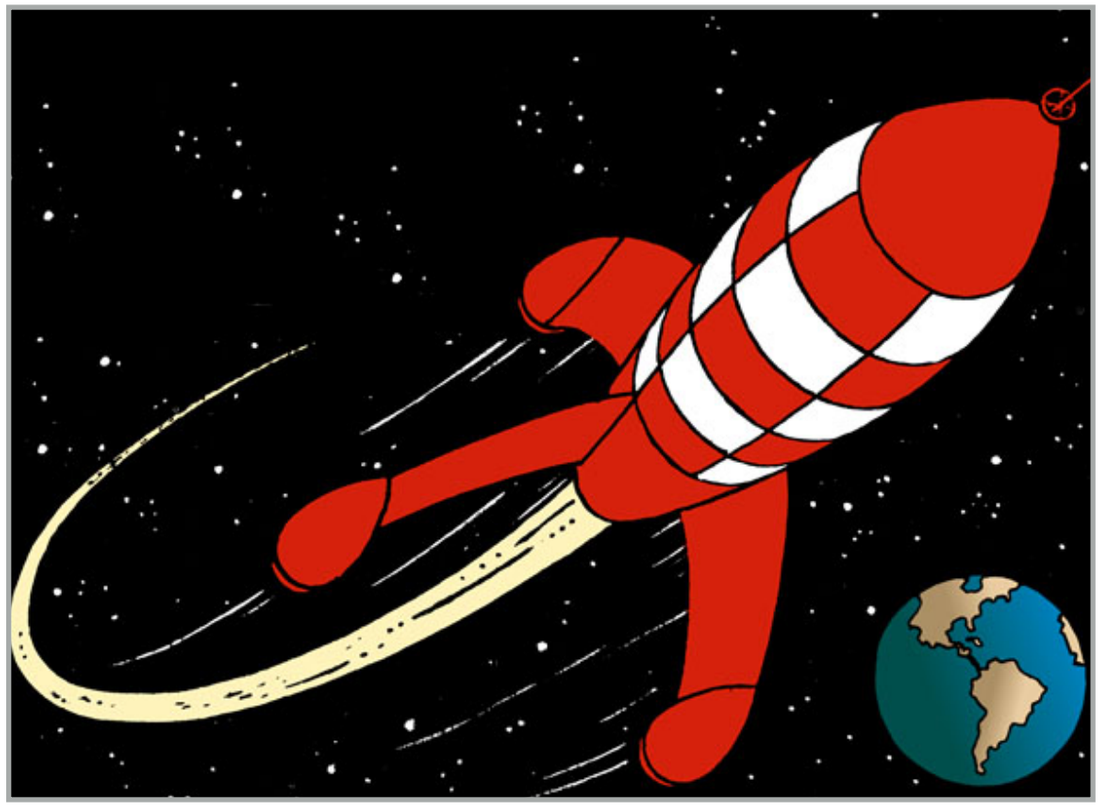
***SWIFT**: Predictive Fast Reroute*



Thomas Holterbach
ETH Zürich / CAIDA

SIGCOMM
24th August 2017

swift.ethz.ch

Joint work with

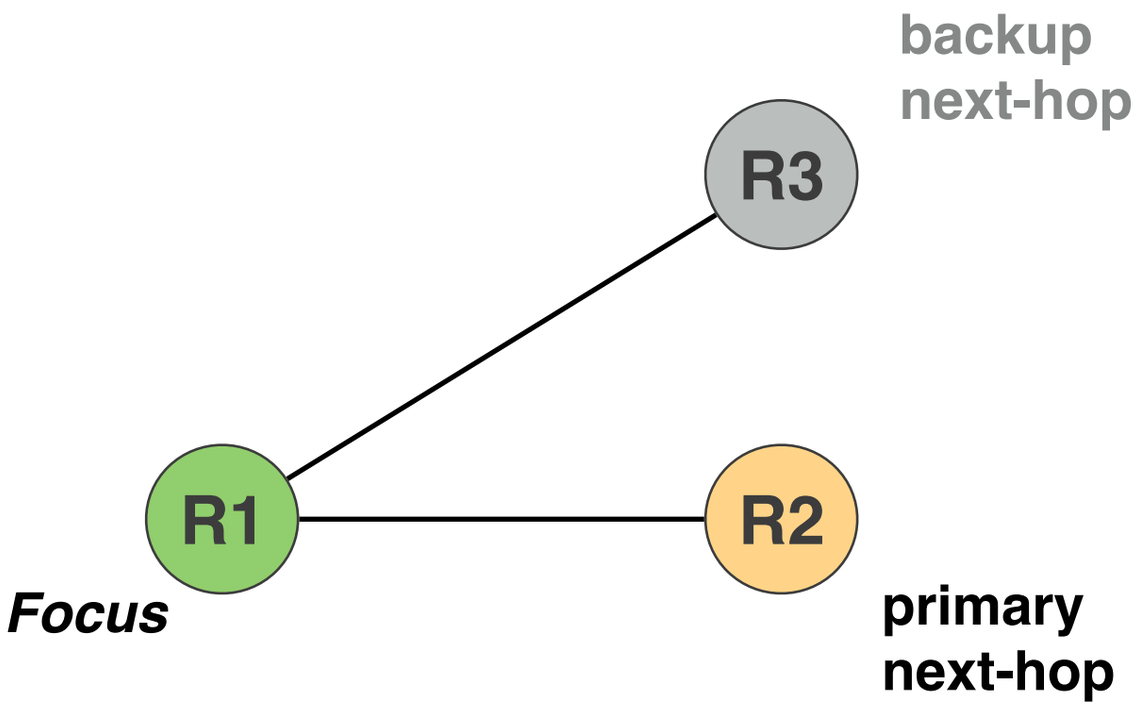
Stefano Vissicchio UCLondon
Alberto Dainotti CAIDA, UC San Diego
Laurent Vanbever ETH Zürich

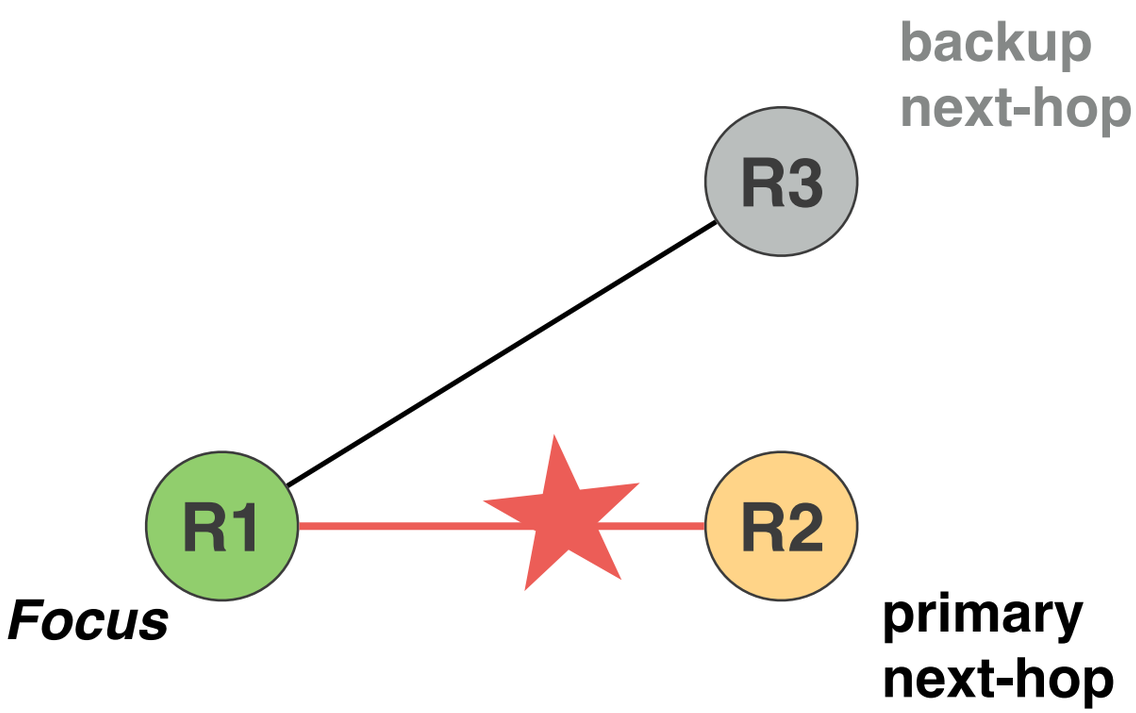
2.5 min

2.5 min

worst-case convergence time of a router upon an Internet failure

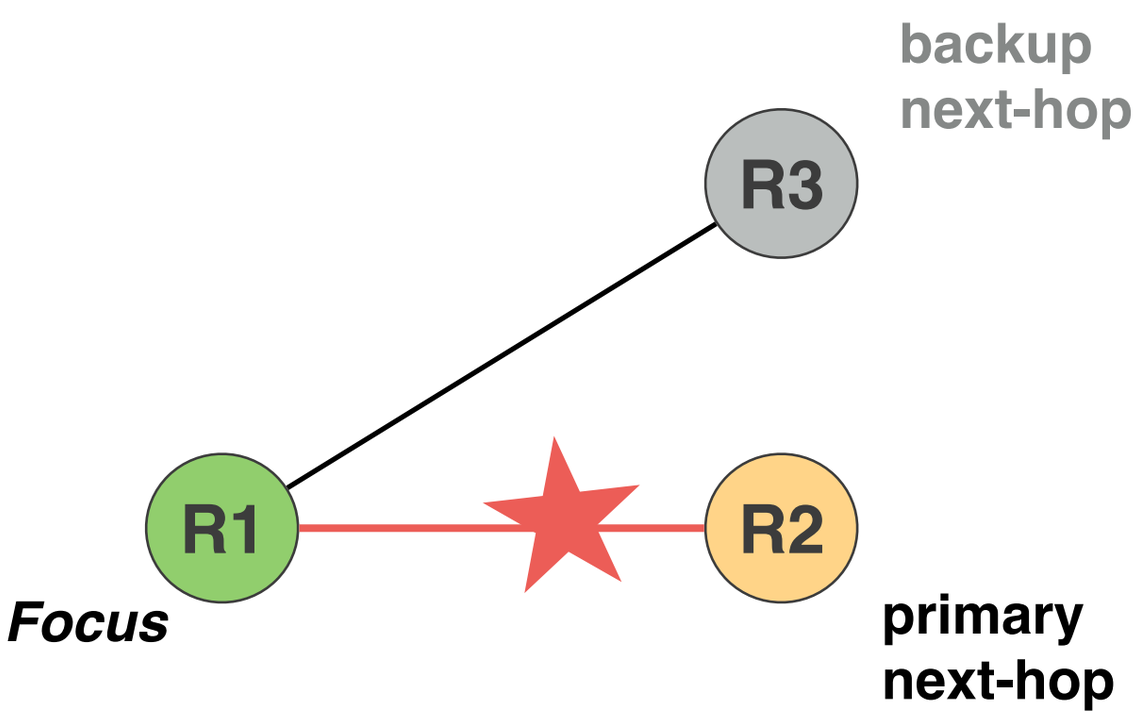
Let’s consider first a simple example




backup
next-hop
R3
R1
R2
Focus
primary
next-hop

BGP convergence

is a two-phase process



Phase #1

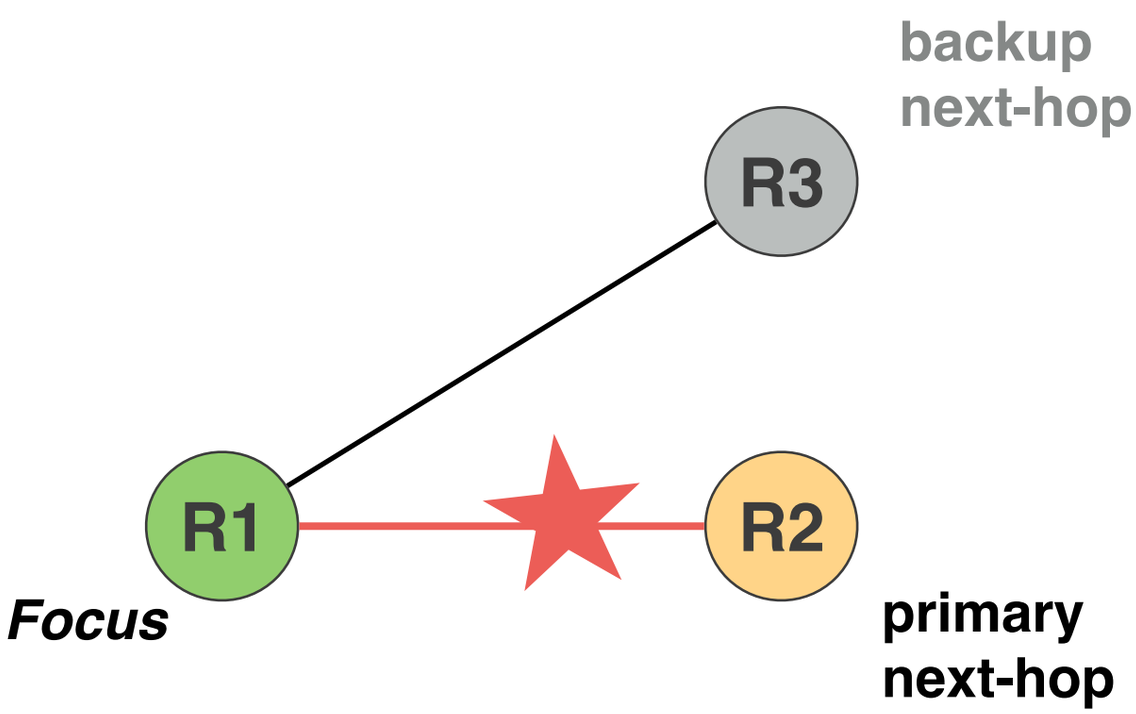
Learning
about the failure

Phase #2

Updating
forwarding entries

BGP convergence

is a two-phase process



R1's forwarding table

pos.	prefix	NH
1	1.0.0.0/24	2
2	1.0.1.0/24	2
3	1.0.2.0/24	2
…	…	…
300K	100.0.0.0/24	2
…	…	…
600K	200.0.0.0/24	2

Phase #1

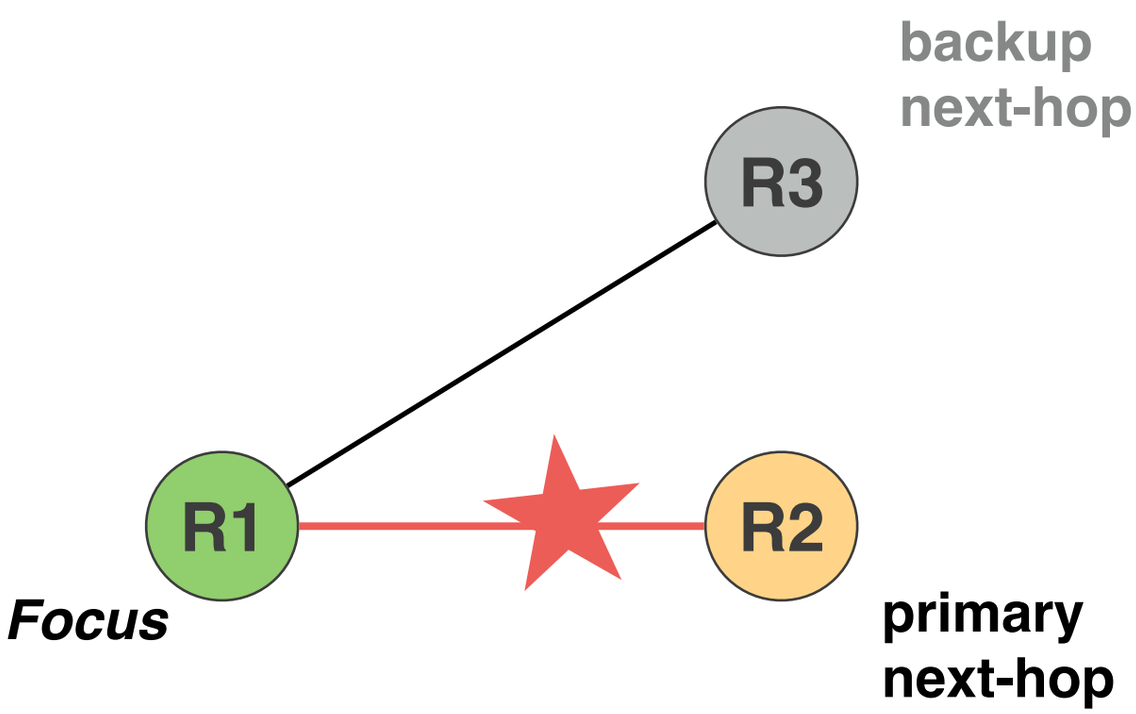
Learning
about the failure

Phase #2

Updating
forwarding entries

BGP convergence

is a two-phase process



R1's forwarding table

pos.	prefix	NH
1	1.0.0.0/24	**3**
2	1.0.1.0/24	2
3	1.0.2.0/24	2
…	…	…
300K	100.0.0.0/24	2
…	…	…
600K	200.0.0.0/24	2

Phase #1

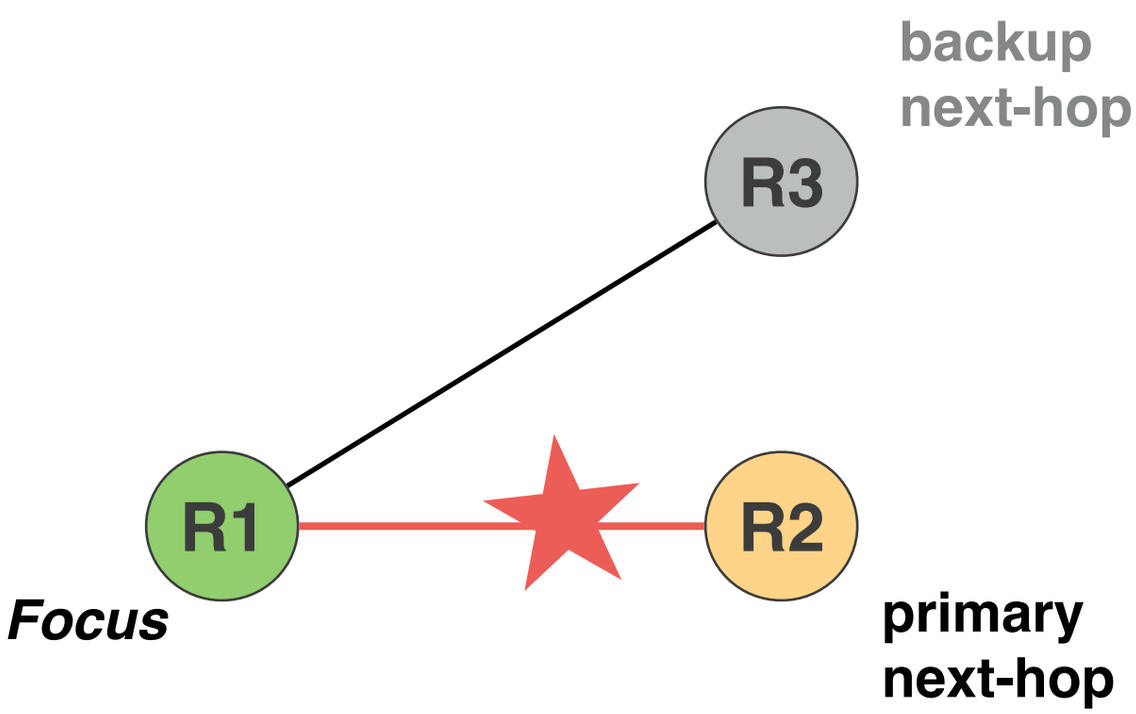
Learning
about the failure

Phase #2

Updating
forwarding entries

BGP convergence

is a two-phase process



R1's forwarding table

pos.	prefix	NH
1	1.0.0.0/24	**3**
2	1.0.1.0/24	**3**
3	1.0.2.0/24	2
…	…	…
300K	100.0.0.0/24	2
…	…	…
600K	200.0.0.0/24	2

Phase #1

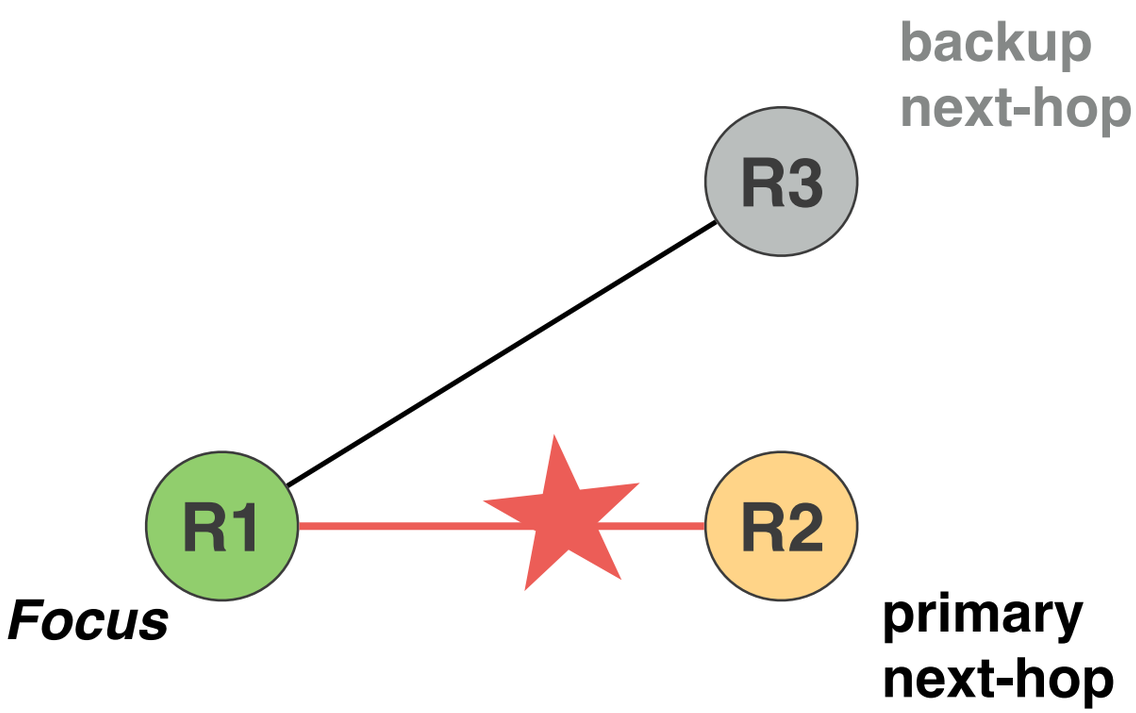
Learning
about the failure

Phase #2

Updating
forwarding entries

BGP convergence

is a two-phase process



R1's forwarding table

pos.	prefix	NH
1	1.0.0.0/24	**3**
2	1.0.1.0/24	**3**
3	1.0.2.0/24	**3**
…	…	…
300K	100.0.0.0/24	2
…	…	…
600K	200.0.0.0/24	2

Phase #1

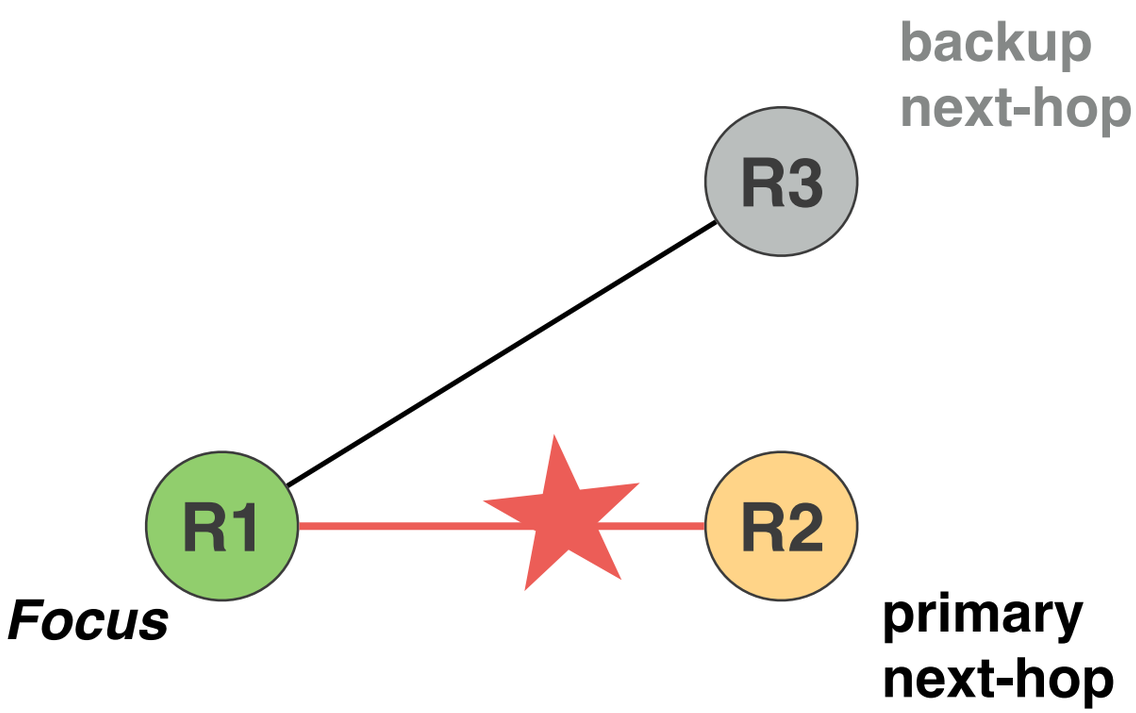
Learning
about the failure

Phase #2

Updating
forwarding entries

BGP convergence

is a two-phase process



R1's forwarding table

pos.	prefix	NH
1	1.0.0.0/24	**3**
2	1.0.1.0/24	**3**
3	1.0.2.0/24	**3**
…	…	…
300K	100.0.0.0/24	**3**
…	…	…
600K	200.0.0.0/24	2

Phase #1

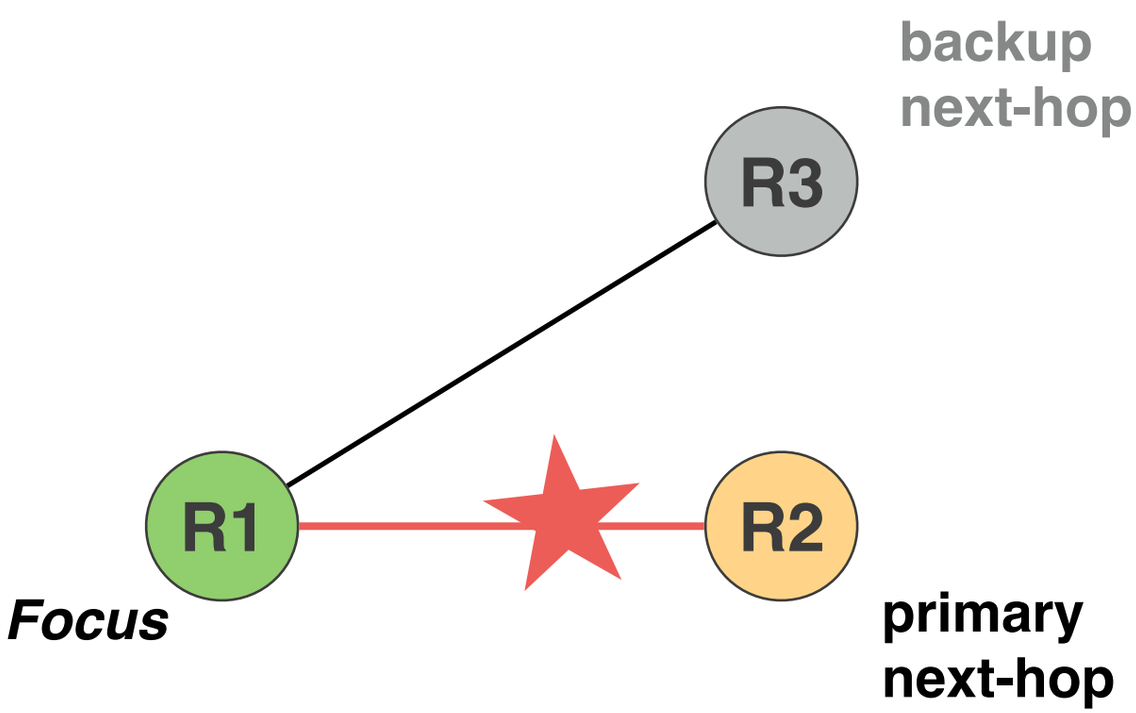
Learning
about the failure

Phase #2

Updating
forwarding entries

BGP convergence

is a two-phase process



R1's forwarding table

pos.	prefix	NH
1	1.0.0.0/24	**3**
2	1.0.1.0/24	**3**
3	1.0.2.0/24	**3**
…	…	…
300K	100.0.0.0/24	**3**
…	…	…
600K	200.0.0.0/24	**3**

Phase #1

Learning
about the failure

Phase #2

Updating
forwarding entries


Google
BGP Fast Reroute
About 3,710 results (0.08 sec)

