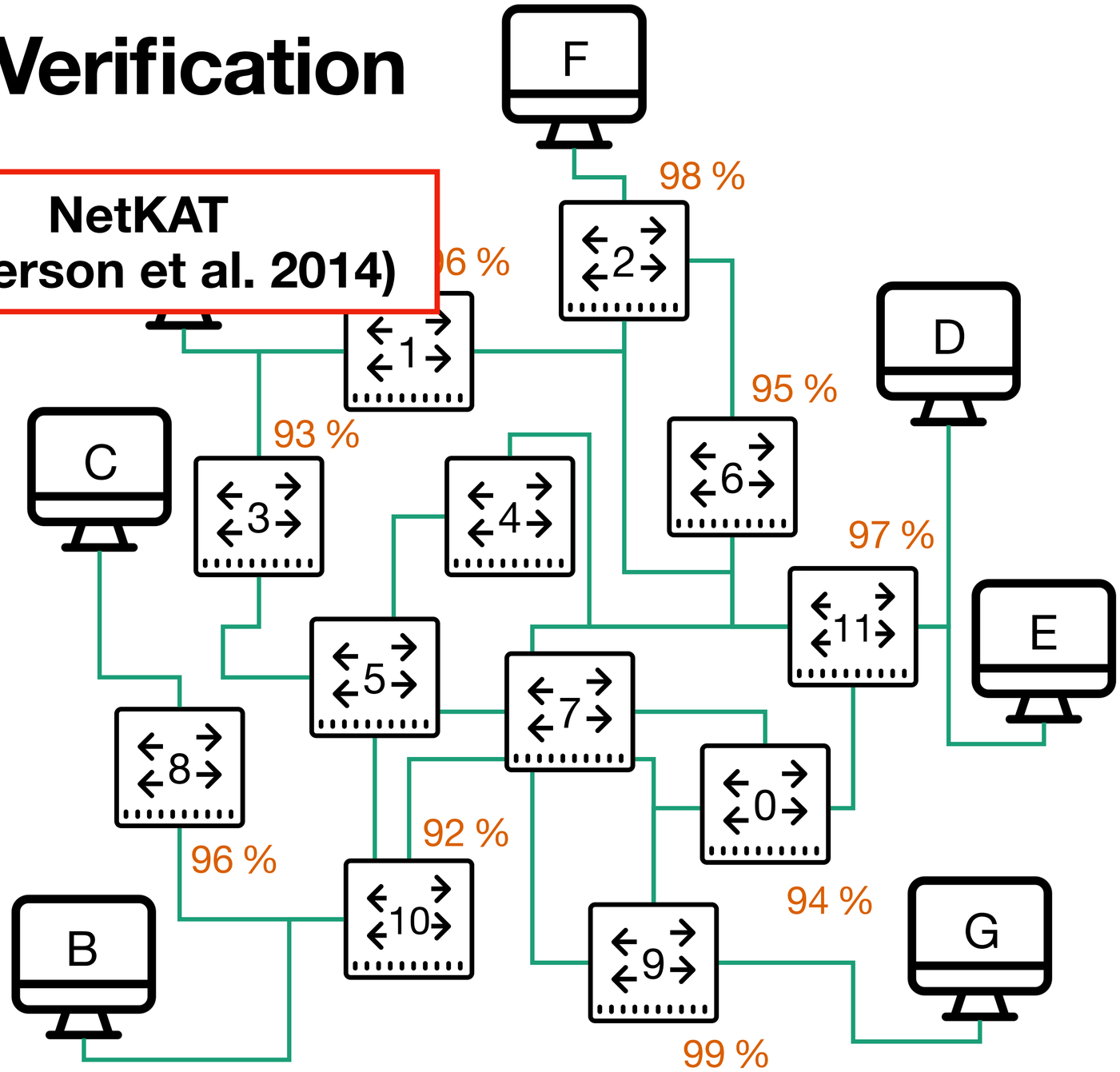
Automatically verify:

Can all hosts deliver packets to one another?

Does all traffic get delivered with at least 90% reliability?

Can host A deliver packets to host B within 5ms?

NetKAT (Anderson et al. 2014)



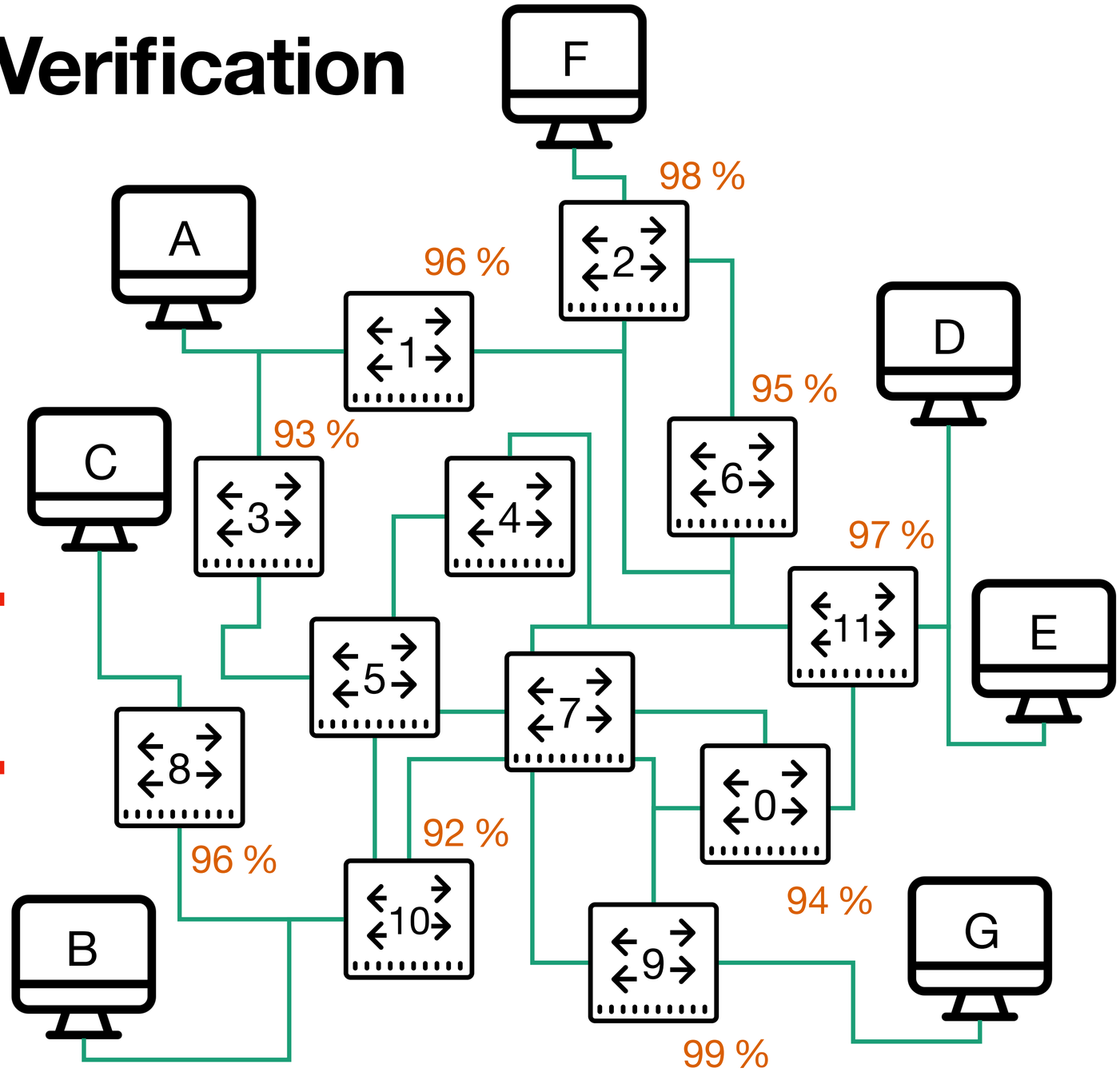
Quantitative Network Verification

Automatically verify:

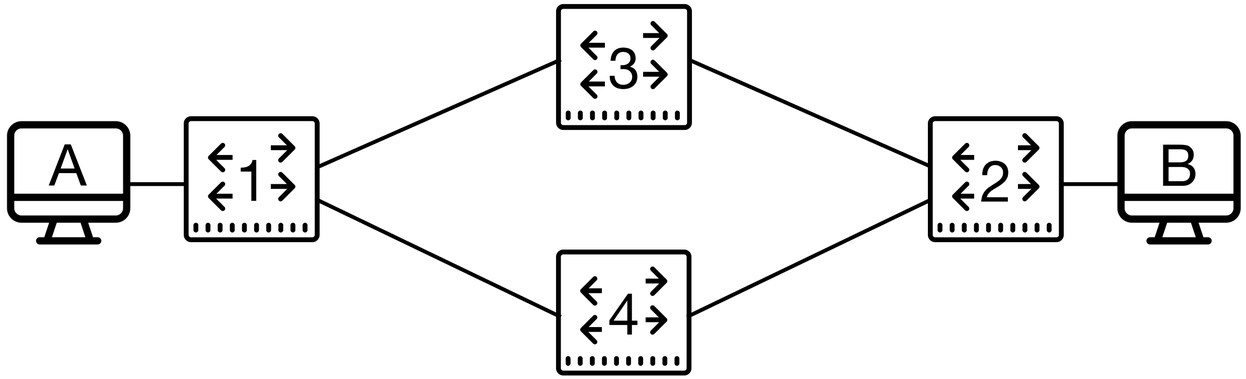
Can all hosts deliver packets to one another?

Does all traffic get delivered with at least 90% reliability?

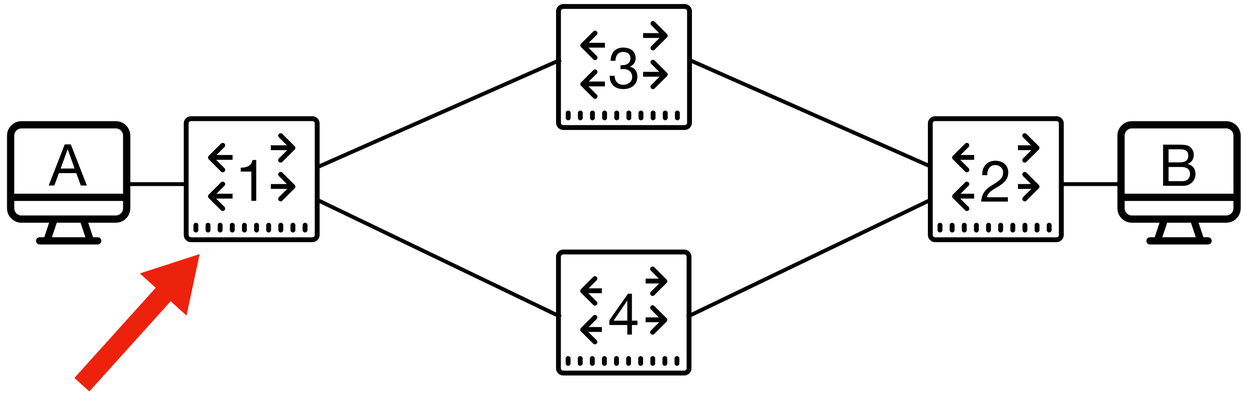
Can host A deliver packets to host B within 5ms?



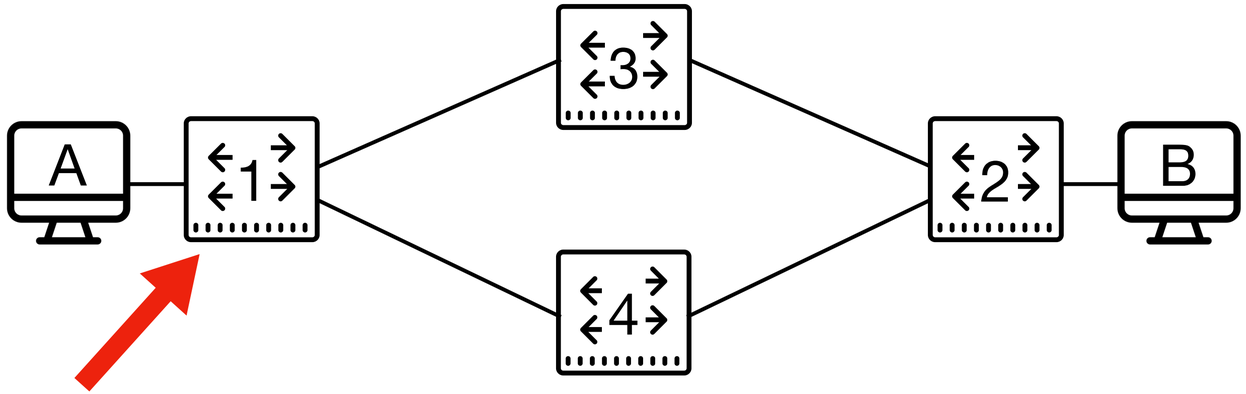
Modeling Network Behavior



Modeling Network Behavior

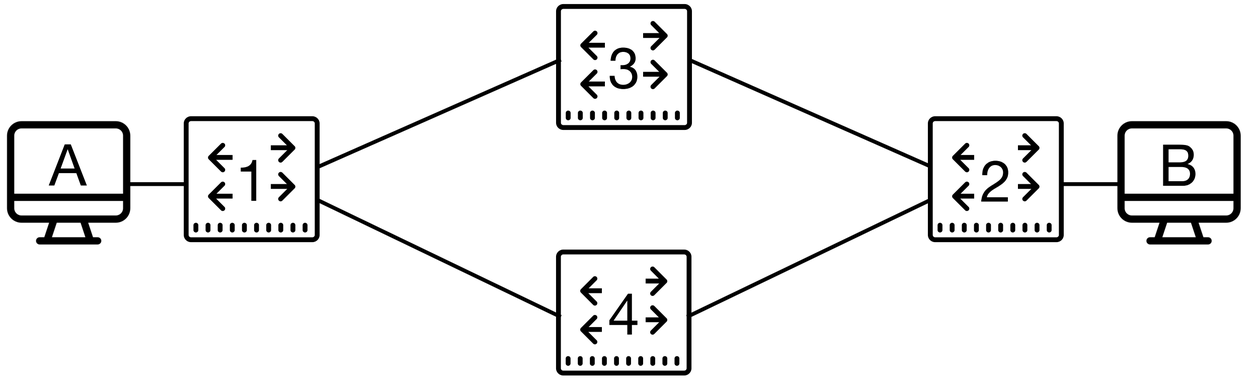


Modeling Network Behavior




```
p1 ≜
  if dst = B then
    sw <- 3 ⊕ sw <- 4
  else ...
```

Modeling Network Behavior

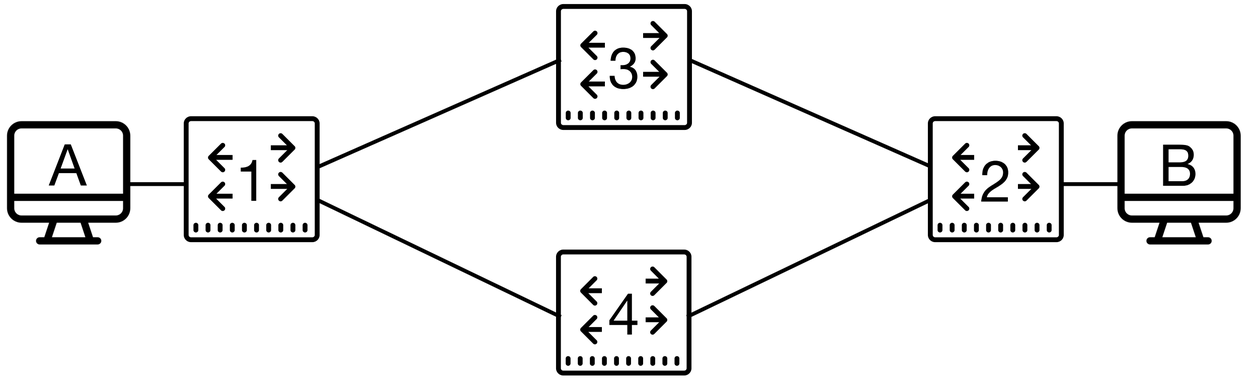


$p_1 \triangleq$

```
 if dst = B then
  sw <- 3 ⊕ sw <- 4
 else ...
```

$p_2 \triangleq \ldots$ $p_3 \triangleq \ldots$ $p_4 \triangleq \ldots$

Modeling Network Behavior



$p_1 \triangleq$

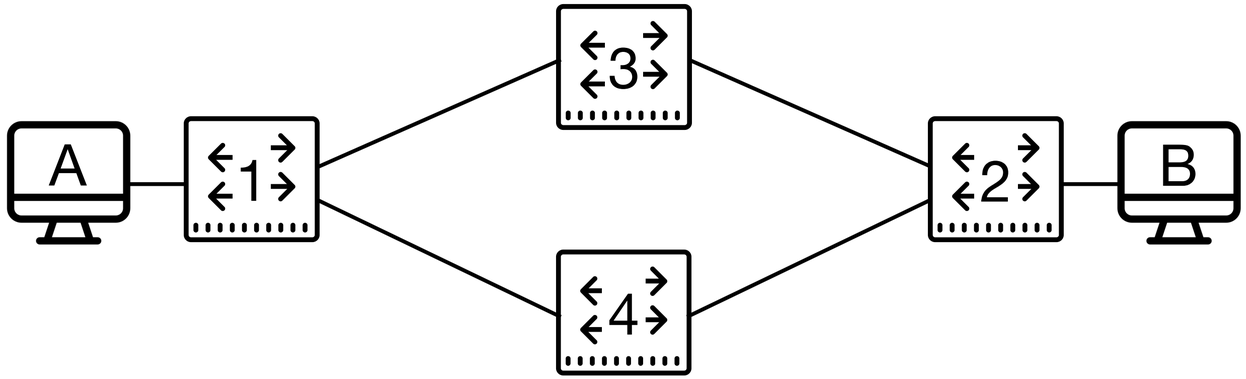
```
if dst = B then
  sw <- 3 ⊕ sw <- 4
else ...
```

$p_2 \triangleq$ ... $p_3 \triangleq$ ... $p_4 \triangleq$ ...

$p \triangleq$

```
if sw = 1 then p1
else if sw = 2 then p2
else if sw = 3 then p3
else if sw = 4 then p4
```

Modeling Network Behavior

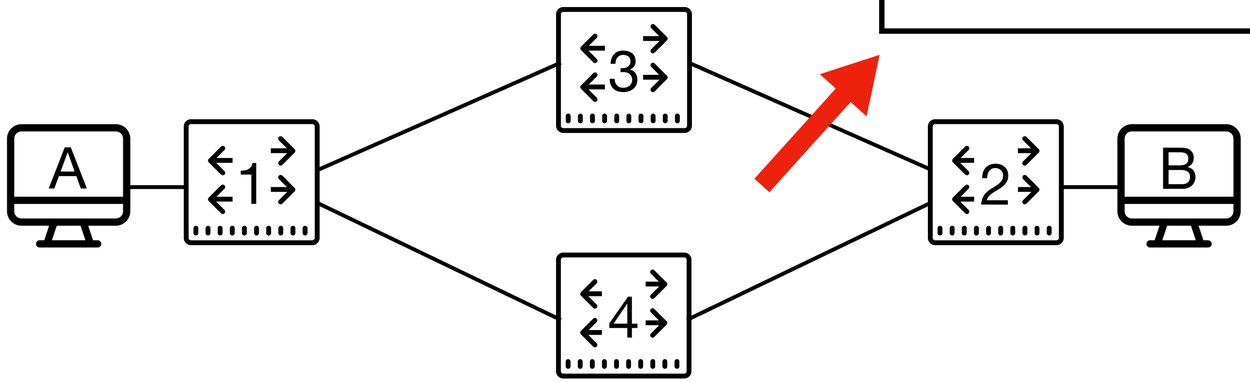



```
p1 ≜
  if dst = B then
    sw <- 3 ⊕ sw <- 4
  else ...
```

```
p2 ≜ ...    p3 ≜ ...    p4 ≜ ...

p ≜       if sw = 1 then p1
      else if sw = 2 then p2
      else if sw = 3 then p3
      else if sw = 4 then p4

net ≜ ( p )*
```

Modeling Network Behavior

Can all hosts deliver packets to one another?



