SWIFT

Predictive Fast Reroute upon Remote BGP Disruptions

The Fortune Teller by Caravaggio, 1595. Louvre, Paris

Laurent Vanbever

Networked Systems, D-ITET

D-INFK Lunch Seminar

December 5 2016

25.9 seconds

25.9 seconds

maximum monthly downtime

under a 99.999% SLA

25.9 seconds

maximum monthly downtime under a 99.999% SLA

>2.5 minutes

25.9 seconds

maximum monthly downtime
under a 99.999% SLA

>2.5 minutes

measured router downtime
upon large Internet failures

25.9 seconds

maximum monthly downtime
under a 99.999% SLA

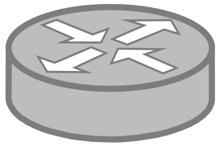
>2.5 minutes

measured router downtime
upon large Internet failures

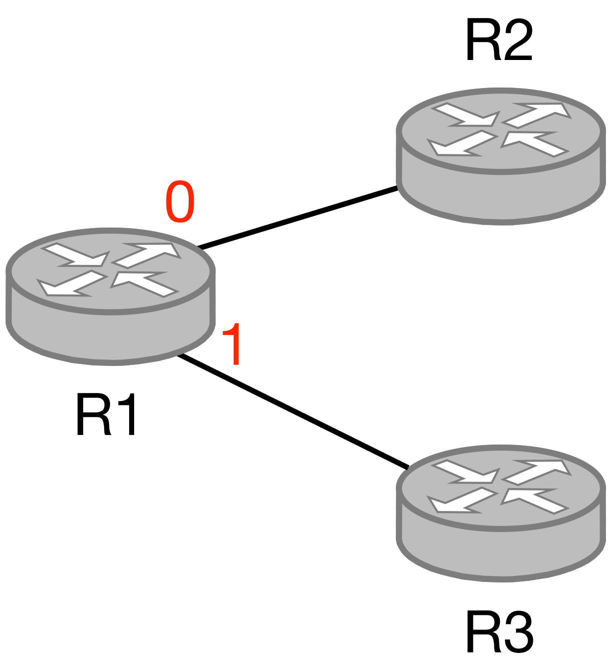
3 seconds

25.9 seconds	maximum monthly downtime under a 99.999% SLA
>2.5 minutes	measured router downtime upon large Internet failures
3 seconds	*SWIFT* measured downtime upon large Internet failures

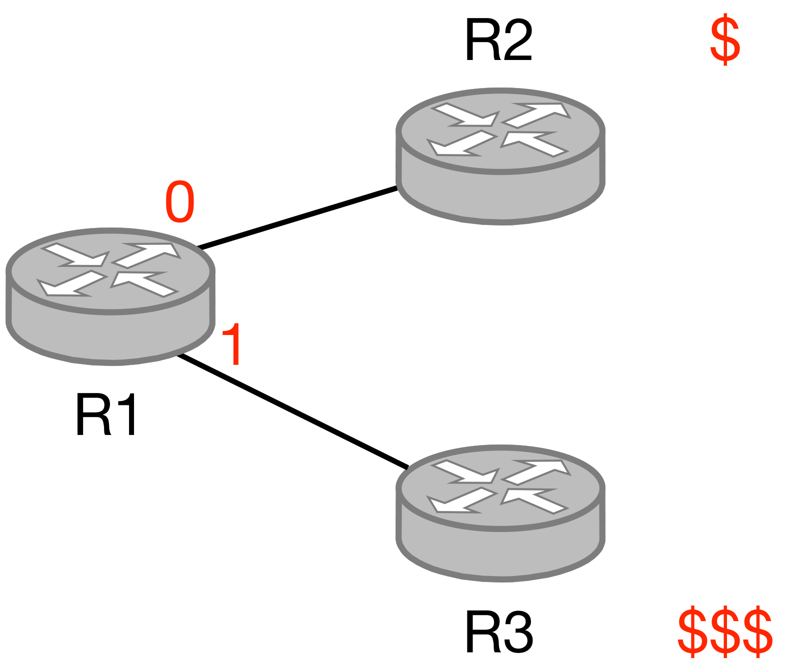
IP routers are slow to converge upon remote link and node failures