In practice, multiple techniques are often used to speed up the convergence time

BGP timers

BFD, …

PIC

MPLS, …

Phase #1

Learning
about the failure

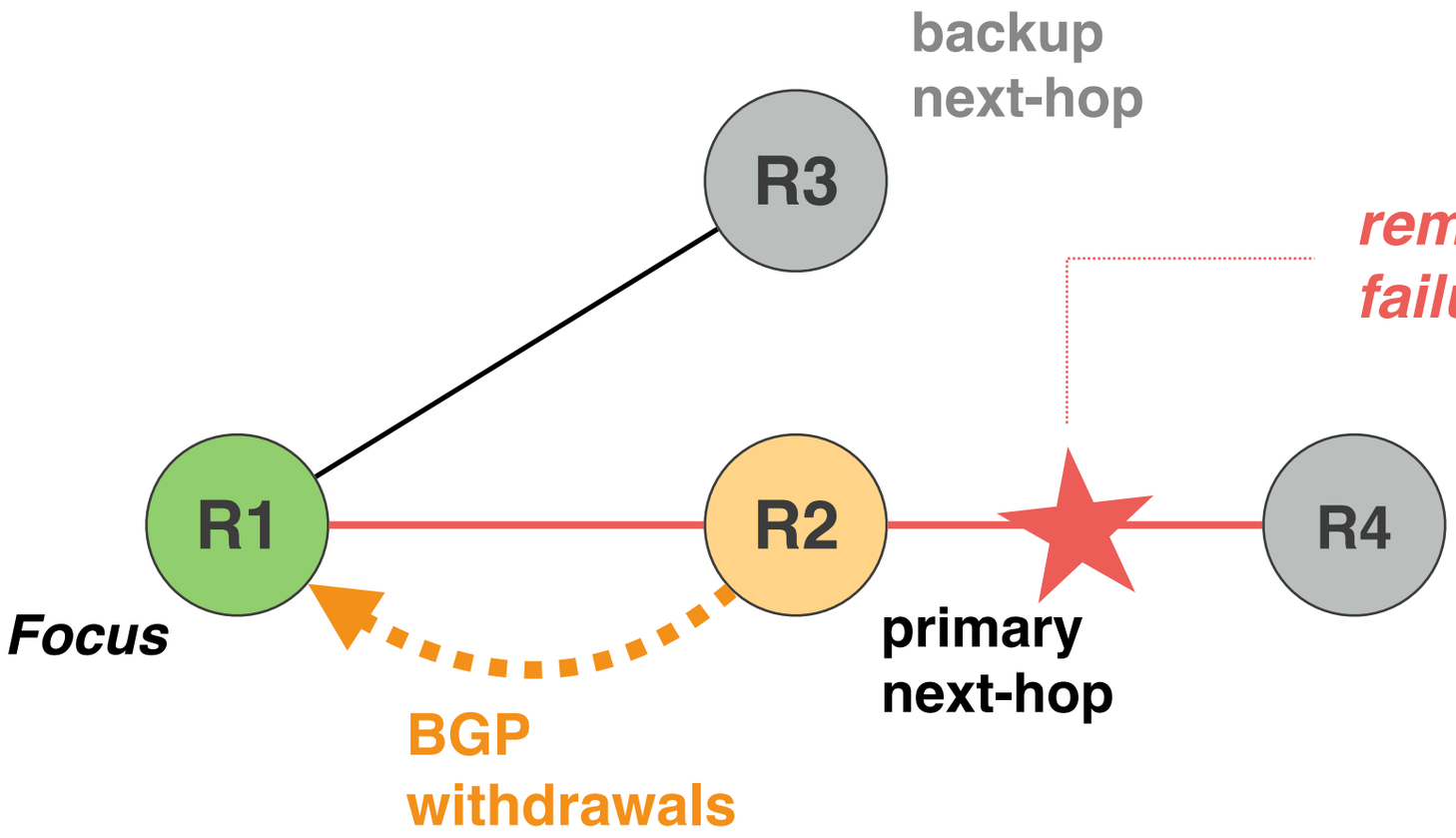
Phase #2

Updating
forwarding entries

About 3,710 results (**0.08** sec)

none of them work upon remote outages

Current fast reroute techniques do not work upon remote outages



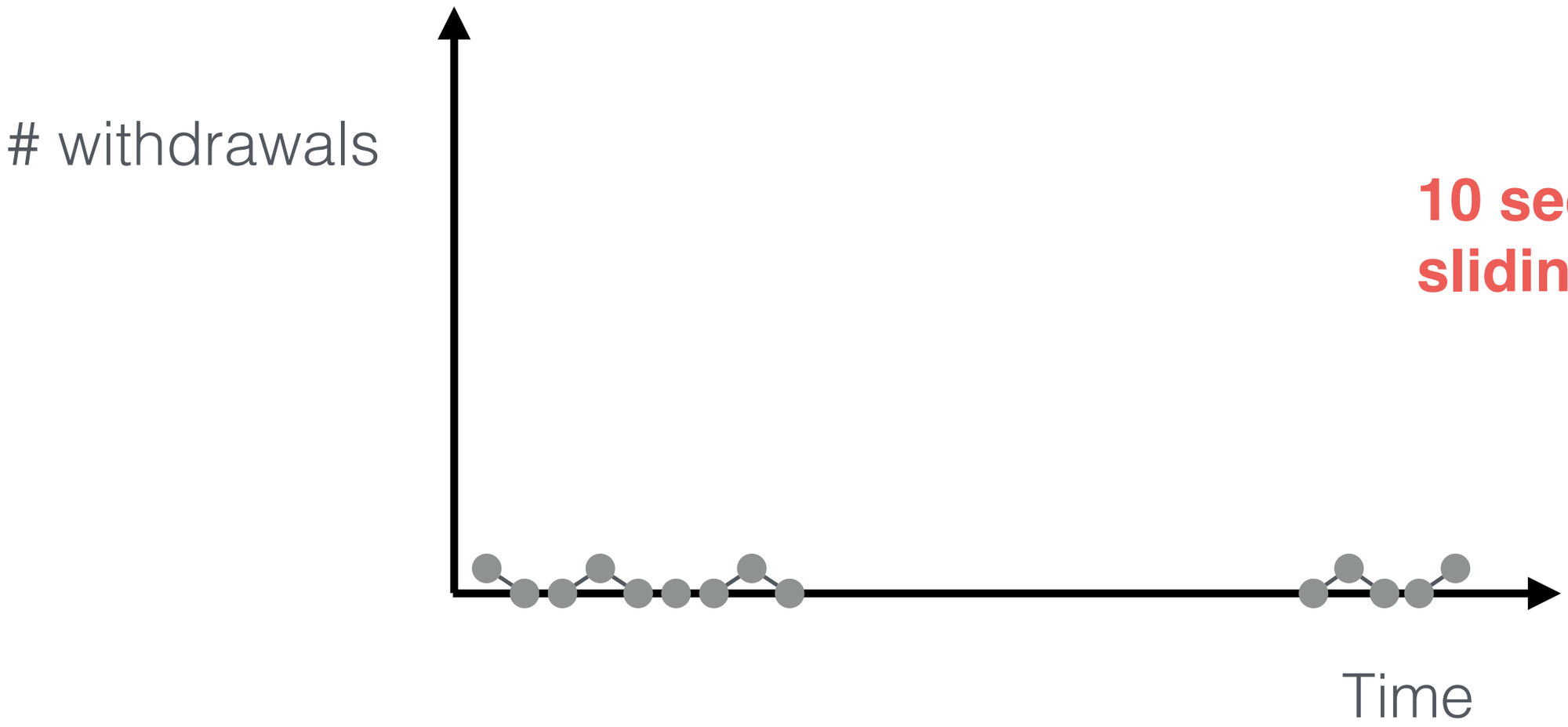
Remote outages are numerous

Remote outages are numerous

We analyzed BGP sessions from RouteViews and RIPE RIS collectors, and looked for **bursts of BGP withdrawals**

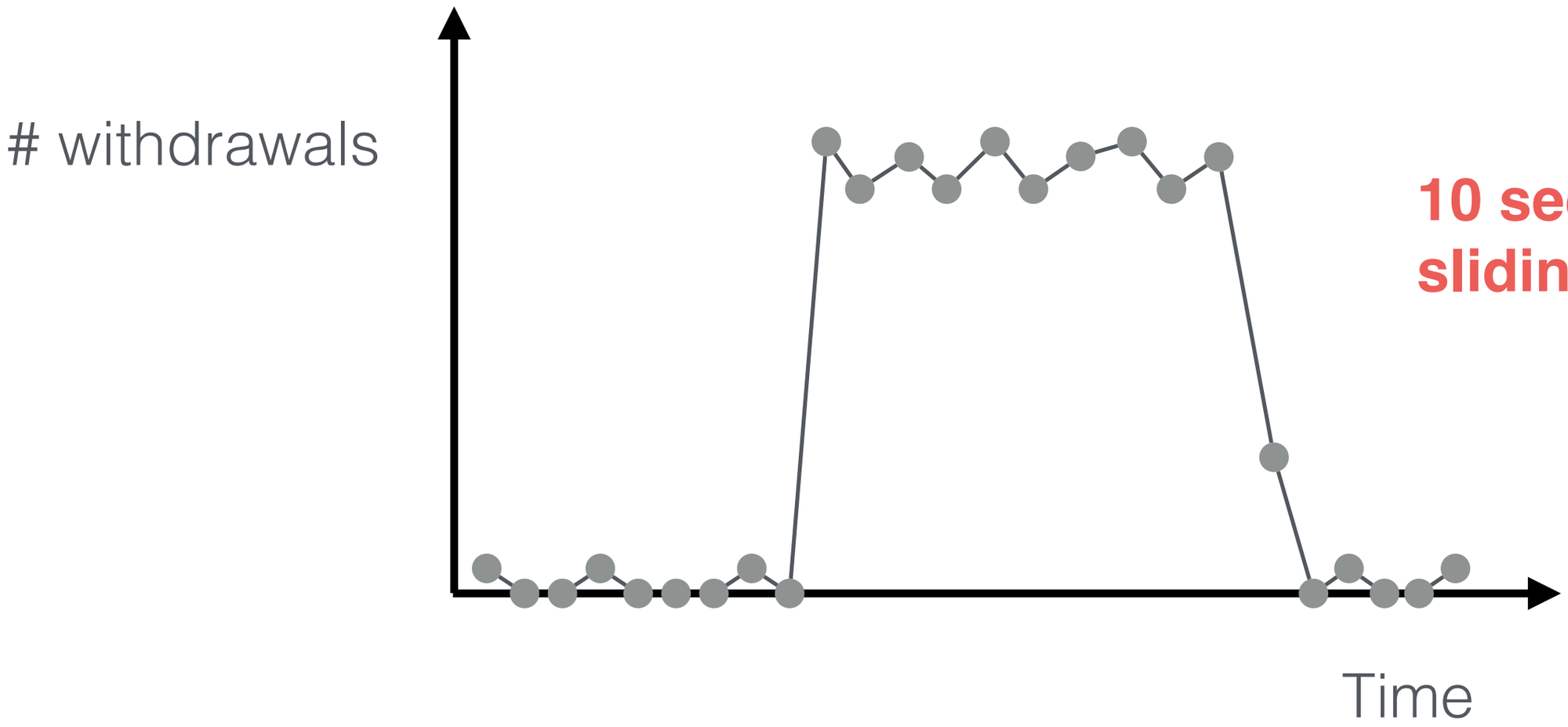
Remote outages are numerous

We analyzed BGP sessions from RouteViews and RIPE RIS collectors, and looked for **bursts of BGP withdrawals**



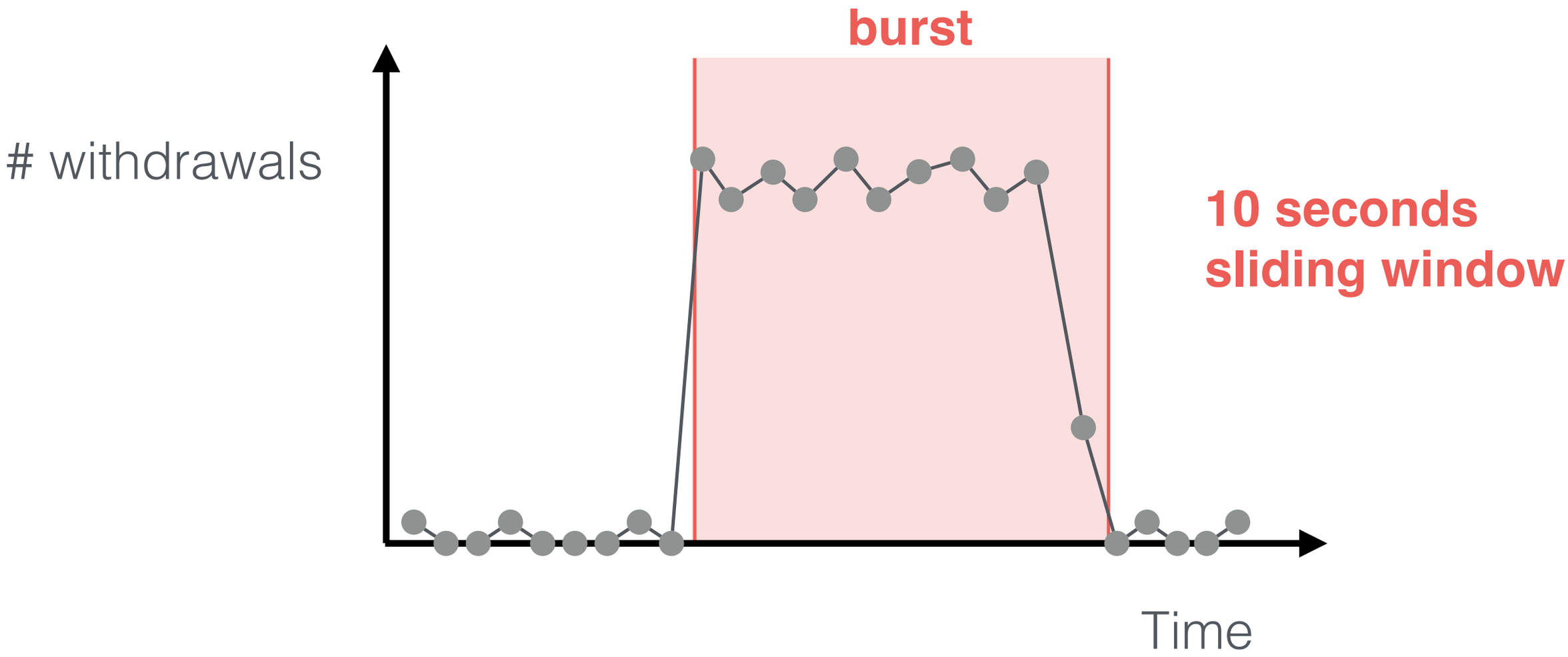
Remote outages are numerous

We analyzed BGP sessions from RouteViews and RIPE RIS collectors, and looked for **bursts of BGP withdrawals**



Remote outages are numerous

We analyzed BGP sessions from RouteViews and RIPE RIS collectors, and looked for **bursts of BGP withdrawals**

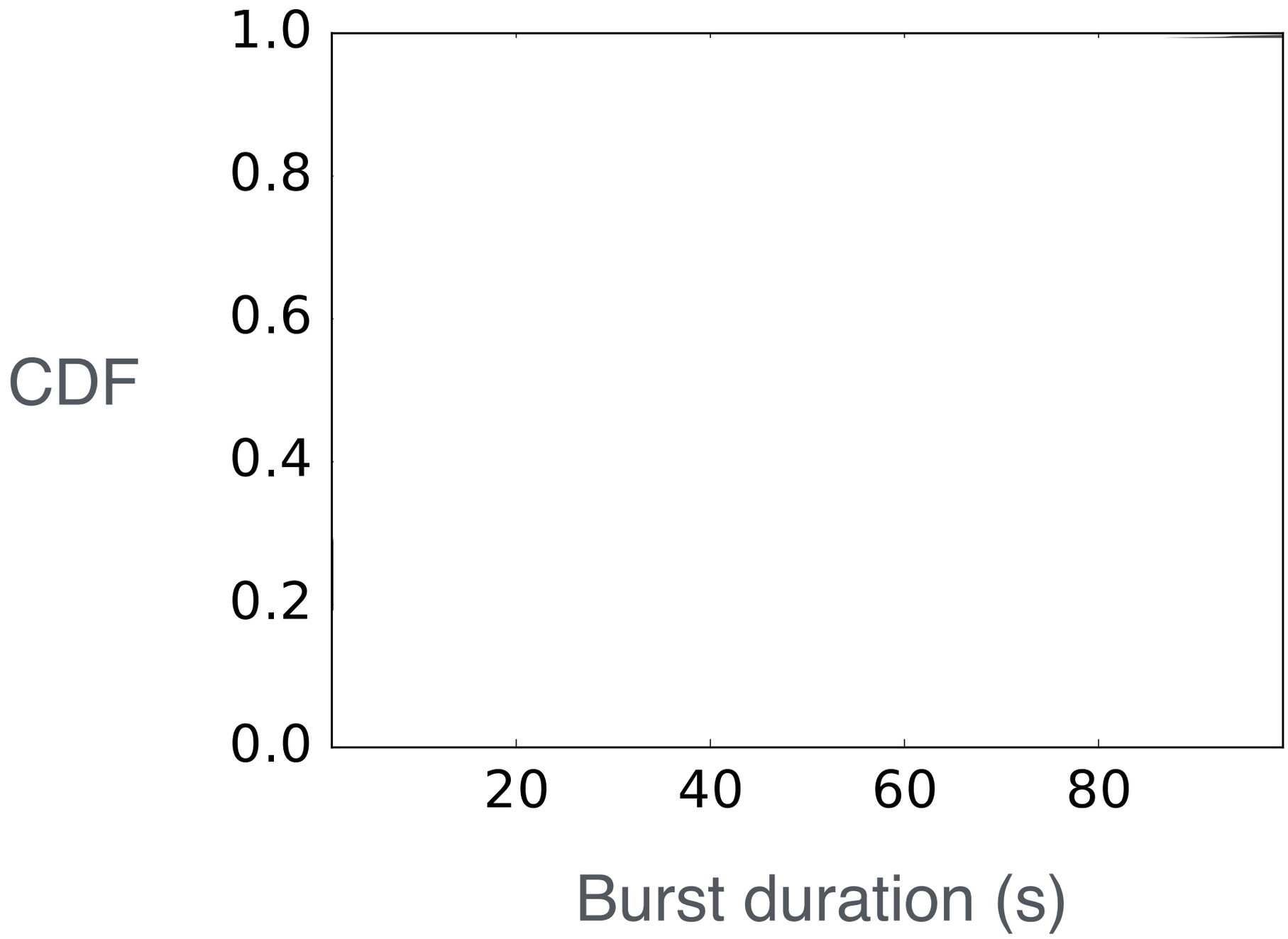


Remote outages are numerous

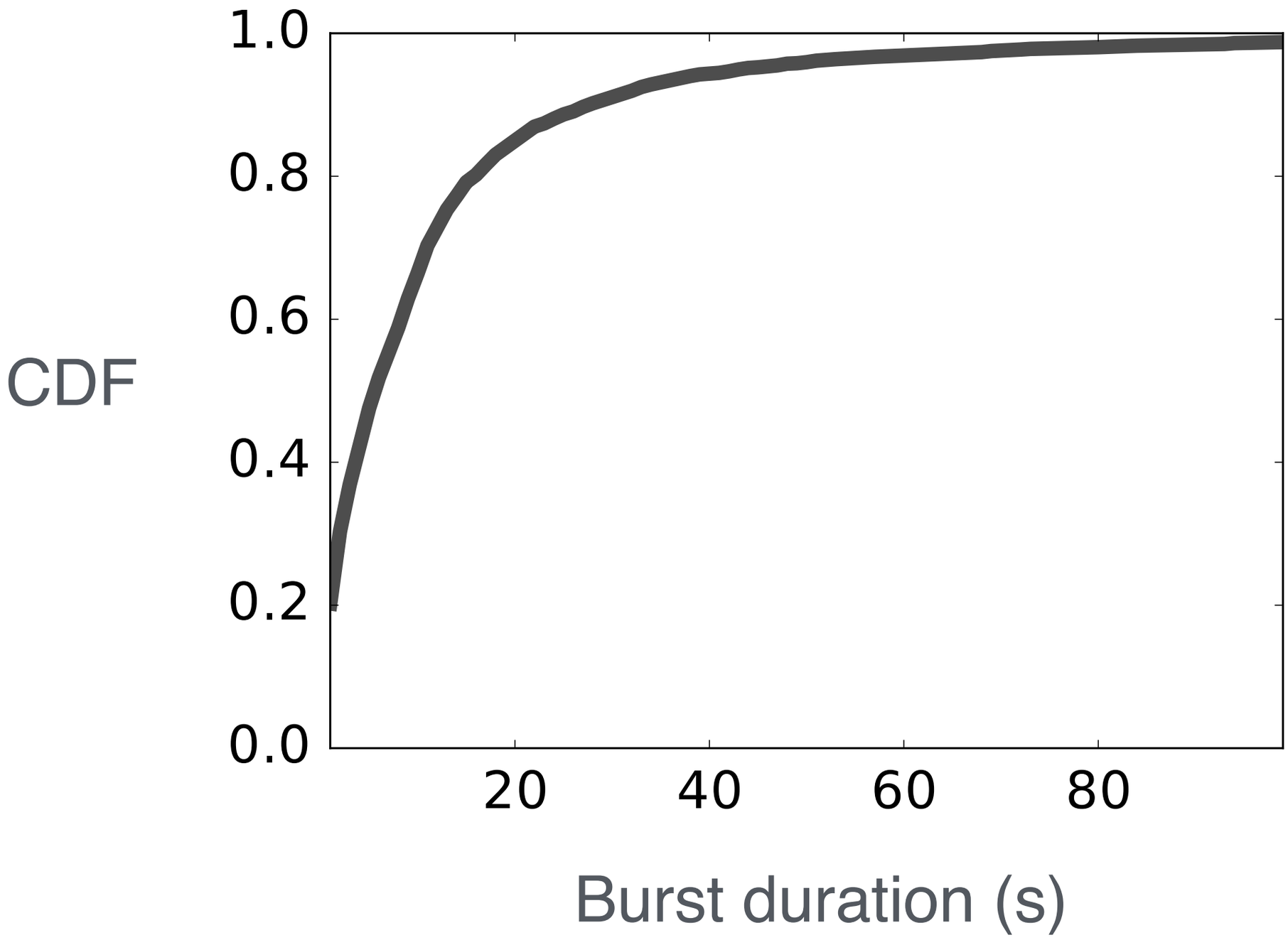
We analyzed BGP sessions from RouteViews and RIPE RIS collectors, and looked for **bursts of BGP withdrawals**

We found **~3,000 bursts** of withdrawals
in November 2016 and on 213 BGP sessions