$p_1 \triangleq$

```
if dst = B then
  sw <- 3 ⊕ sw <- 4
else ...
```

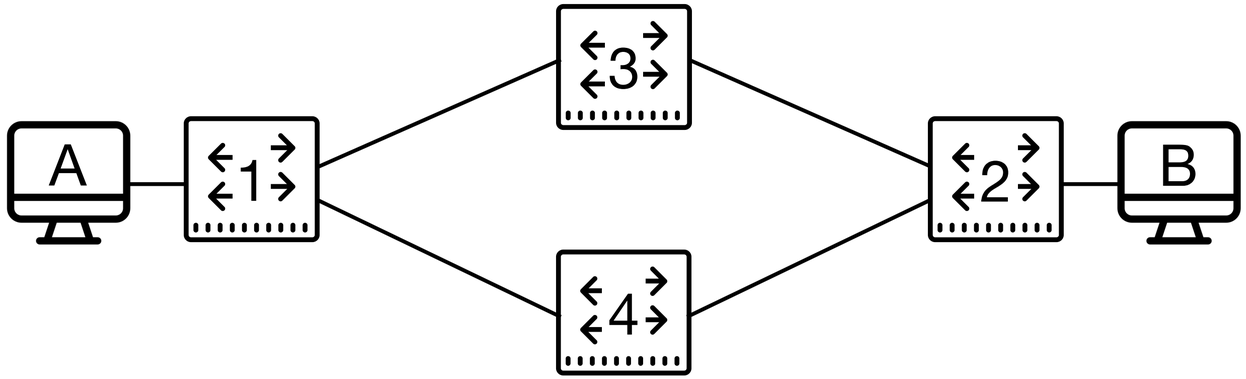
$p_2 \triangleq$ ... $p_3 \triangleq$ ... $p_4 \triangleq$ ...

$p \triangleq$

```
if sw = 1 then p1
else if sw = 2 then p2
else if sw = 3 then p3
else if sw = 4 then p4
```

$net \triangleq (p)^*$

Modeling Quantitative Network Behavior



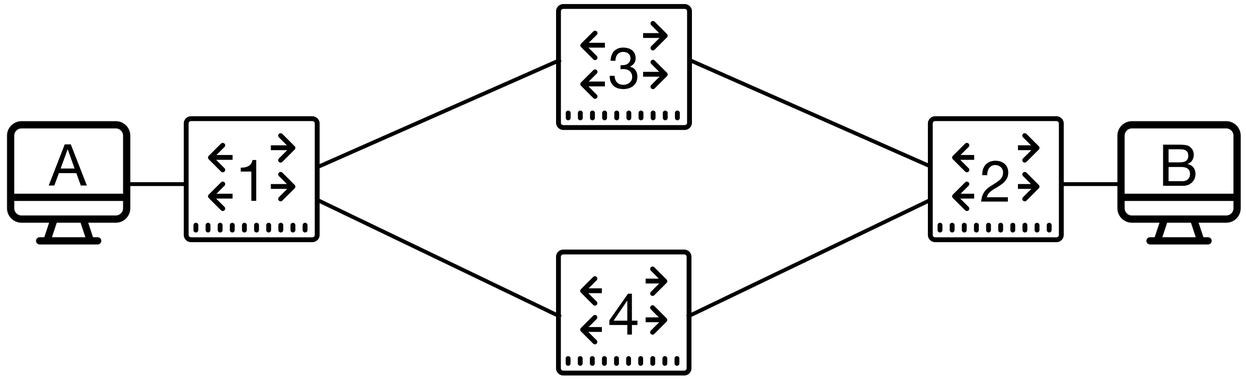
$p_1 \triangleq$

```
if dst = B then
  sw <- 3 ⊕ sw <- 4
else ...
```

$p_2 \triangleq$ ... $p_3 \triangleq$ ... $p_4 \triangleq$ ...

```
p ≜ if sw = 1 then p1
    else if sw = 2 then p2
    else if sw = 3 then p3
    else if sw = 4 then p4

net ≜ ( p )*
```

Modeling Quantitative Network Behavior



$p_1 \triangleq$

```
if dst = B then
  sw <- 3 ⊕ sw <- 4
else ...
```

$p_2 \triangleq$ ... $\triangleq$ ...

$p \triangleq$ p_1

2

3

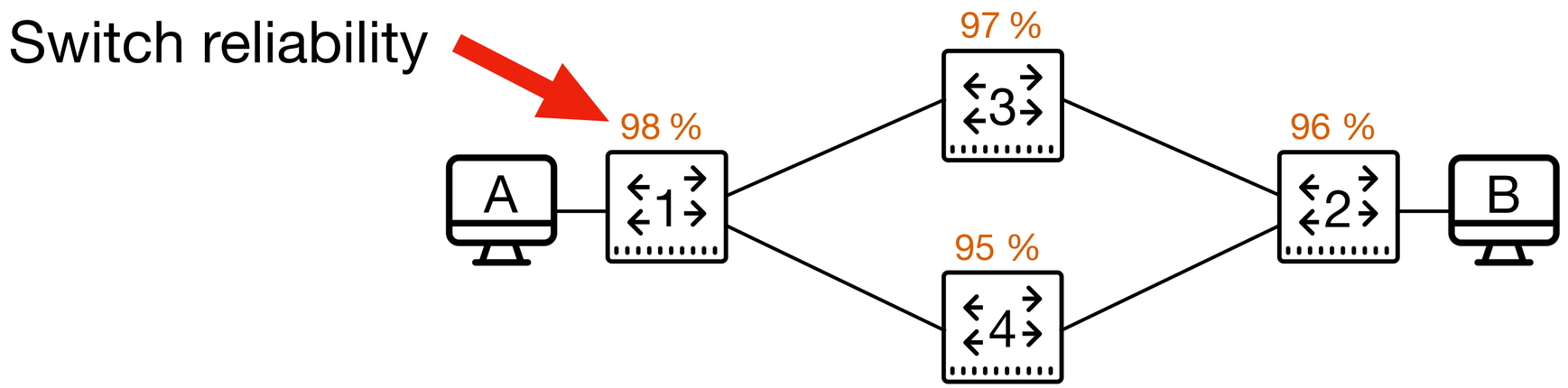
4

net

r ⊙ p

weighting:
“do p with weight r”

Modeling Quantitative Network Behavior



$p_1 \triangleq$

```
if dst = B then
  sw <- 3 ⊕ sw <- 4
else ...
```

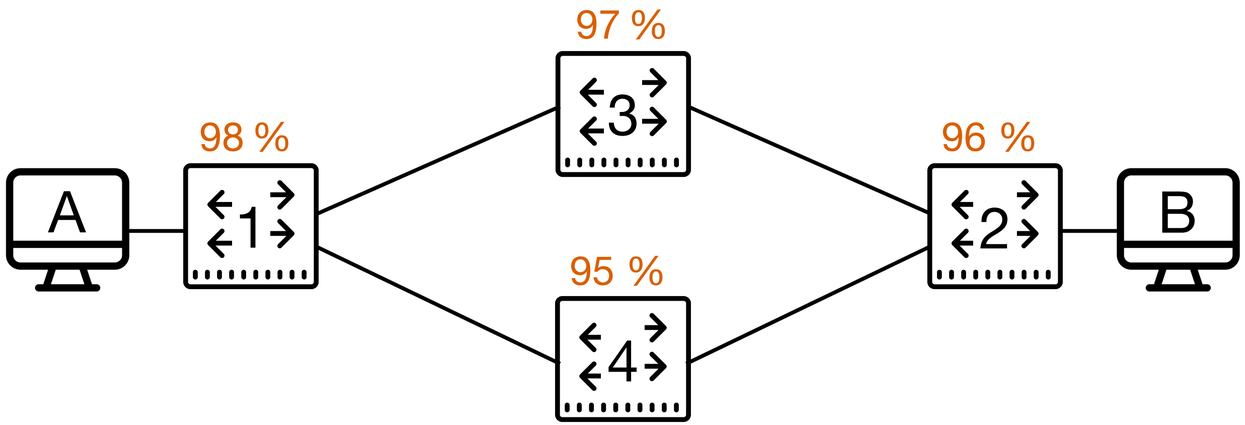
$p_2 \triangleq$ ... $p_3 \triangleq$ ... $p_4 \triangleq$ ...

$p \triangleq$

```
if sw = 1 then p1
else if sw = 2 then p2
else if sw = 3 then p3
else if sw = 4 then p4
```

$net \triangleq (p)^*$

Modeling Quantitative Network Behavior



$p_1 \triangleq$

```
if dst = B then
  sw <- 3 ⊕ sw <- 4
else ...
```

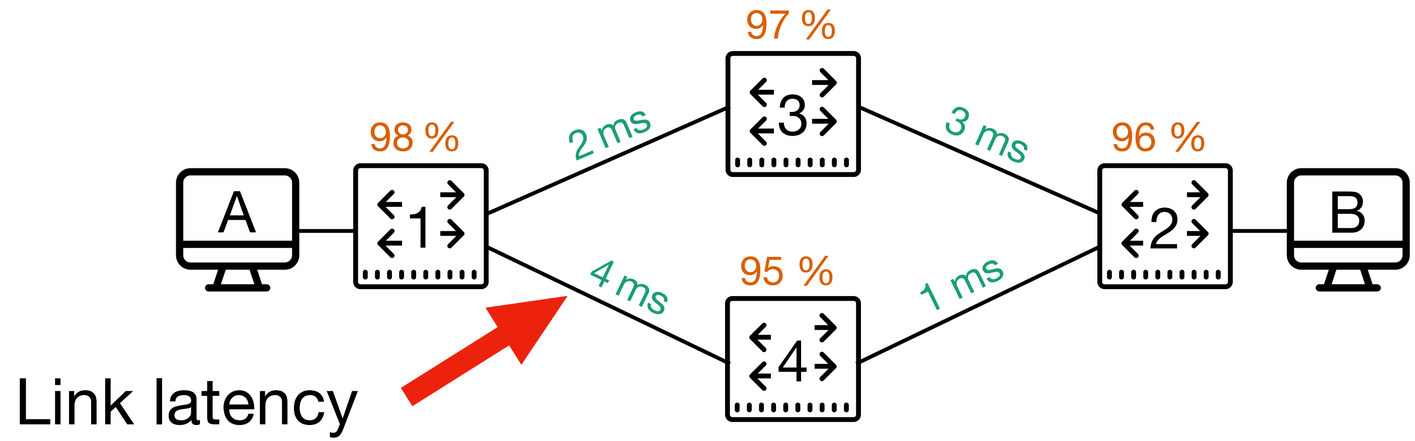
$p_2 \triangleq$ ... $p_3 \triangleq$ ... $p_4 \triangleq$ ...

$p \triangleq$

```
if sw = 1 then 98% ⊙ p1
else if sw = 2 then 96% ⊙ p2
else if sw = 3 then 97% ⊙ p3
else if sw = 4 then 95% ⊙ p4
```

$net \triangleq (p)^*$

Modeling Quantitative Network Behavior



$p_1 \triangleq$

```
if dst = B then
  sw <- 3 ⊕ sw <- 4
else ...
```

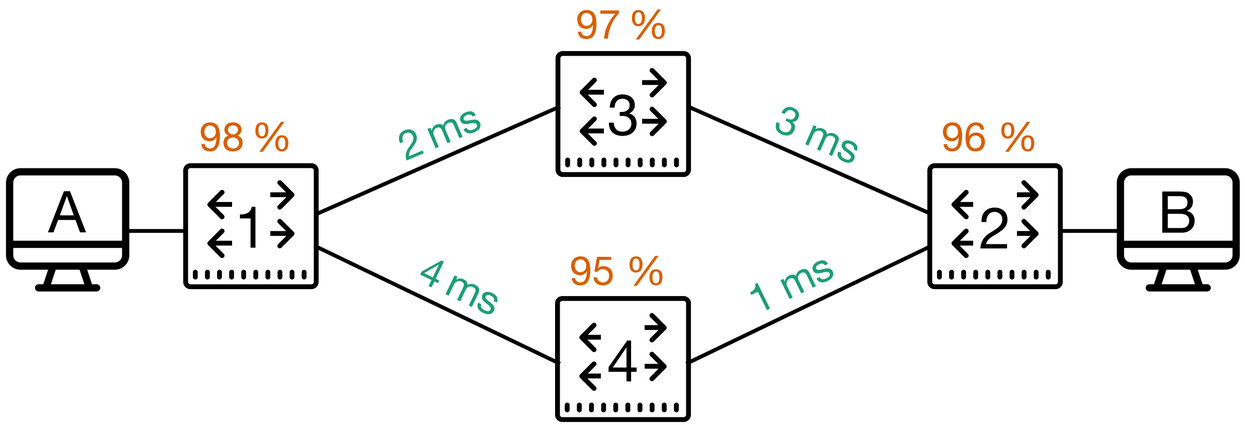
$p_2 \triangleq$ ... $p_3 \triangleq$ ... $p_4 \triangleq$ ...

$p \triangleq$

```
if sw = 1 then p1
else if sw = 2 then p2
else if sw = 3 then p3
else if sw = 4 then p4
```

$net \triangleq (p)^*$

Modeling Quantitative Network Behavior



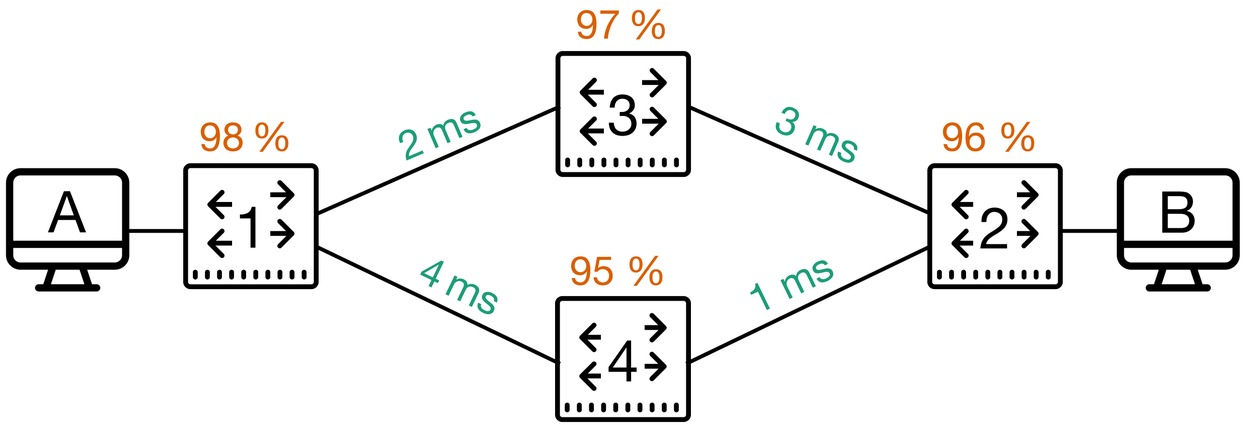
$p_1 \triangleq$

```
  if dst = B then

     (2ms ⊙ sw <- 3) ⊕
          (4ms ⊙ sw <- 4)

  else ...
```

$p_2 \triangleq$ ... $p_3 \triangleq$ ... $p_4 \triangleq$ ...

```
p ≜       if sw = 1 then p1
      else if sw = 2 then p2
      else if sw = 3 then p3
      else if sw = 4 then p4

net ≜ ( p )*
```

Modeling Quantitative Network Behavior

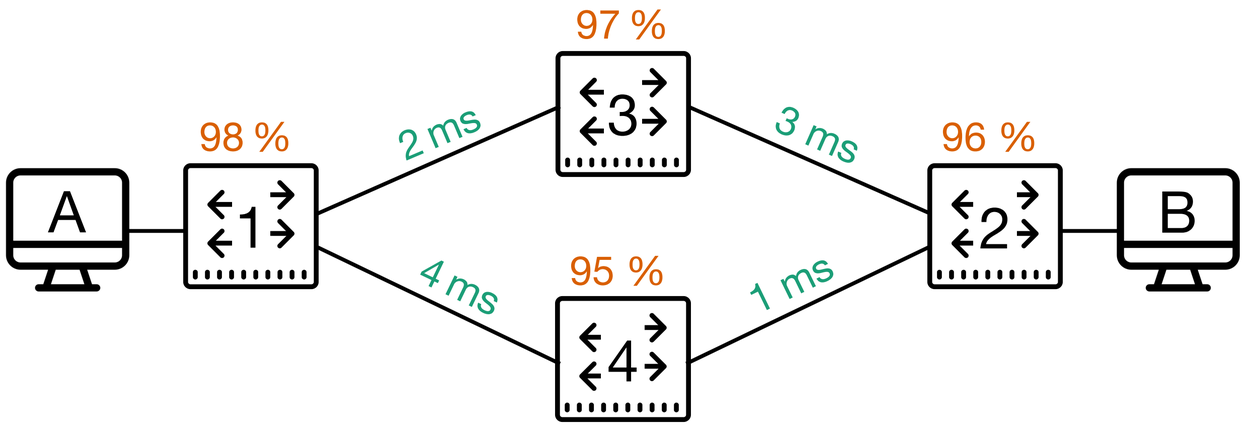


weighted choice

```
p1 ≜
  if dst = B then
     (2ms ⊙ sw <- 3) ⊕
          (4ms ⊙ sw <- 4)
  else ...
```

```
p2 ≜ ...    p3 ≜ ...    p4 ≜ ...

p ≜       if sw = 1 then p1
     else if sw = 2 then p2
     else if sw = 3 then p3
     else if sw = 4 then p4

net ≜ ( p )*
```