R1

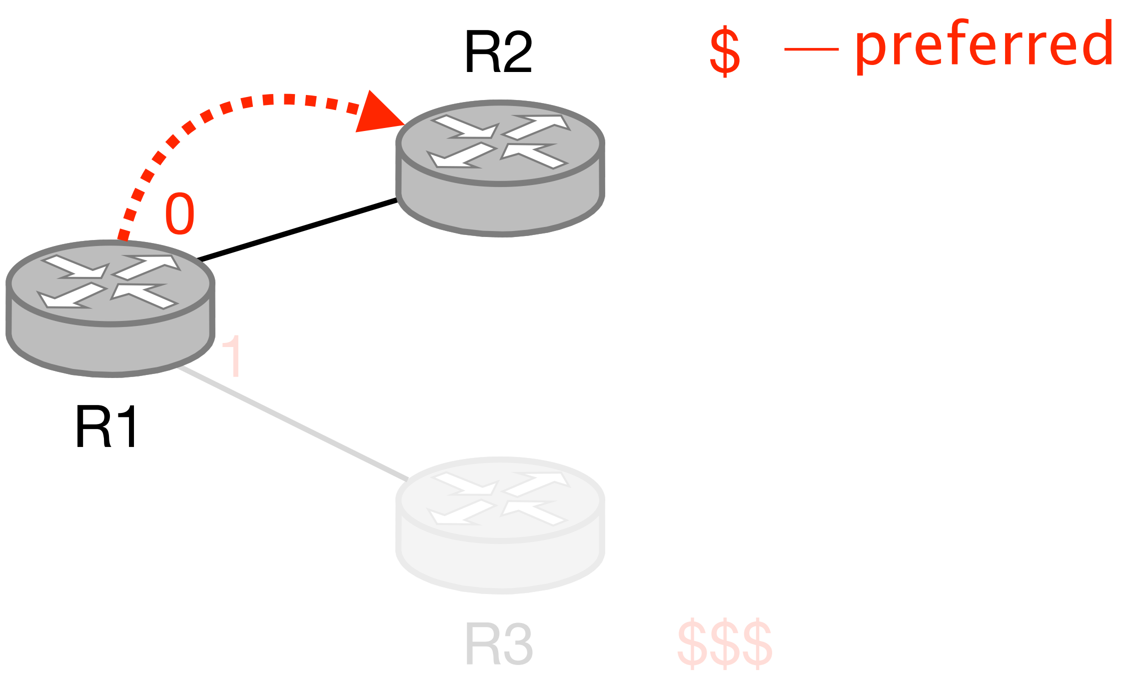

R2
0
1
R1
R3

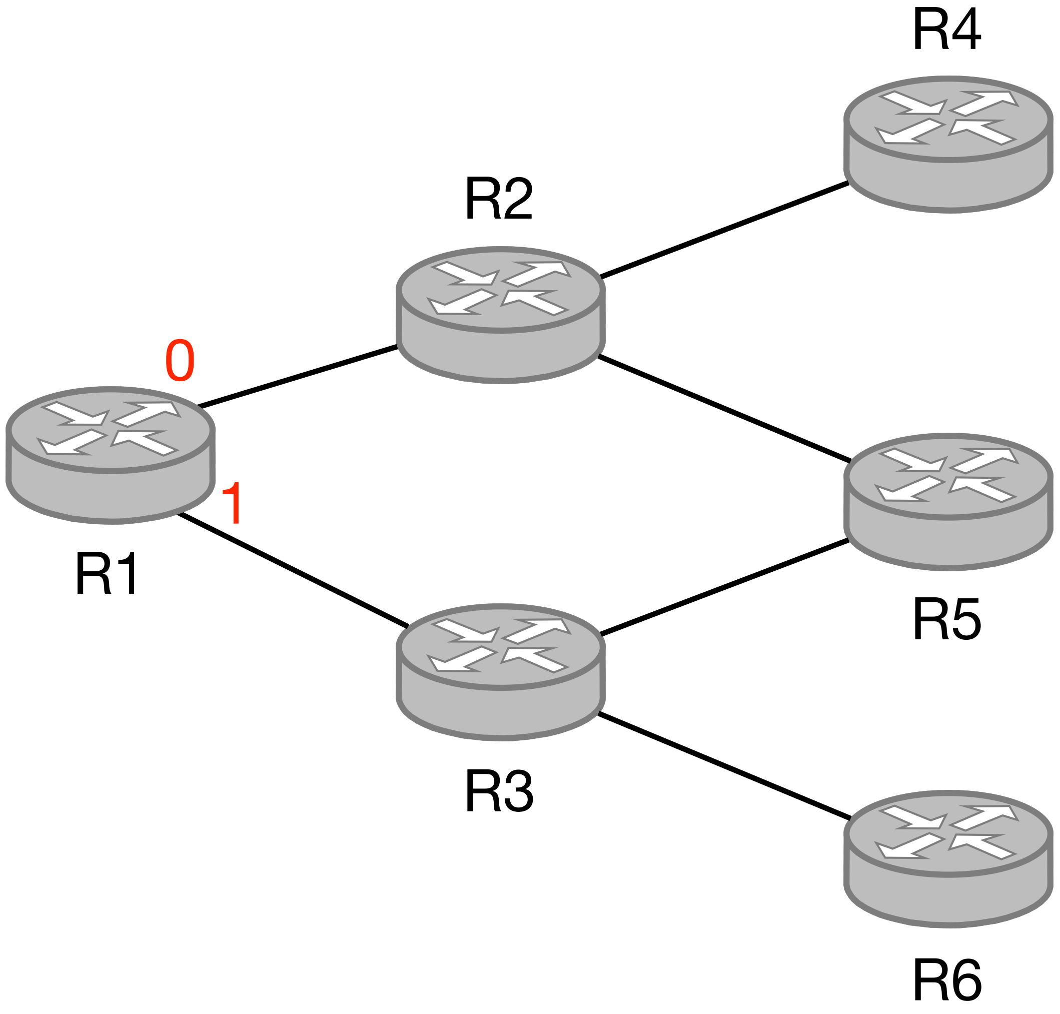
R1 prefers to send traffic via R2 when possible, as it is much cheaper than via R3