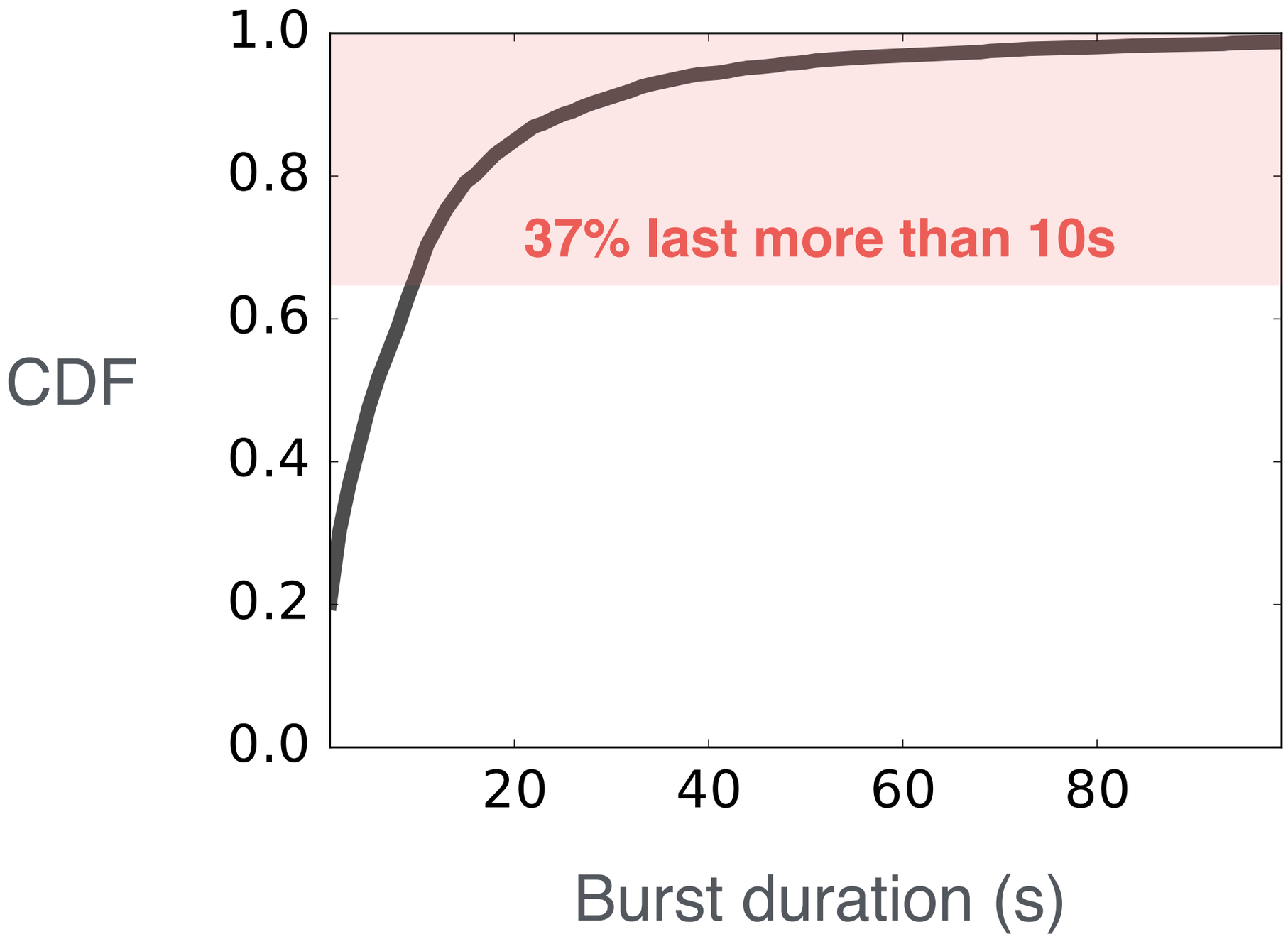
Remote outages propagate slowly



Remote outages propagate slowly

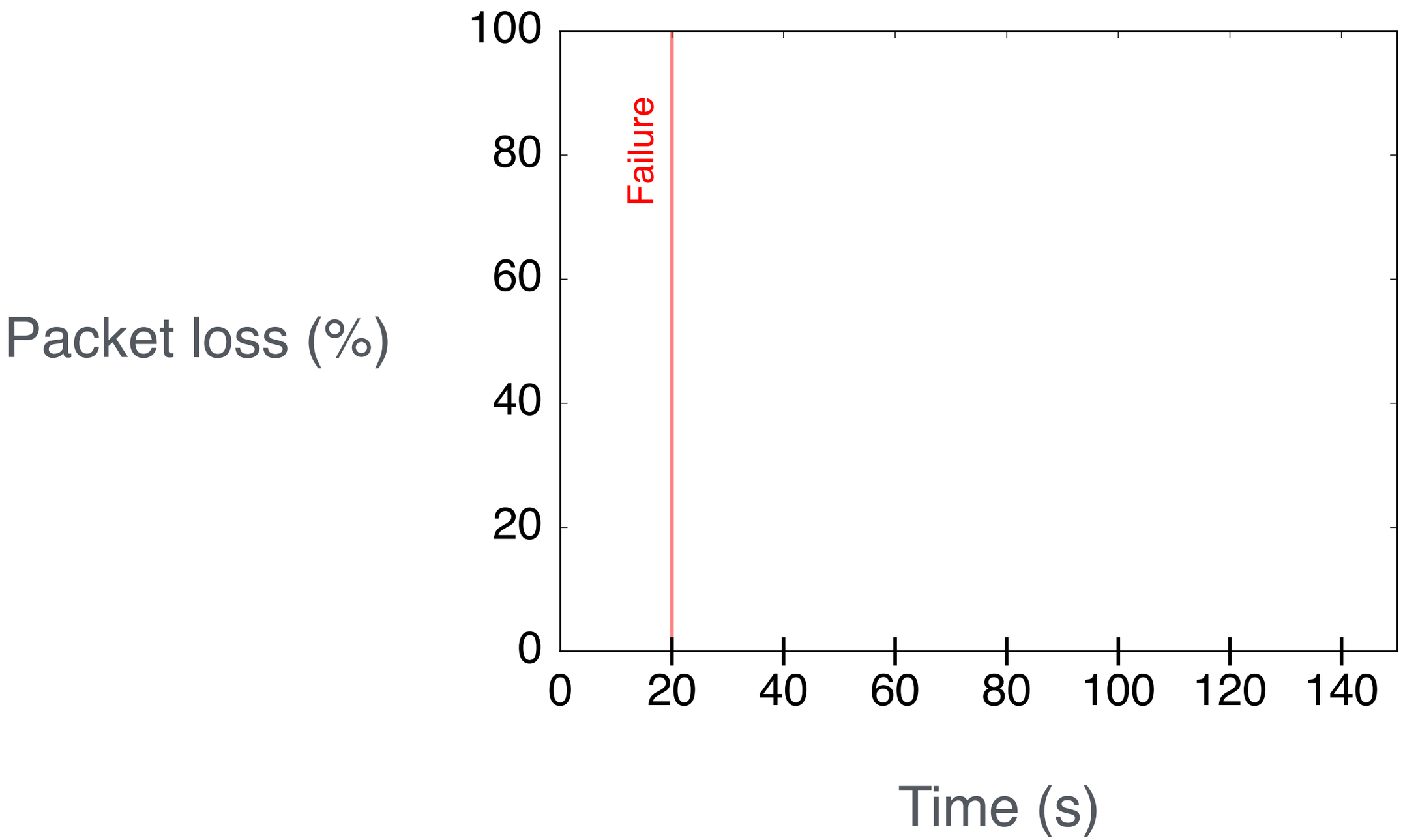


Remote outages propagate slowly

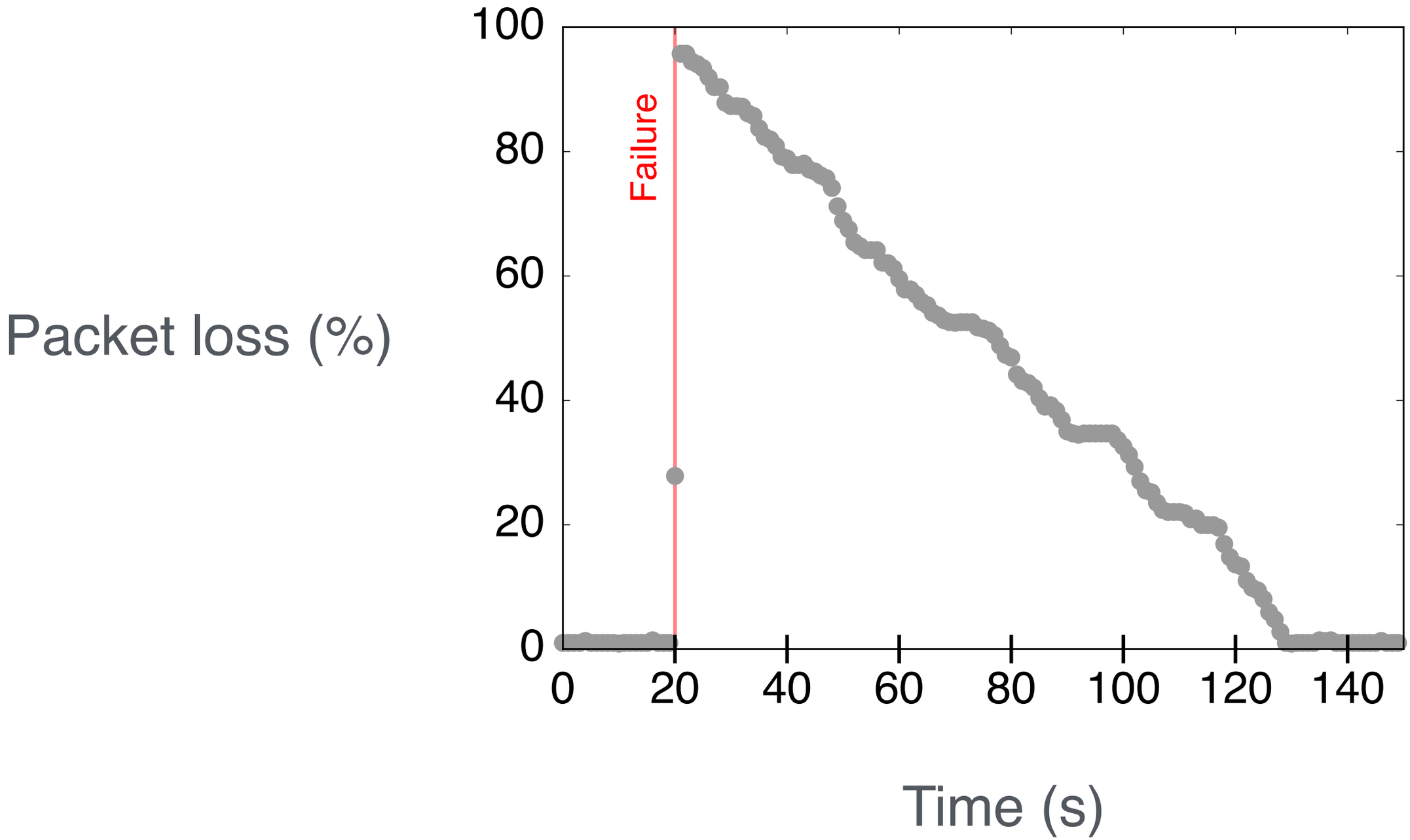


Remote outages result in downtime

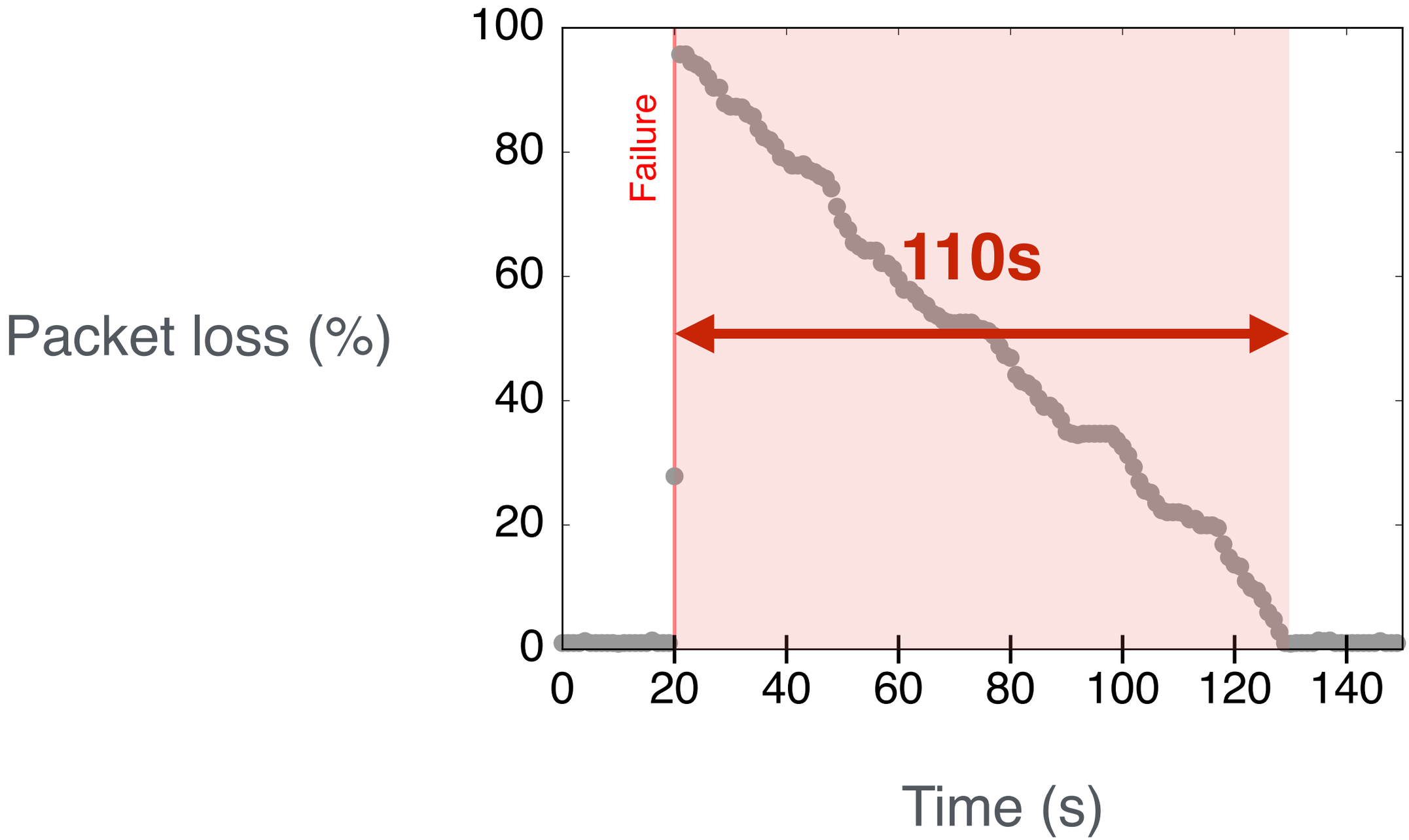
Remote outages result in downtime



Remote outages result in downtime



Remote outages result in downtime



SWIFT: Predictive Fast Reroute Framework

SWIFT: Predictive Fast Reroute Framework

works upon **remote** (and local) outages

SWIFT: Predictive Fast Reroute

1. ***SWIFT*** reduces the learning time of a withdrawal from 13s to 2s using a control-plane prediction algorithm

SWIFT: Predictive Fast Reroute

1. ***SWIFT*** reduces the learning time of a withdrawal from 13s to 2s using a control-plane prediction algorithm

2. ***SWIFT*** fast reroutes the affected traffic towards unaffected paths in 40ms using a hierarchical forwarding table

SWIFT: Predictive Fast Reroute

1. ***SWIFT*** reduces the learning time of a withdrawal from 13s to 2s using a control-plane prediction algorithm

2. ***SWIFT*** fast reroutes the affected traffic towards unaffected paths in 40ms using a hierarchical forwarding table

3. ***SWIFT*** is deployable on existing routers

SWIFT: Predictive Fast Reroute

1. ***SWIFT*** reduces the learning time of a withdrawal from 13s to 2s using a control-plane prediction algorithm

2. ***SWIFT*** fast reroutes the affected traffic towards unaffected paths in 40ms using a hierarchical forwarding table

3. ***SWIFT*** is deployable on existing routers

SWIFT predicts the extent of an outage
using fast, yet precise Root Cause Analysis

Existing RCA techniques *

accuracy

vantage points

detection speed

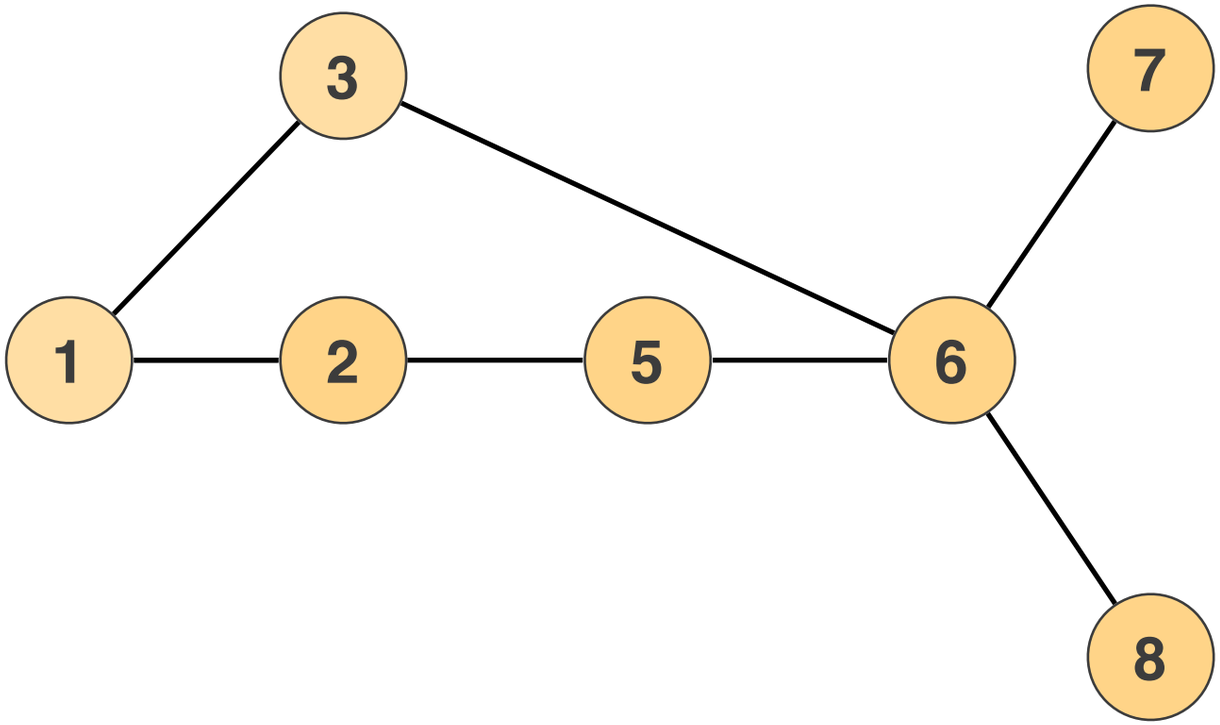
* Lifeguard, SIGCOMM’14
  Poiroot, SIGCOMM’13
  Feldmann et al., SIGCOMM’04

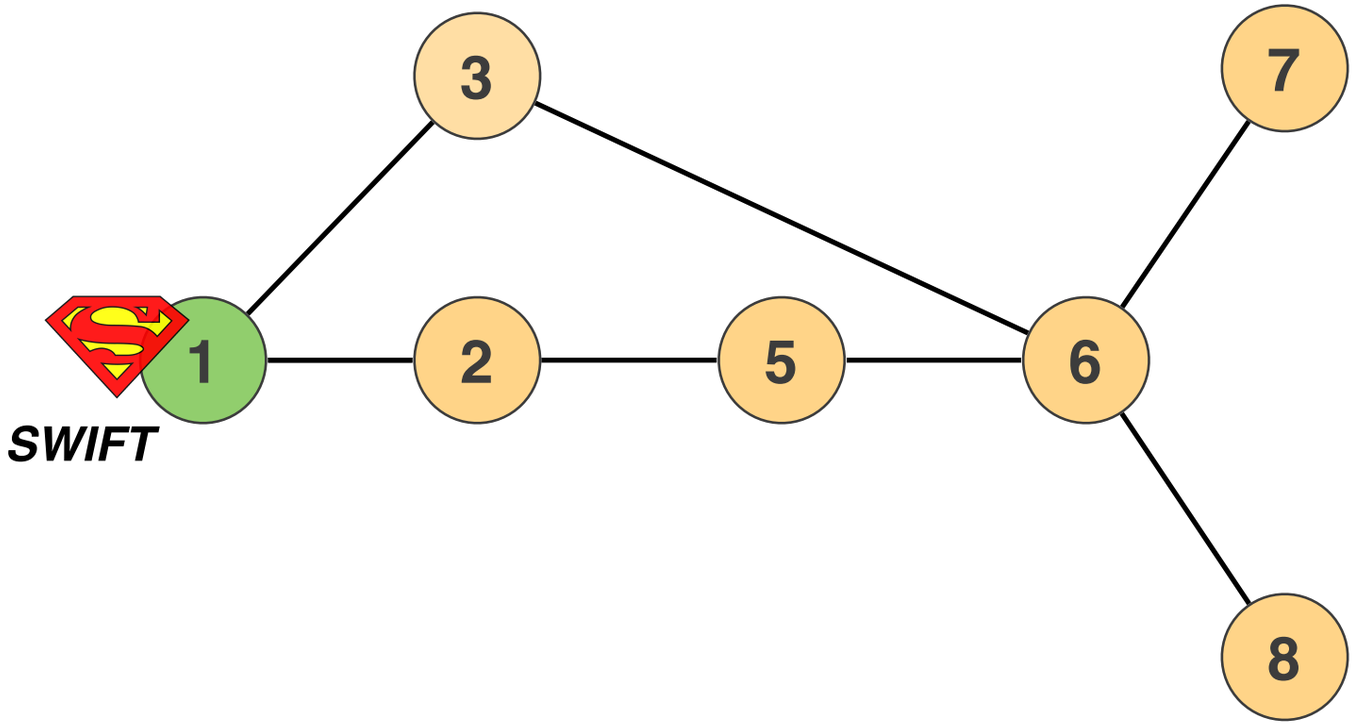
	Existing RCA techniques *
accuracy	link/AS
# vantage points	multiple
detection speed	O(minute)

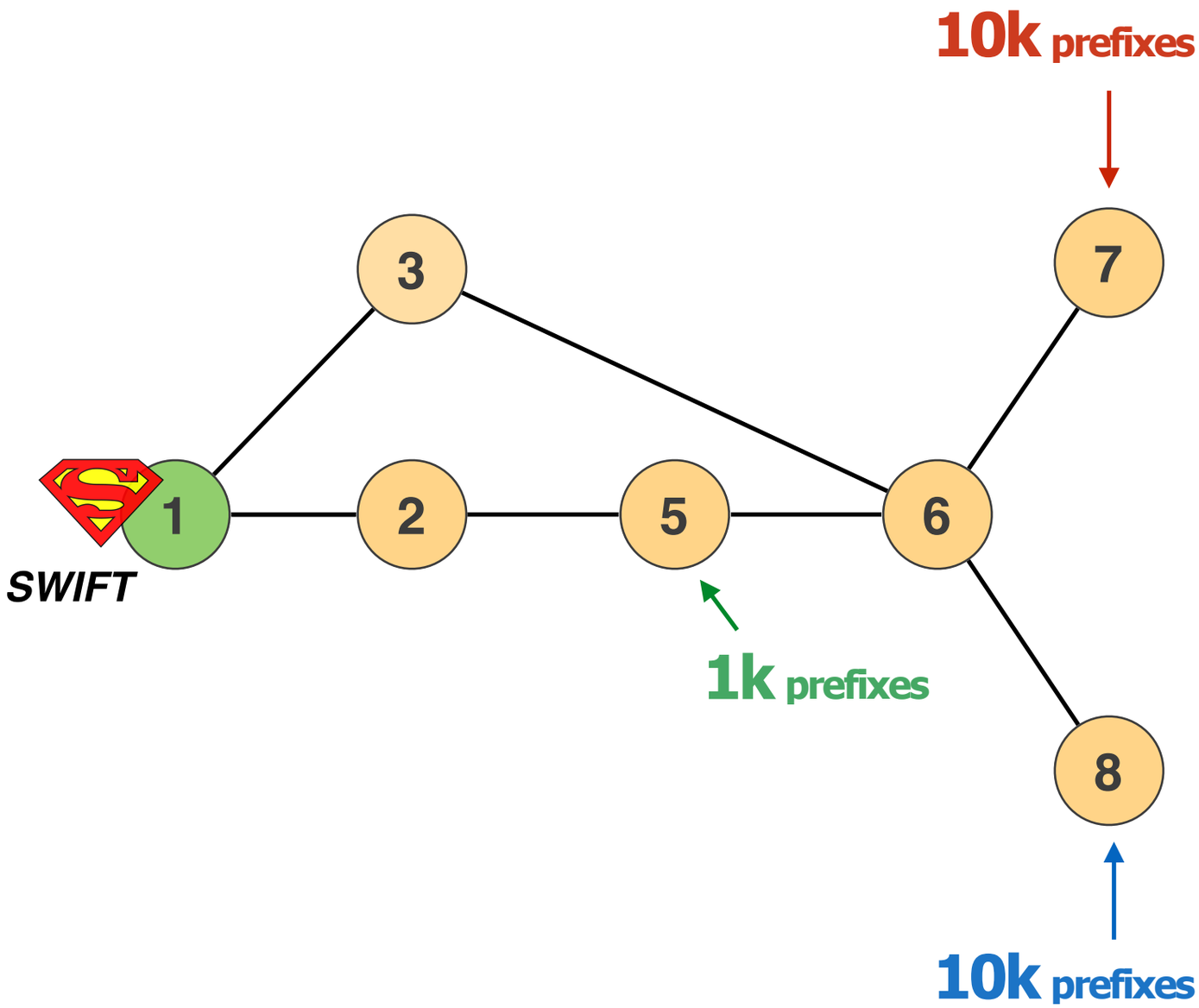
* Lifeguard, SIGCOMM'14
Poiroot, SIGCOMM'13
Feldmann et al., SIGCOMM'04

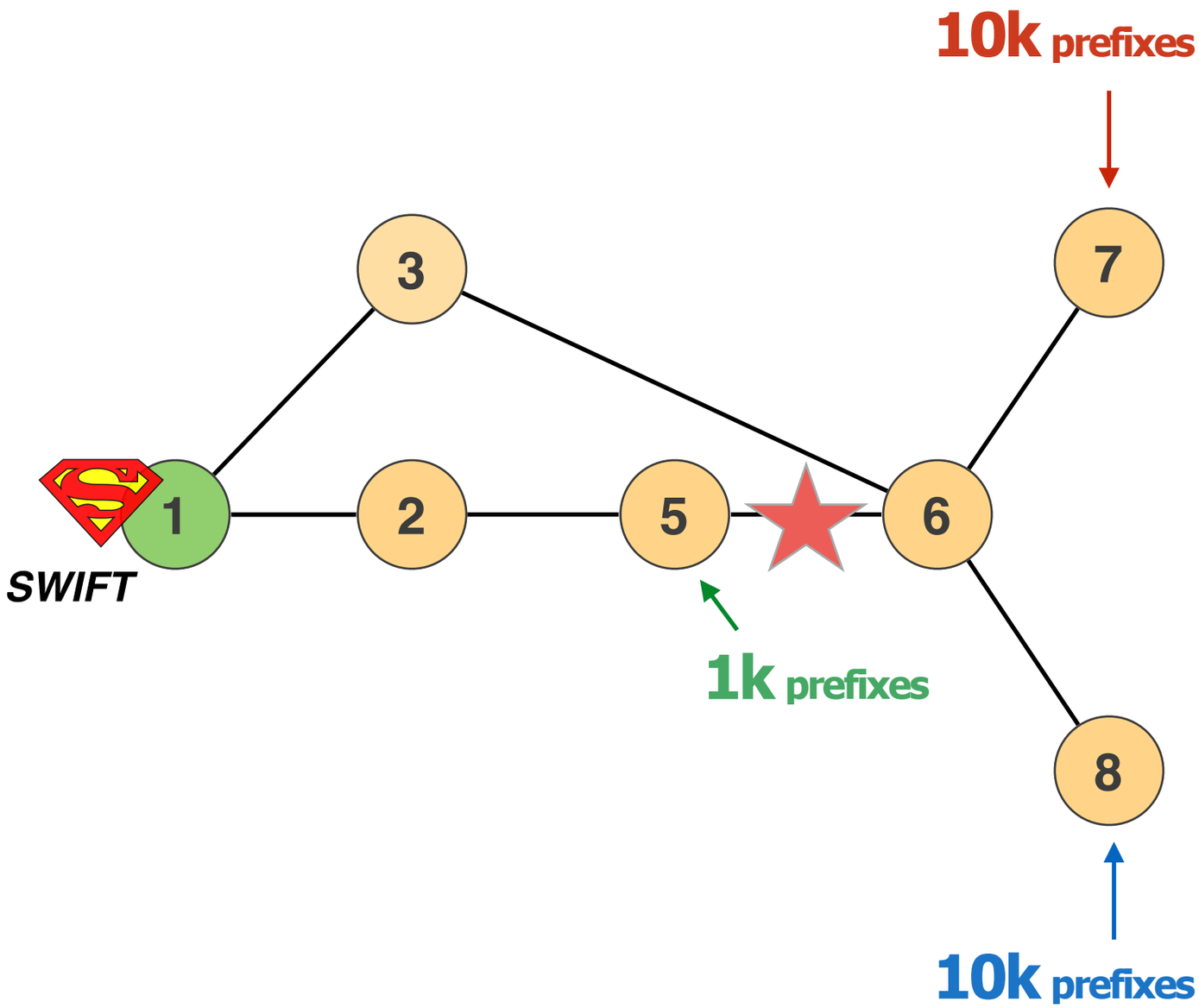
	Existing RCA techniques *	***SWIFT***
accuracy	link/AS	region
# vantage points	multiple	1
detection speed	O(minute)	**O(second)**

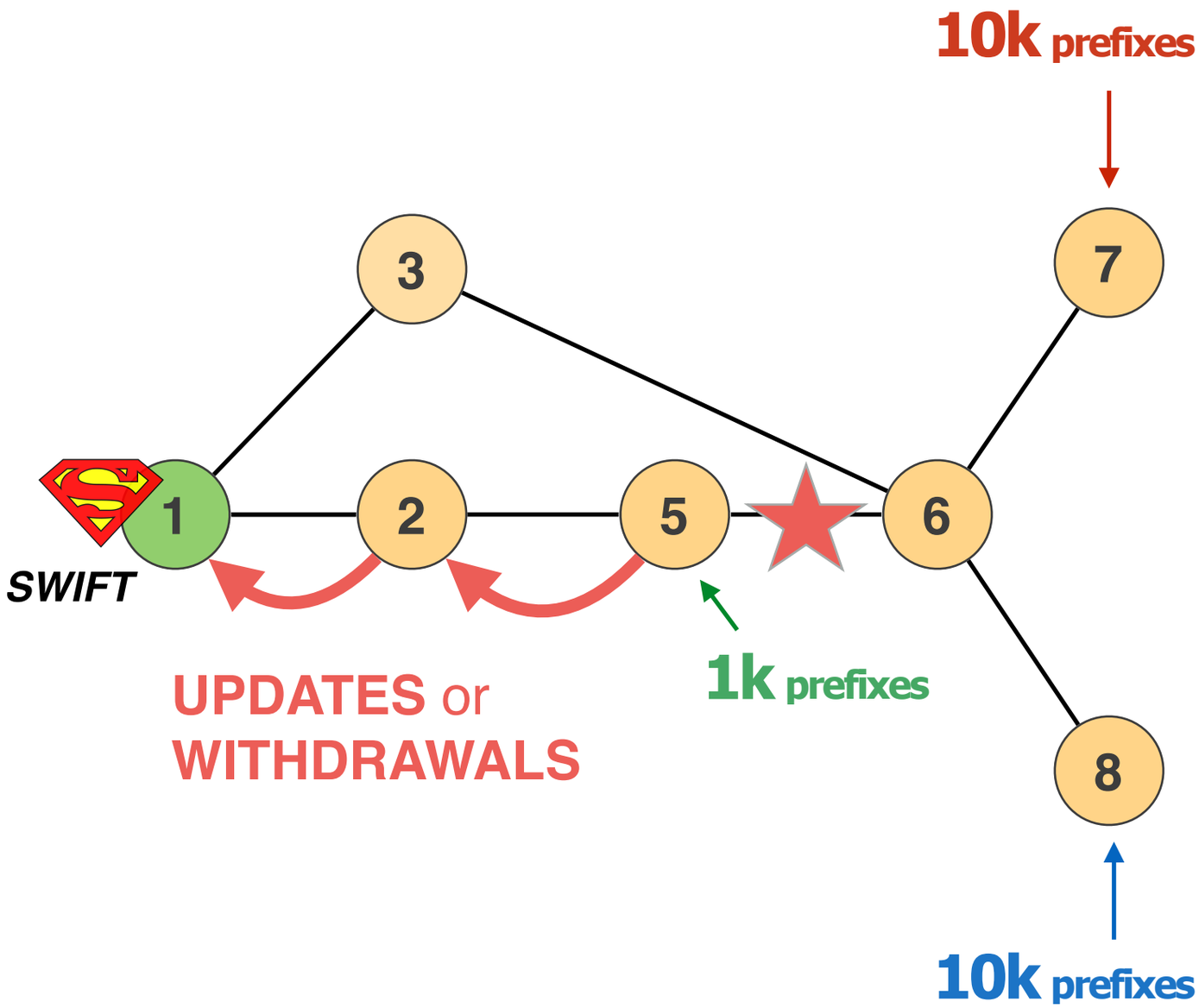
* Lifeguard, SIGCOMM'14
Poiroot, SIGCOMM'13
Feldmann et al., SIGCOMM'04