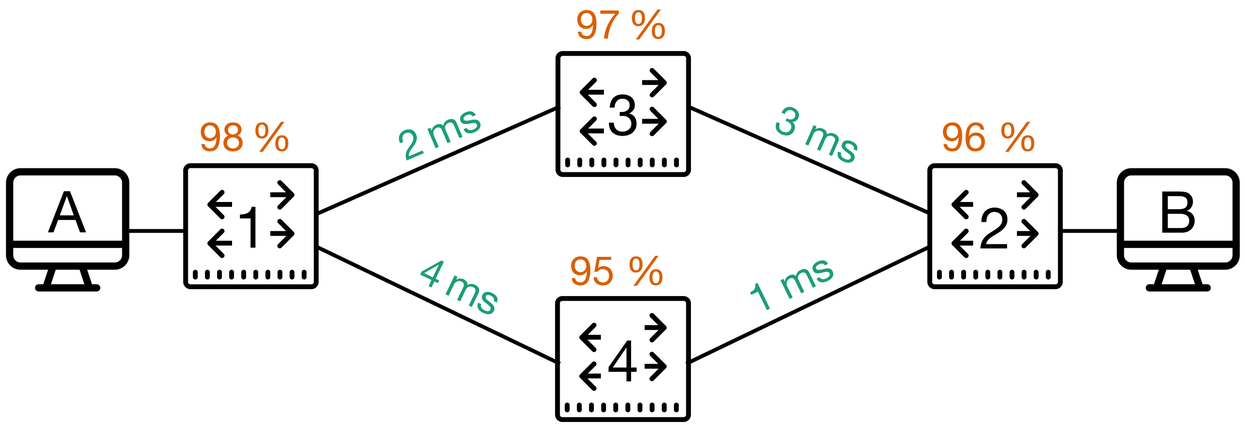
Modeling Quantitative Network Behavior



Weights of network (and operations $\odot$, $\oplus$) drawn from a **semiring**

$$\mathcal{S} \triangleq (S, +, \cdot, 0, 1)$$

Modeling Quantitative Network Behavior

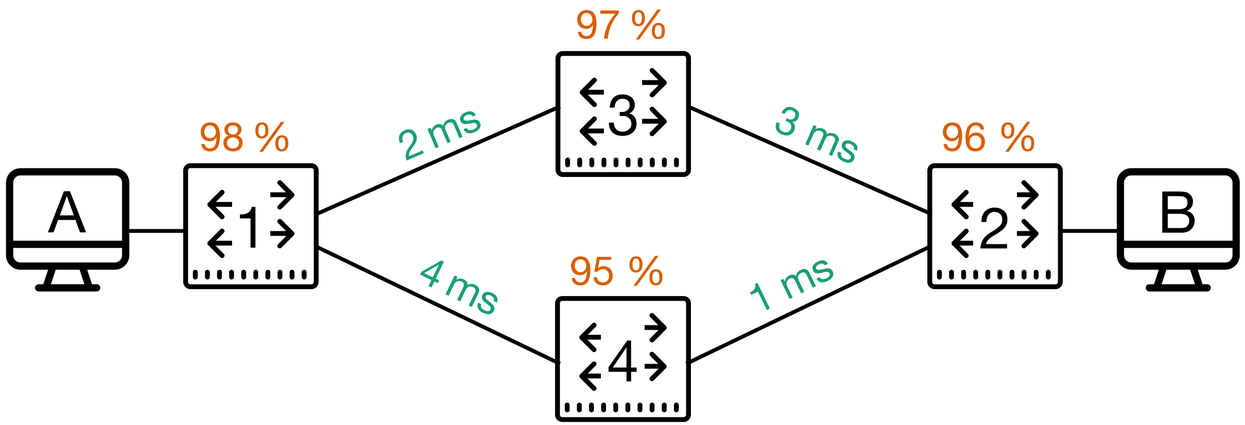


Weights of network (and operations $\odot$, $\oplus$) drawn from a **semiring**

$$\mathcal{S} \triangleq (S, +, \cdot, 0, 1)$$

- Combining/accumulating weights along a path (multiplication)

Modeling Quantitative Network Behavior

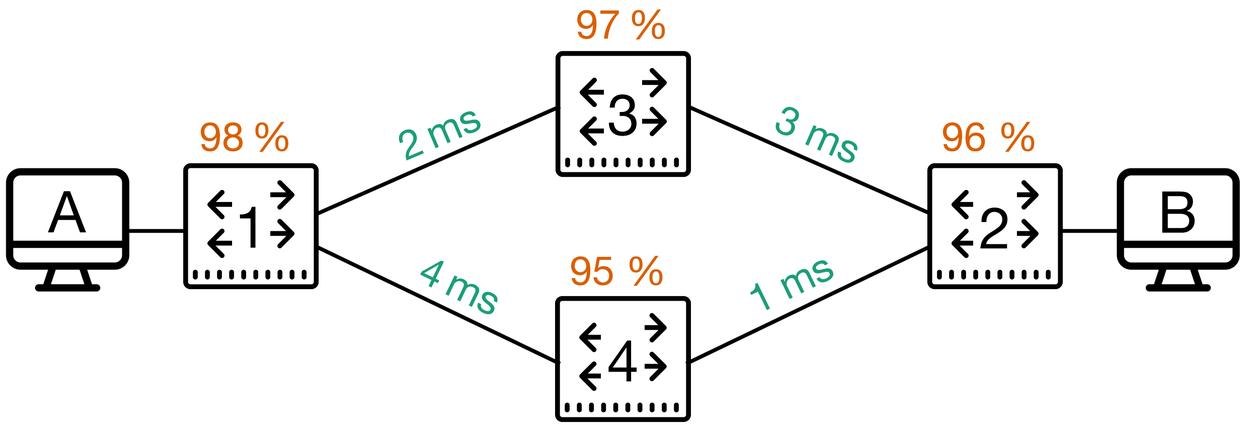


Weights of network (and operations $\odot$, $\oplus$) drawn from a **semiring**

$$\mathcal{S} \triangleq (S, +, \cdot, 0, 1)$$

- Combining/accumulating weights along a path (multiplication)
- Choosing between alternate paths (addition)

Modeling Quantitative Network Behavior

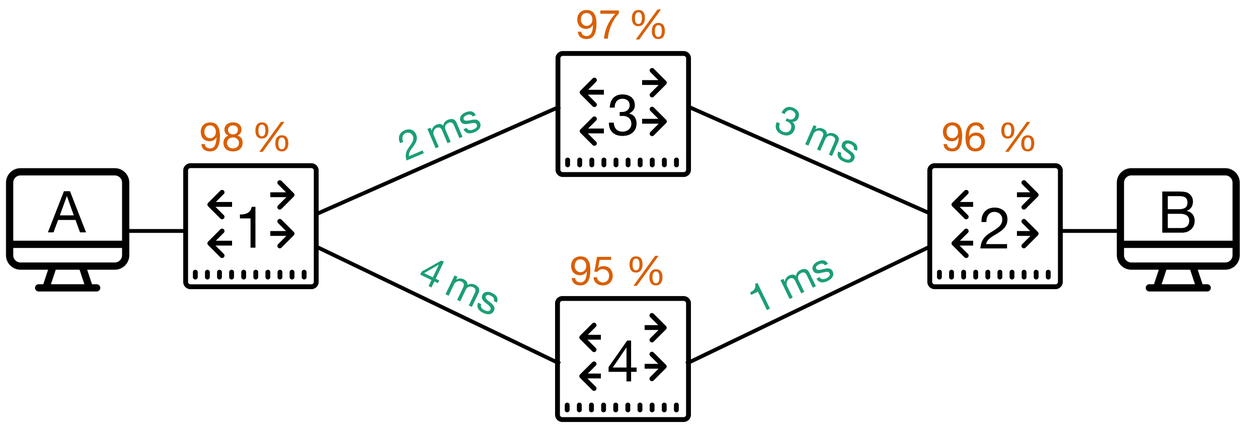


Weights of network (and operations $\odot$, $\oplus$) drawn from a **semiring**

$$\mathcal{S} \triangleq (S, +, \cdot, 0, 1)$$

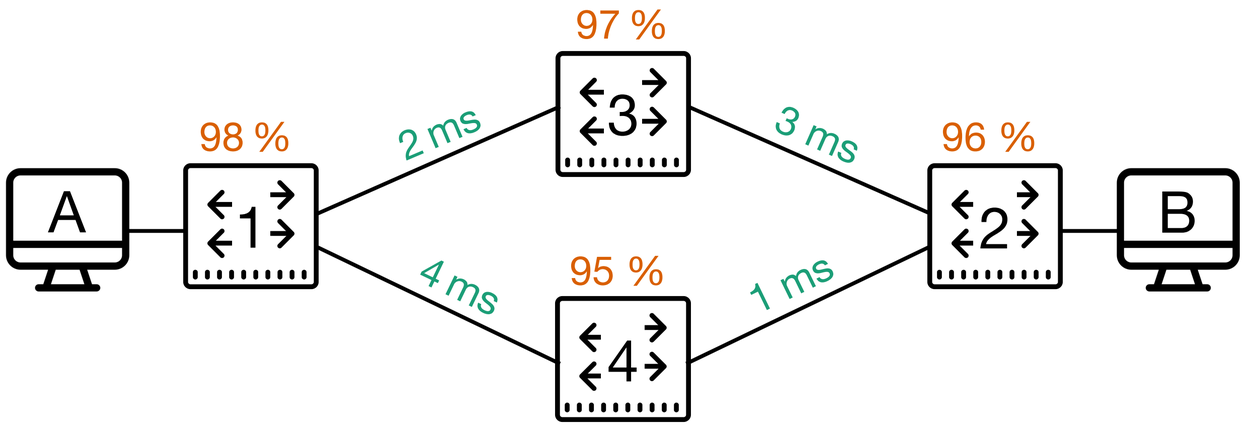
- Combining/accumulating weights along a path (multiplication)
- Choosing between alternate paths (addition)
- **Quantitative behavior we are modeling varies based on the choice of semiring**

Modeling Quantitative Network Behavior



Tropical Semiring
$(\mathbb{N}^{+\infty}, \texttt{min}, +, +\infty, 0)$
to model **Best-Case Latency**

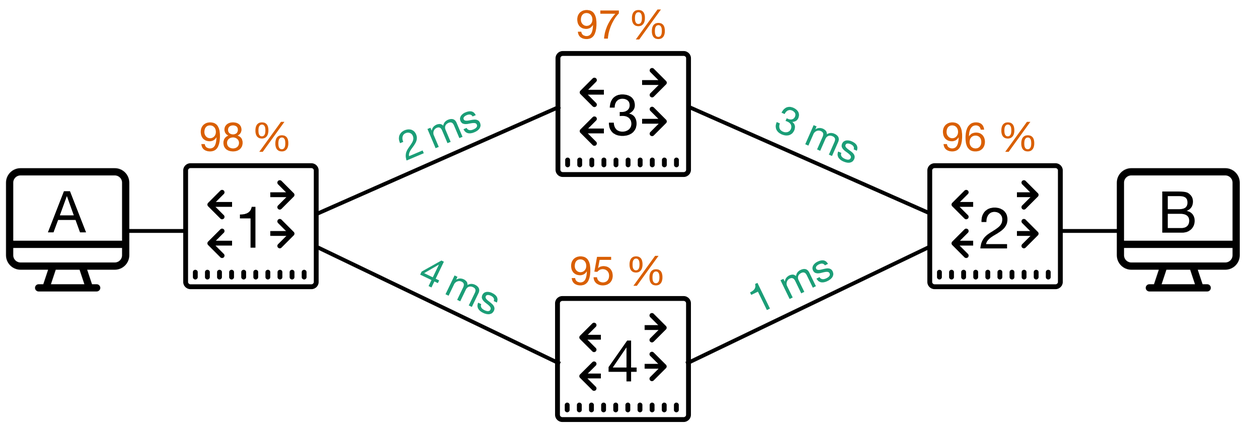
Modeling Quantitative Network Behavior



Tropical Semiring
$(\mathbb{N}^{+\infty}, \texttt{min}, +, +\infty, 0)$
to model **Best-Case Latency**

- weighted choice := `min` (choose better latency)

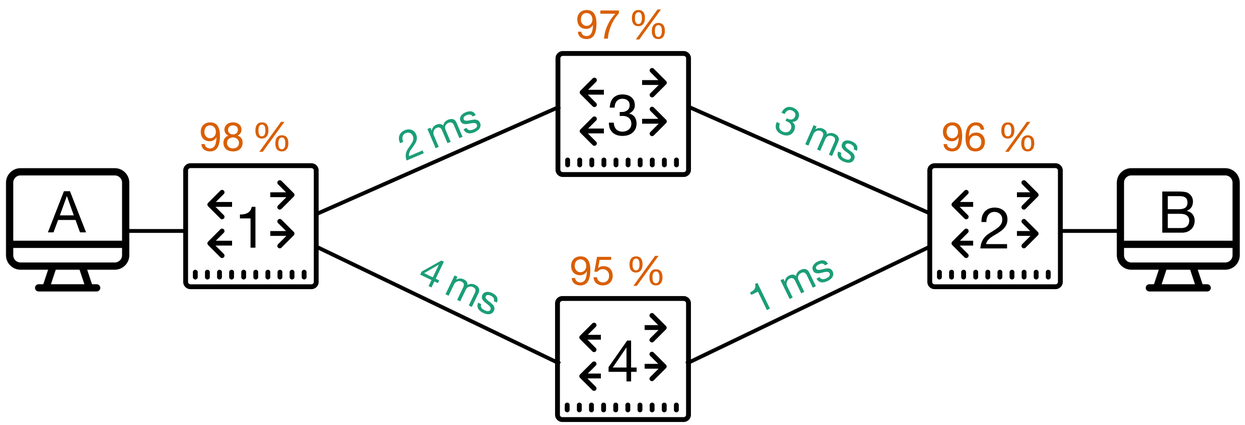
Modeling Quantitative Network Behavior



Tropical Semiring
$(\mathbb{N}^{+\infty}, \texttt{min}, +, +\infty, 0)$
to model **Best-Case Latency**

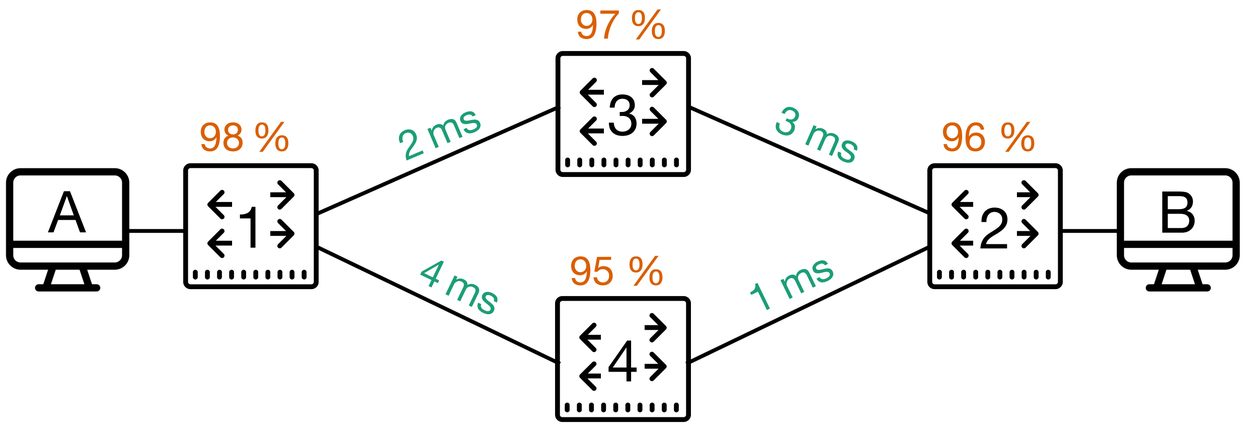
- weighted choice := `min` (choose better latency)
- path accumulation := + (add latencies together)

Modeling Quantitative Network Behavior



Viterbi Semiring
`([0,1], max, ·, 0, 1)`
to model **Best-Case Reliability**

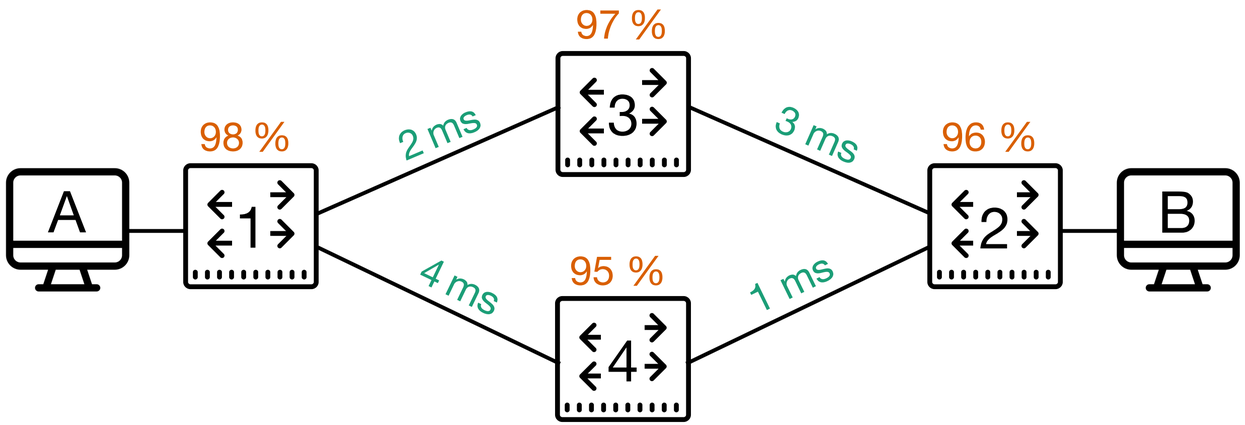
Modeling Quantitative Network Behavior



Viterbi Semiring
`([0,1], max, ·, 0, 1)`
to model **Best-Case Reliability**

- weighted choice := `max` (choose better reliability)

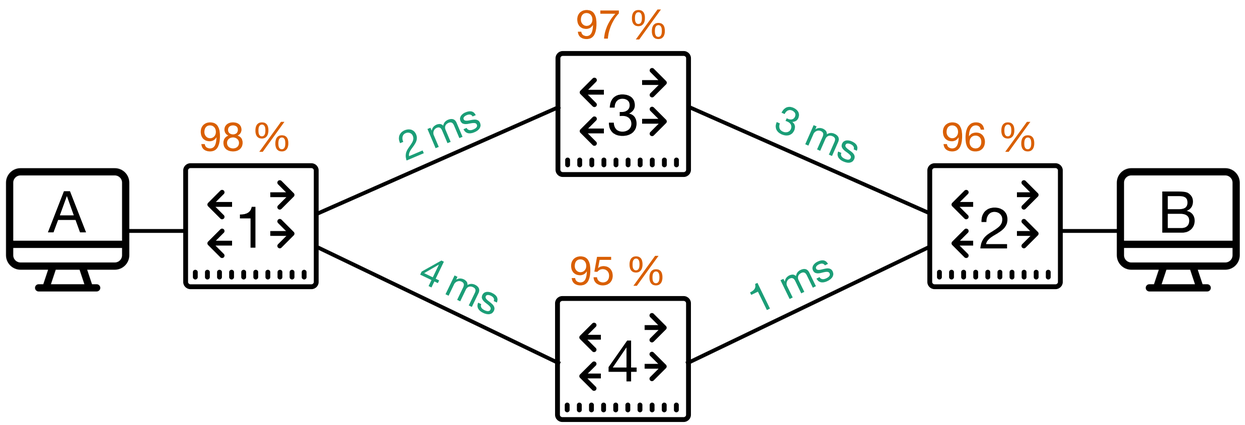
Modeling Quantitative Network Behavior



Viterbi Semiring
`([0,1], max, ·, 0, 1)`
to model **Best-Case Reliability**

- weighted choice := `max` (choose better reliability)
- path accumulation := · (multiply probabilities together)

Modeling Quantitative Network Behavior

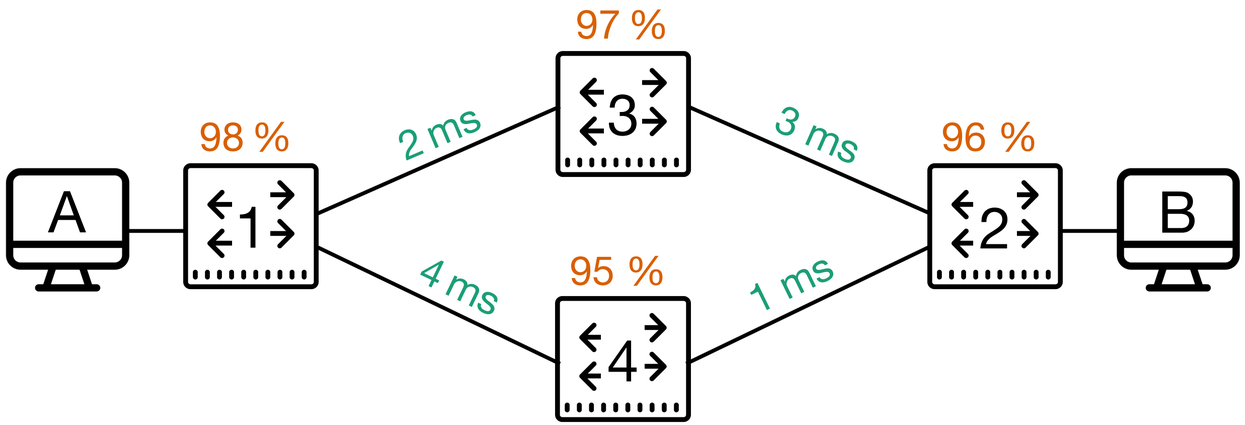


Worst

Viterbi Semiring
`([0,1], max, ·, 0, 1)`
to model ~~**Best**~~**-Case Reliability**