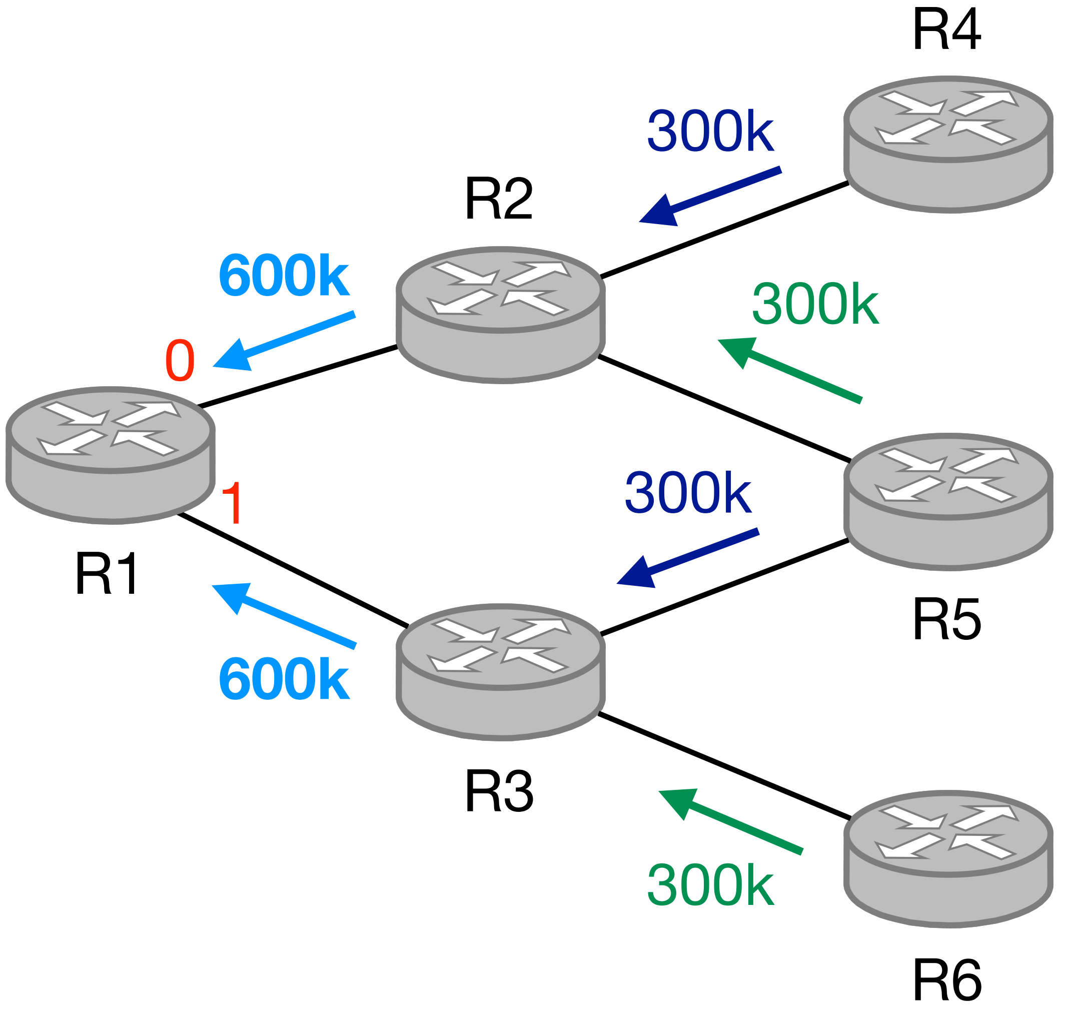


R1 forwards traffic to R2

for any destination R2 advertises

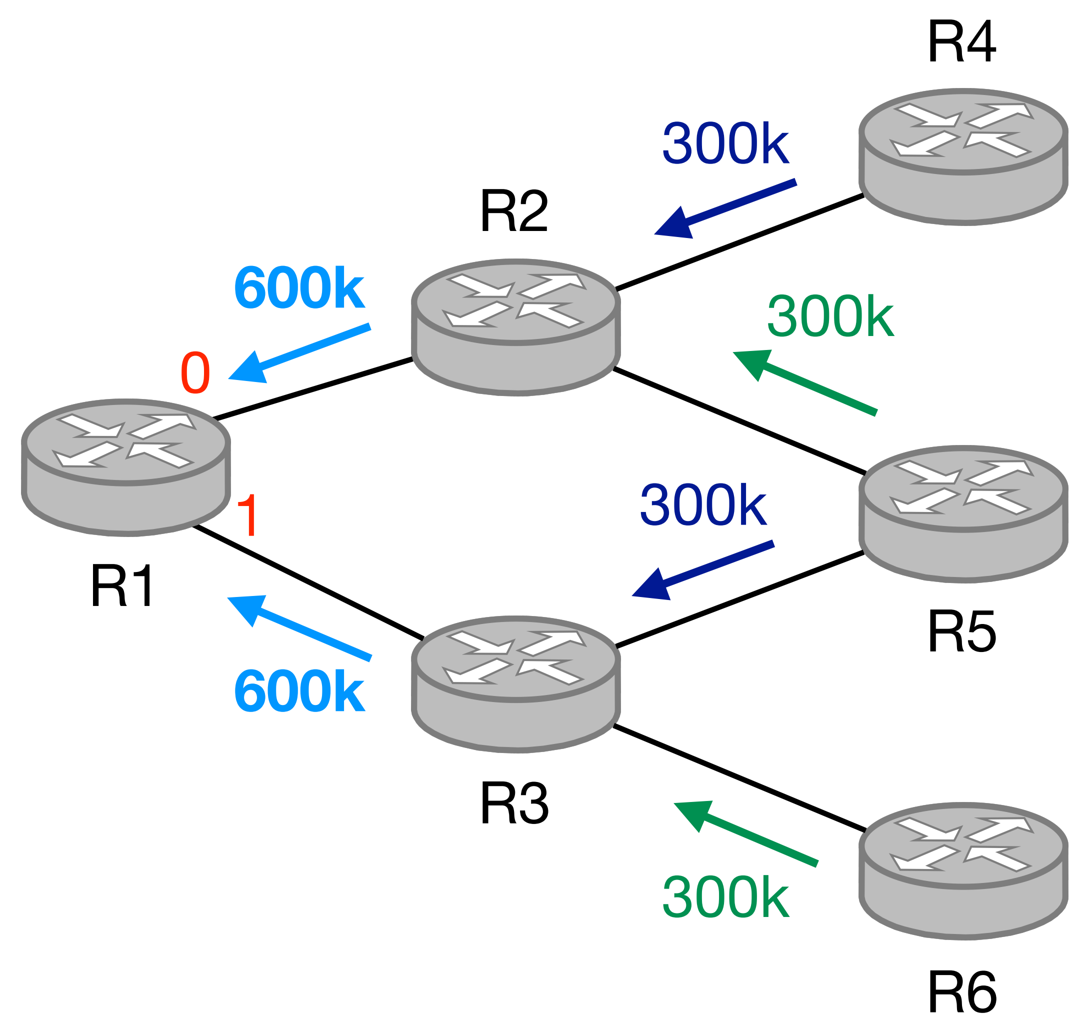