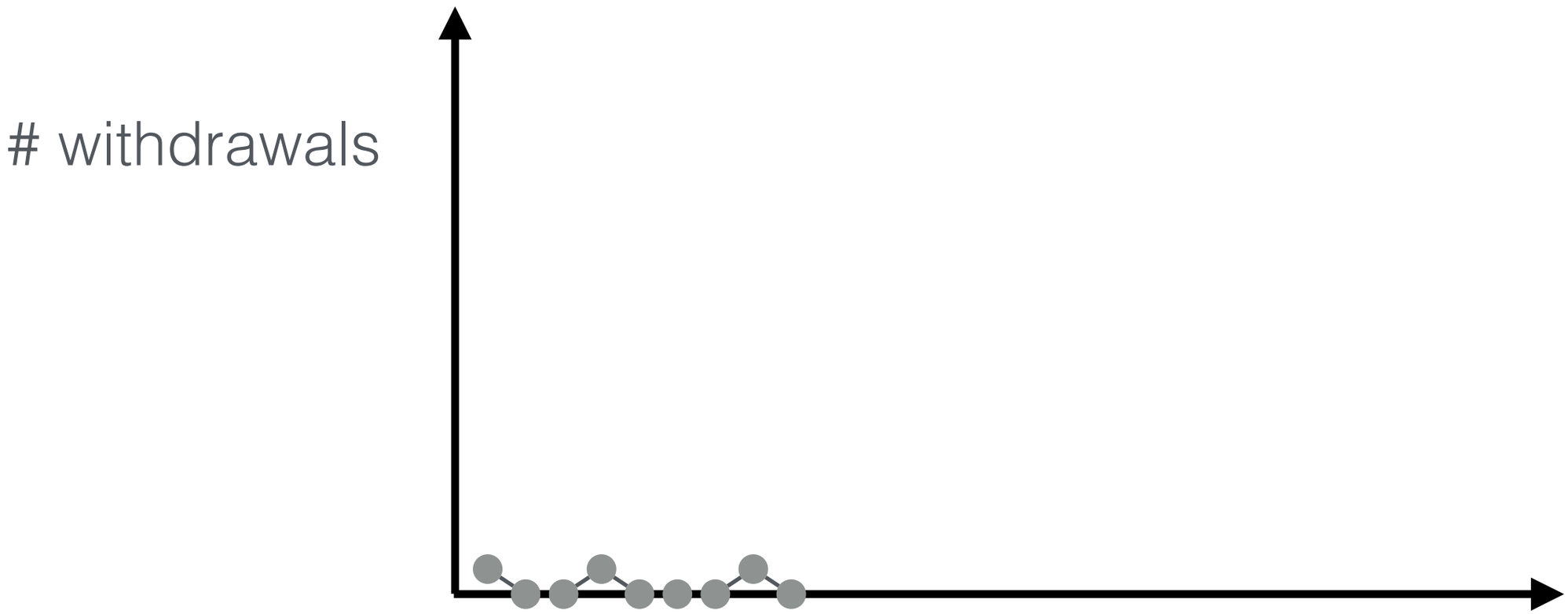

3
7
1
2
5
6
8


3
7
1
2
5
6
8
SWIFT


10k prefixes
3
7
1
2
5
6
SWIFT
1k prefixes
8
10k prefixes


10k prefixes
3
7
1
2
5
6
SWIFT
1k prefixes
8
10k prefixes


10k prefixes
3
7
1
2
5
6
8
SWIFT
UPDATES or
WITHDRAWALS
1k prefixes
10k prefixes



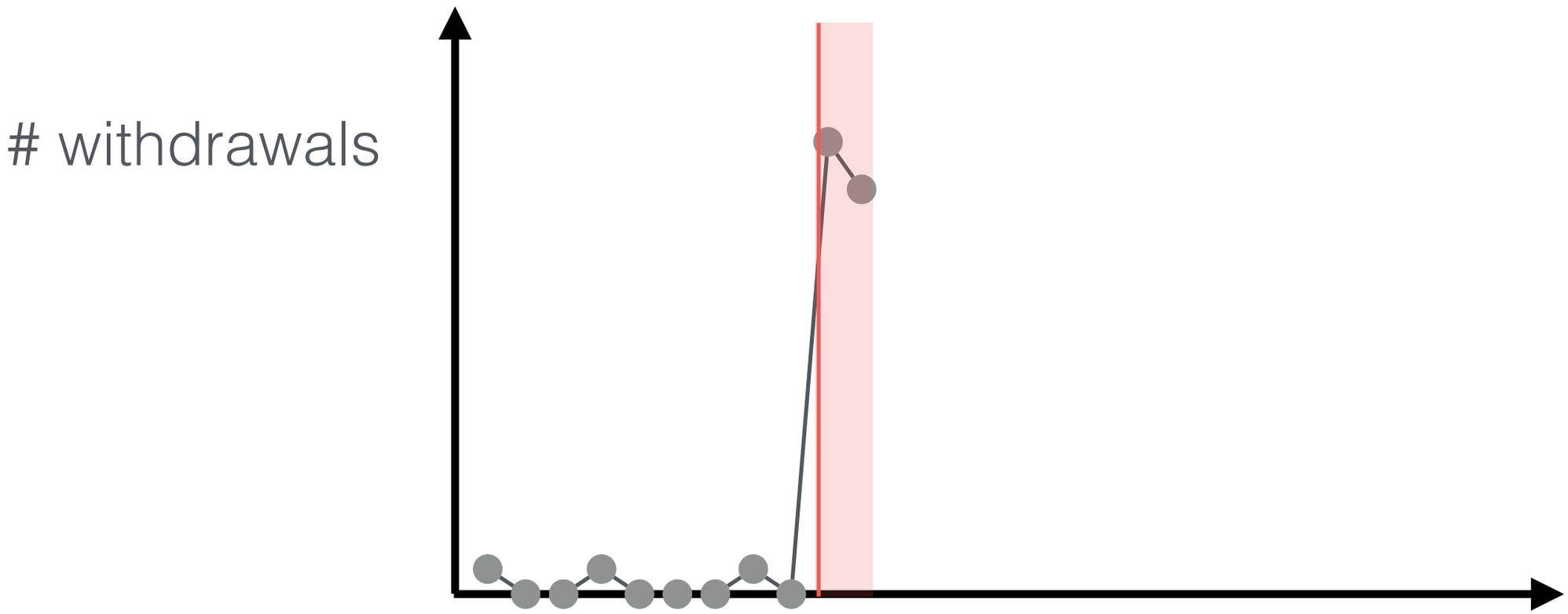
#1 monitor the stream of BGP messages

#2 detect and
locate the outage

Links failure
probability

(A,B)	0.3
⋮	⋮
(A,D)	**0.9**

→ Link **(A,D)** is dead



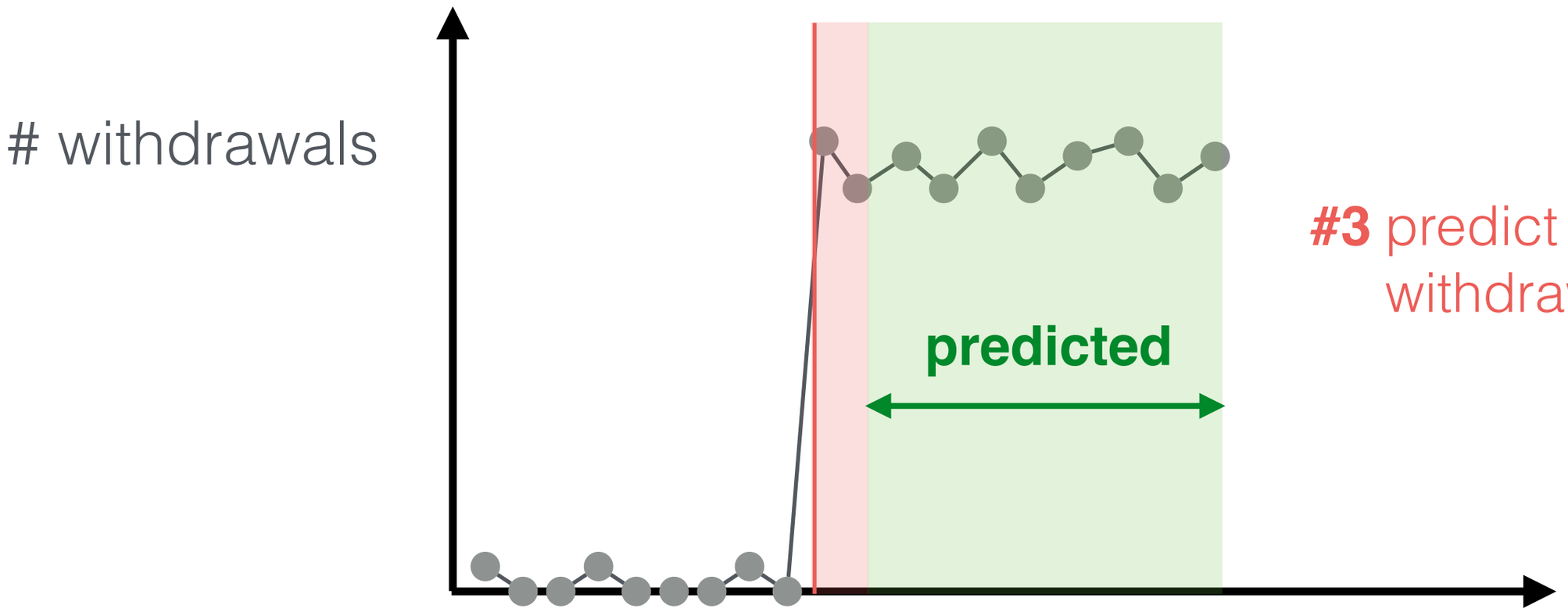
#1 monitor the stream
of BGP messages

#2 detect and
locate the outage

Links failure
probability

(A,B)	0.3
⋮	⋮
(A,D)	**0.9**

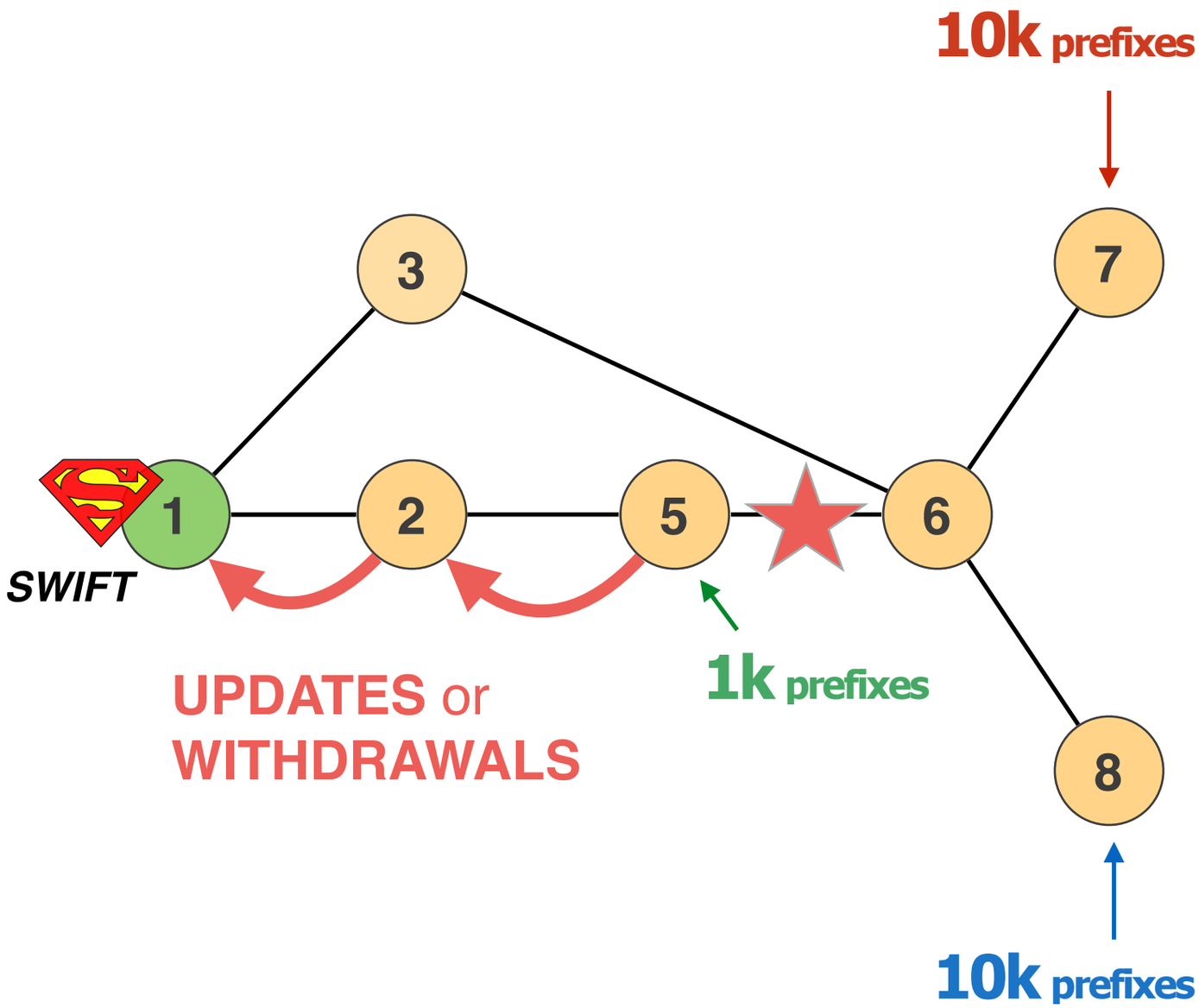
→ Link **(A,D)** is dead

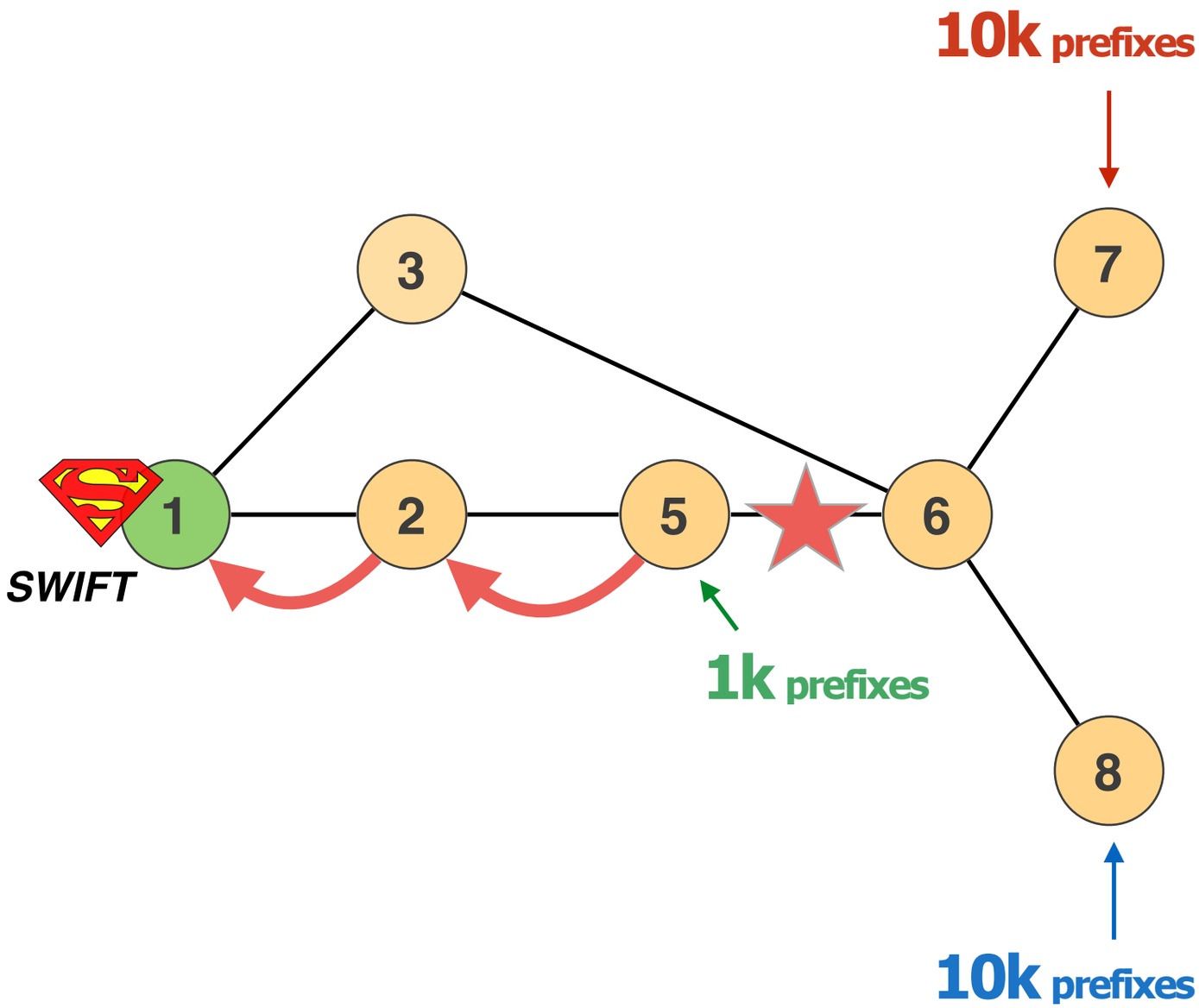


#3 predict future
withdrawals

#1 monitor the stream
of BGP messages

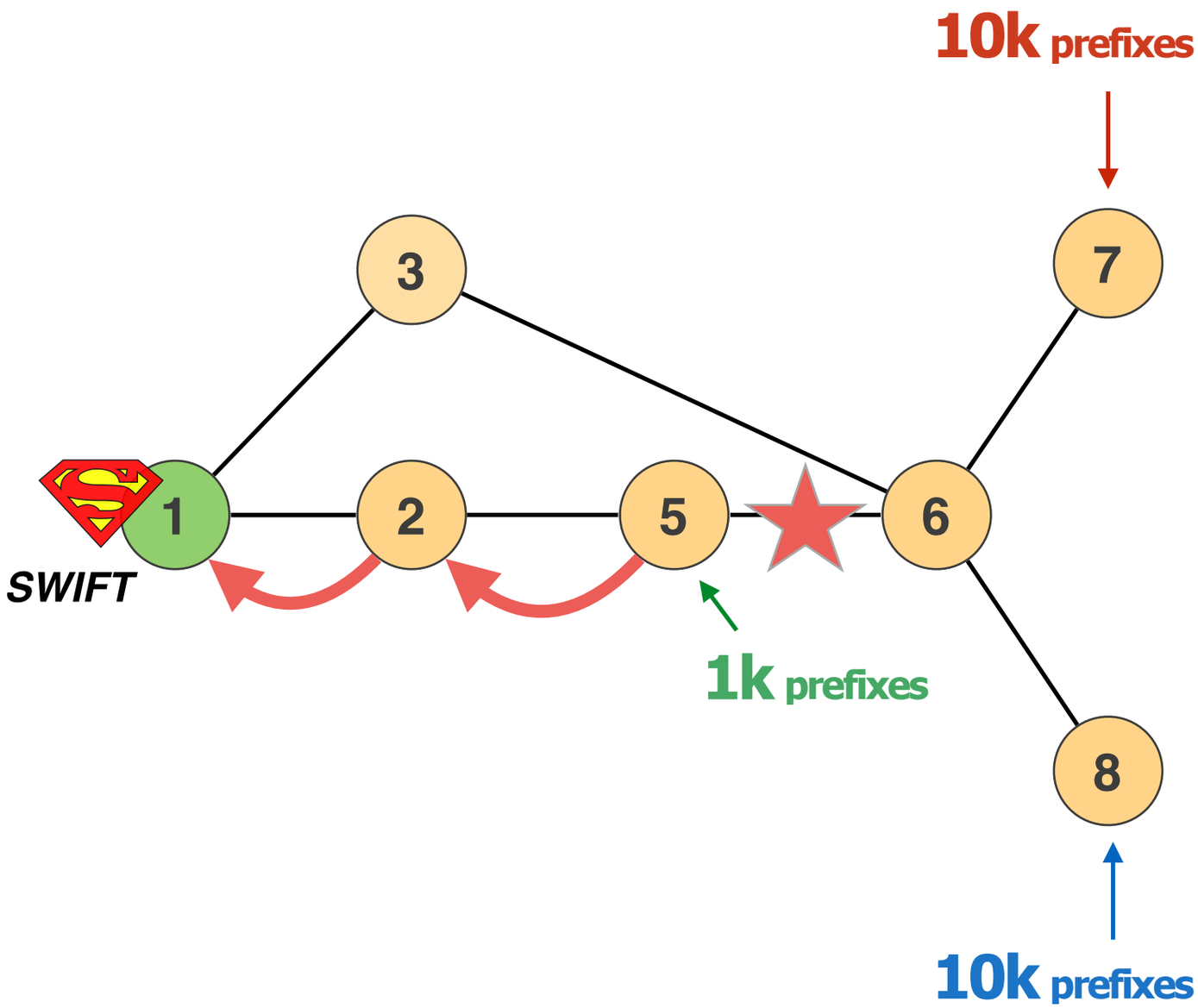
SWIFT link failure inference algorithm leverages the affected and unaffected prefixes





candidates set

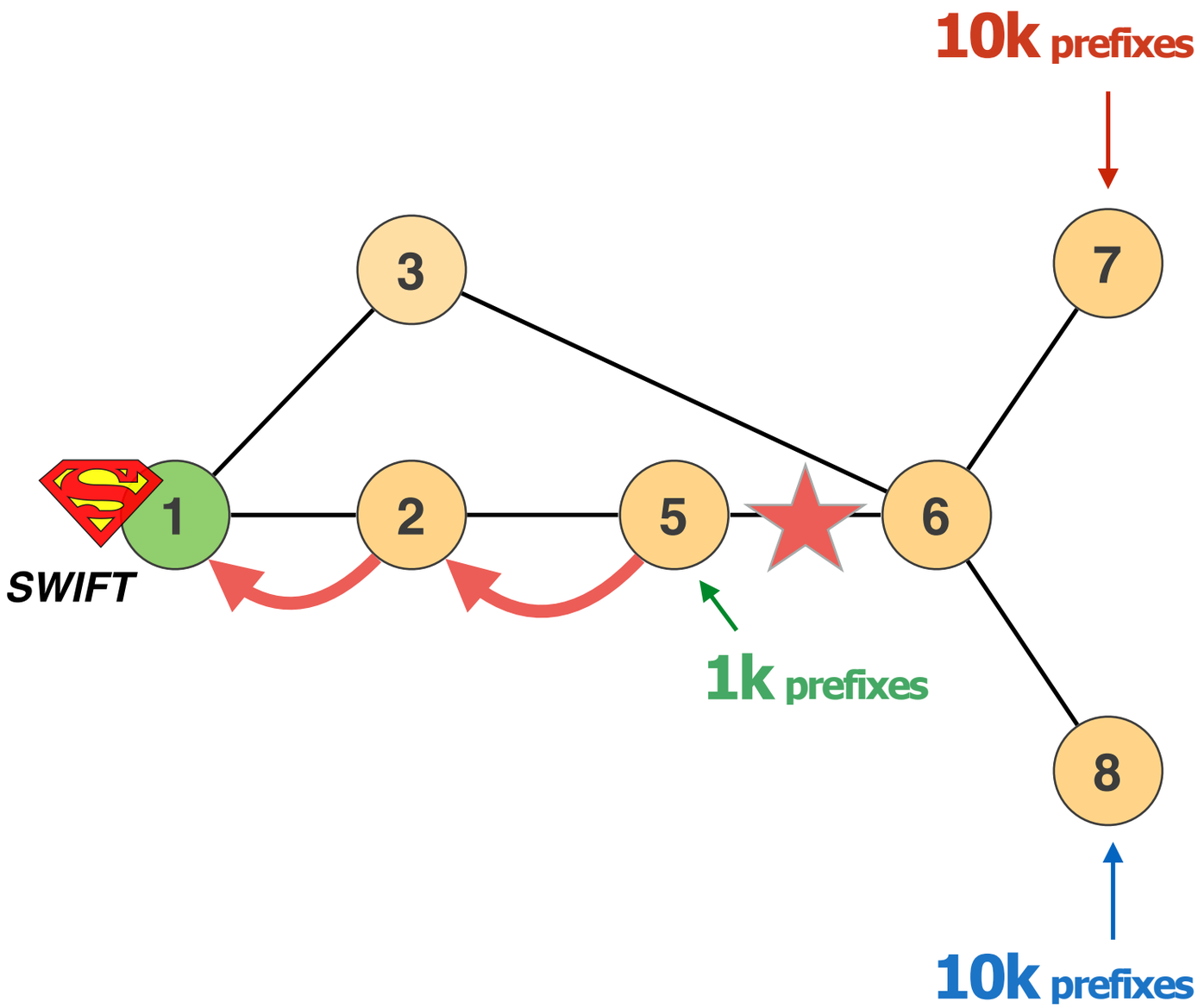
AS links (2,5) (5,6) (6,7) (6,8)



	candidates set
AS links	(2,5) (5,6) (6,7) (6,8)

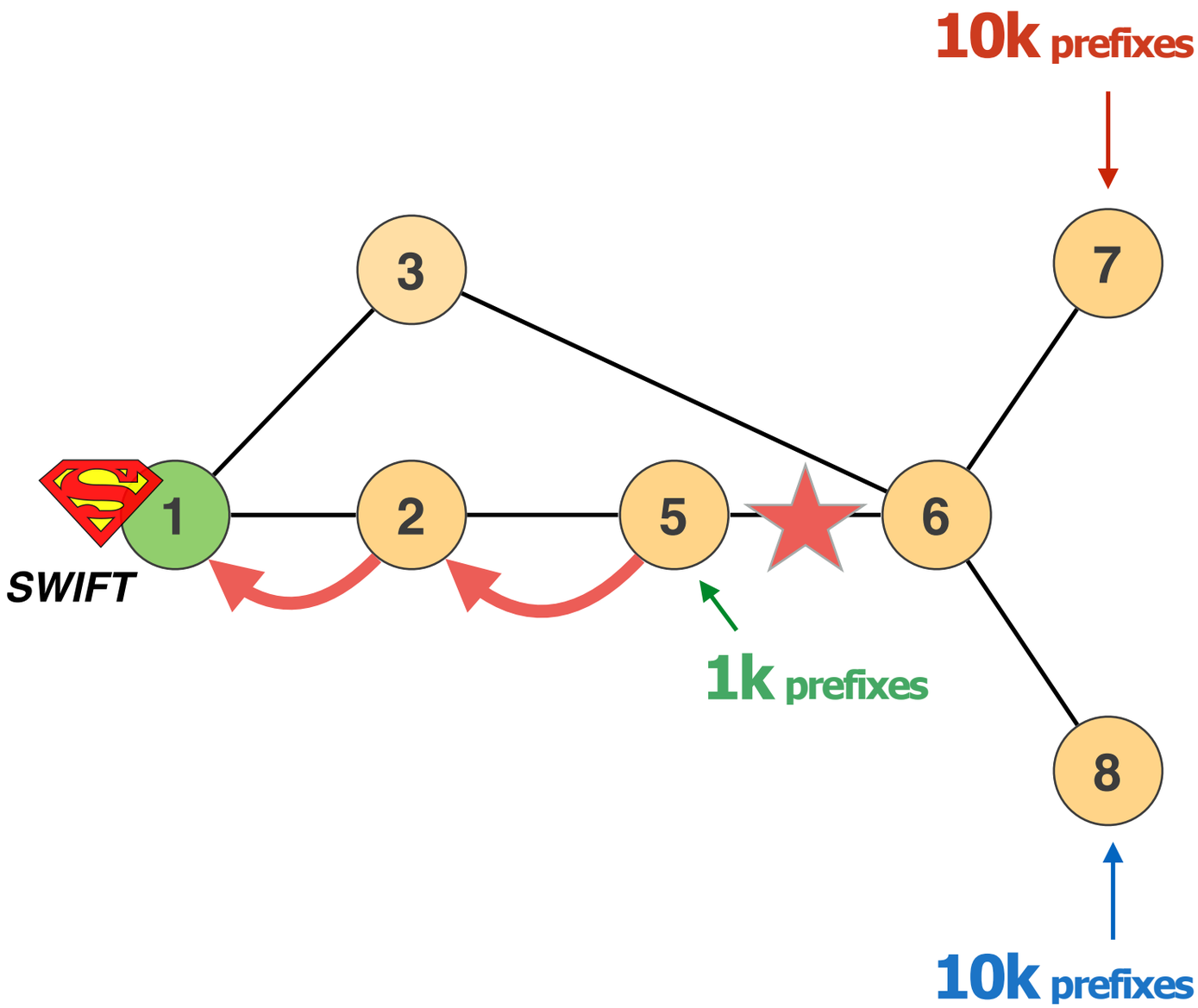
affected prefixes

from AS7	(2,5) (5,6) (6,7) ~~(6,8)~~



	candidates set
AS links	(2,5) (5,6) (6,7) (6,8)

affected prefixes	
from AS7	(2,5) (5,6) (6,7) ~~(6,8)~~
from AS8	(2,5) (5,6) ~~(6,7)~~ ~~(6,8)~~



candidates set

AS links (2,5) (5,6) (6,7) (6,8)

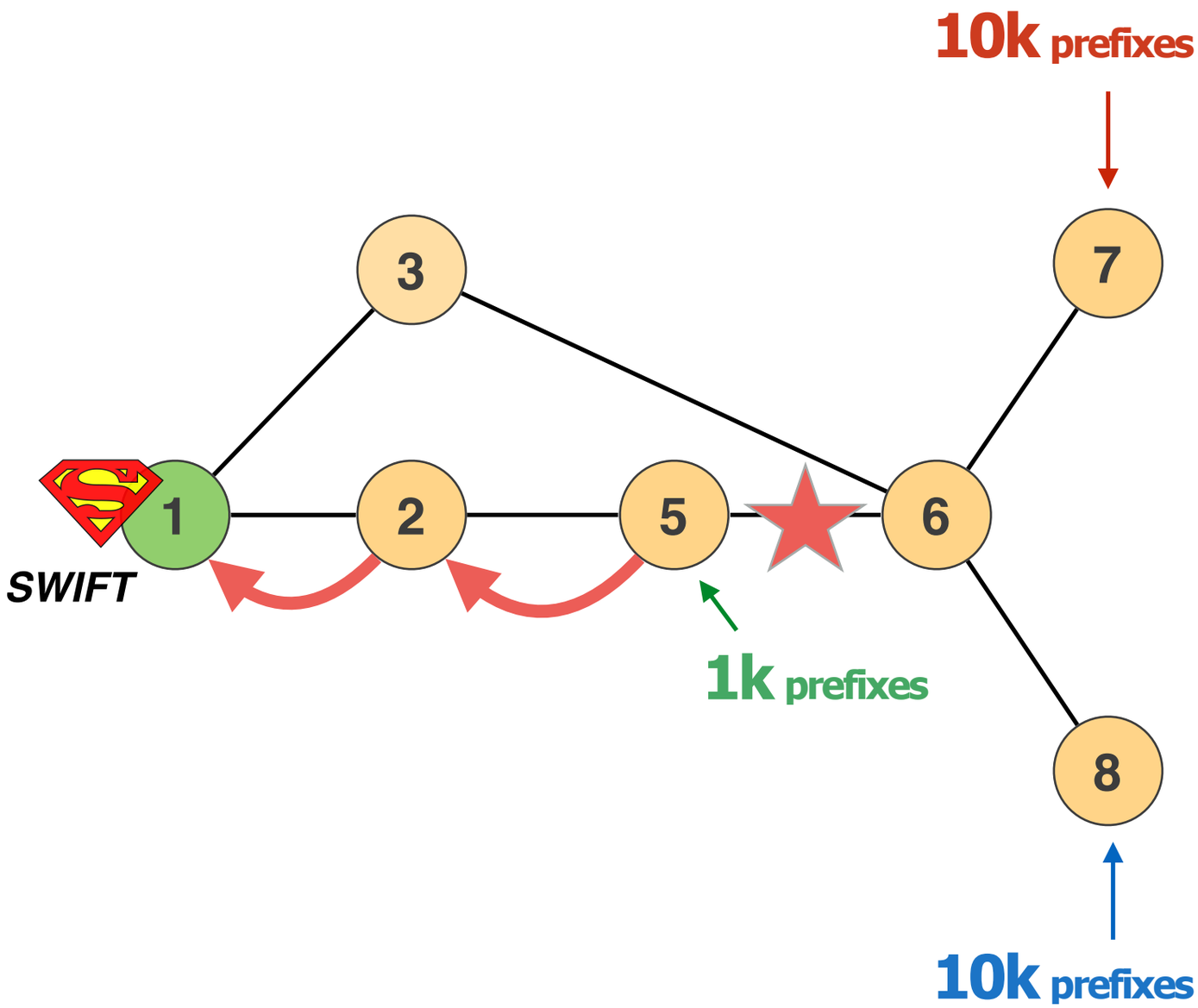
affected prefixes

from AS7 (2,5) (5,6) (6,7) ~~(6,8)~~

from AS8 (2,5) (5,6) ~~(6,7)~~ ~~(6,8)~~

unaffected prefixes

from AS5 ~~(2,5)~~ (5,6) ~~(6,7)~~ ~~(6,8)~~



candidates set

AS links (2,5) (5,6) (6,7) (6,8)

affected prefixes

from AS7 (2,5) (5,6) (6,7) ~~(6,8)~~

from AS8 (2,5) (5,6) ~~(6,7)~~ ~~(6,8)~~

unaffected prefixes

from AS5 ~~(2,5)~~ (5,6) ~~(6,7)~~ ~~(6,8)~~

is the failed link

In practice, SWIFT uses two metrics

Computed on a per AS-link basis

Withdrawals share

$WS(l, t)$

Fraction of withdrawals crossing link l

Path sharing

$PS(l, t)$

Proportion of prefixes withdrawn on link l

In practice, SWIFT uses two metrics
Computed on a per AS-link basis

$$FS(l, t) = \text{Withdrawals share } WS(l, t) \times \text{Path sharing } PS(l, t)$$

Fit Score for link l

Fraction of withdrawals crossing link l

Proportion of prefixes withdrawn on link l

Theorem Under perfect conditions, SWIFT always returns a set of links including the failed link

Theorem

Under perfect conditions, SWIFT always returns a set of links including the failed link

Challenge #1

To be fast, we can't wait for all the withdrawals

SWIFT runs the link inference **early** during the burst in order to predict future withdrawals

Theorem

Under perfect conditions, SWIFT always returns a set of links including the failed link

Challenge #1

To be fast, we can't wait for all the withdrawals

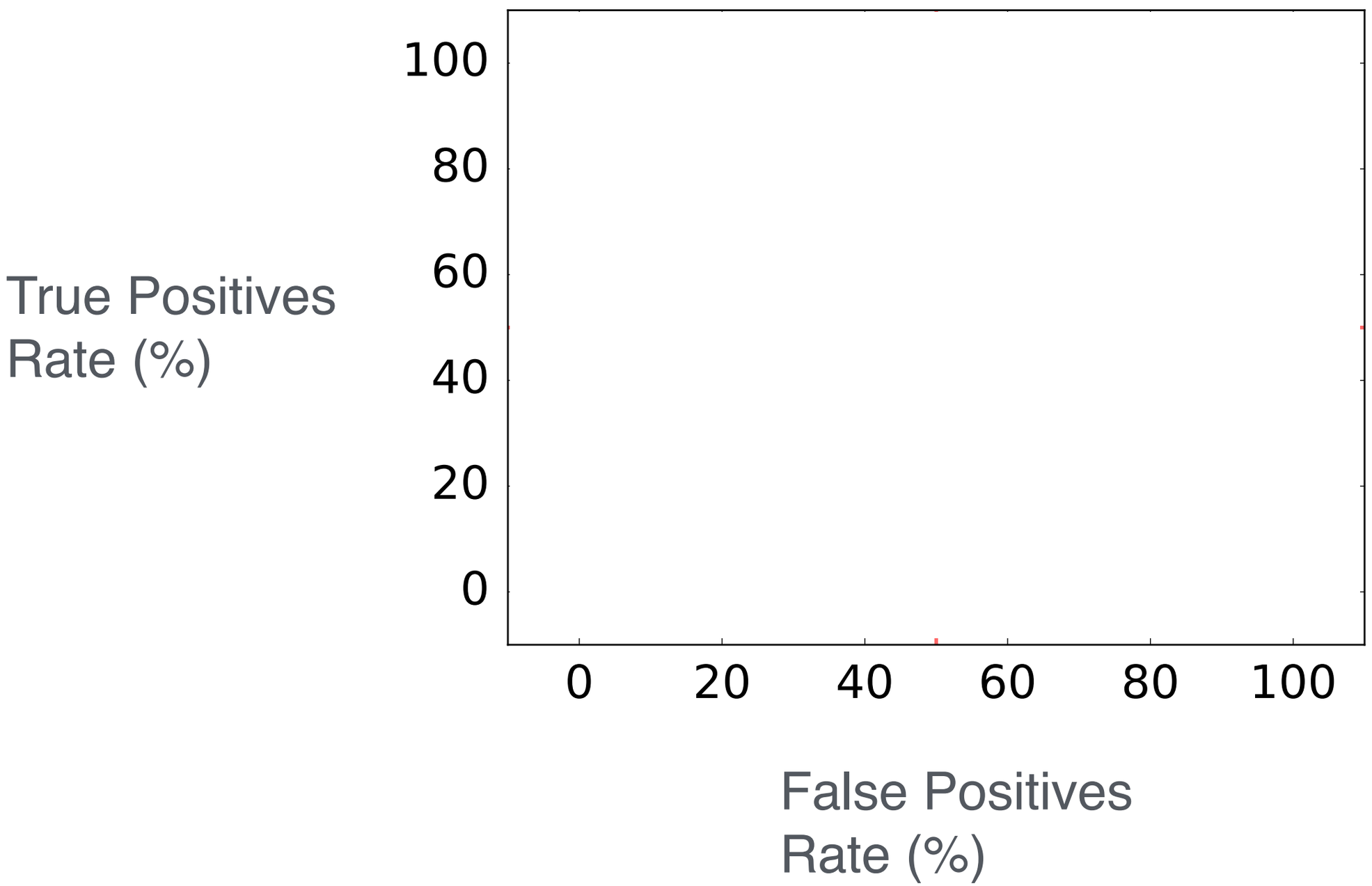
SWIFT runs the link inference **early** during the burst in order to predict future withdrawals