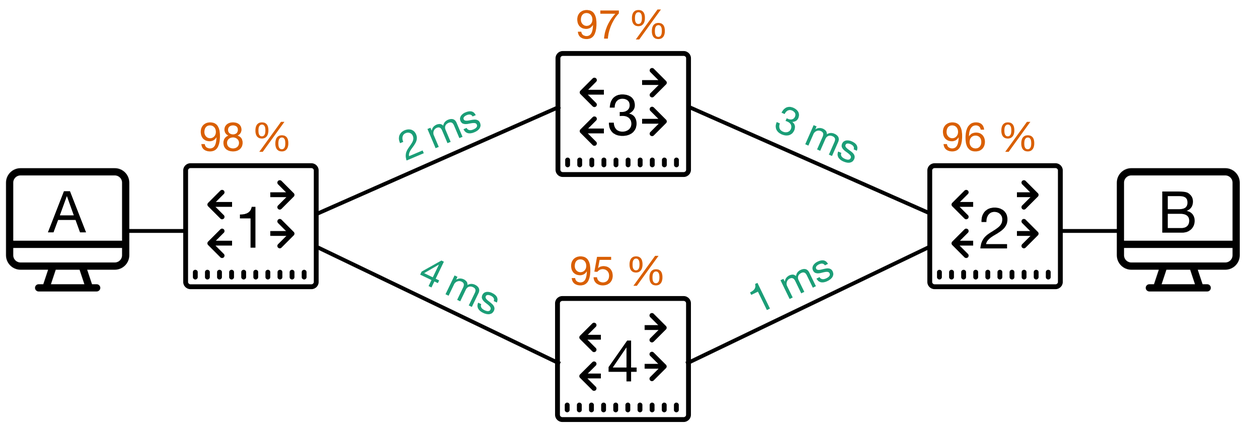
- weighted choice := `max` (choose better reliability)
- path accumulation := · (multiply probabilities together)

Modeling Quantitative Network Behavior



Probabilistic-union Semiring
$([0,1] \cup \{-\infty\}, \max, \uplus, -\infty, 0)$
to model **Worst-Case Reliability**

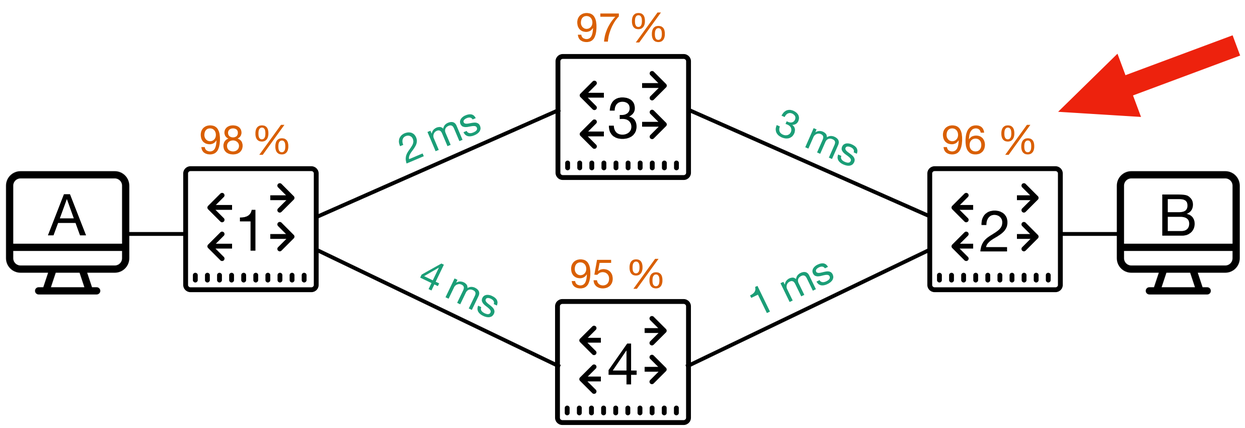
Modeling Quantitative Network Behavior



Probabilistic-union Semiring
$([0,1] \cup \{-\infty\}, \max, \uplus, -\infty, 0)$
to model **Worst-Case Reliability**

- weighted choice := $\max$ (choose worse failure rate)

Modeling Quantitative Network Behavior

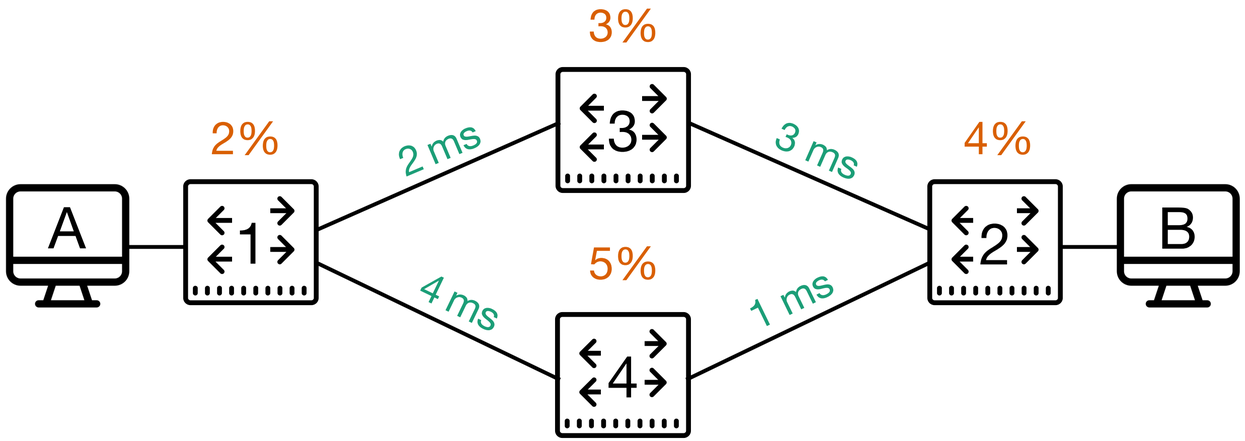


96% success rate
=
4% failure rate

Probabilistic-union Semiring
$([0,1] \cup \{-\infty\}, \max, \uplus, -\infty, 0)$
to model **Worst-Case Reliability**

- weighted choice := $\max$ (choose worse failure rate)

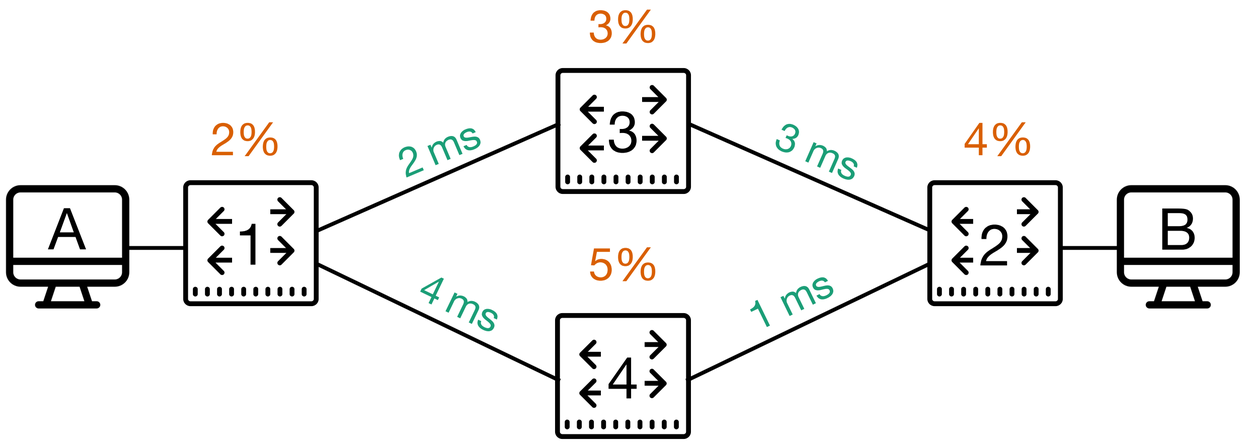
Modeling Quantitative Network Behavior



Probabilistic-union Semiring
$([0,1] \cup \{-\infty\}, \max, \uplus, -\infty, 0)$
to model **Worst-Case Reliability**

- weighted choice := $\max$ (choose worse failure rate)

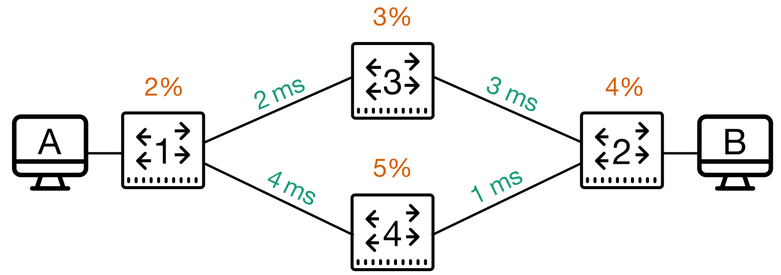
Modeling Quantitative Network Behavior



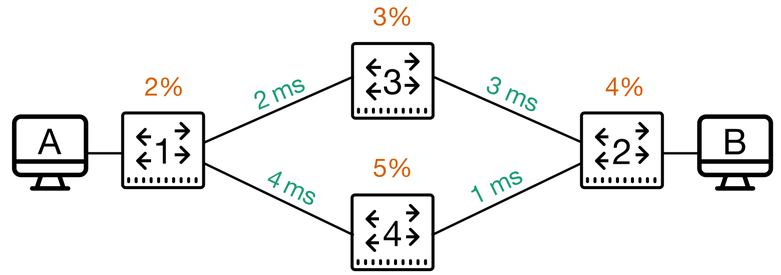
Probabilistic-union Semiring
$([0,1] \cup \{-\infty\}, \max, \uplus, -\infty, 0)$
to model **Worst-Case Reliability**

- weighted choice := $\max$ (choose worse failure rate)
- path accumulation := $\uplus$ (probability of union of 2 events: p + q - pq)

From Modeling to Verification

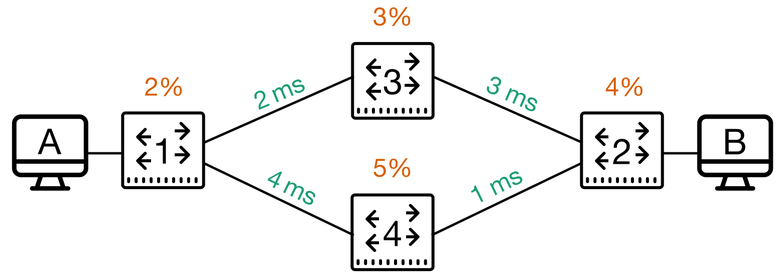


From Modeling to Verification



Program `p` takes input packet `π` and assigns a *weight* to every output trace `h`

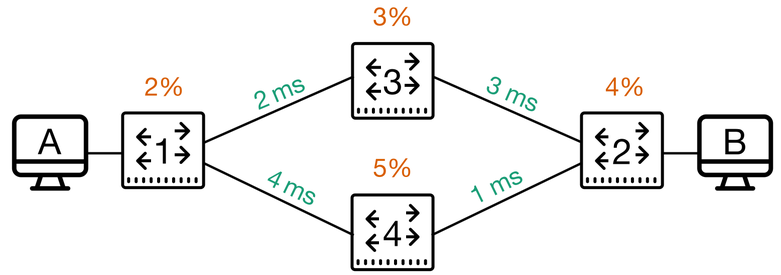
From Modeling to Verification



Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ \mathtt{0},\ \mathtt{1})$$

From Modeling to Verification

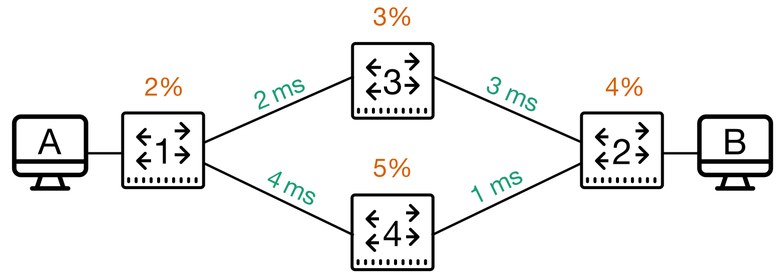


Program $\mathtt{p}$ takes input packet $\mathtt{\pi}$ and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\mathtt{\pi})(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ \mathtt{0},\ \mathtt{1})$$

Note:

From Modeling to Verification



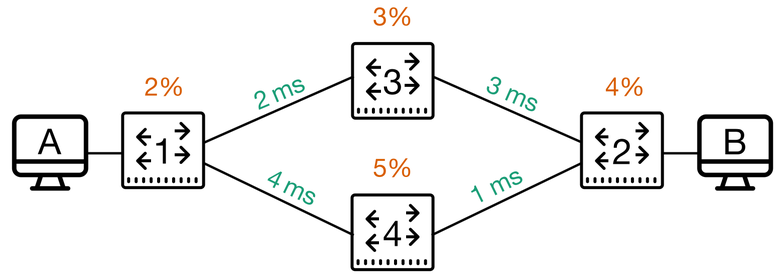
Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

Note:

- Semantics are *purely functional* (i.e., input packet, output trace)

From Modeling to Verification



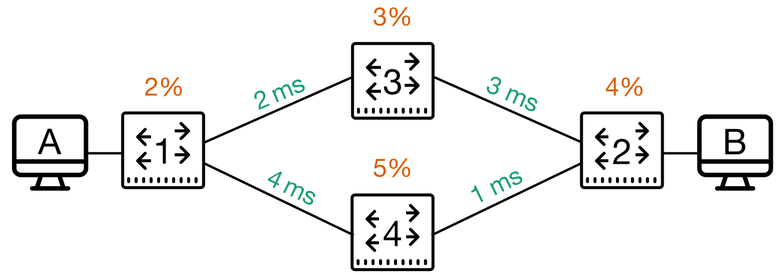
Program $\mathtt{p}$ takes input packet $\mathtt{\pi}$ and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\mathtt{\pi})(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

Note:

- Semantics are *purely functional* (i.e., input packet, output trace)
  - No support for queuing / congestion

From Modeling to Verification



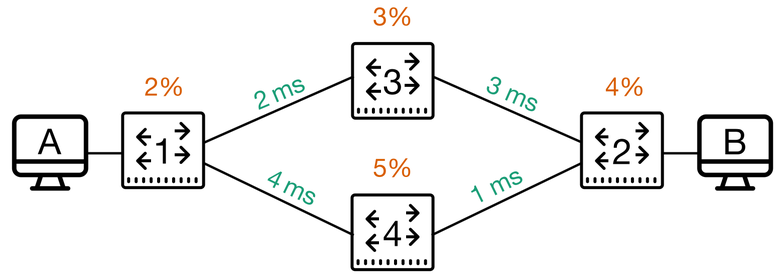
Program $\mathtt{p}$ takes input packet $\mathtt{\pi}$ and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\mathtt{\pi})(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

Note:

- Semantics are *purely functional* (i.e., input packet, output trace)
  - No support for queuing / congestion
- Still suitable for many **Quantitative Verification Questions**

From Modeling to Verification



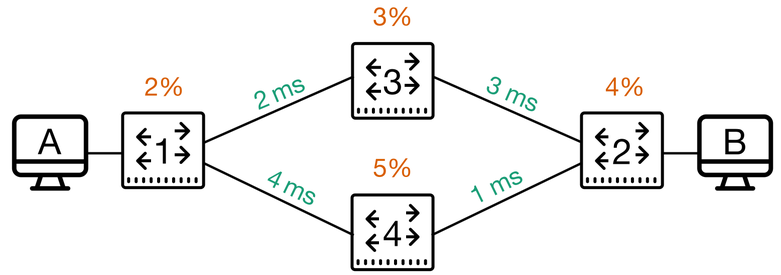
Program $\mathtt{p}$ takes input packet $\mathtt{\pi}$ and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\mathtt{\pi})(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

Note:

- Semantics are *purely functional* (i.e., input packet, output trace)
  - No support for queuing / congestion
- Still suitable for many **Quantitative Verification Questions**
  - modern network hardware has made packet-level interactions less important

From Modeling to Verification



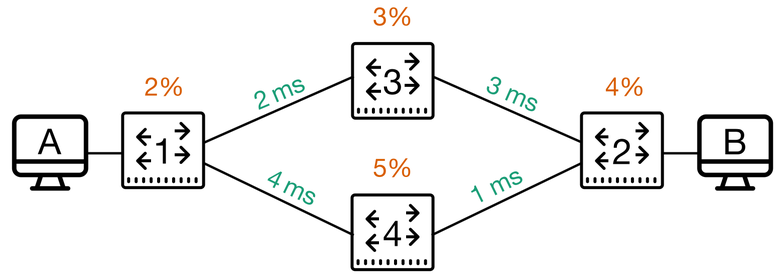
Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

Note:

- Semantics are *purely functional* (i.e., input packet, output trace)
  - No support for queuing / congestion
- Still suitable for many **Quantitative Verification Questions**
  - modern network hardware has made packet-level interactions less important
  - many quantities are independent of congestion (e.g., reliability, security)

From Modeling to Verification

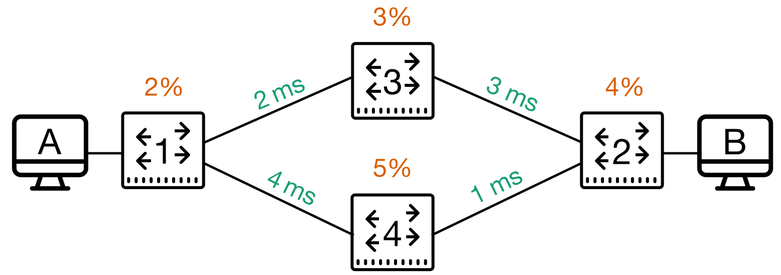


Program $\mathtt{p}$ takes input packet $\mathtt{\pi}$ and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\mathtt{\pi})(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ \mathtt{0},\ \mathtt{1})$$

Quantitative Verification Questions

From Modeling to Verification



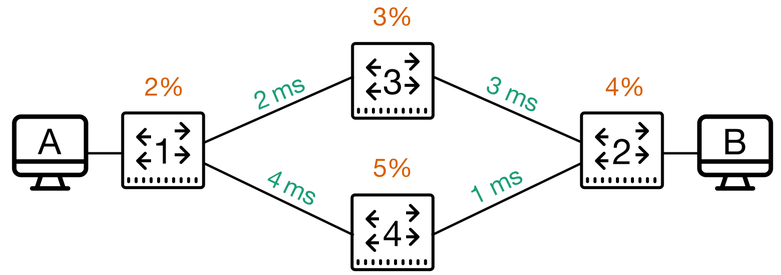
Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ \mathtt{0},\ \mathtt{1})$$

Quantitative Verification Questions

$\mathtt{r}$**-safety**
(Does all network traffic have weight *at most* $\mathtt{r}$)?

From Modeling to Verification



Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

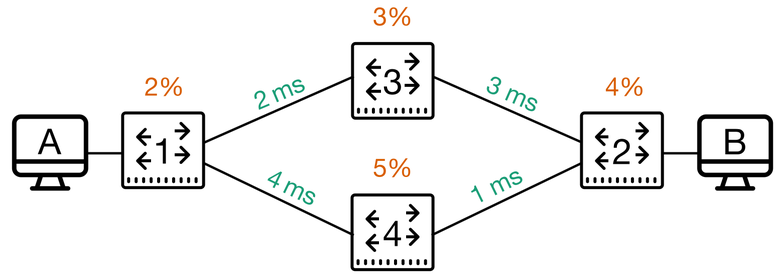
$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ \mathtt{0},\ \mathtt{1})$$

Quantitative Verification Questions

$\mathtt{r}$-safety
(Does all network traffic have weight *at most* $\mathtt{r}$)?

Does all traffic get delivered with at least 90% reliability?

From Modeling to Verification



Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

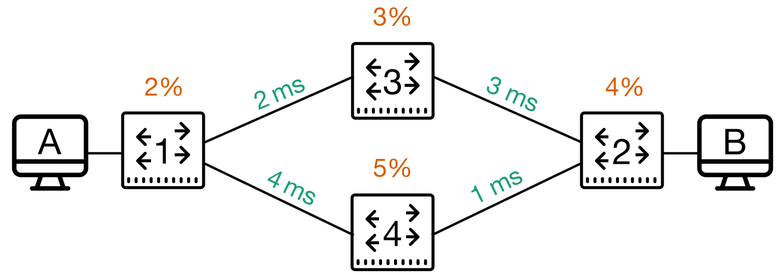
Quantitative Verification Questions

$\mathtt{r}$**-safety**
(Does all network traffic have weight *at most* $\mathtt{r}$)?

$$\forall \pi, \mathtt{h}:\ [\![\mathtt{p}]\!](\pi)(\mathtt{h}) \leq \mathtt{r}$$

Does all traffic get delivered with at least 90% reliability?

From Modeling to Verification



Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

Quantitative Verification Questions

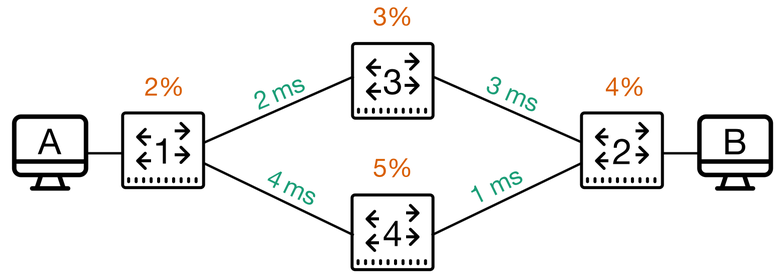
$\mathtt{r}$-safety
(Does all network traffic have weight *at most* $\mathtt{r}$)?

$$\forall \pi, \mathtt{h}:\ [\![\mathtt{p}]\!](\pi)(\mathtt{h}) \leq \mathtt{r}$$

$\mathtt{r}$-reachability
(Does there exist network traffic with weight *at least* $\mathtt{r}$)?

Does all traffic get delivered with at least 90% reliability?

From Modeling to Verification



Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

Quantitative Verification Questions

$\mathtt{r}$-safety
(Does all network traffic have weight *at most* $\mathtt{r}$)?

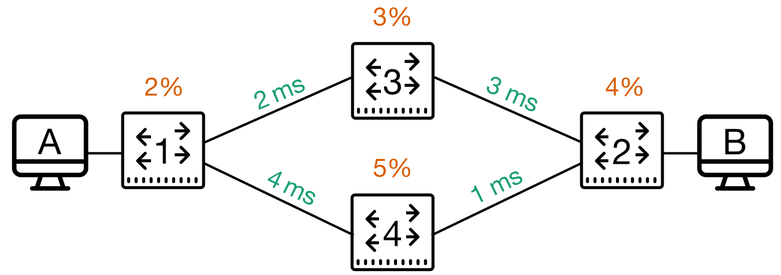
$$\forall \pi, \mathtt{h}:\ [\![\mathtt{p}]\!](\pi)(\mathtt{h}) \leq \mathtt{r}$$

Does all traffic get delivered with at least 90% reliability?

$\mathtt{r}$-reachability
(Does there exist network traffic with weight *at least* $\mathtt{r}$)?

Can host A deliver packets to host B within 5ms?

From Modeling to Verification



Program $\mathtt{p}$ takes input packet π and assigns a *weight* to every output trace $\mathtt{h}$

$$[\![\mathtt{p}]\!](\pi)(\mathtt{h}) \in \mathcal{S} \qquad\qquad \mathcal{S} \triangleq (\mathtt{S},\ +,\ \cdot,\ 0,\ 1)$$

Quantitative Verification Questions

$\mathtt{r}$-safety
(Does all network traffic have weight *at most* $\mathtt{r}$)?

$$\forall \pi, \mathtt{h}:\ [\![\mathtt{p}]\!](\pi)(\mathtt{h}) \leq \mathtt{r}$$

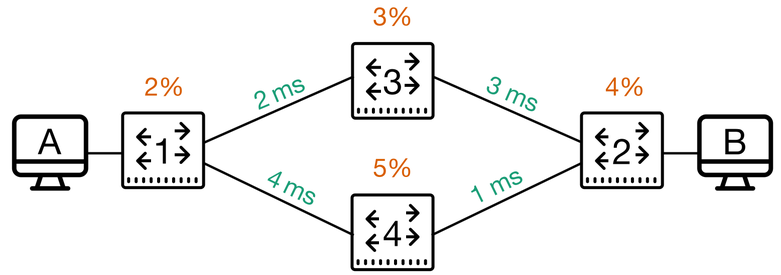
Does all traffic get delivered with at least 90% reliability?

$\mathtt{r}$-reachability
(Does there exist network traffic with weight *at least* $\mathtt{r}$)?

$$\exists \pi, \mathtt{h}:\ [\![\mathtt{p}]\!](\pi)(\mathtt{h}) \geq \mathtt{r}$$

Can host A deliver packets to host B within 5ms?

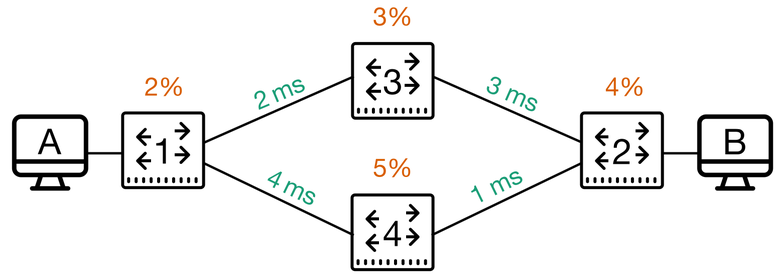
From Modeling to Verification



$$\forall \pi, h: \; [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \; [\![p]\!](\pi)(h) \geq r$$

From Modeling to Verification

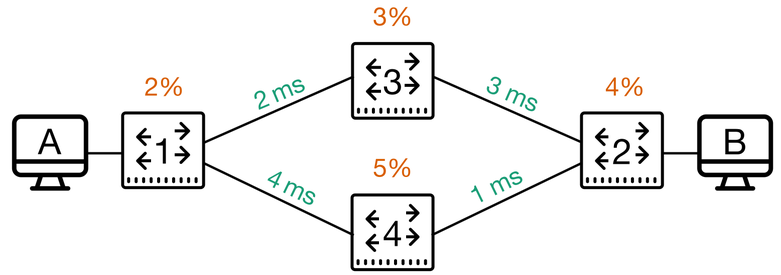


$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification
Need to compute semantics for:

From Modeling to Verification



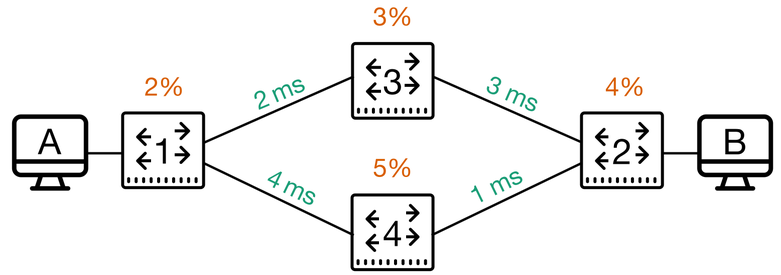
$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification
Need to compute semantics for:

- Infinitely many input packets

From Modeling to Verification



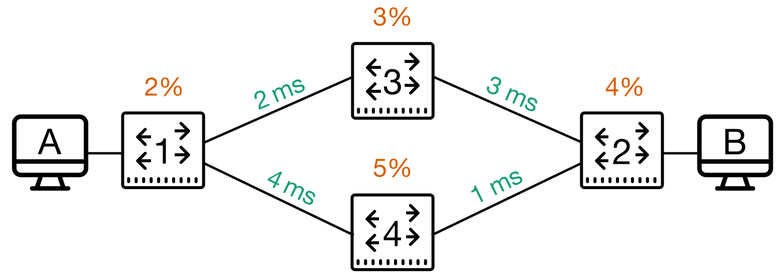
$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification
Need to compute semantics for:

- Infinitely many input packets
- Programs with (potentially) unbounded iteration

From Modeling to Verification



$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

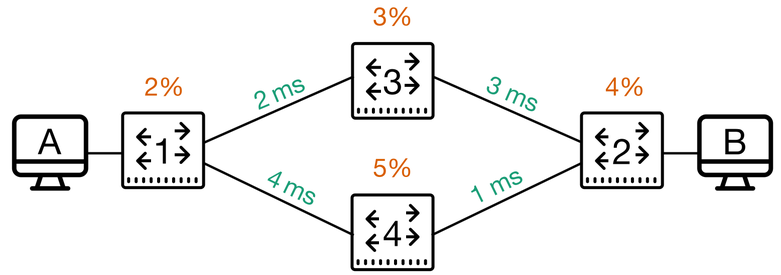
$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification
Need to compute semantics for:

- Infinitely many input packets
- Programs with (potentially) unbounded iteration

Approach
Automata-based model checking

From Modeling to Verification



$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

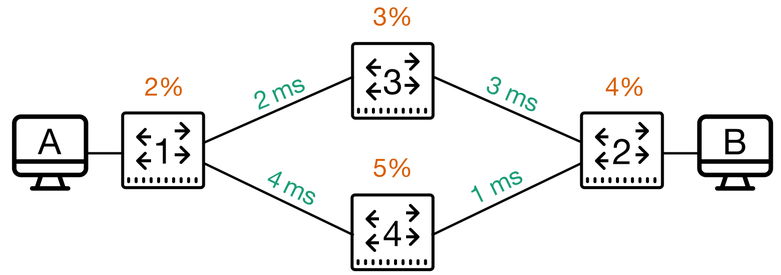
Automatic Verification
Need to compute semantics for:

- Infinitely many input packets
- Programs with (potentially) unbounded iteration

Approach
Automata-based model checking

Challenge

From Modeling to Verification



$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification
Need to compute semantics for:

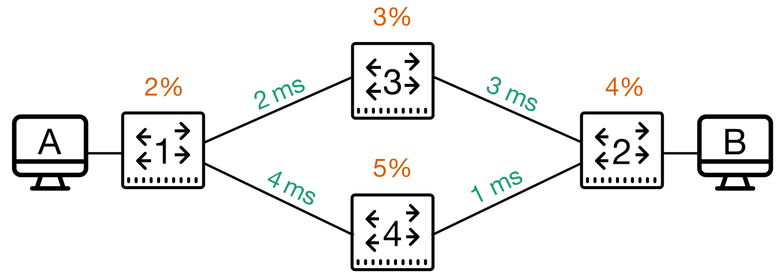
- Infinitely many input packets
- Programs with (potentially) unbounded iteration

Approach
Automata-based model checking

Challenge

- wNetKAT has *stateful packet transitions*

From Modeling to Verification



$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification
Need to compute semantics for:

- Infinitely many input packets
- Programs with (potentially) unbounded iteration

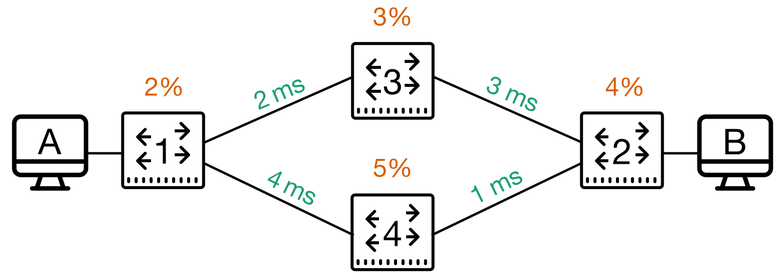
Approach
Automata-based model checking

Challenge

- wNetKAT has *stateful packet transitions*

```
if dst = B then sw <- 3
```

From Modeling to Verification



$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification
Need to compute semantics for:

- Infinitely many input packets
- Programs with (potentially) unbounded iteration

Approach
Automata-based model checking

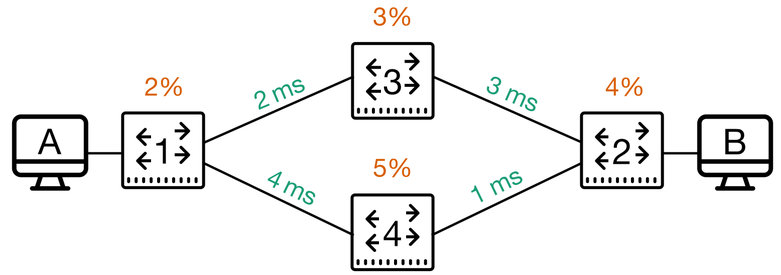
Challenge

- wNetKAT has *stateful packet transitions*

```
if dst = B then sw <- 3
```

  - need to keep track of “previous packet state”

From Modeling to Verification



$$\forall \pi, h: \; [\![p]\!](\pi)(h) \leq r$$

$$\exists \pi, h: \; [\![p]\!](\pi)(h) \geq r$$

Automatic Verification
Need to compute semantics for:

- Infinitely many input packets
- Programs with (potentially) unbounded iteration

Approach
Automata-based model checking

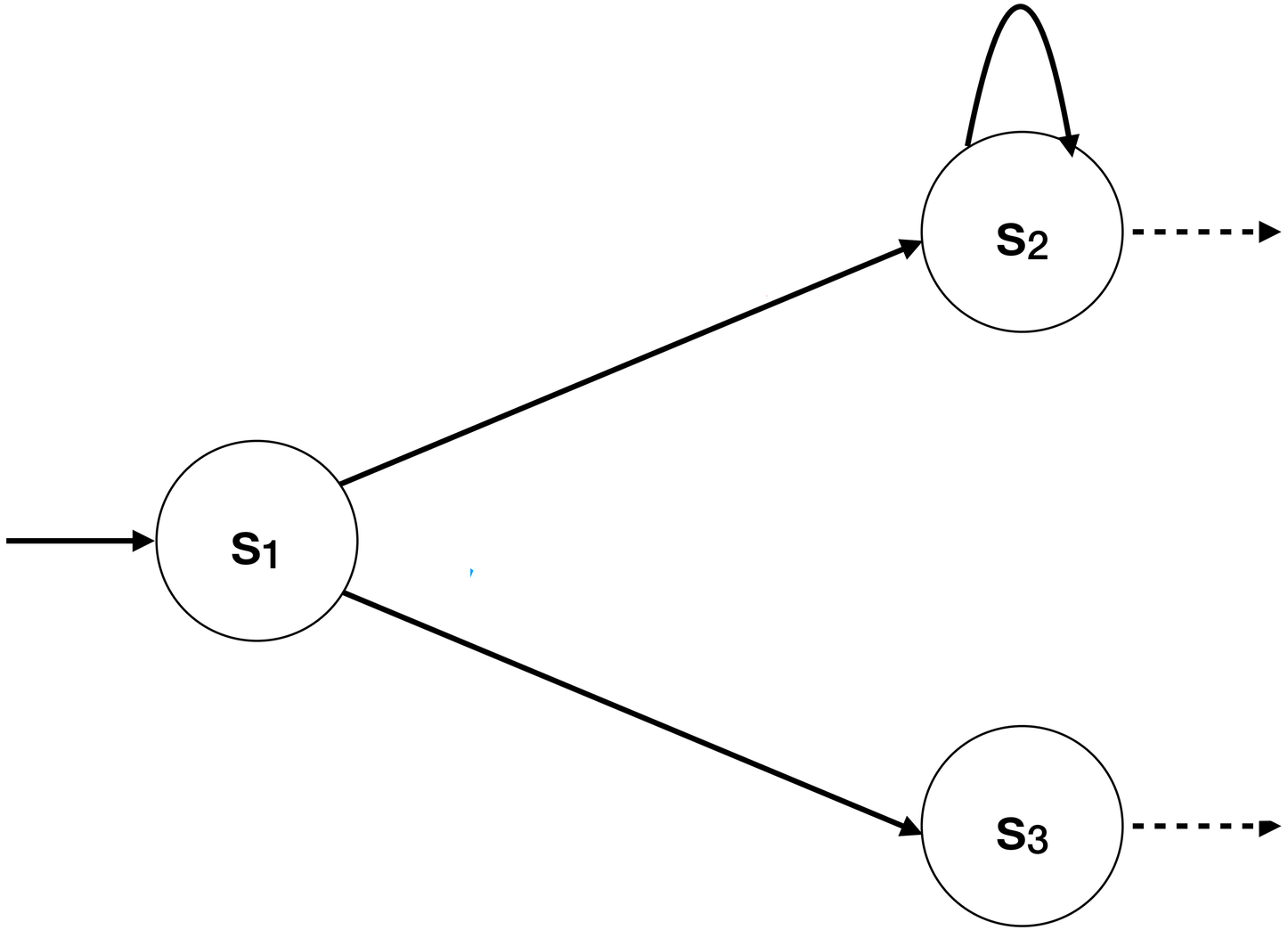
Challenge

- wNetKAT has *stateful packet transitions*

```
if dst = B then sw <- 3
```

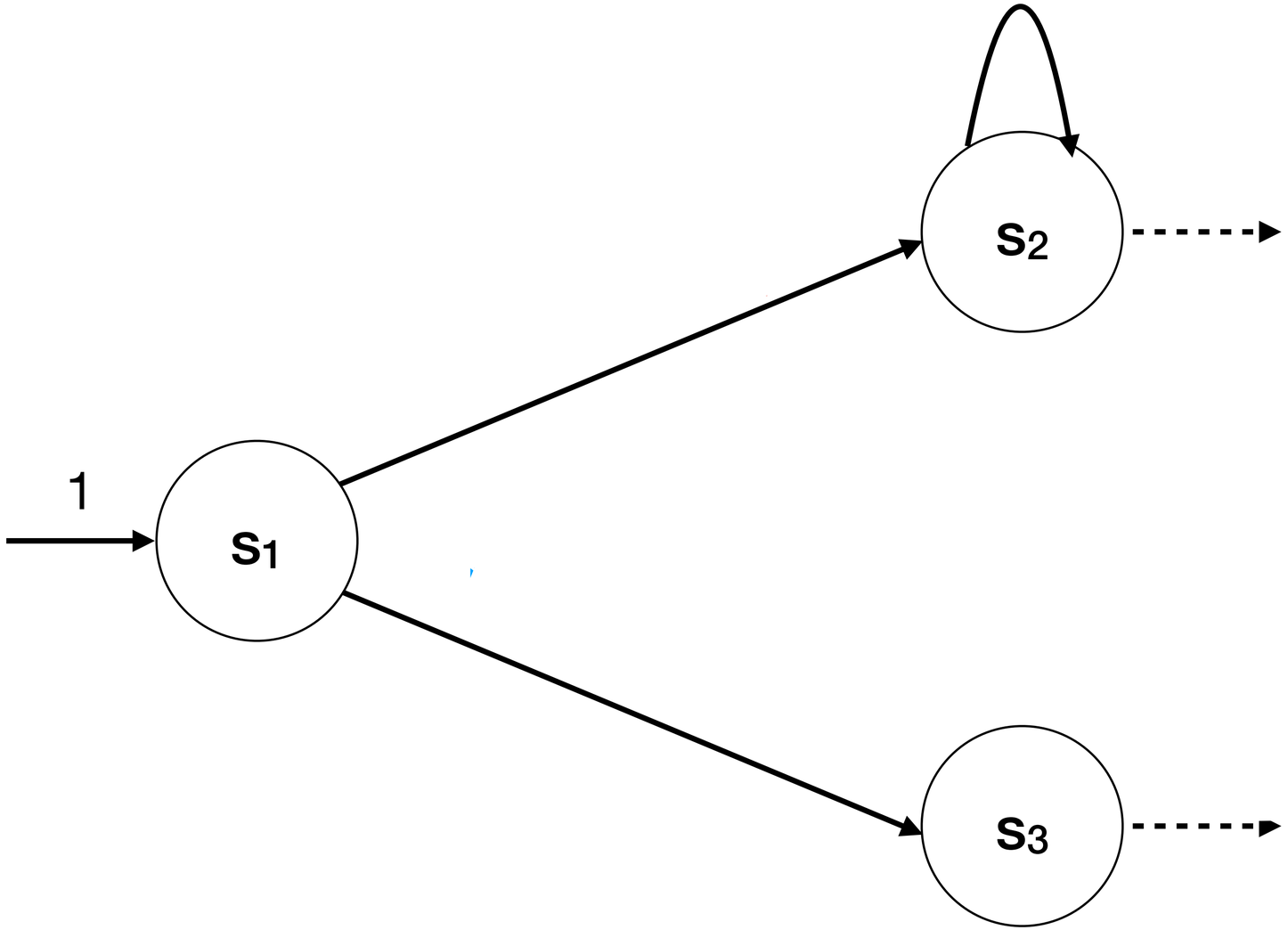
  - need to keep track of “previous packet state”
- Regular automata are not suitable