Challenge #2

An outage can affect multiple AS links

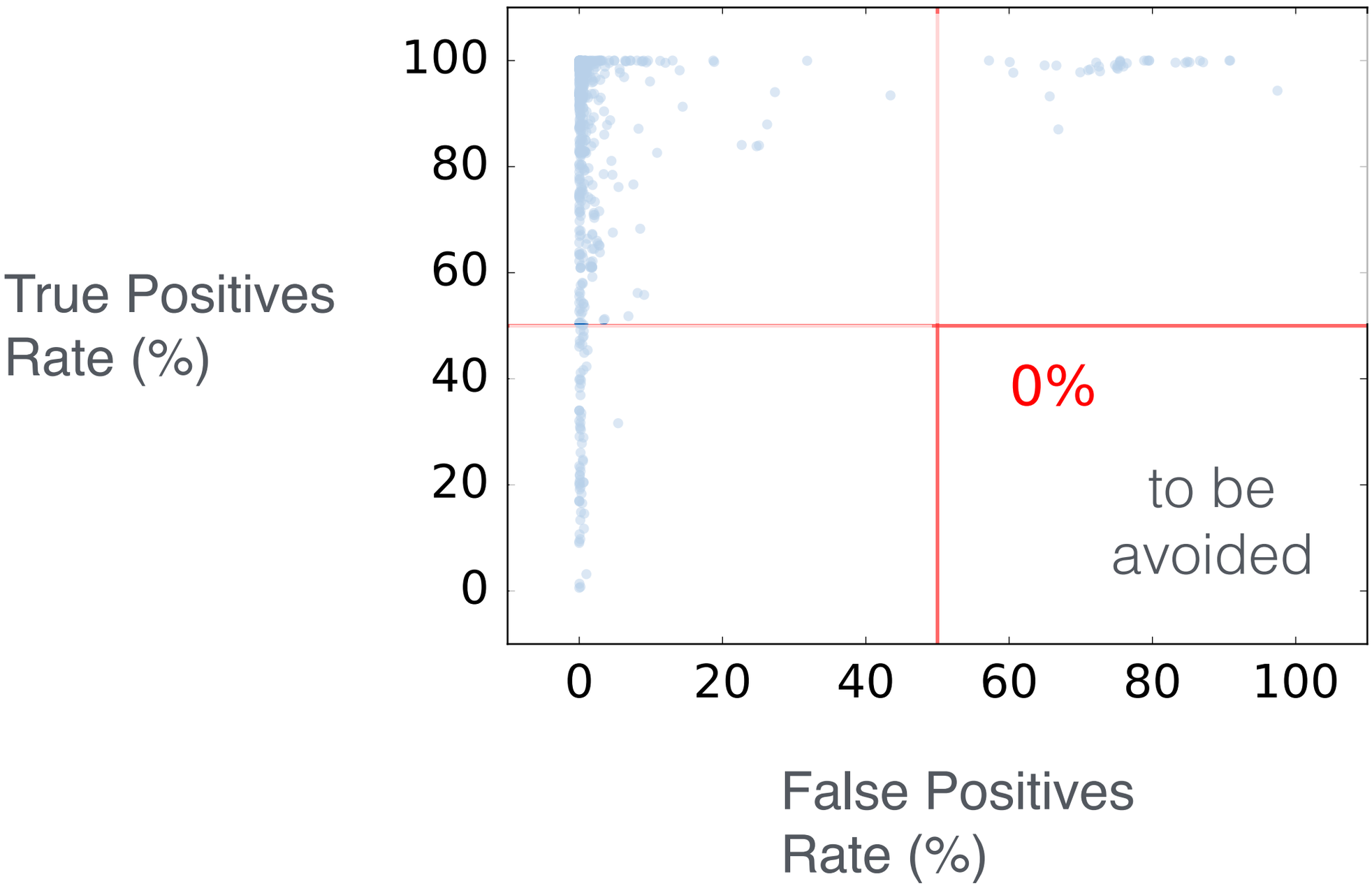
SWIFT link inference algorithm returns a ***set of links****, all with a high fit score*

SWIFT link failure inference is accurate

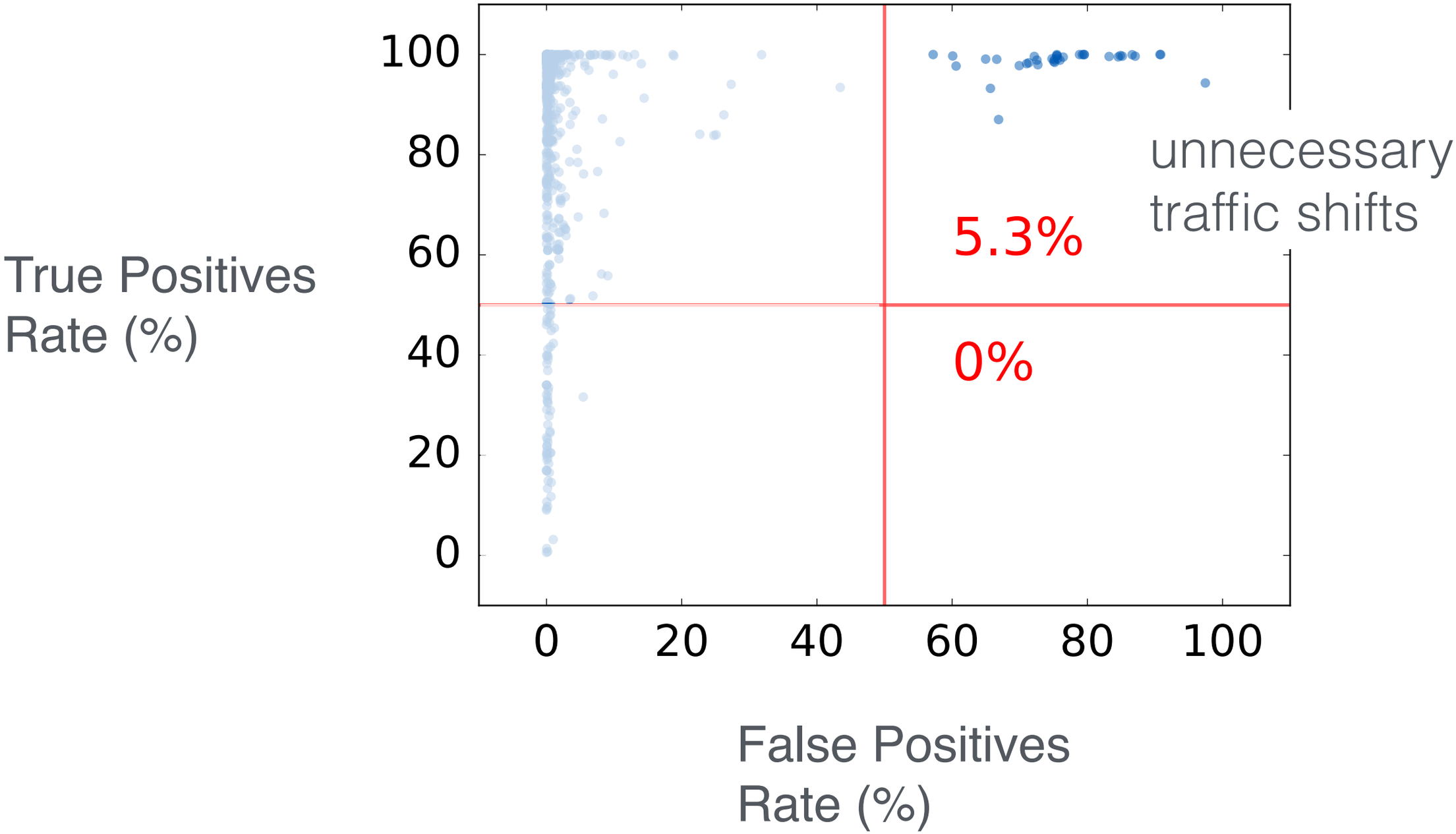


SWIFT link failure inference is accurate

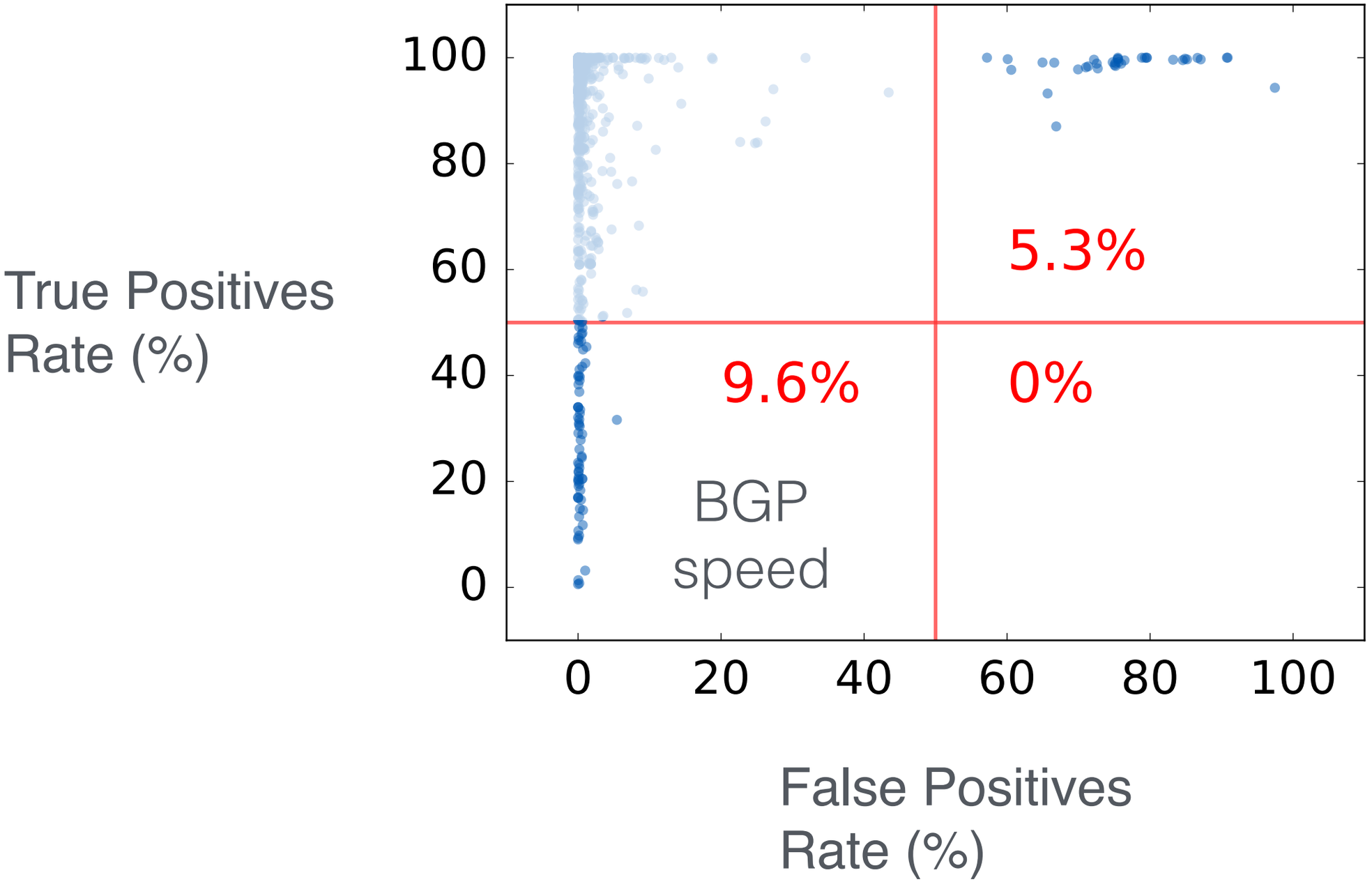
SWIFT link failure inference is accurate



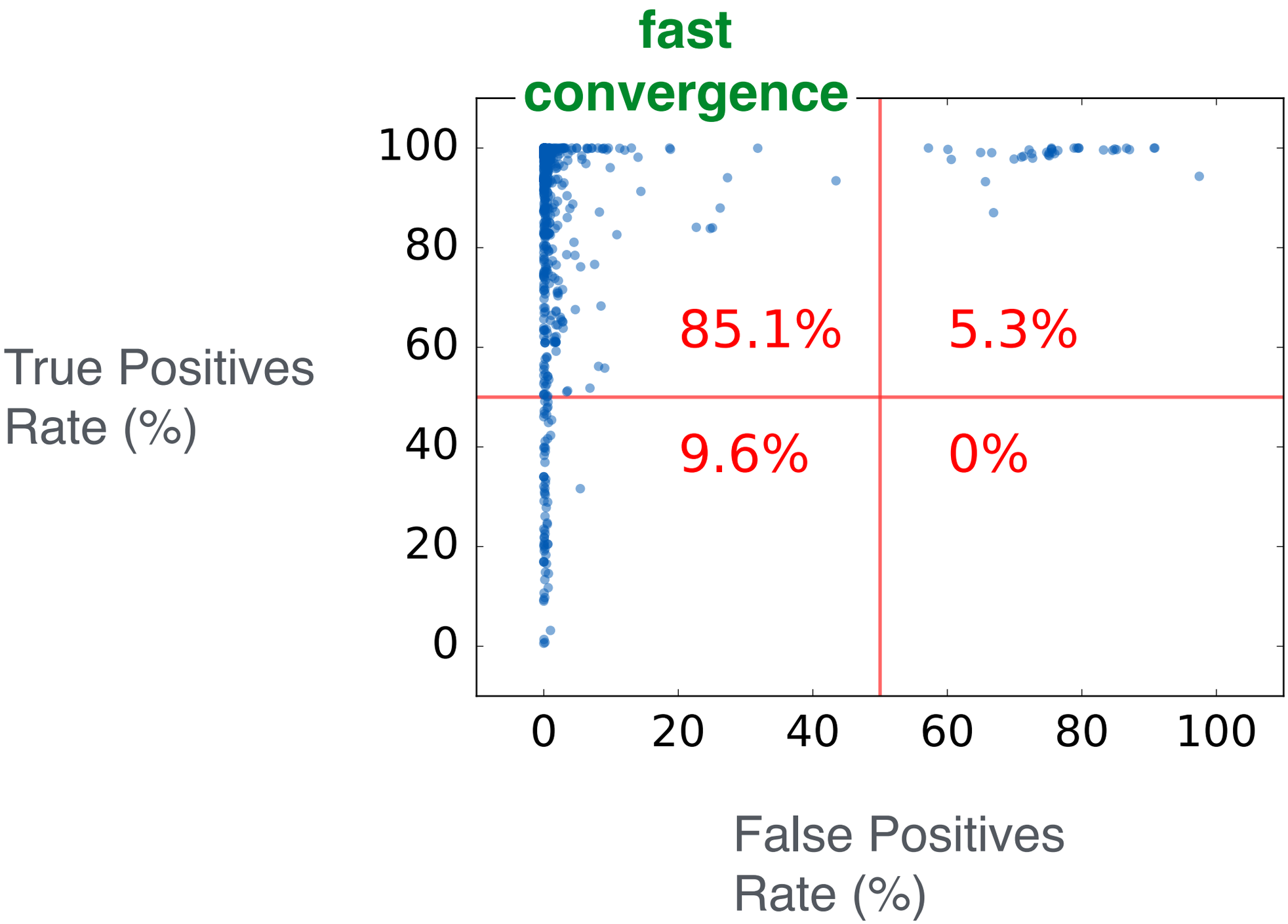
SWIFT link failure inference is accurate



SWIFT link failure inference is accurate



SWIFT link failure inference is accurate



SWIFT is much faster than BGP

	BGP	SWIFT
learning time (median)	**13 sec**	

SWIFT is much faster than BGP

	BGP	SWIFT
learning time (median)	**13 sec**	**2 sec**

SWIFT: Predictive Fast Reroute

1. ***SWIFT*** reduces the learning time of a withdrawal from 13s to 2s using a control-plane prediction algorithm

2. ***SWIFT*** fast reroutes the affected traffic towards unaffected paths in 40ms using a hierarchical forwarding table

3. ***SWIFT*** is deployable on existing routers

SWIFT: Predictive Fast Reroute

1. ***SWIFT*** reduces the learning time of a withdrawal from 13s to 2s using a control-plane prediction algorithm

2. ***SWIFT*** fast reroutes the affected traffic towards unaffected paths in 40ms using a hierarchical forwarding table

3. ***SWIFT*** is deployable on existing routers

Fast rerouting the traffic upon remote outages

has several challenges

Fast rerouting the traffic upon remote outages has several challenges

A remote outage can affect any set of prefixes

Fast rerouting the traffic upon remote outages has several challenges

A remote outage can affect any set of prefixes

A remote outage can affect backup paths

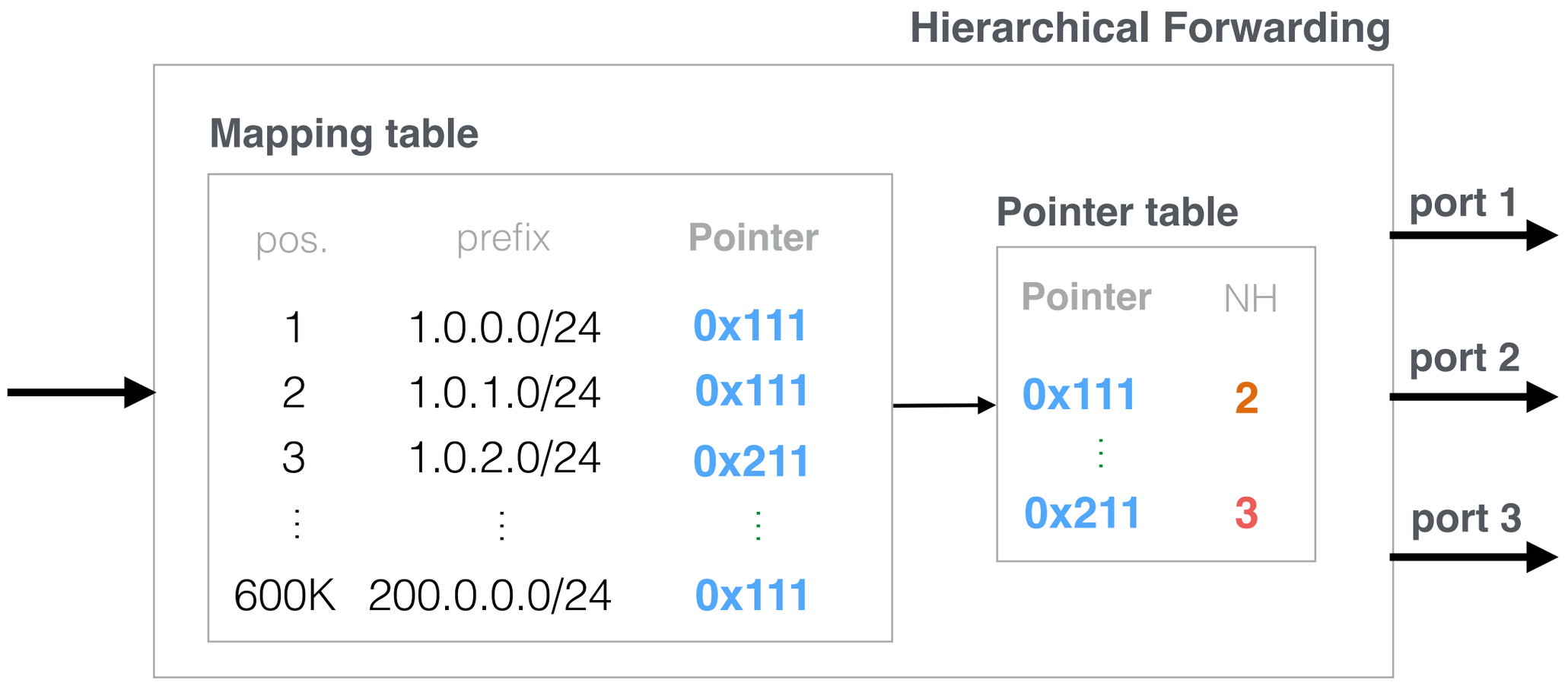
Fast rerouting the traffic upon remote outages
has several challenges

A remote outage can affect any set of prefixes

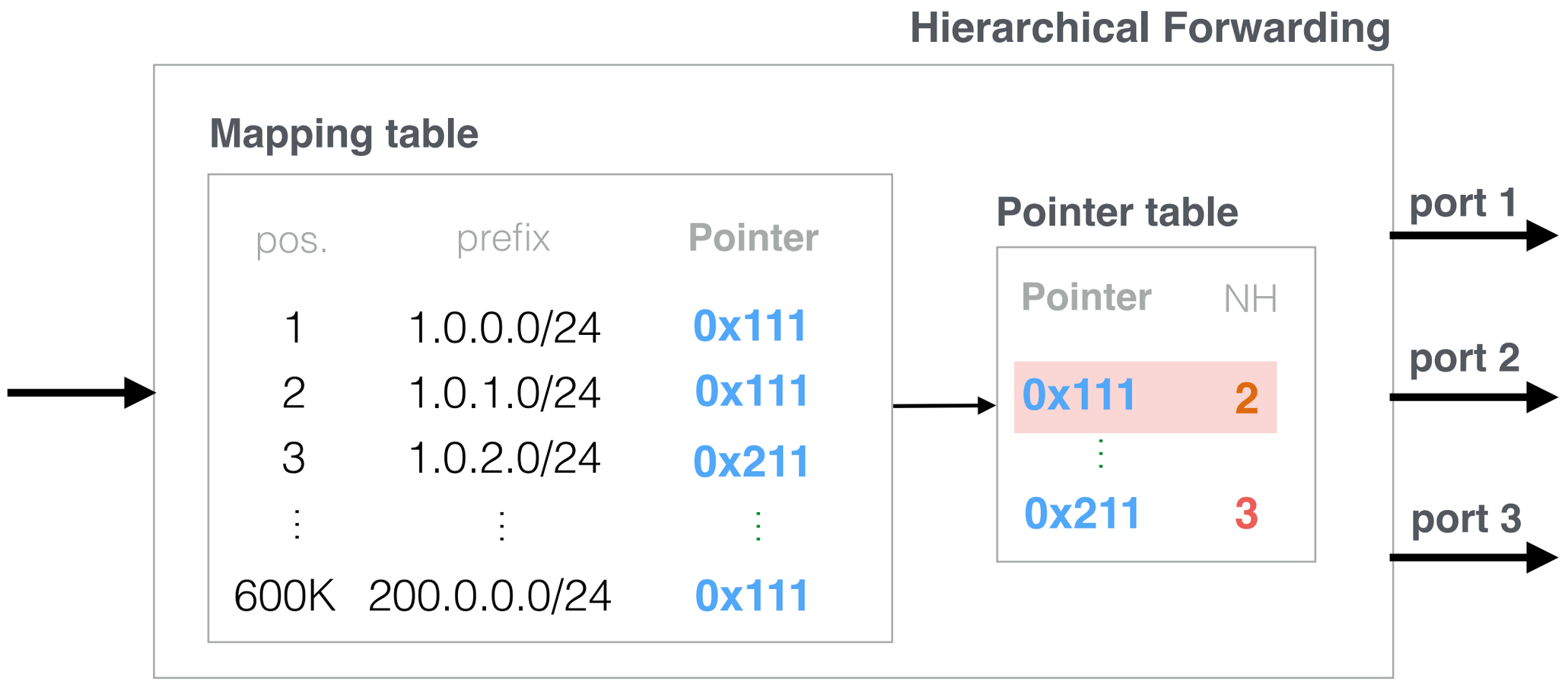
A remote outage can affect backup paths

and we want to be fast

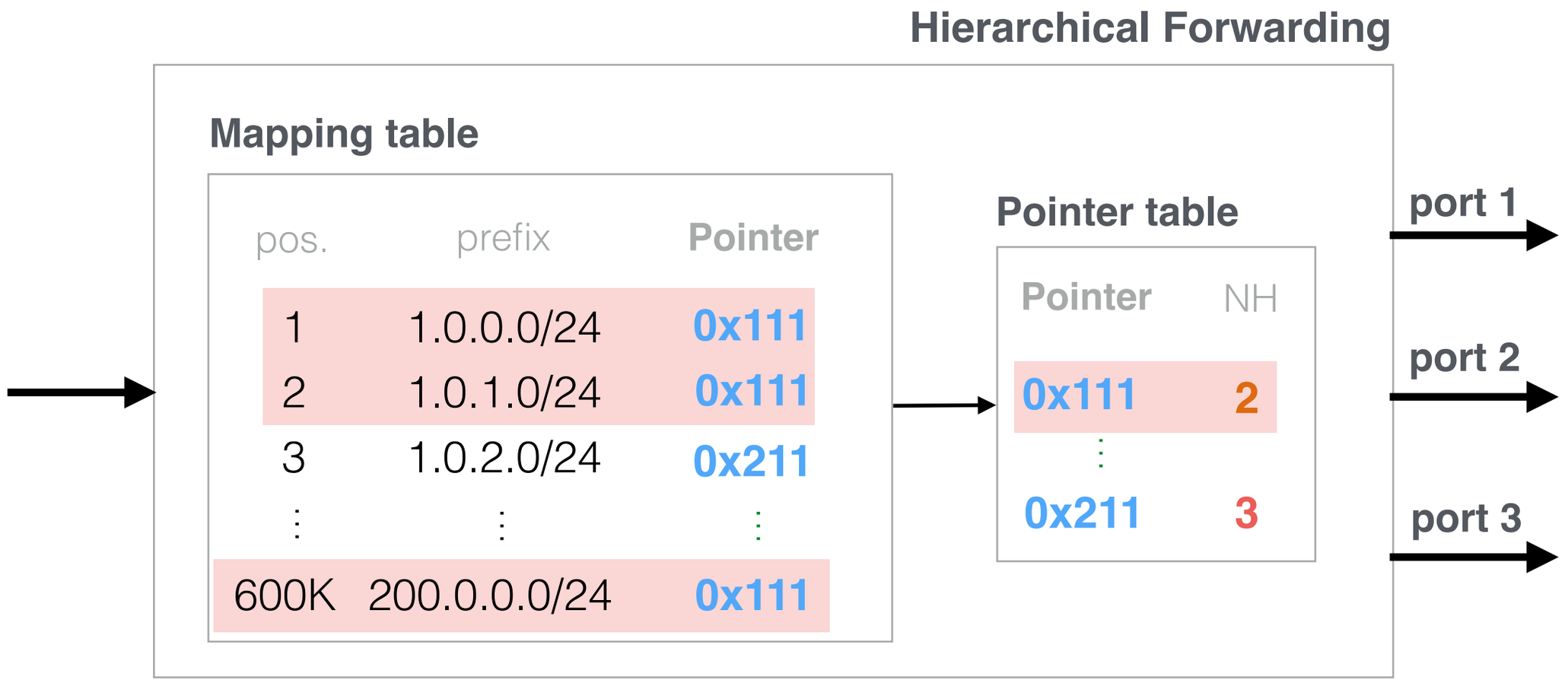
To solve these challenges,
SWIFT relies on a hierarchical forwarding table



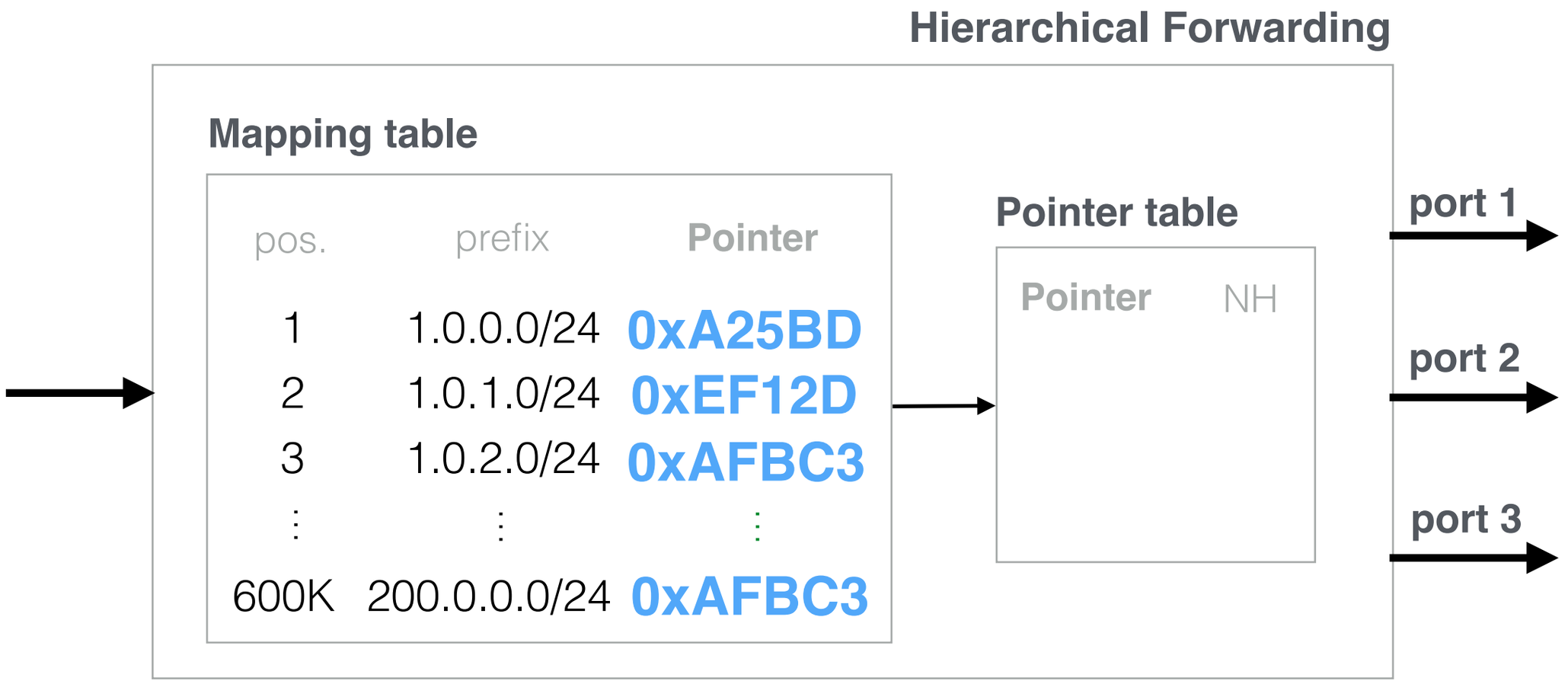
To solve these challenges,
SWIFT relies on a hierarchical forwarding table