wNetKAT Automata



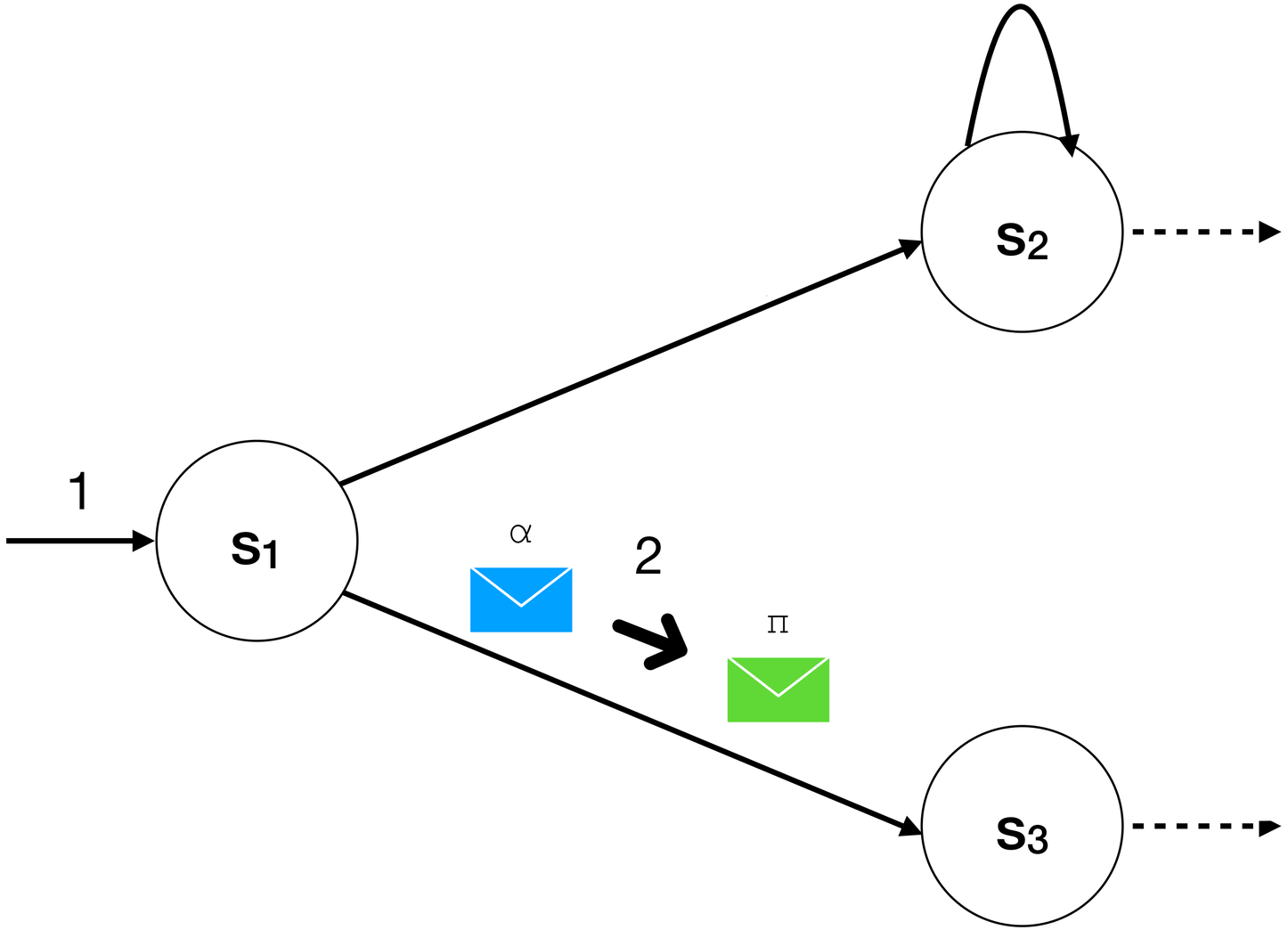
Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$

wNetKAT Automata



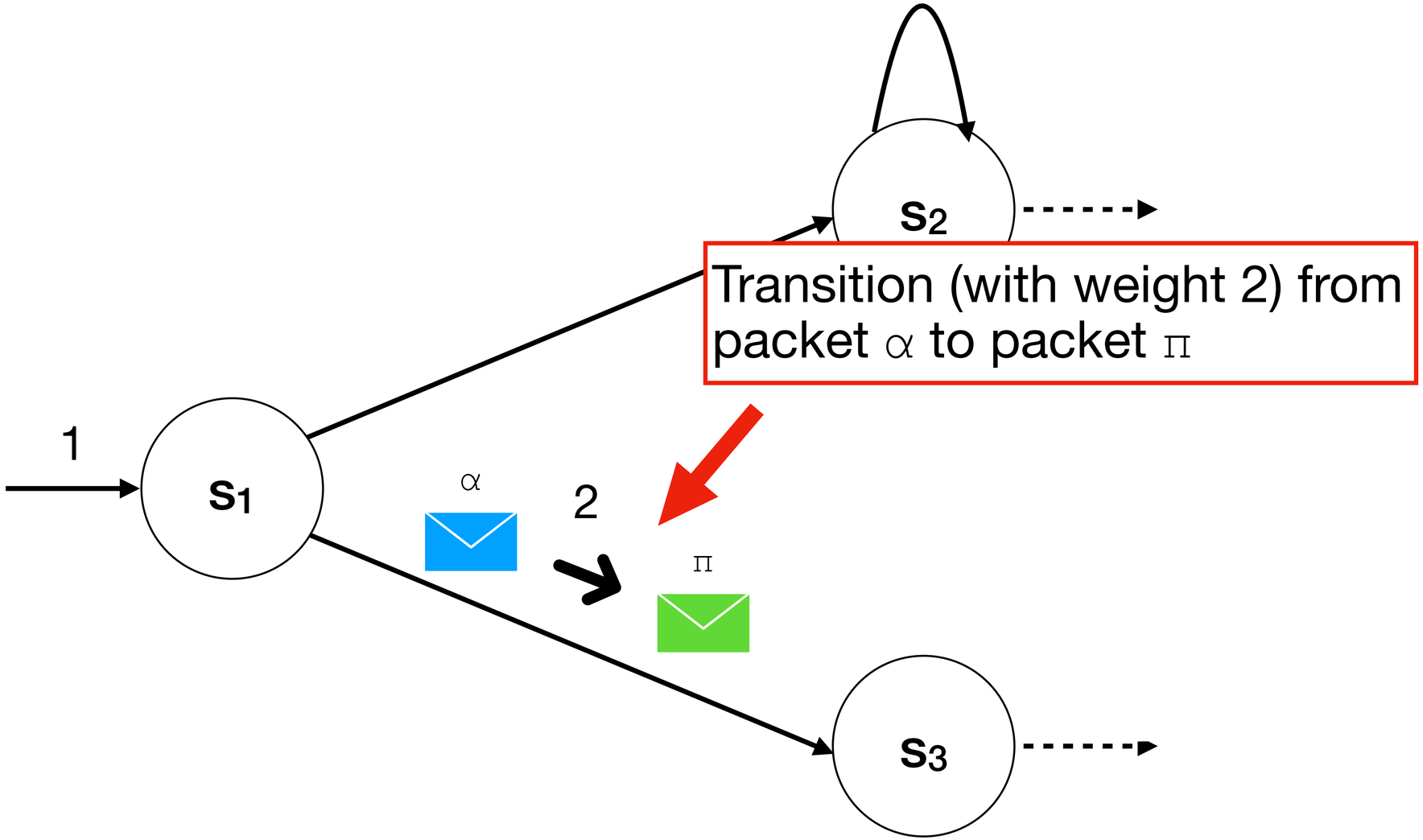
Tropical Semiring
$(\mathbb{N}^{+\infty}, \texttt{min}, +, +\infty, 0)$

wNetKAT Automata



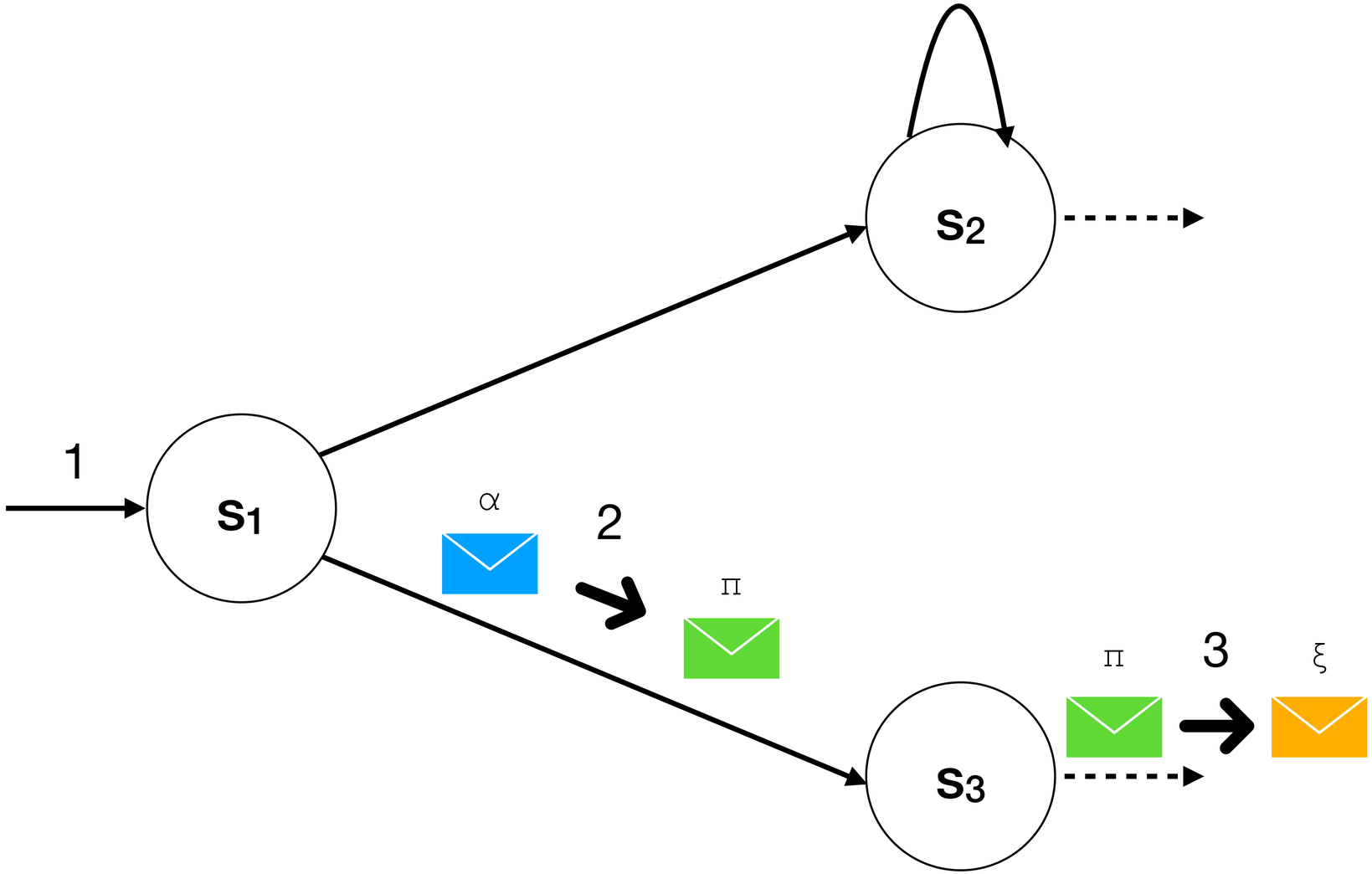
Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$

wNetKAT Automata



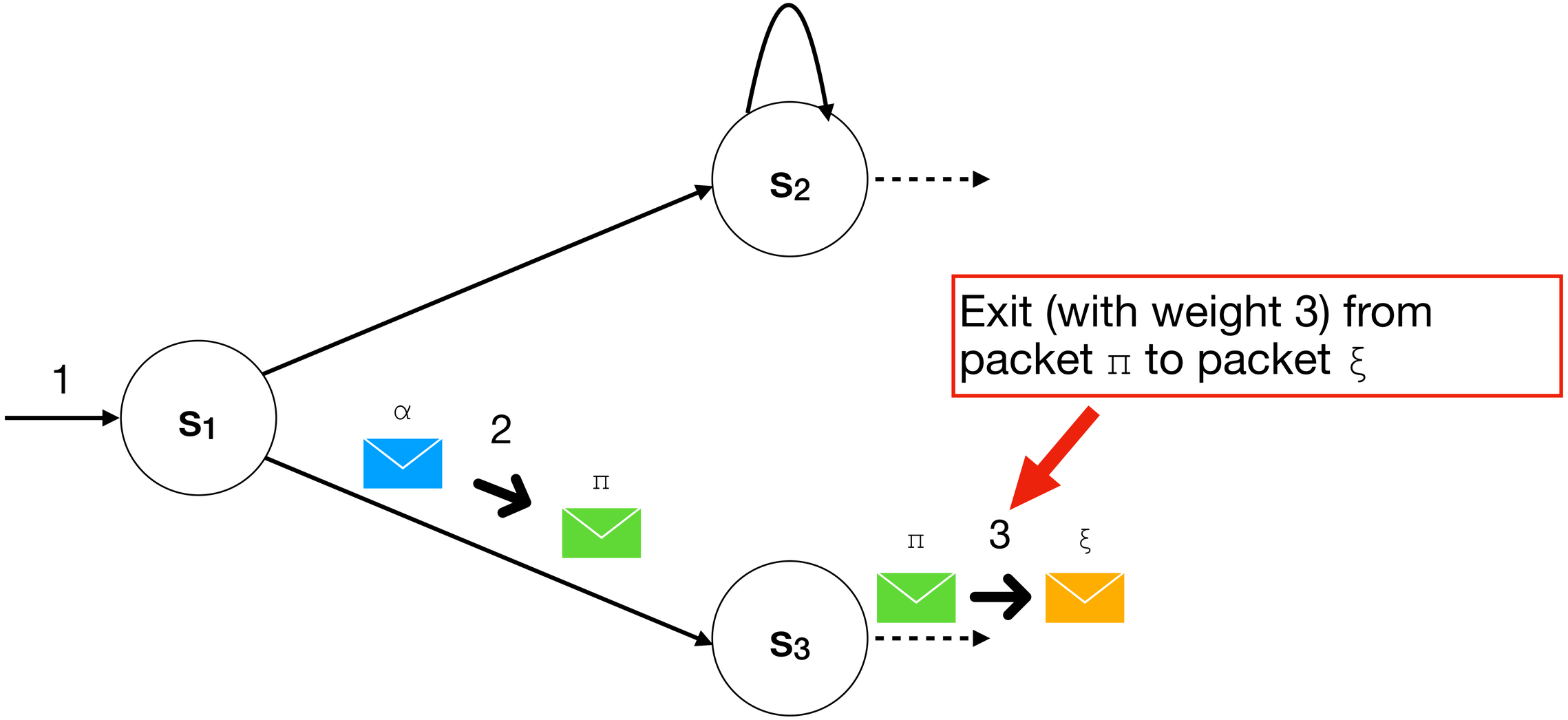
Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$

wNetKAT Automata



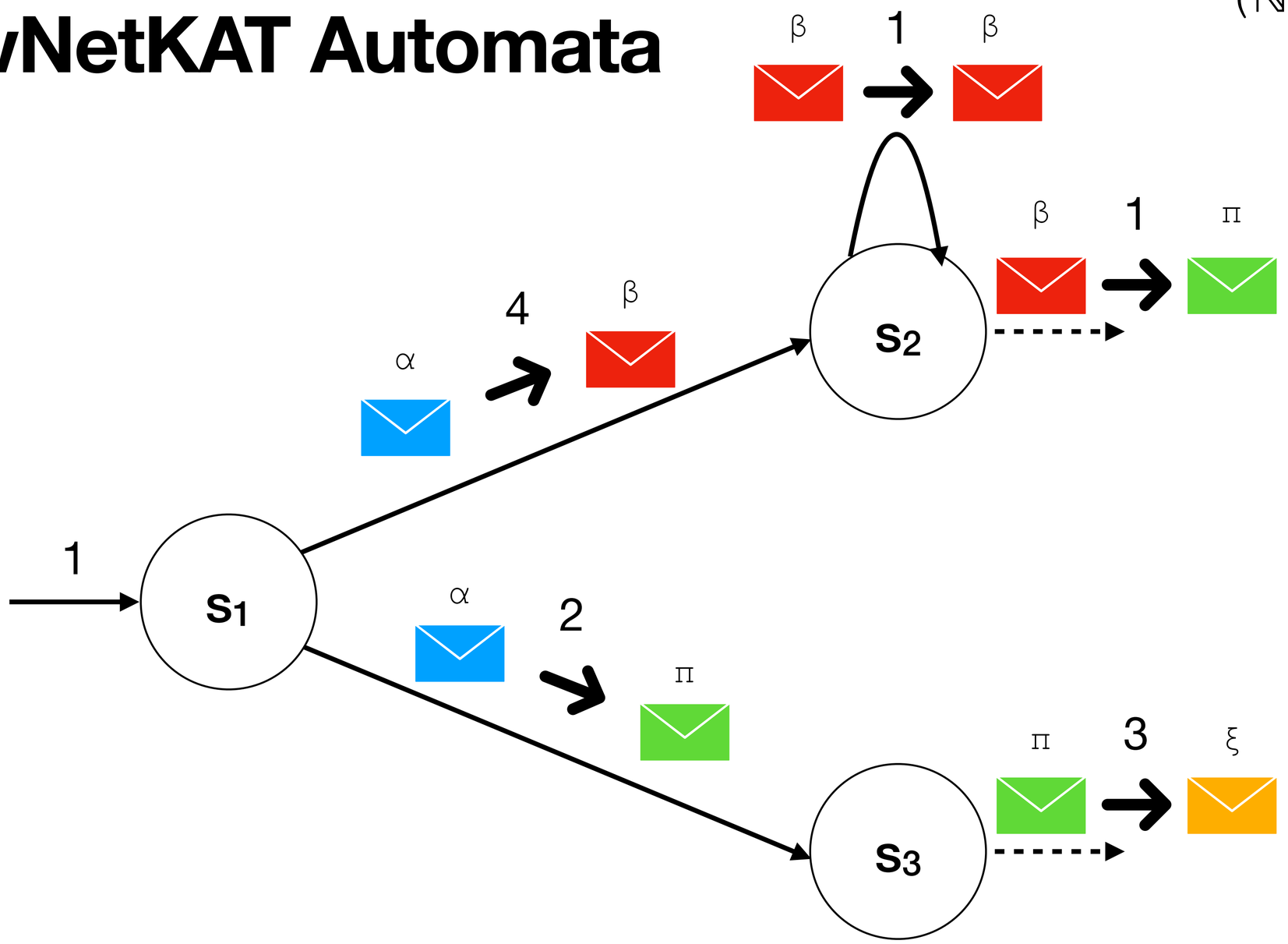
Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$

wNetKAT Automata



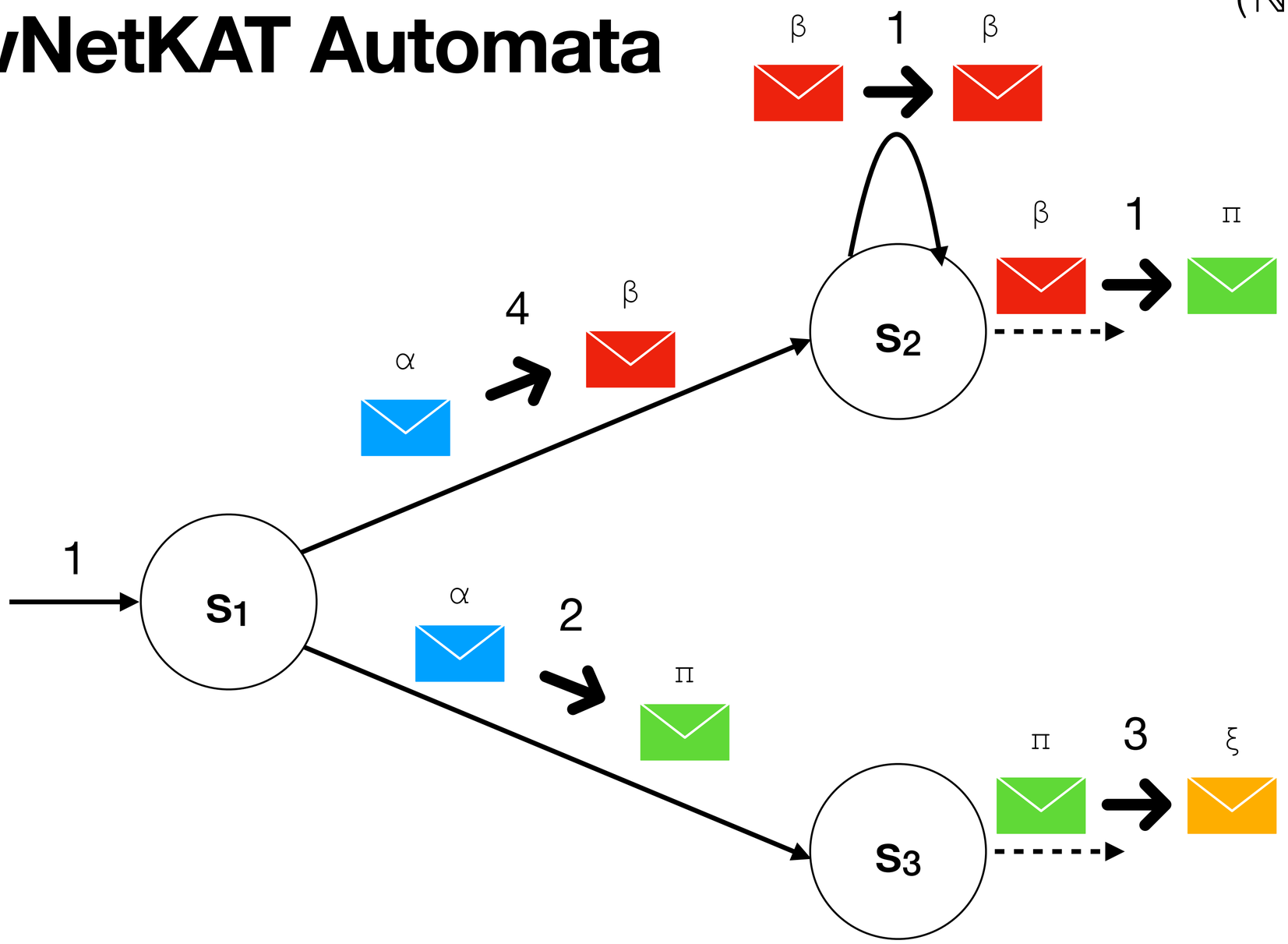
wNetKAT Automata

Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$



wNetKAT Automata

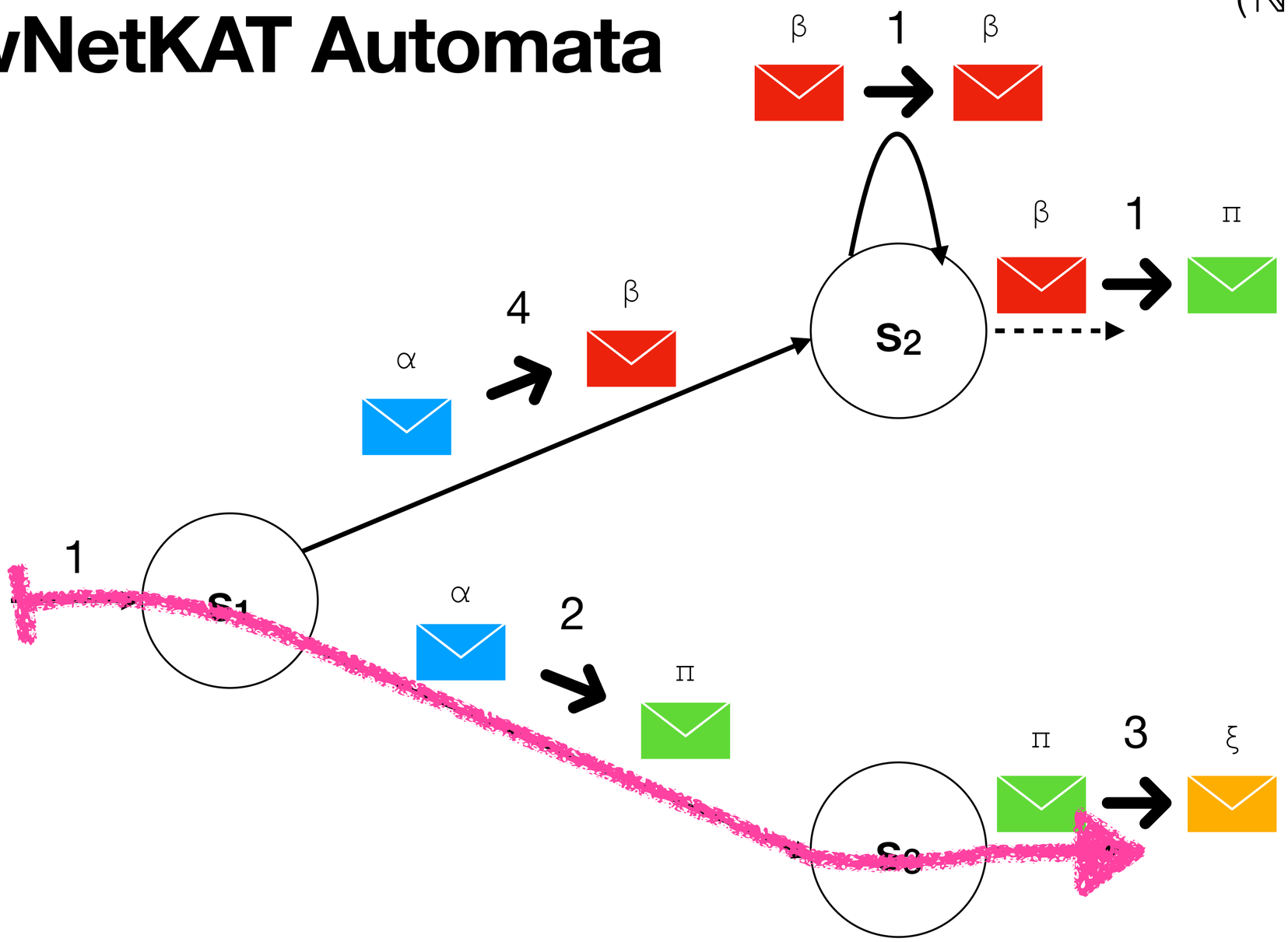
Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$



Language Accepted:

wNetKAT Automata

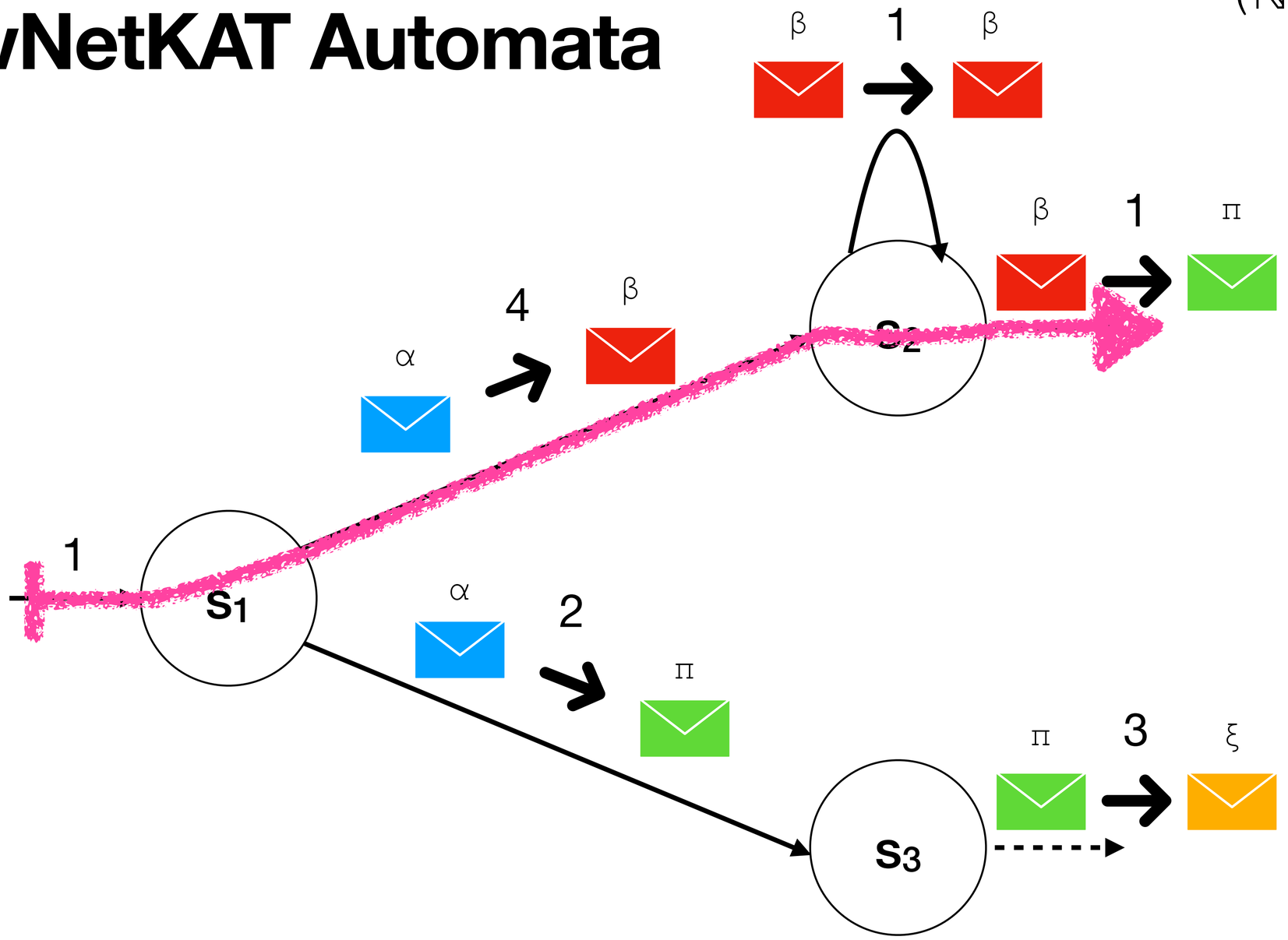
Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$



Language Accepted:

$\alpha\ \pi\ \xi \mapsto 6$

wNetKAT Automata



Tropical Semiring

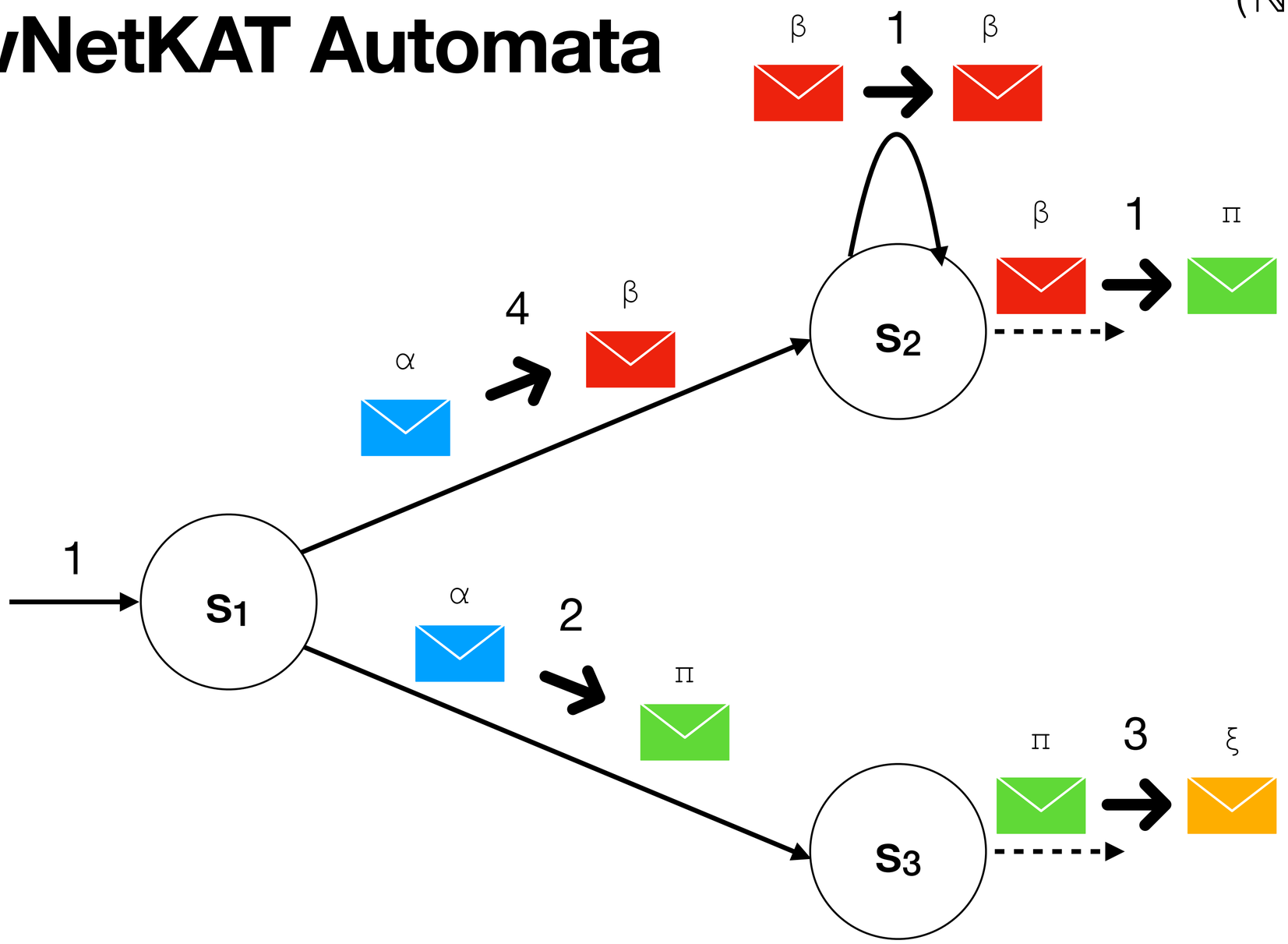
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$

Language Accepted:

$\alpha\ \pi\ \xi \mapsto 6$

$\alpha\ \beta\ \pi \mapsto 6$

wNetKAT Automata



Tropical Semiring

$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$

Language Accepted:

$\alpha\ \pi\ \xi \mapsto 6$

$\alpha\ \beta\ \pi \mapsto 6$

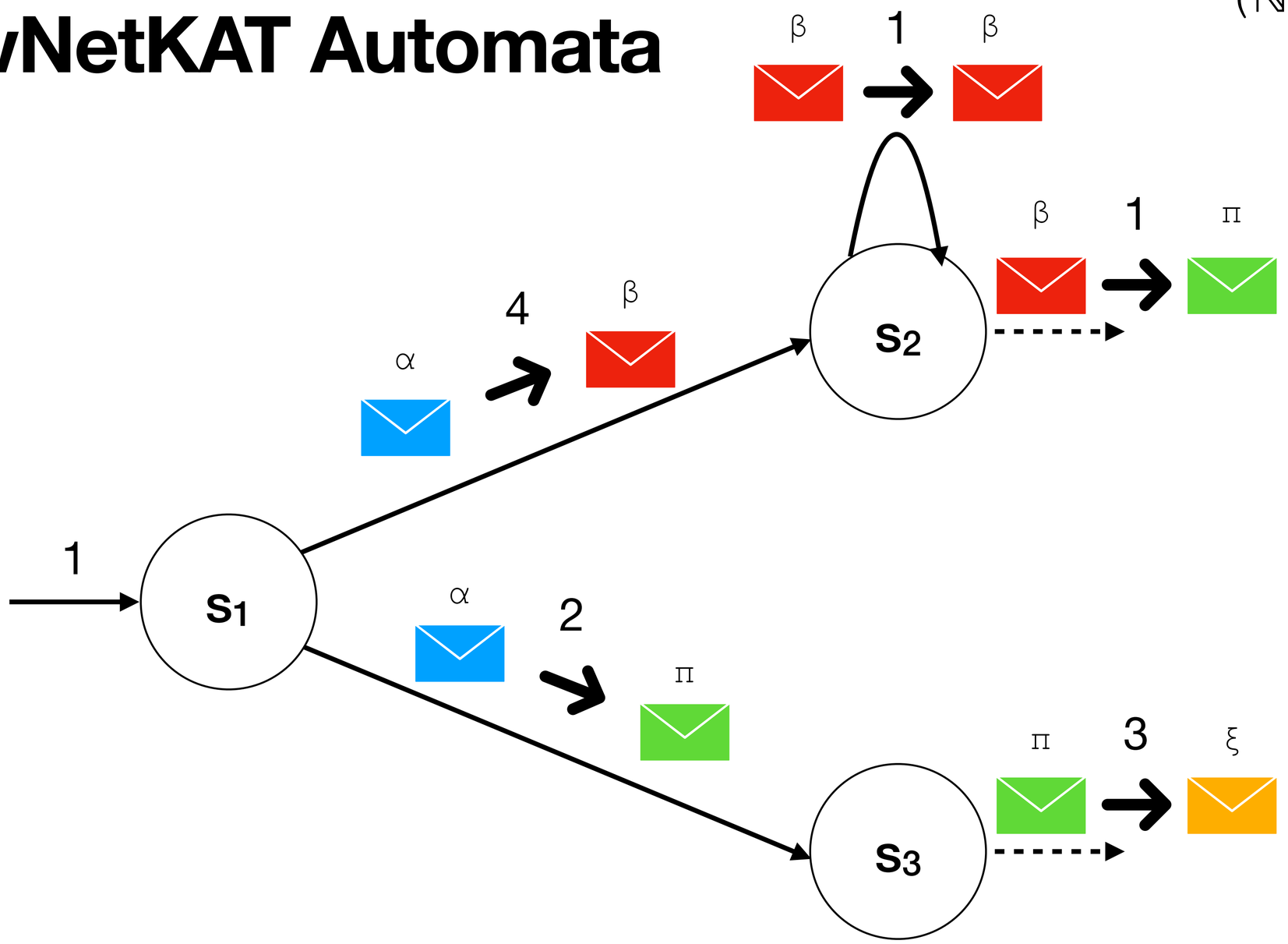
$\alpha\ \beta\ \beta\ \pi \mapsto 7$

$\alpha\ \beta\ \beta\ \beta\ \pi \mapsto 8$

...

wNetKAT Automata

Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$



Language Accepted:

$\alpha\ \pi\ \xi \mapsto 6$

$\alpha\ \beta\ \pi \mapsto 6$

$\alpha\ \beta\ \beta\ \pi \mapsto 7$

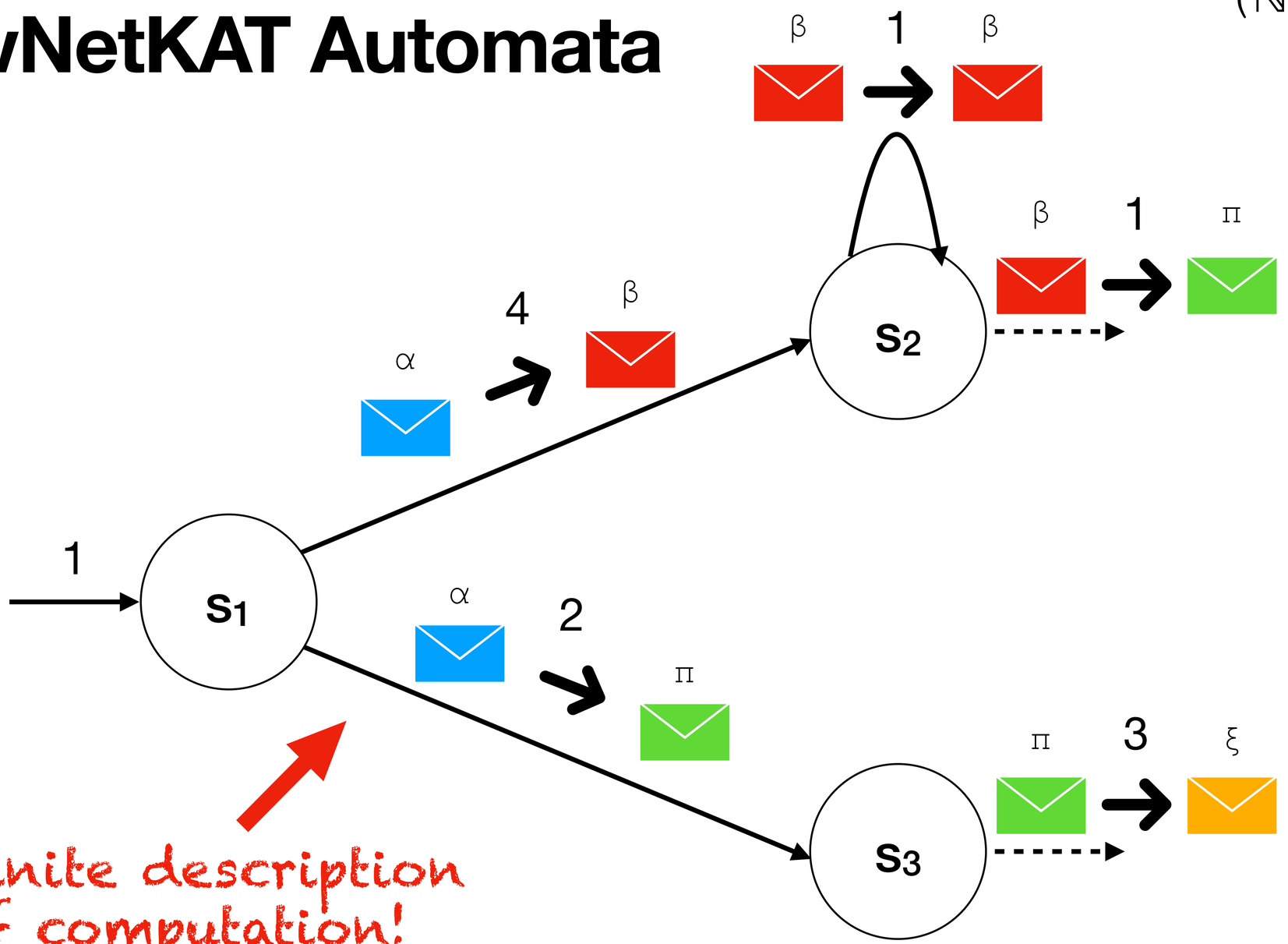
$\alpha\ \beta\ \beta\ \beta\ \pi \mapsto 8$

...

Complete network traces

wNetKAT Automata

Tropical Semiring
$(\mathbb{N}^{+\infty}, \min, +, +\infty, 0)$



Language Accepted:

$\alpha\ \pi\ \xi \mapsto 6$

$\alpha\ \beta\ \pi \mapsto 6$

$\alpha\ \beta\ \beta\ \pi \mapsto 7$

$\alpha\ \beta\ \beta\ \beta\ \pi \mapsto 8$

...

Finite description of computation!

Complete network traces

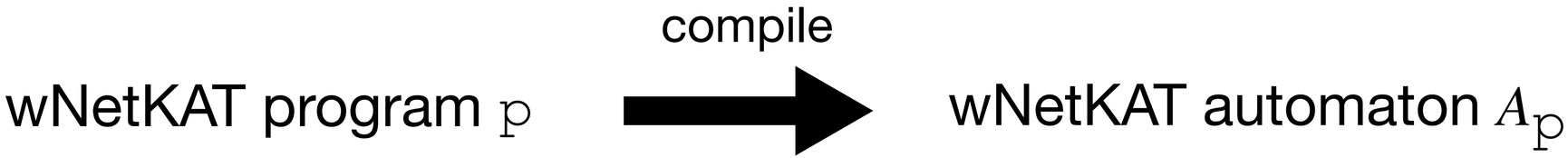
Automatic Verification

wNetKAT program $\mathtt{p}$

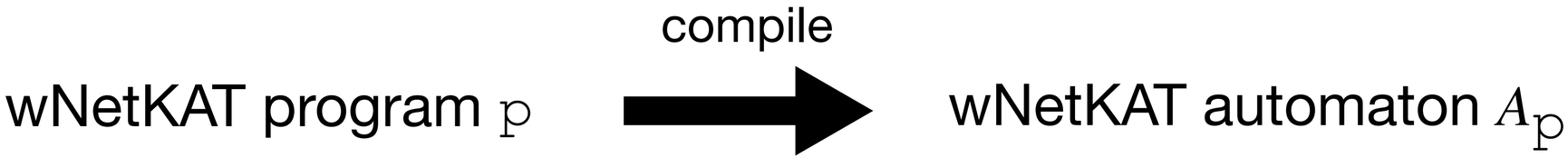
Automatic Verification

wNetKAT program `p` 

Automatic Verification

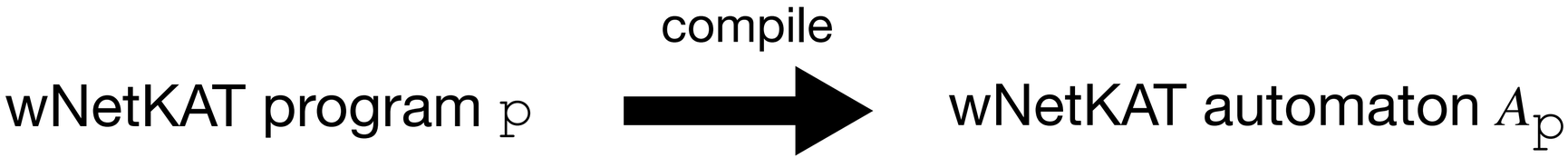


Automatic Verification



We show:

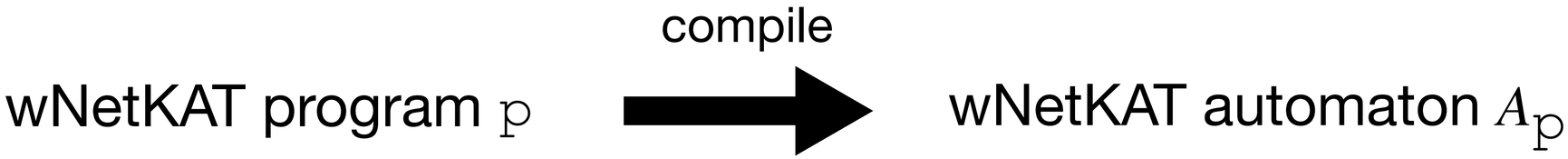
Automatic Verification



We show:

$$[\![\mathrm{p}]\!] \;=\; [\![A_{\mathrm{p}}]\!]$$

Automatic Verification

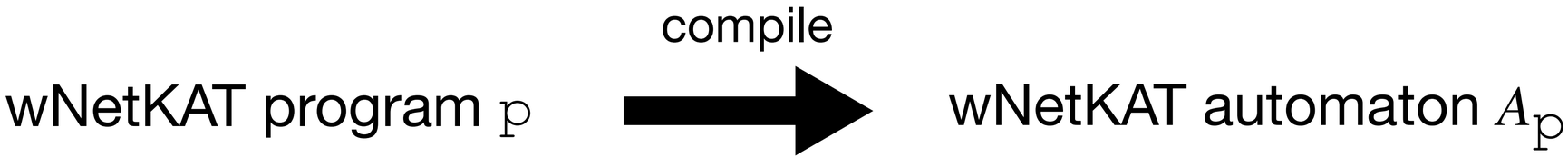


We show:

$$[\![\mathrm{p}]\!] \;=\; [\![A_{\mathrm{p}}]\!]$$

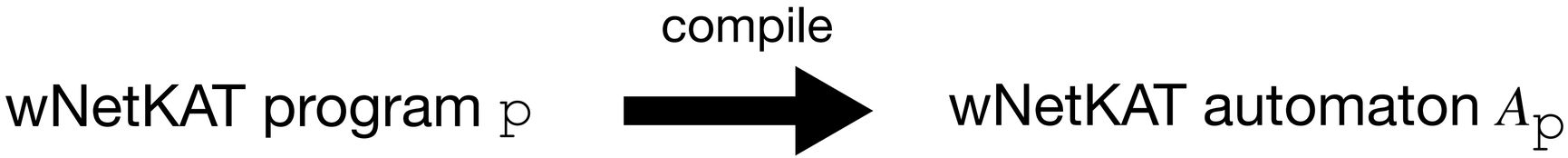
Theorem:
Semantics of any wNetKAT program can be computed by compiling to wNetKAT automata.

Automatic Verification



Theorem:
Semantics of any wNetKAT program can be computed by compiling to wNetKAT automata.

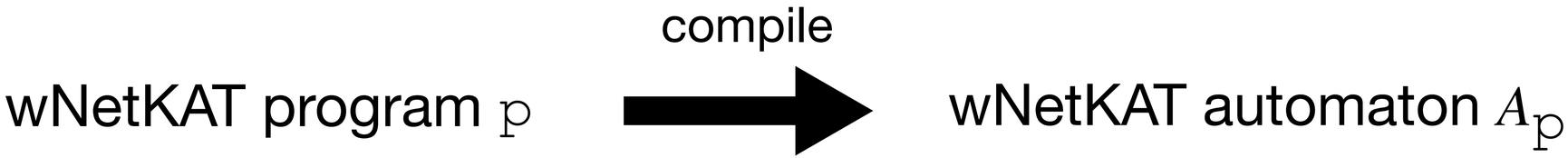
Automatic Verification



Theorem:
Semantics of any wNetKAT program can be computed by compiling to wNetKAT automata.

⇒ Automatic verification of r-safety / r-reachability can be done at the level of wNetKAT automata

Automatic Verification



Theorem:
Semantics of any wNetKAT program can be computed by compiling to wNetKAT automata.

⇒ Automatic verification of $\mathtt{r}$-safety / $\mathtt{r}$-reachability can be done at the level of wNetKAT automata

Several properties of automata generally become undecidable in the presence of weights!

Automatic Verification

We develop decision procedures over wNetKAT automata to establish decidability of:

Automatic Verification

We develop decision procedures over wNetKAT automata to establish decidability of:

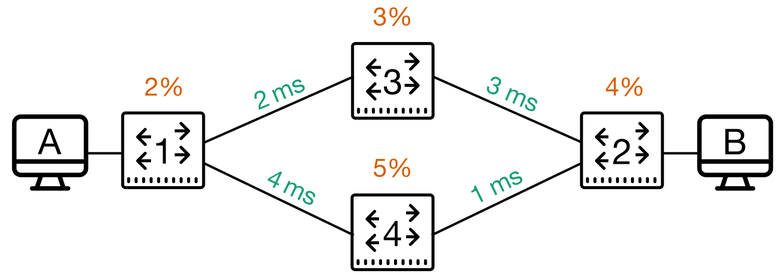
`r`**-safety**
(Does all network traffic have weight *at most* `r`)?

$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

`r`**-reachability**
(Does there exist network traffic with weight *at least* `r`)?

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification



We develop decision procedures over wNetKAT automata to establish decidability of:

`r`-safety
(Does all network traffic have weight *at most* `r`)?

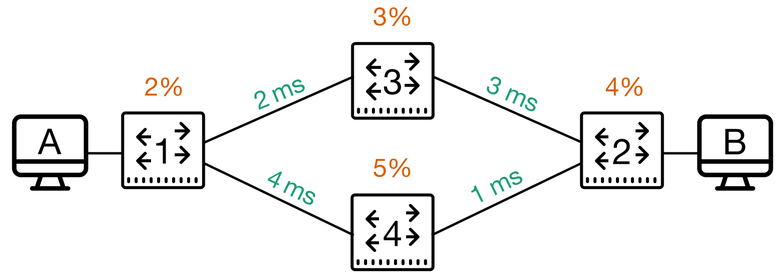
$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

Does all traffic get delivered with at least 90% reliability?

`r`-reachability
(Does there exist network traffic with weight *at least* `r`)?

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification



We develop decision procedures over wNetKAT automata to establish decidability of:

`r`-safety
(Does all network traffic have weight *at most* `r`)?

$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

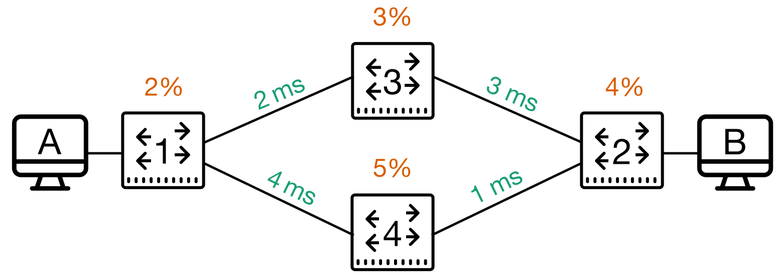
Does all traffic get delivered with at least 90% reliability?

If no, provide a trace violating property

`r`-reachability
(Does there exist network traffic with weight *at least* `r`)?

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Automatic Verification



We develop decision procedures over wNetKAT automata to establish decidability of:

`r`**-safety**
(Does all network traffic have weight *at most* `r`)?

$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

Does all traffic get delivered with at least 90% reliability?

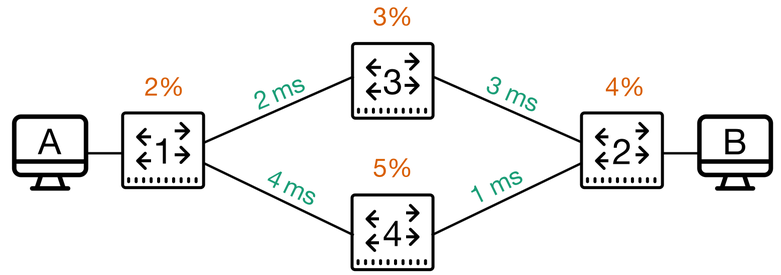
If no, provide a trace violating property

`r`**-reachability**
(Does there exist network traffic with weight *at least* `r`)?

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Can host A deliver packets to host B within 5ms?

Automatic Verification



We develop decision procedures over wNetKAT automata to establish decidability of:

`r`-safety
(Does all network traffic have weight *at most* `r`)?

$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

Does all traffic get delivered with at least 90% reliability?

If no, provide a trace violating property

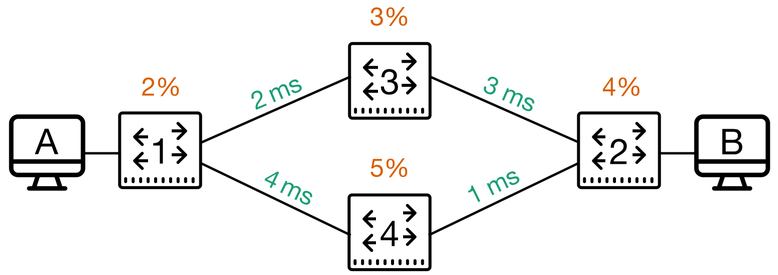
`r`-reachability
(Does there exist network traffic with weight *at least* `r`)?

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Can host A deliver packets to host B within 5ms?

If yes, provide a trace satisfying property

Automatic Verification



Generic over a broad class of semirings

We develop decision procedures over wNetKAT automata to establish decidability of:

`r`**-safety**
(Does all network traffic have weight *at most* `r`)?

$$\forall \pi, h: \ [\![p]\!](\pi)(h) \leq r$$

Does all traffic get delivered with at least 90% reliability?

If no, provide a trace violating property

`r`**-reachability**
(Does there exist network traffic with weight *at least* `r`)?

$$\exists \pi, h: \ [\![p]\!](\pi)(h) \geq r$$

Can host A deliver packets to host B within 5ms?

If yes, provide a trace satisfying property

Thank You!

See Paper For:

- Semantics of wNetKAT
- Compilation to automata
- Decision procedures
- Case study

Current/Future Work:

- Lean mechanization
- Symbolic representations of automata
- Efficient implementations of procedures