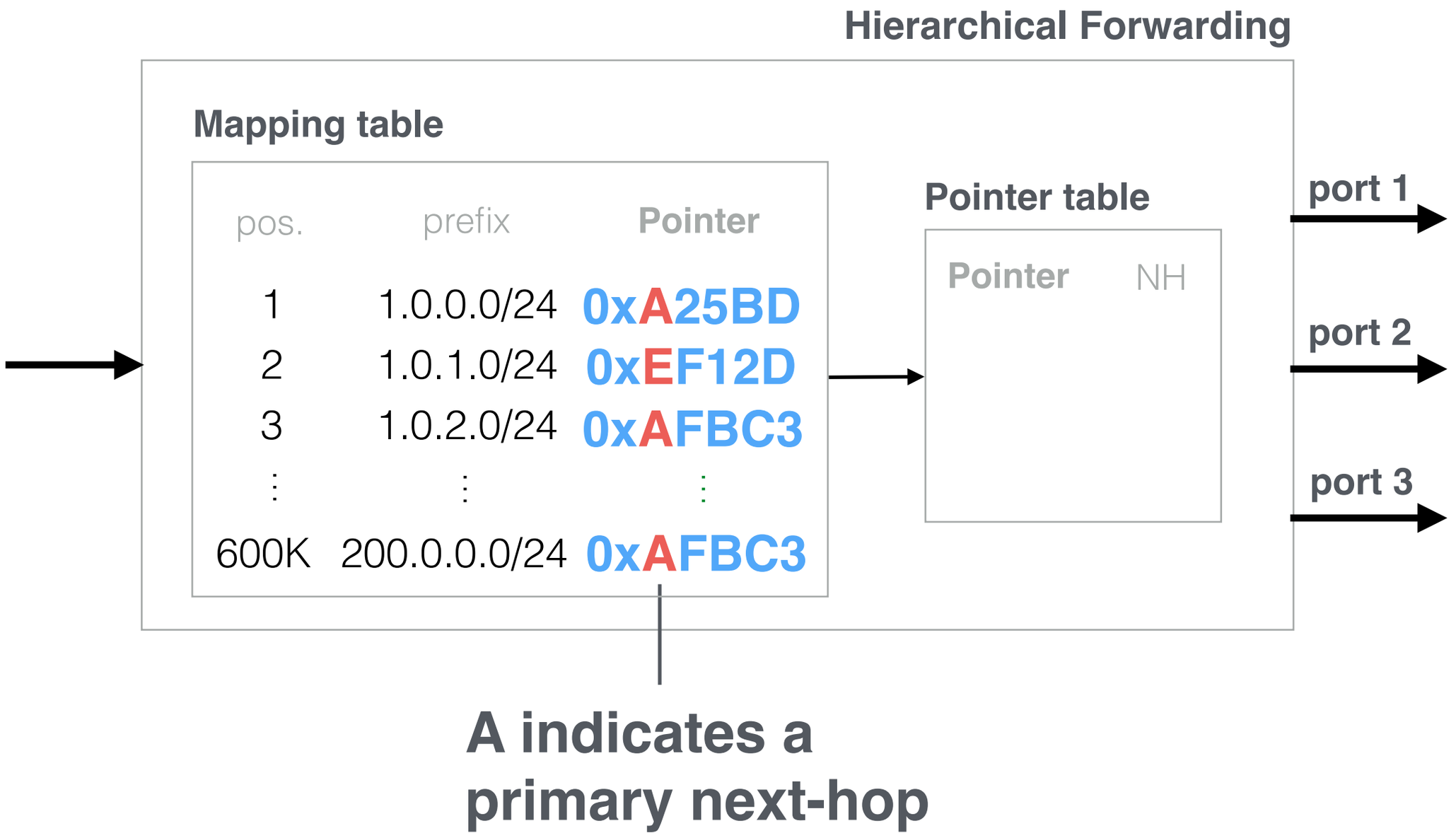
To solve these challenges,
SWIFT relies on a hierarchical forwarding table



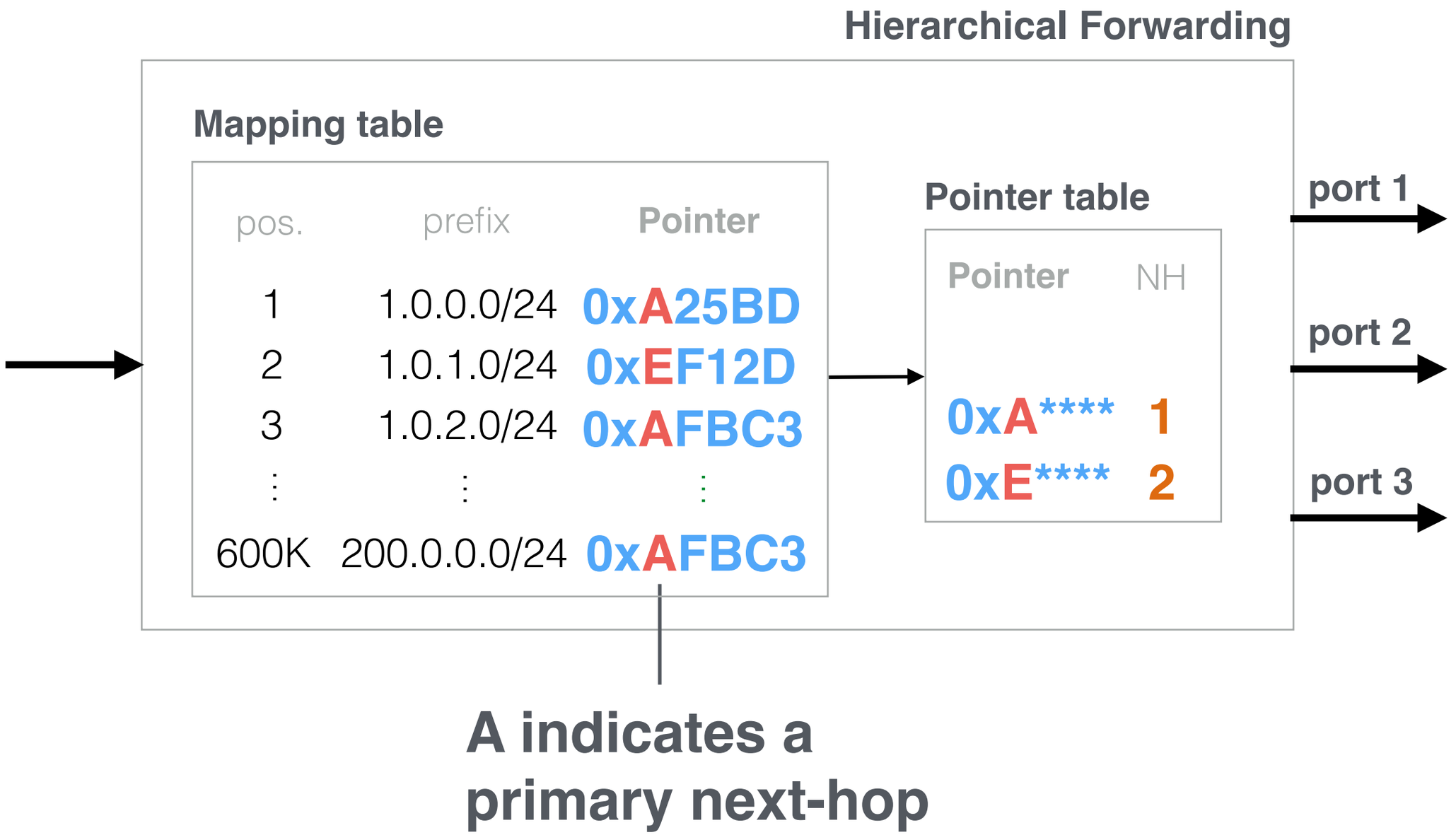
SWIFT uses custom pointers



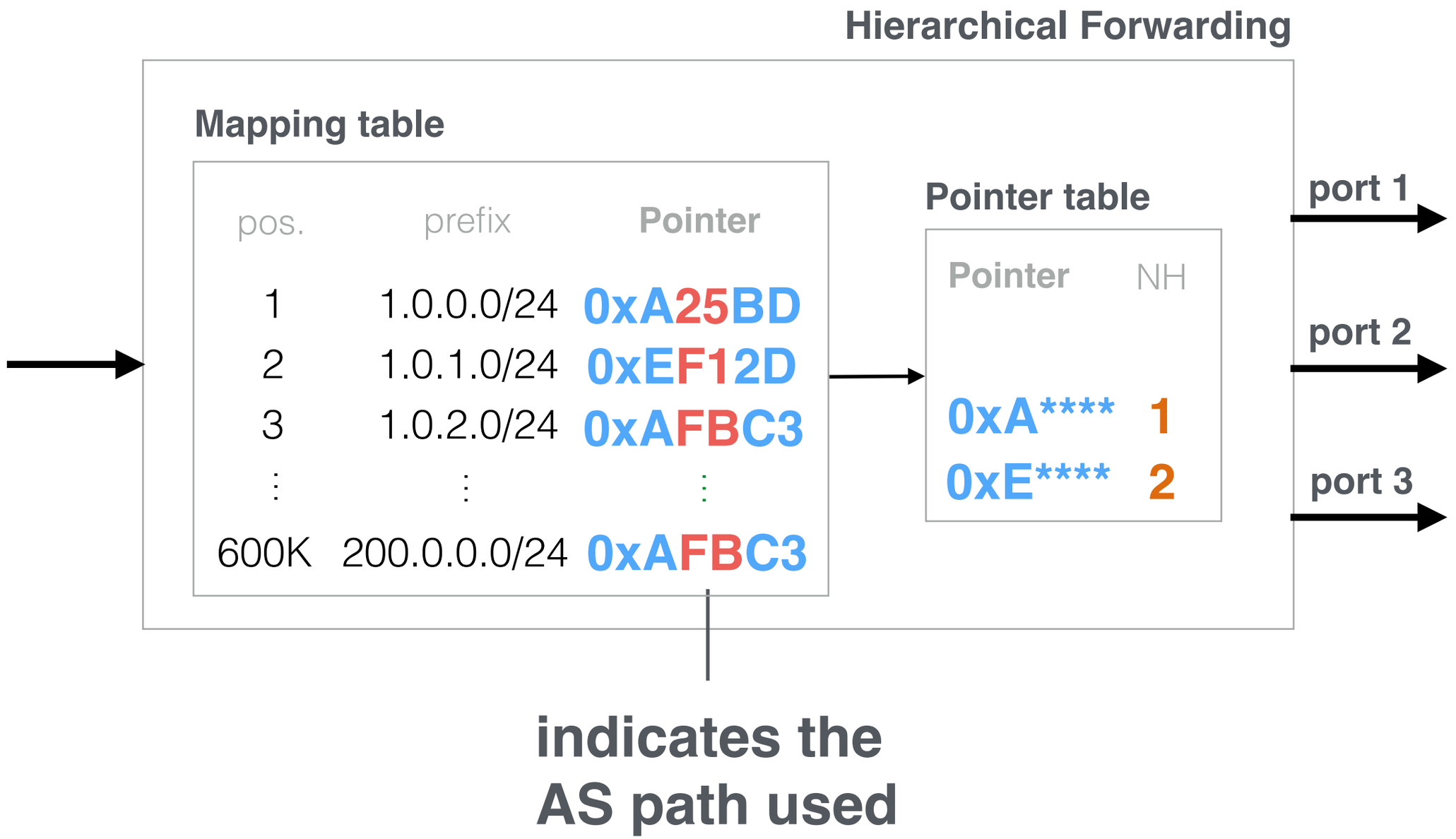
SWIFT uses custom pointers



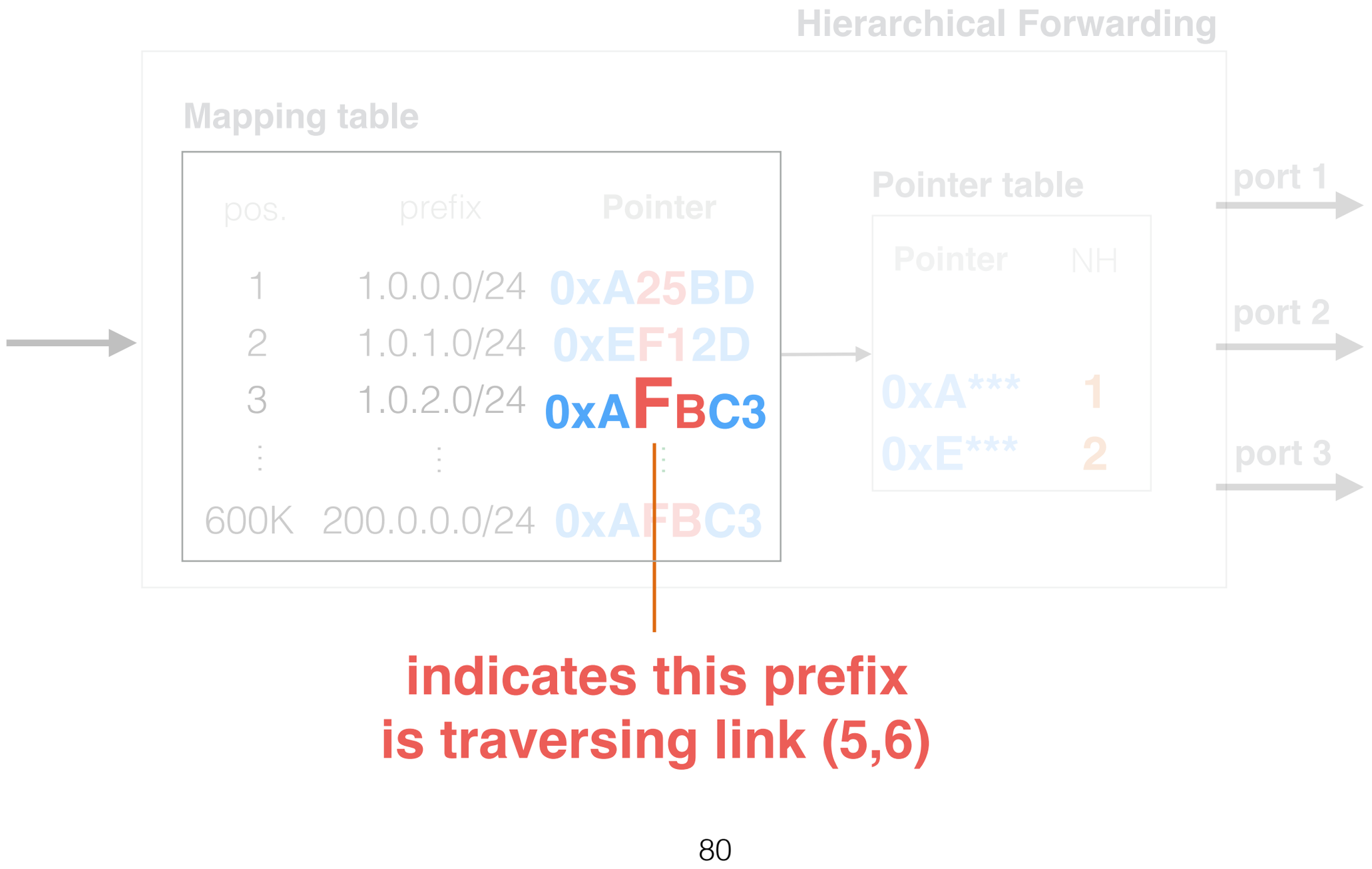
SWIFT uses custom pointers



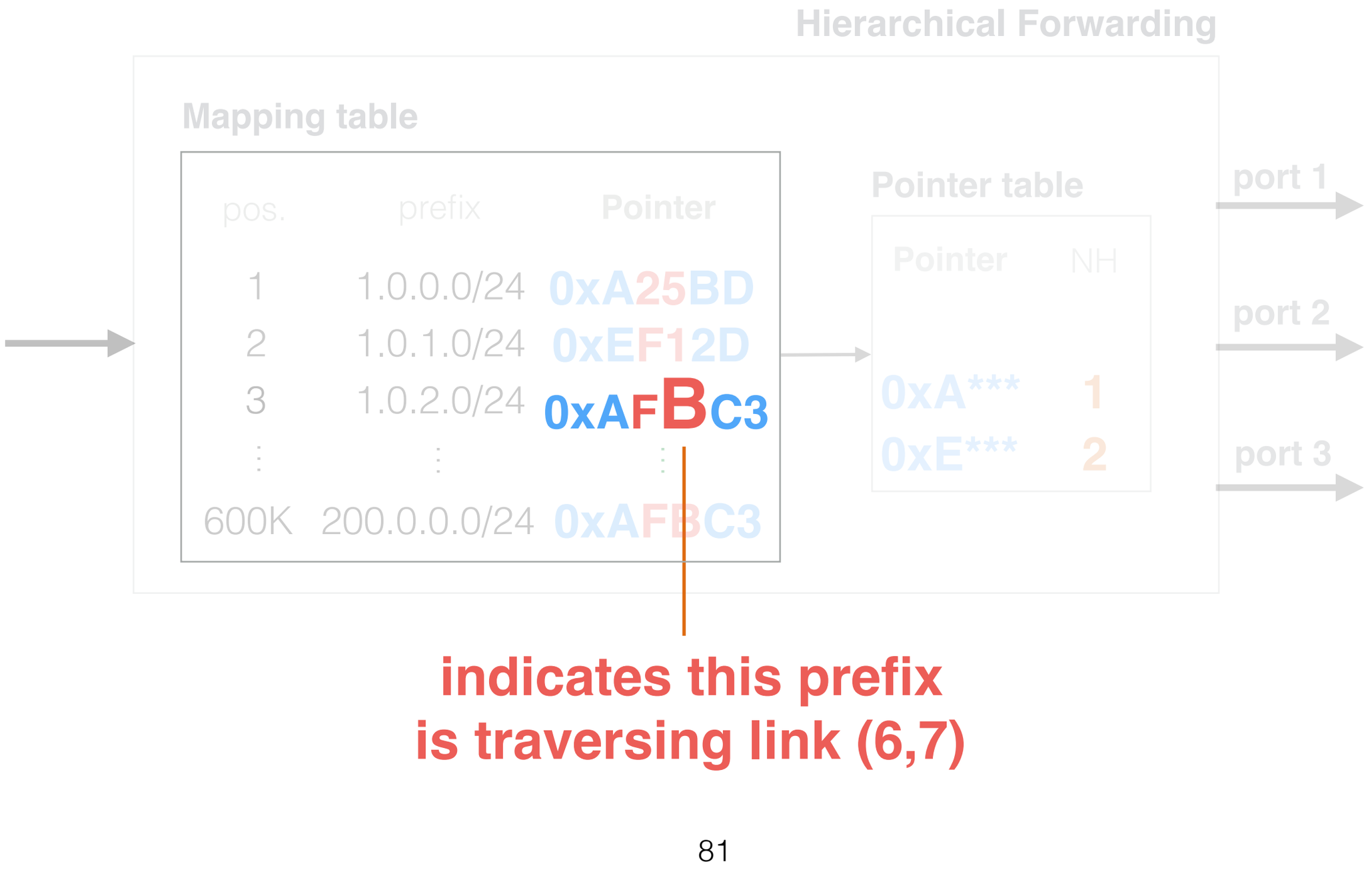
SWIFT uses custom pointers



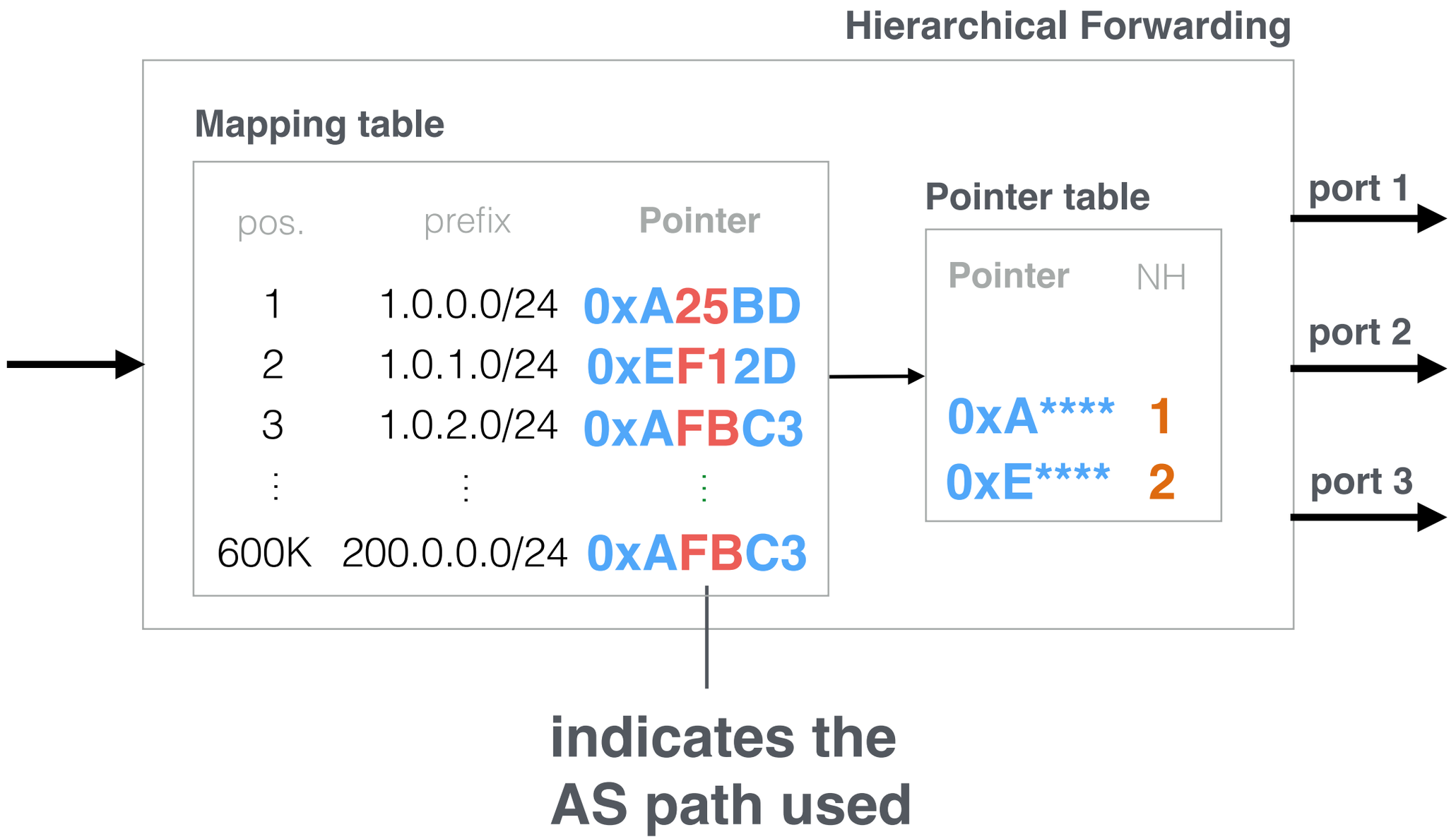
SWIFT uses custom pointers



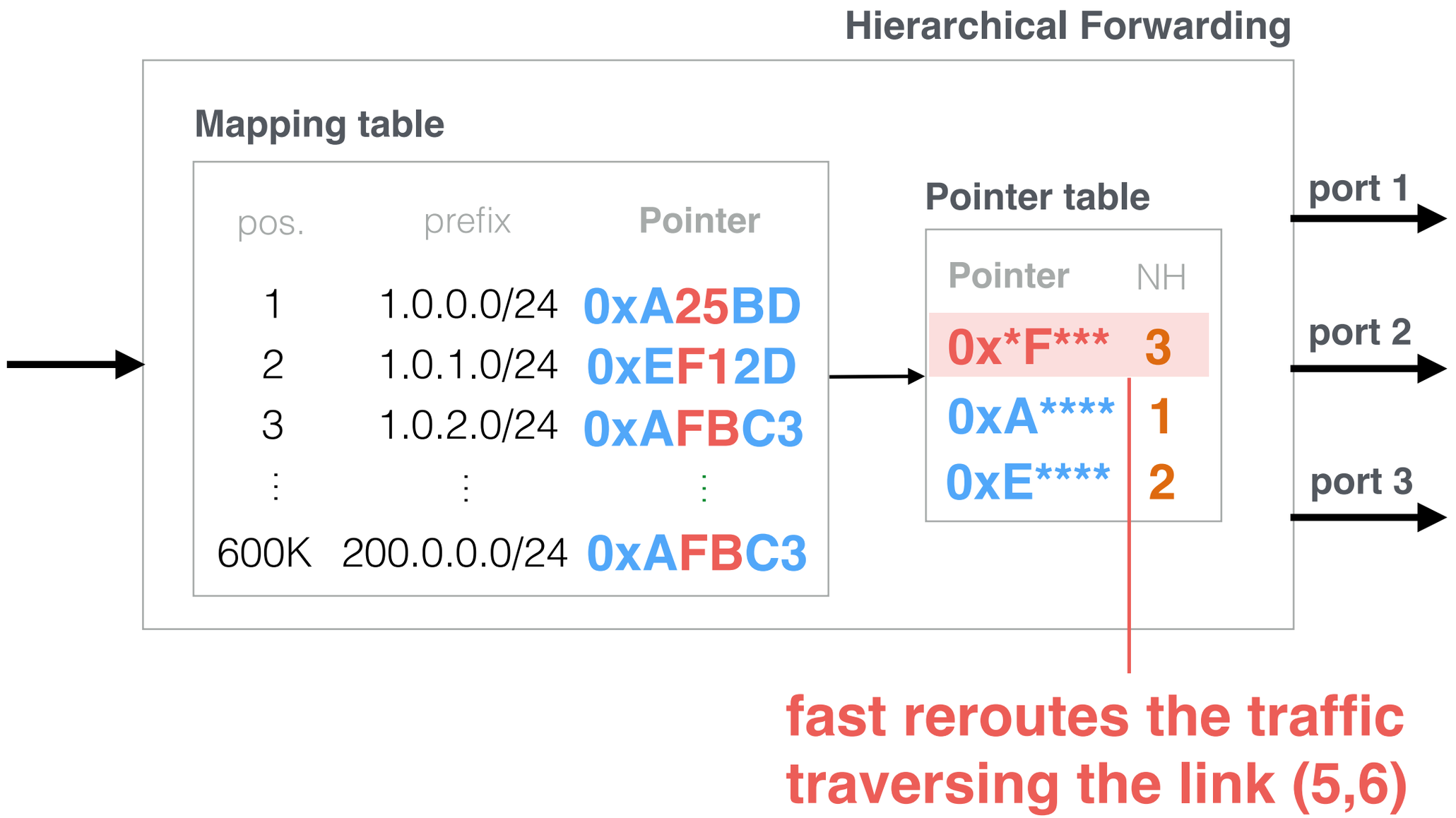
SWIFT uses custom pointers



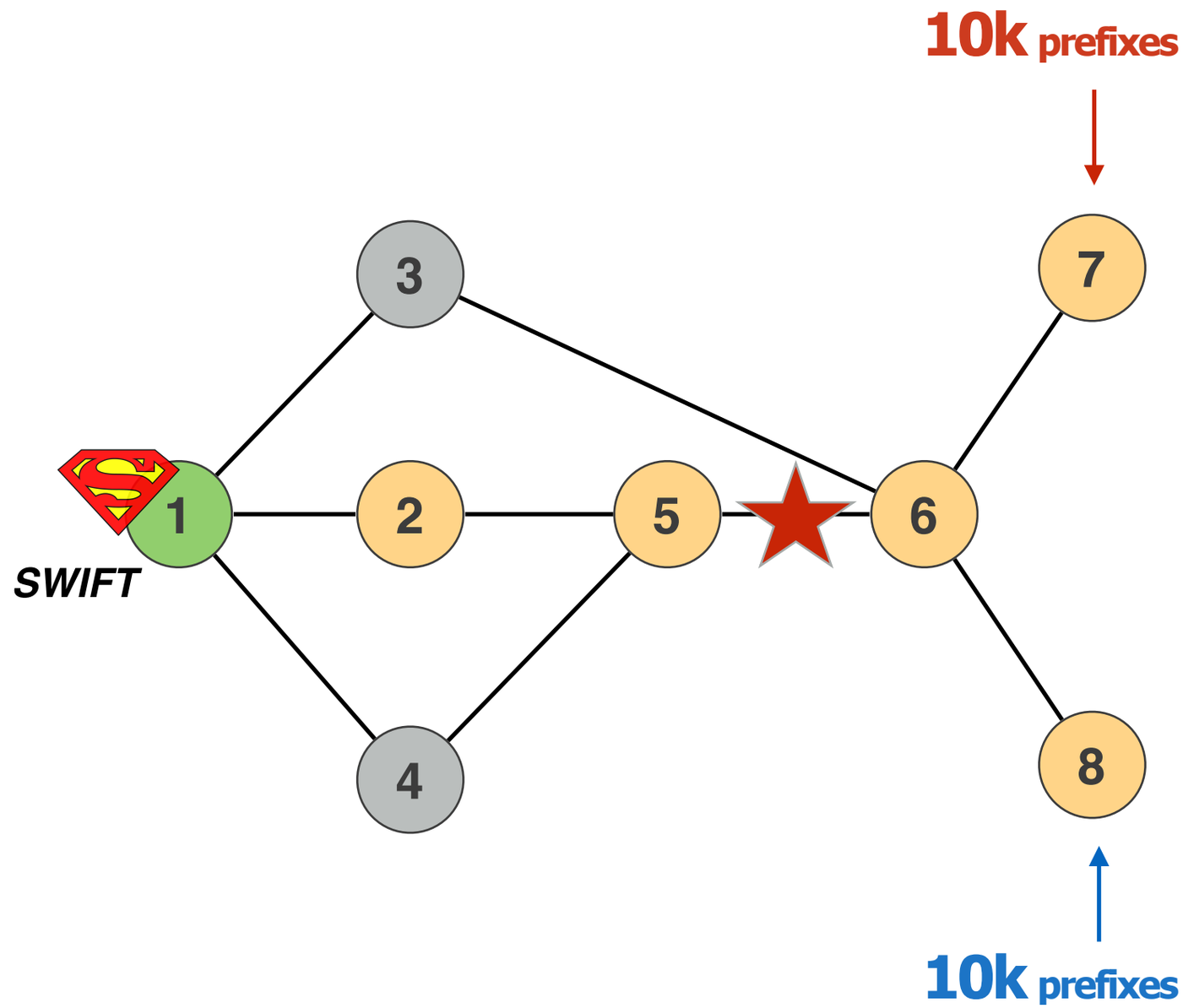
SWIFT uses custom pointers



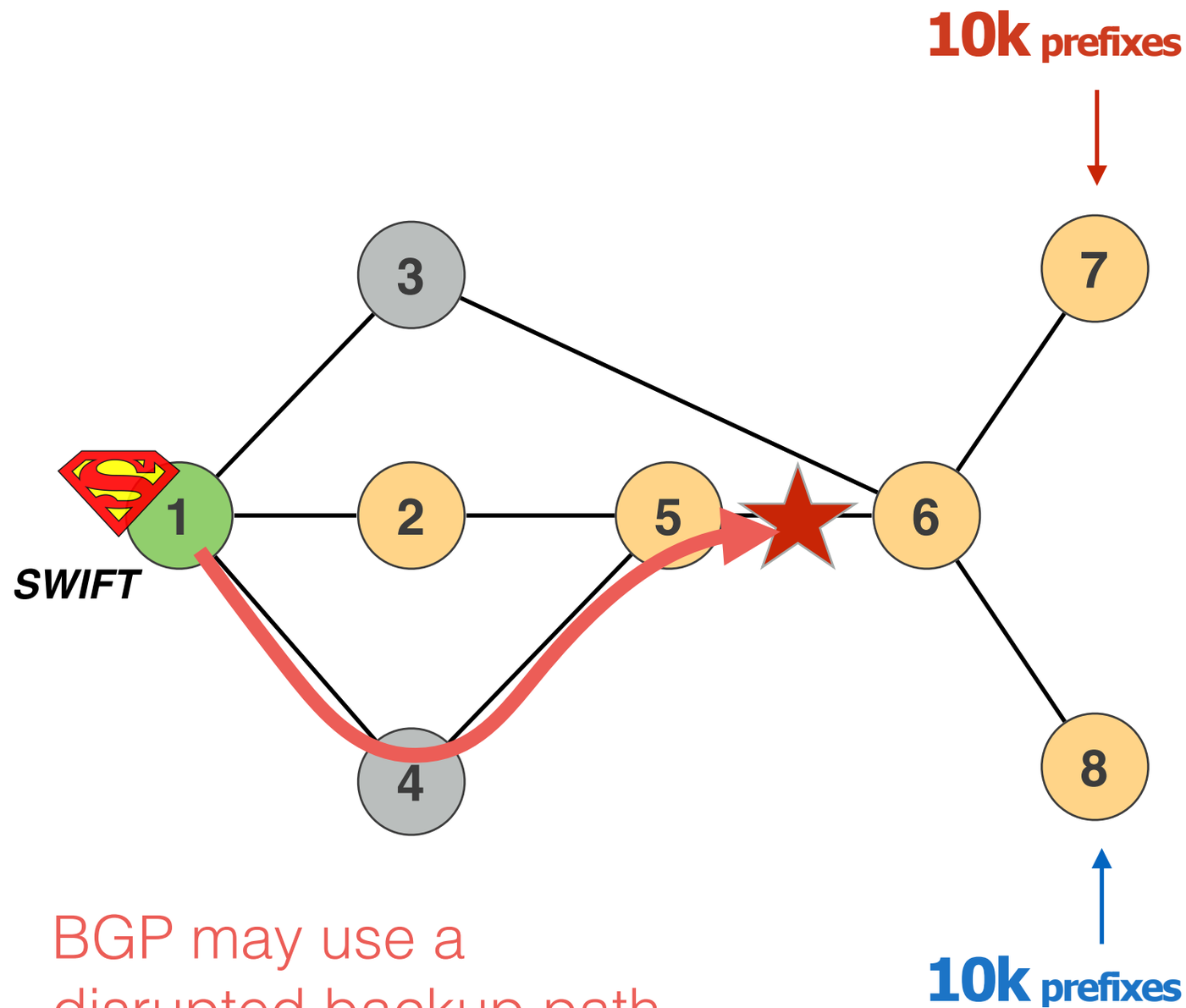
SWIFT uses custom pointers



SWIFT reroutes the affected traffic
to unaffected backup paths

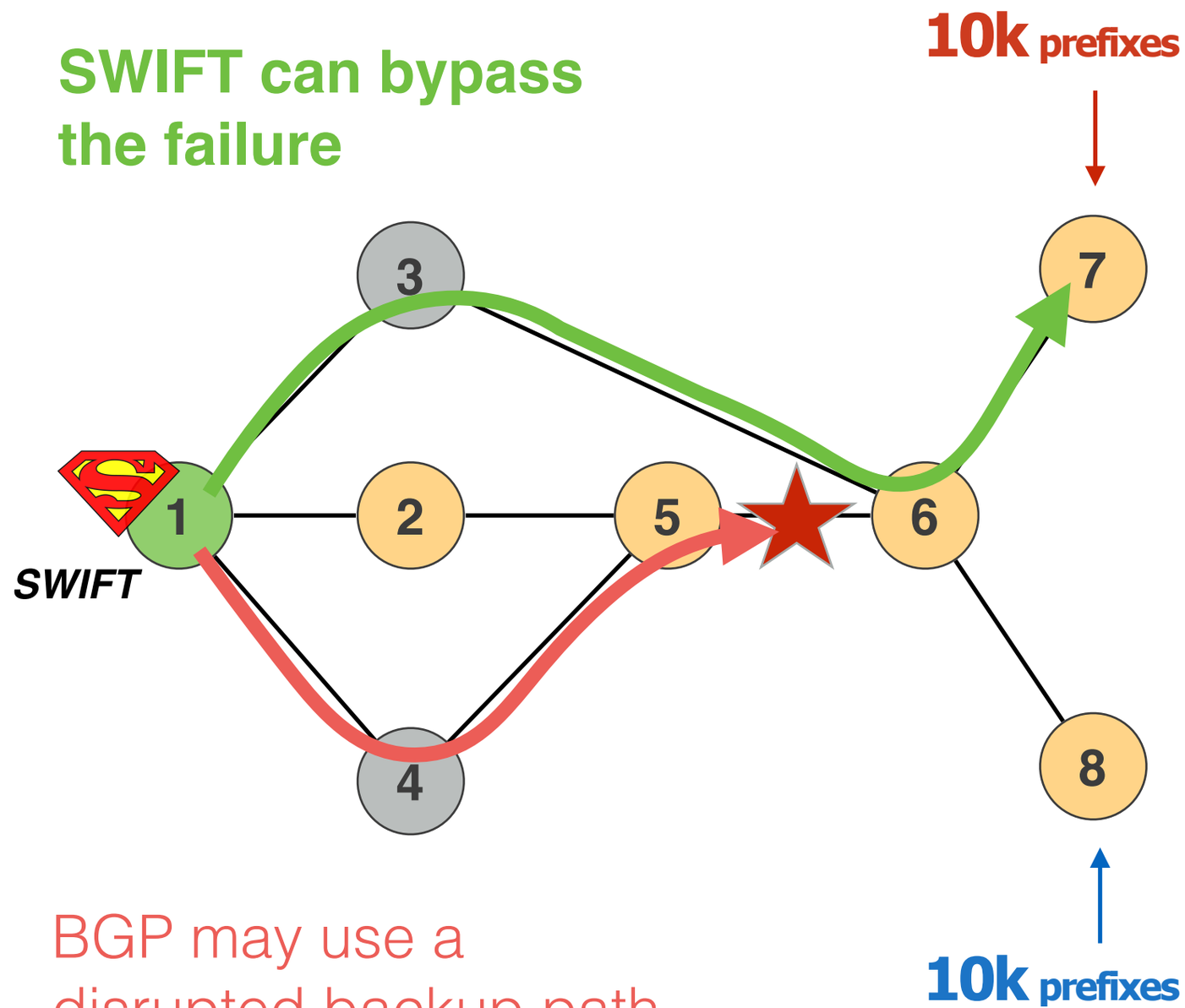


SWIFT reroutes the affected traffic to unaffected backup paths



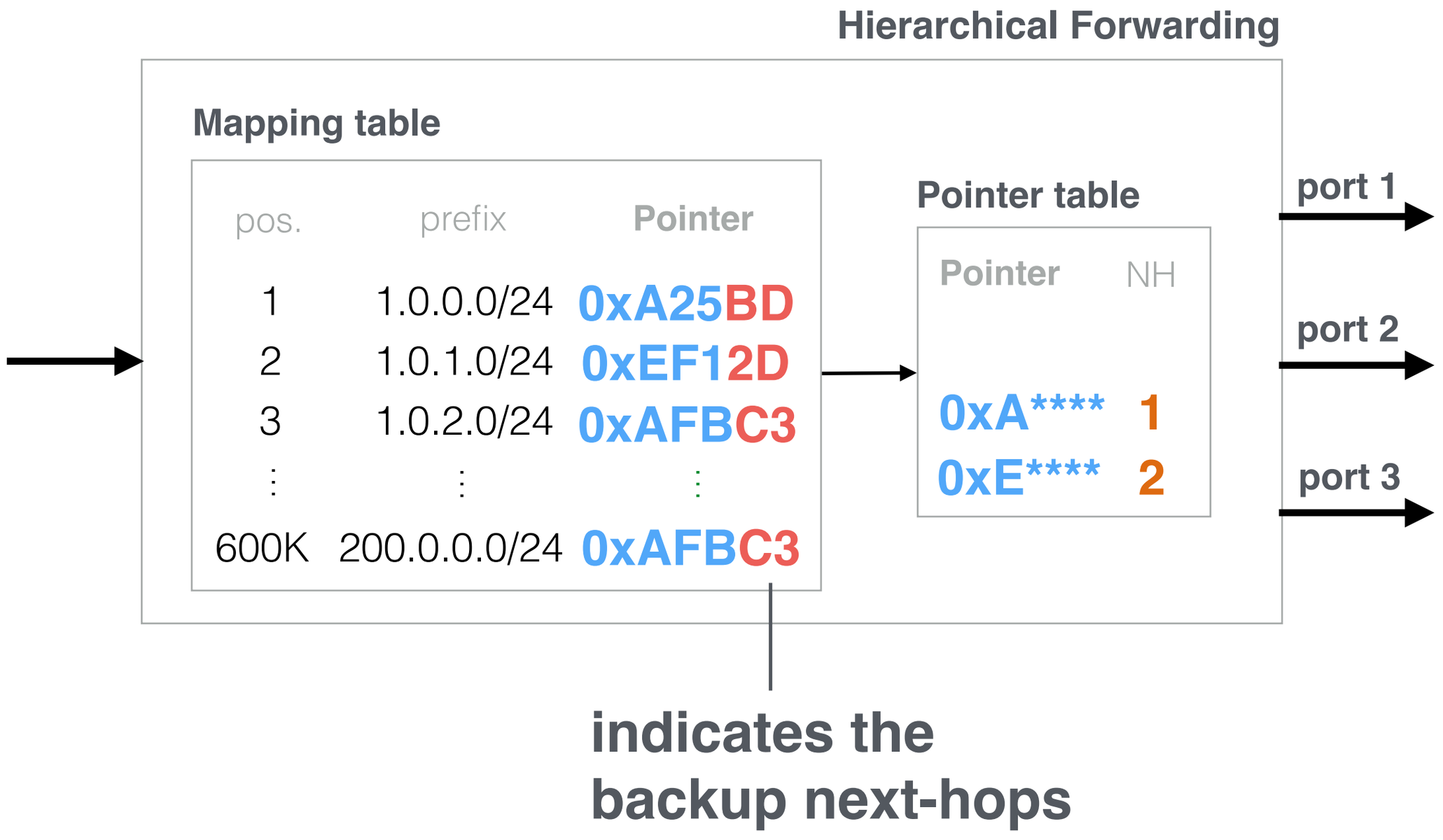
BGP may use a disrupted backup path

SWIFT reroutes the affected traffic
to unaffected backup paths

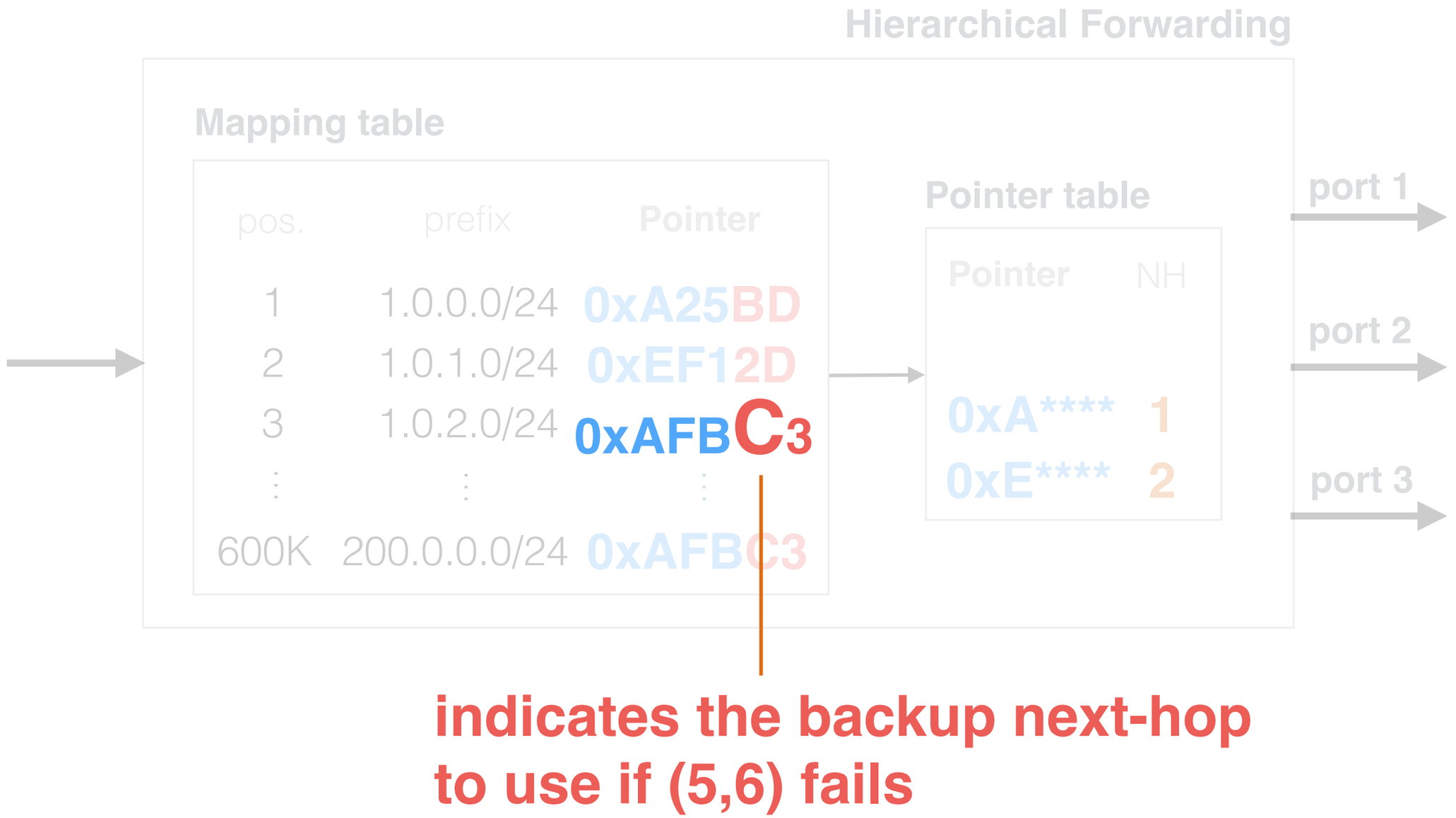


BGP may use a
disrupted backup path

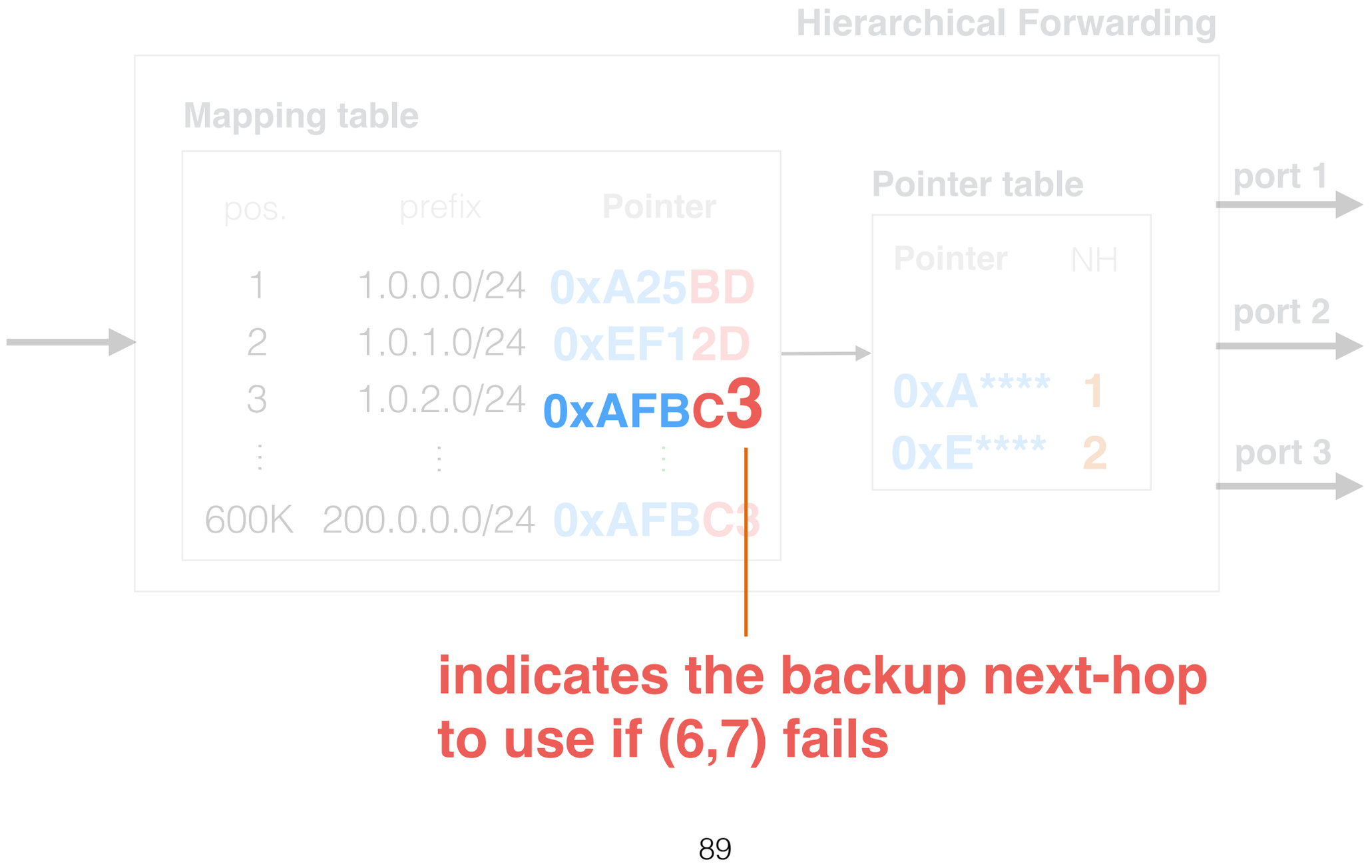
SWIFT encodes the next-hops in the pointers too



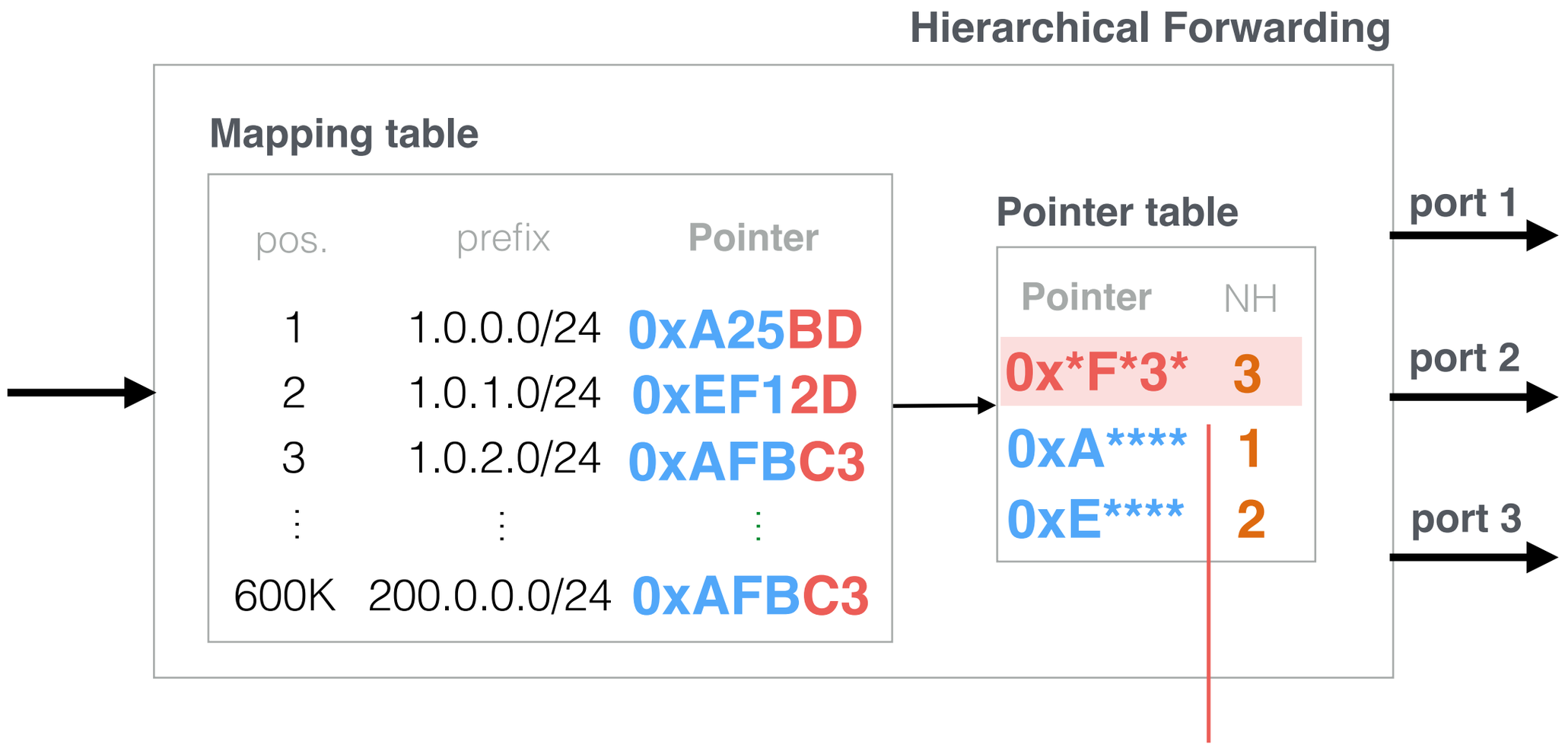
SWIFT encodes the next-hops in the pointers too



SWIFT encodes the next-hops in the pointers too



SWIFT encodes the next-hops in the pointers too



Pointers are data-plane tags with limited size

Pointers are data-plane tags with limited size

as-path encoding

primary and backup next-hops

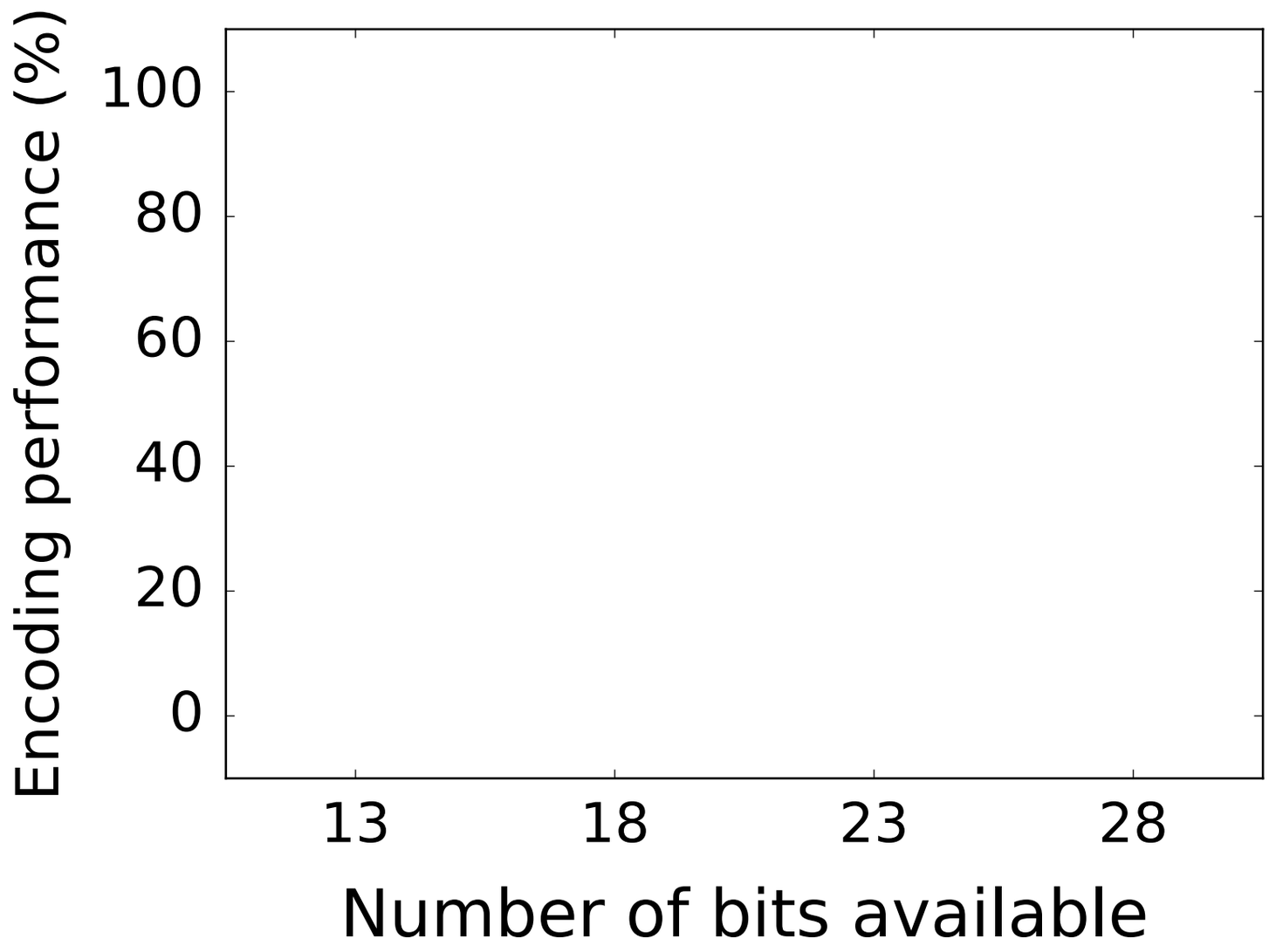
Pointers are data-plane tags with limited size

as-path encoding

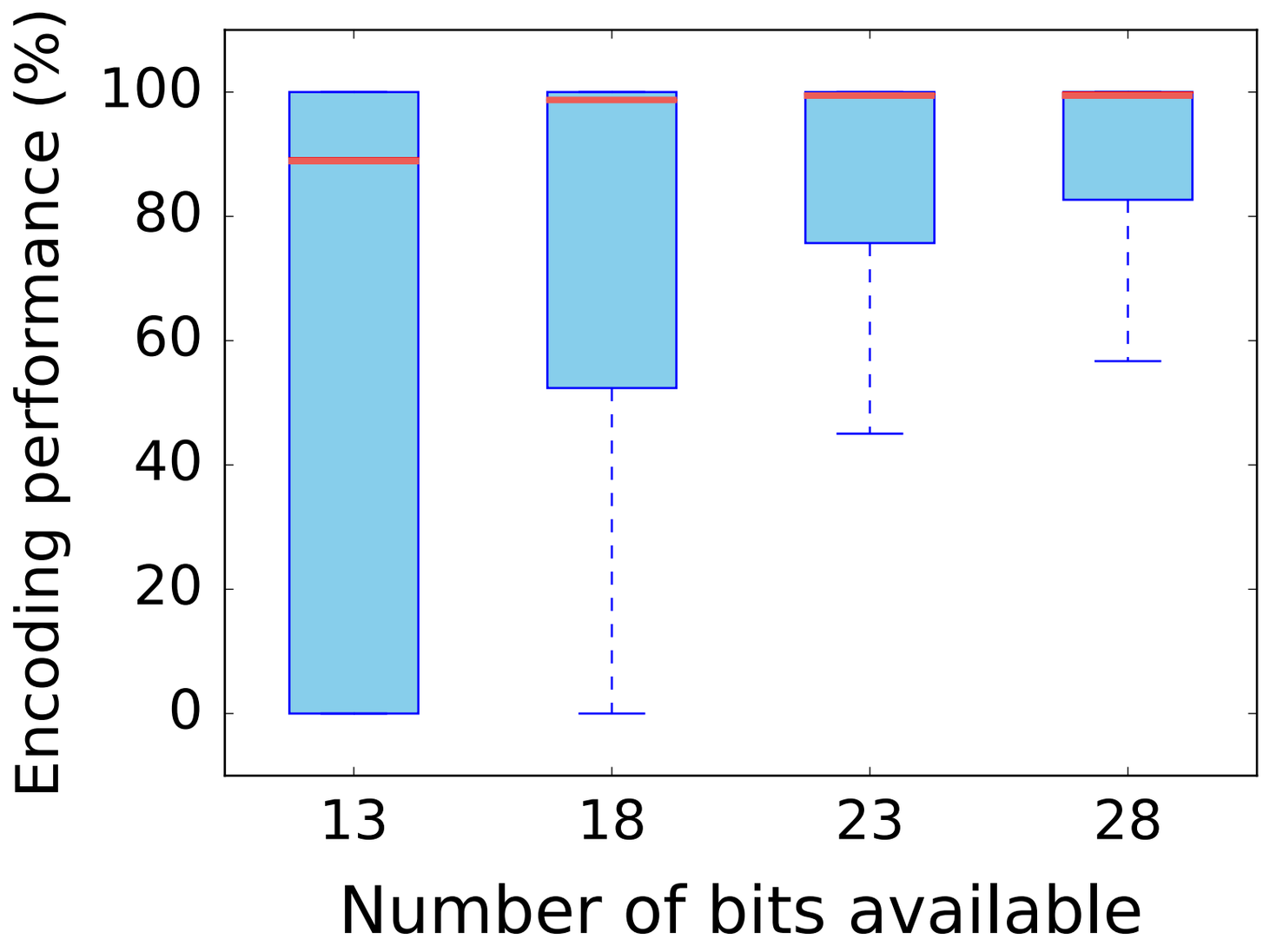
primary and backup next-hops

SWIFT only encodes the most important AS links

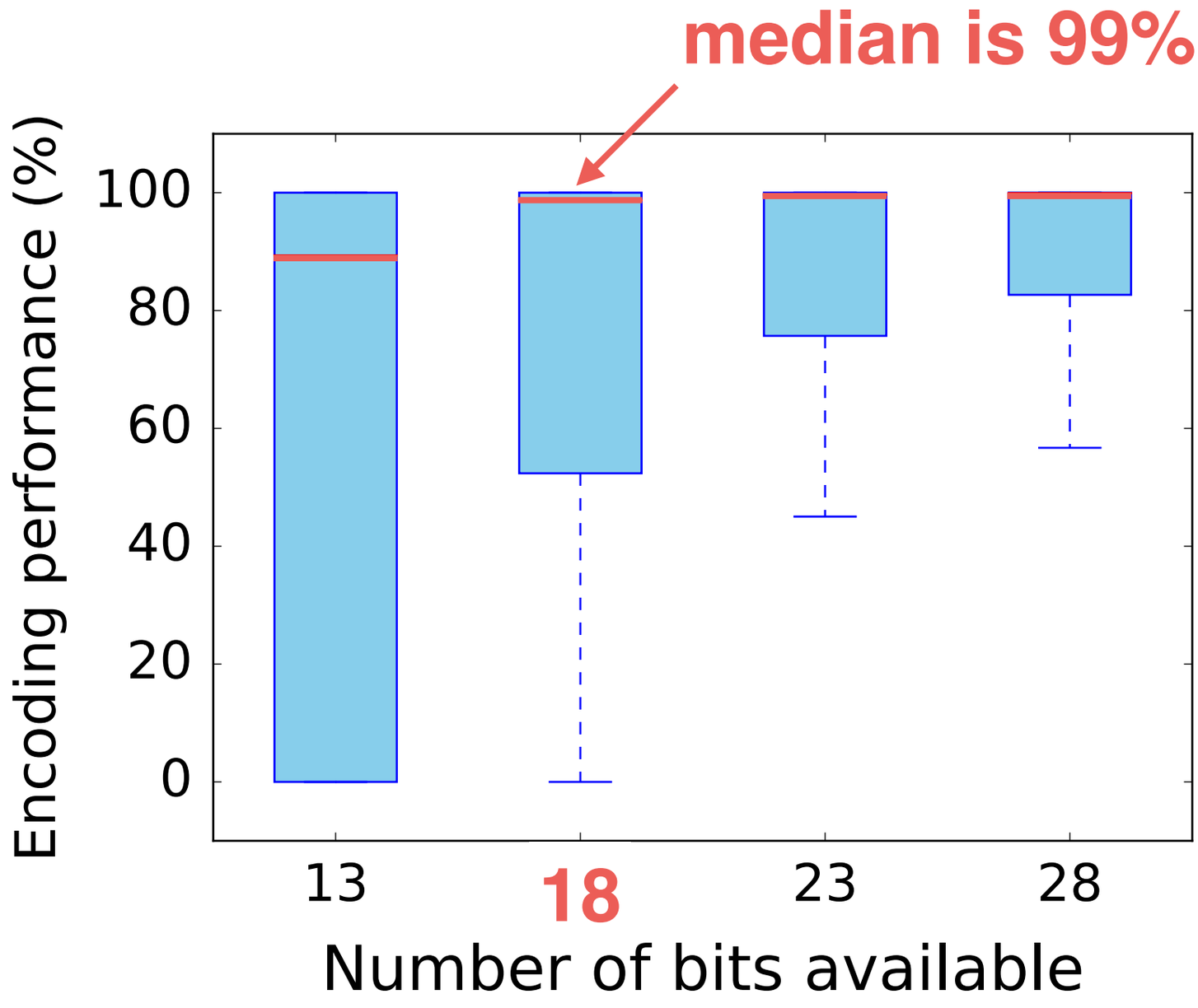
SWIFT as path encoding algorithm enables to reroute most of the affected prefixes, with few bits only



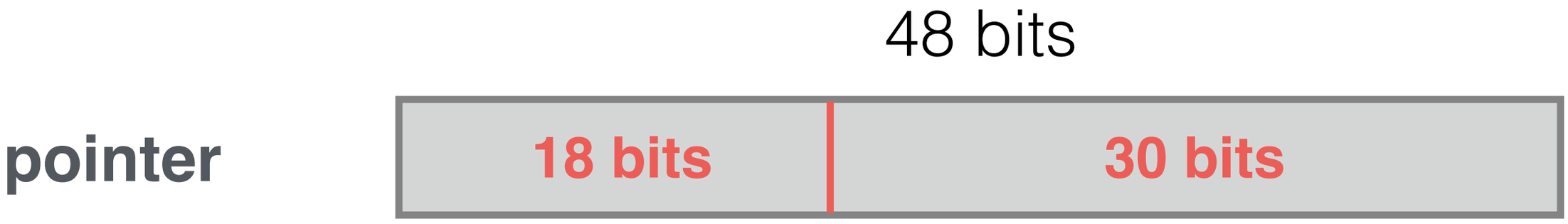
SWIFT as path encoding algorithm enables to reroute most of the affected prefixes, with few bits only



SWIFT as path encoding algorithm enables to reroute most of the affected prefixes, with few bits only



Pointers are data-plane tags with limited size



as-path encoding

primary and backup next-hops

SWIFT requires less than 100 forwarding updates to reroute (with 16 backups)

SWIFT requires less than 100 forwarding updates to reroute (with 16 backups)

40 ms

SWIFT: Predictive Fast Reroute

1. ***SWIFT*** reduces the learning time of a withdrawal from 13s to 2s using a control-plane prediction algorithm

2. ***SWIFT*** fast reroutes the affected traffic towards unaffected paths in 40ms using a hierarchical forwarding table

3. ***SWIFT*** is deployable on existing routers

SWIFT: Predictive Fast Reroute

1. ***SWIFT*** reduces the learning time of a withdrawal from 13s to 2s using a control-plane prediction algorithm

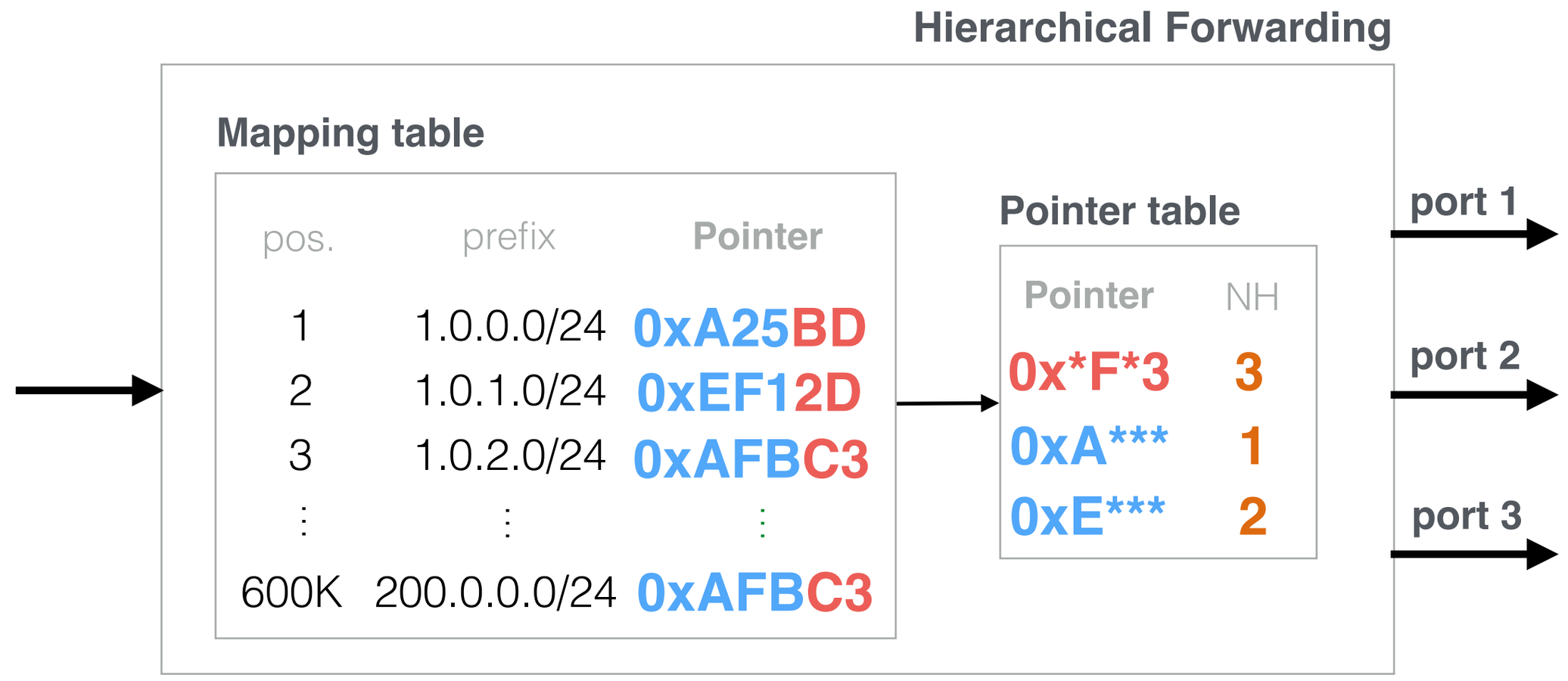
2. ***SWIFT*** fast reroutes the affected traffic towards unaffected paths in 40ms using a hierarchical forwarding table

3. ***SWIFT*** is deployable on existing routers

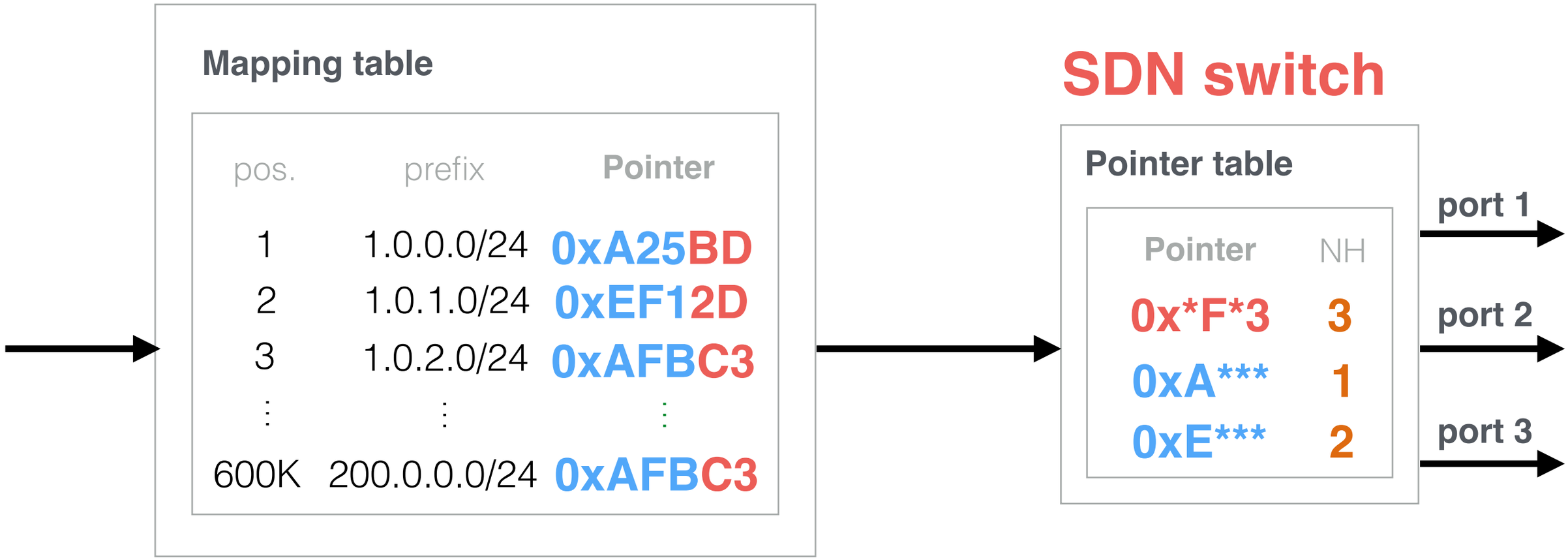
SWIFT is deployable on existing routers equipped with a hierarchical forwarding table

SWIFT is deployable on existing routers equipped with a hierarchical forwarding table

what if we don't have one though?


Hierarchical Forwarding
Mapping table
pos.
prefix
Pointer
1
1.0.0.0/24
0xA25BD
2
1.0.1.0/24
0xEF12D
3
1.0.2.0/24
0xAFBC3
600K
200.0.0.0/24
0xAFBC3
Pointer table
Pointer
NH
0x*F*3
3
0xA***
1
0xE***
2
port 1
port 2
port 3

IP Router


Mapping table
pos.
prefix
Pointer
1
1.0.0.0/24
0xA25BD
2
1.0.1.0/24
0xEF12D
3
1.0.2.0/24
0xAFBC3
600K
200.0.0.0/24
0xAFBC3
SDN switch
Pointer table
Pointer
NH
0x*F*3
3
0xA***
1
0xE***
2
port 1
port 2
port 3

IP Router

Mapping table

pos.	prefix	Dest. MAC
1	1.0.0.0/24	0xA25BD
2	1.0.1.0/24	0xEF12D
3	1.0.2.0/24	0xAFBC3
⋮	⋮	⋮
600K	200.0.0.0/24	0xAFBC3

SDN switch

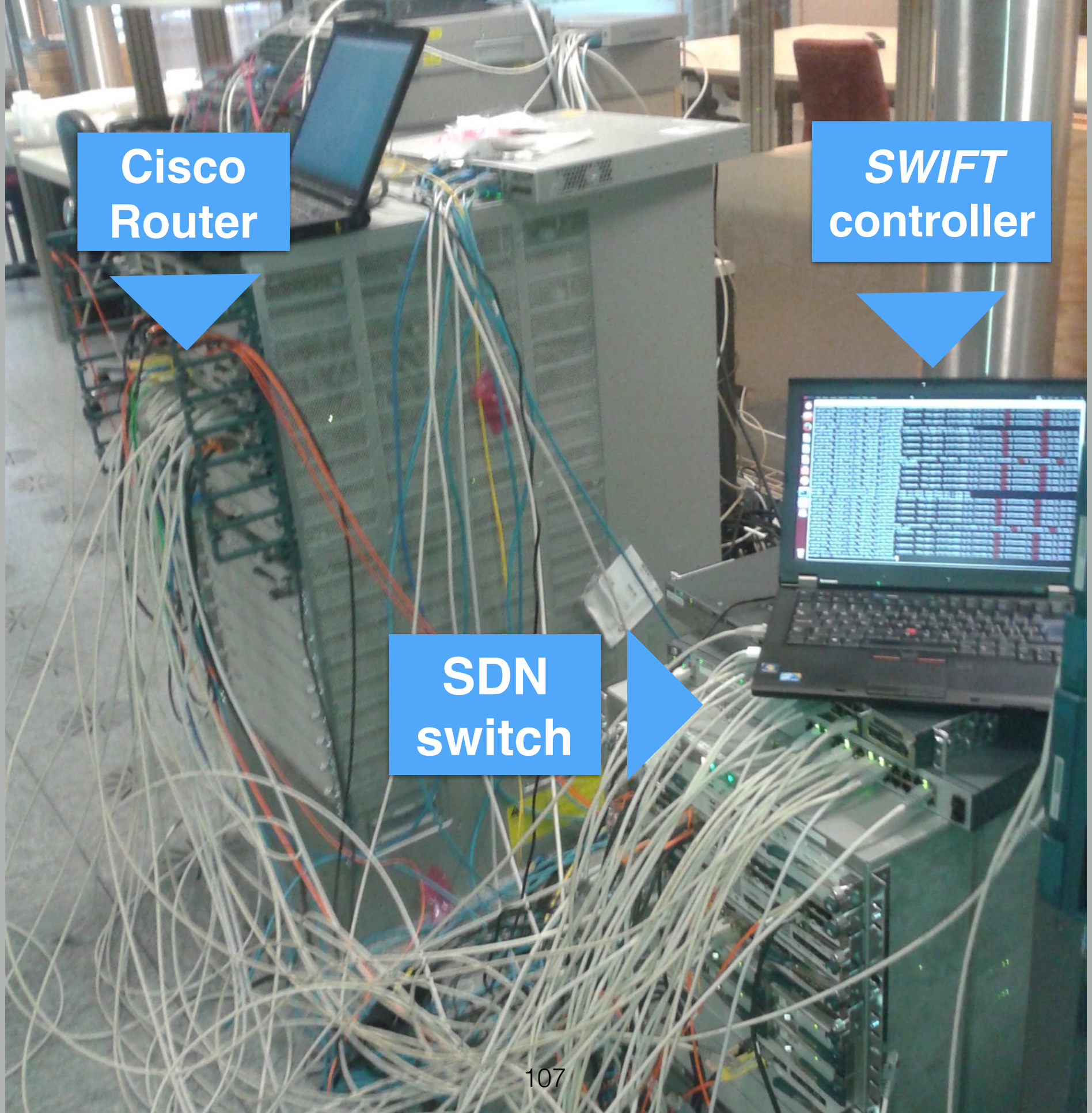
Pointer table

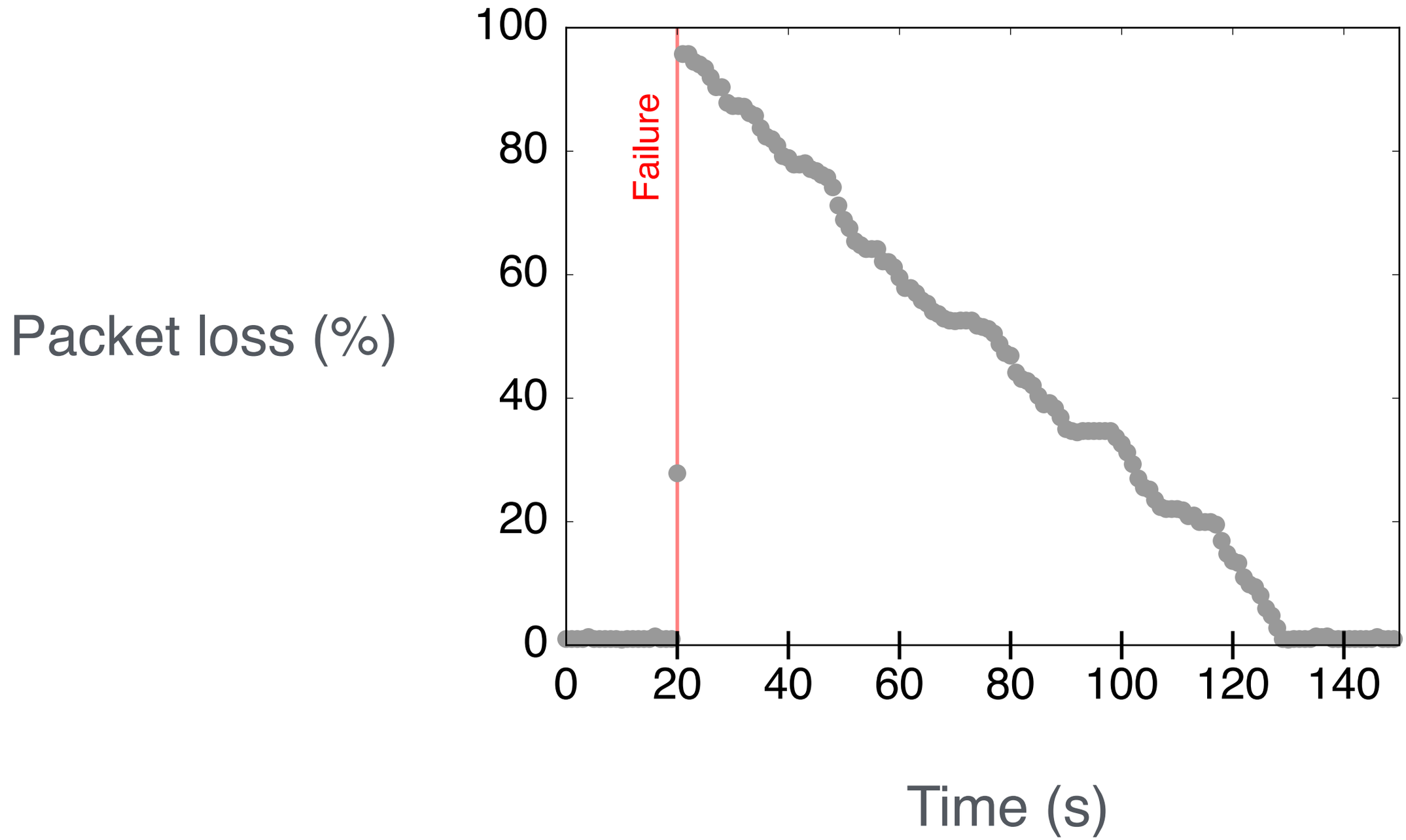
Dest. MAC	NH
0x*F*3	3
0xA***	1
0xE***	2

port 1

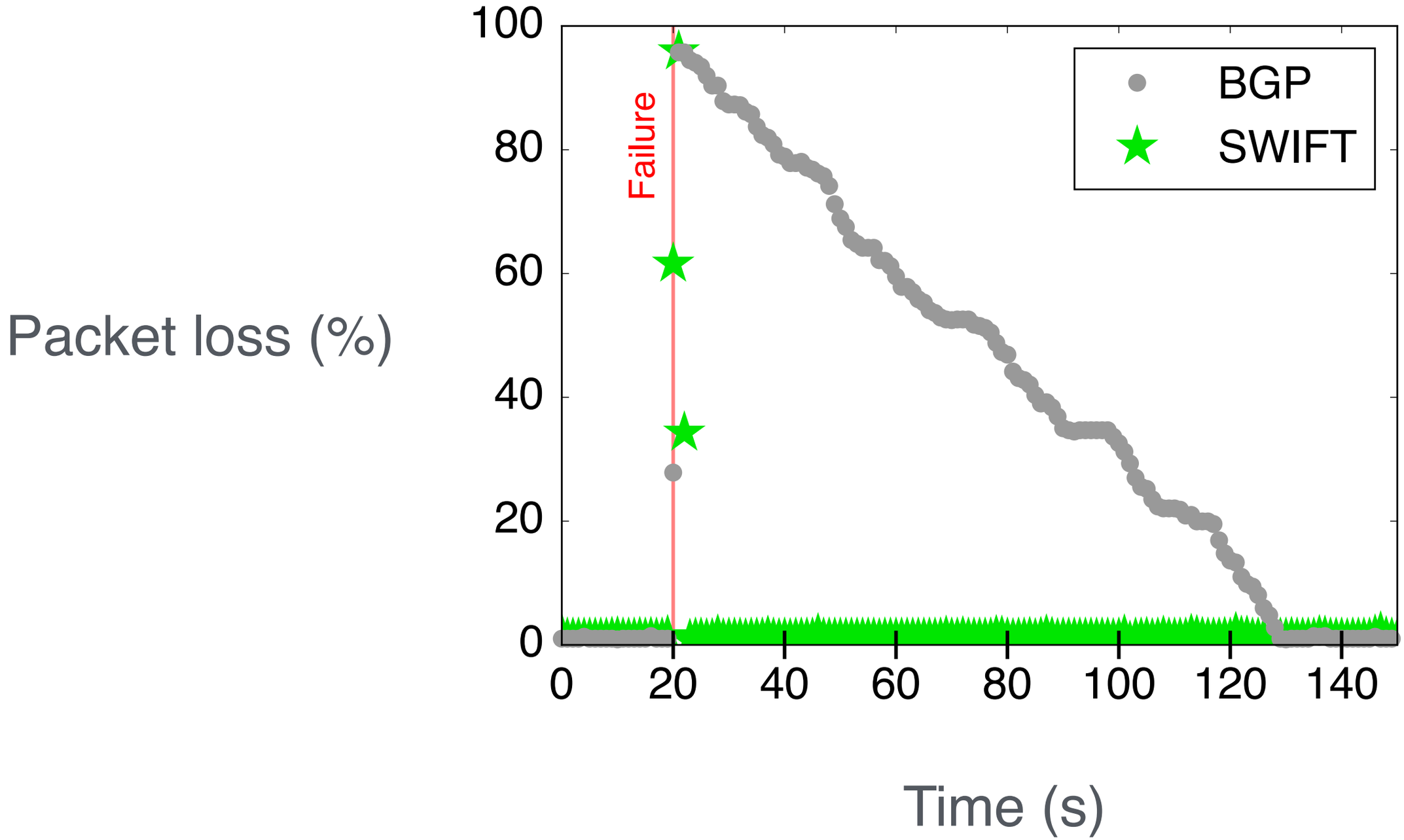
port 2

port 3


Cisco
Router
SWIFT
controller
SDN
switch

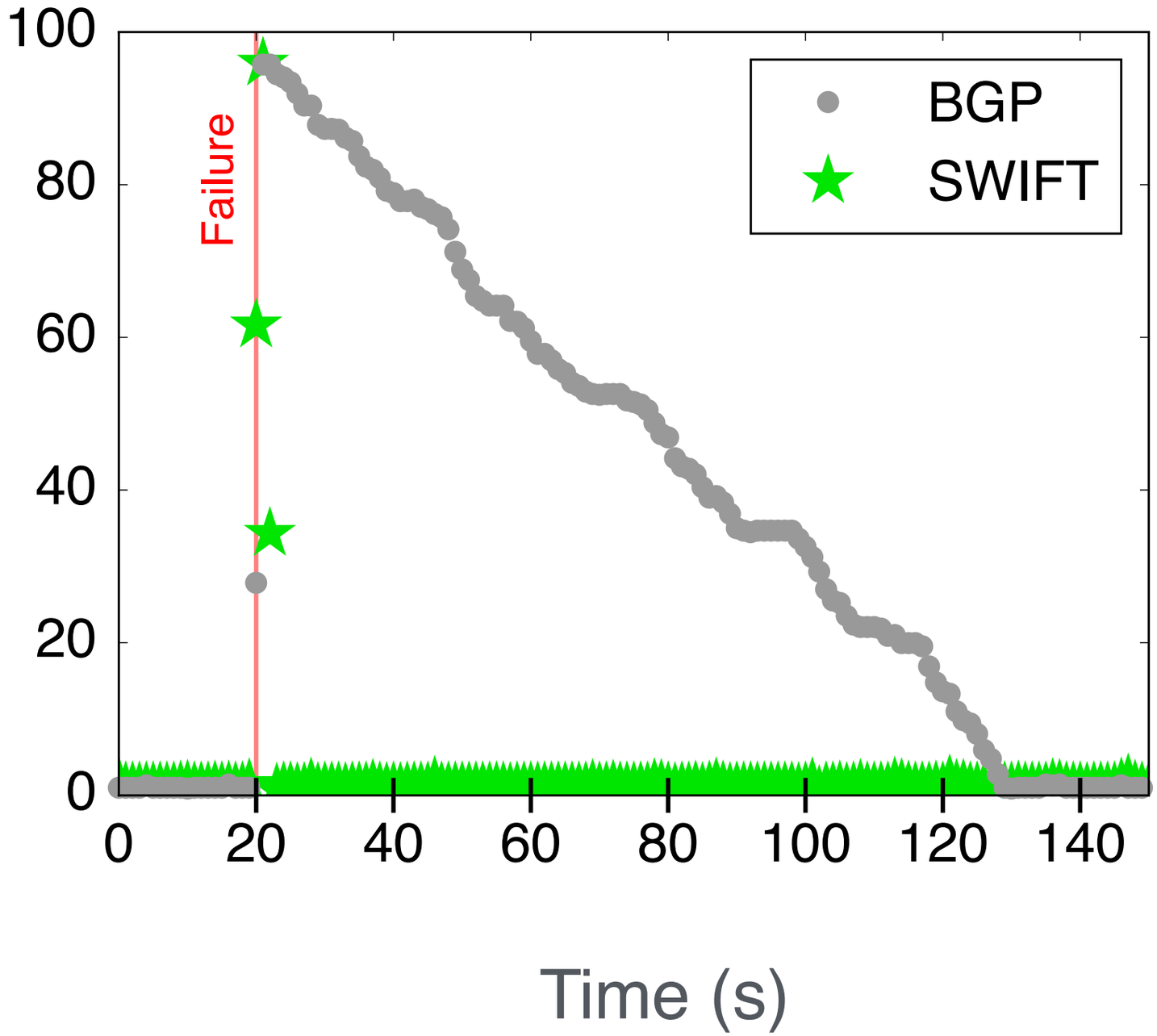

Packet loss (%)
100
80
60
40
20
0
Failure
0
20
40
60
80
100
120
140
Time (s)

SWIFT reduces the convergence time from ~110s to 2s



SWIFT reduces the convergence time from ~110s to 2s

98% speed up improvement



SWIFT: *Predictive Fast Reroute*

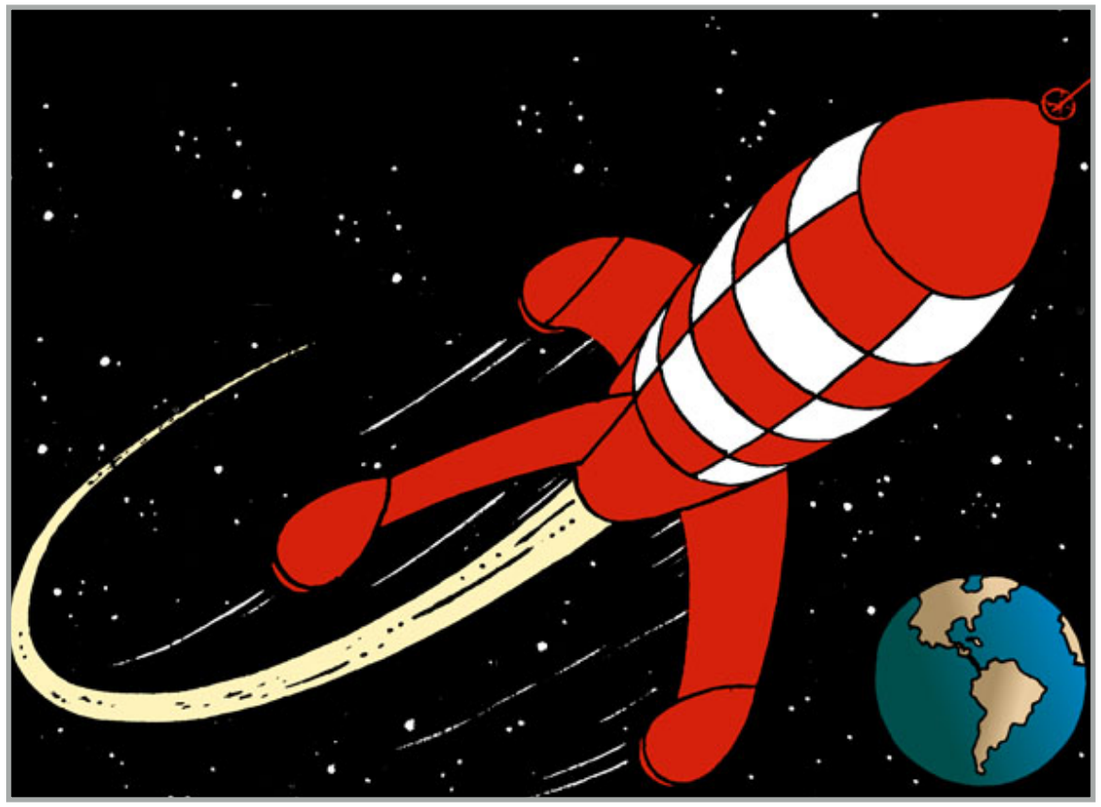
Speeds up the learning phase by predicting the control-plane
85% accuracy early in the burst

Quickly fast reroutes the affected traffic to unaffected paths
for 99% of the prefixes

Is deployable on existing routers
98% speed up improvement

swift.ethz.ch (source code, VM + demo!)

***SWIFT**: Predictive Fast Reroute*



Thomas Holterbach
ETH Zürich / CAIDA

SIGCOMM
24th August 2017

swift.ethz.ch

Joint work with

Stefano Vissicchio **UCLondon**
Alberto Dainotti **CAIDA, UC San Diego**
Laurent Vanbever **ETH Zürich**