R4
R2
0
R1
1
R5
R3
R6


R4
300k
R2
600k
300k
0
R1
1
300k
R5
600k
R3
300k
R6

R1's Forwarding Table

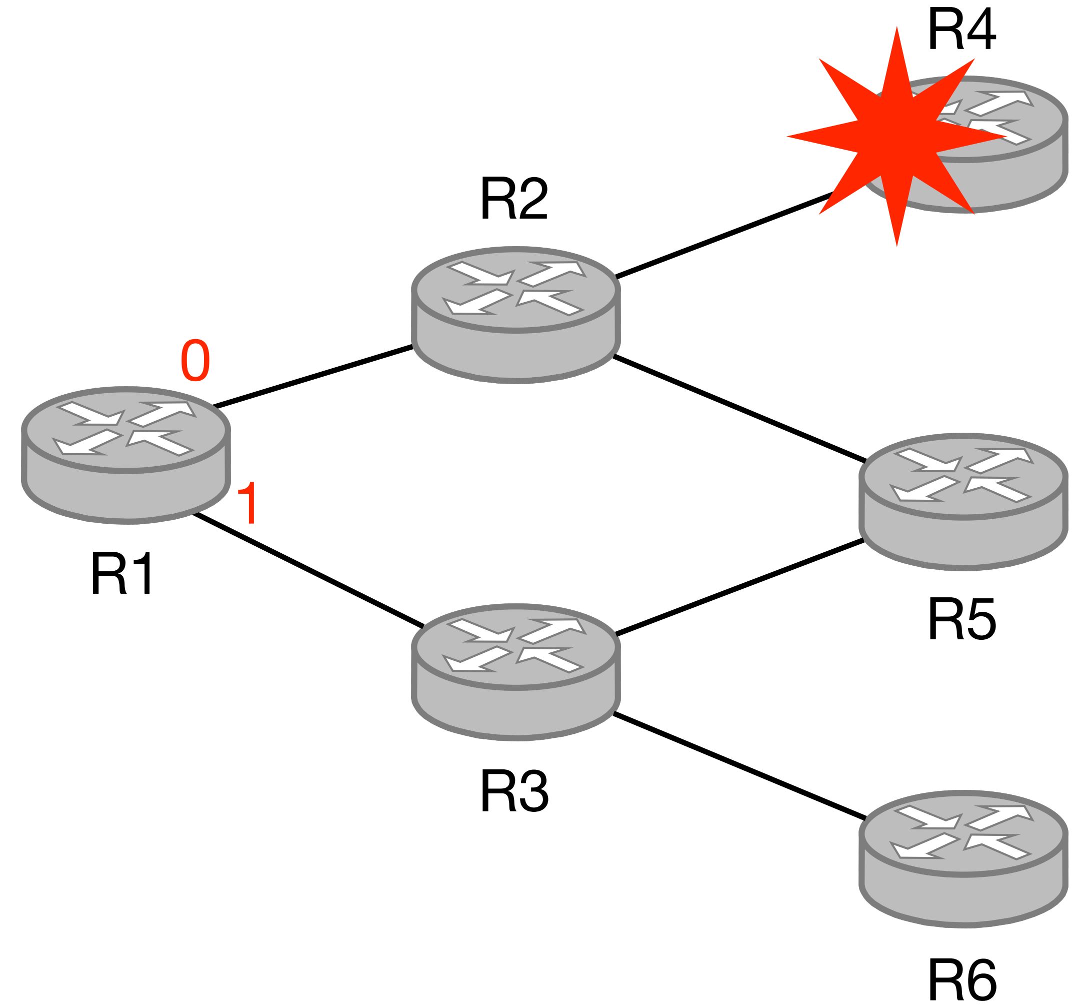
	IP prefix	Next-Hop
1	1.0.0.0/24	0
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



What if R4 fails?

R1's Forwarding Table

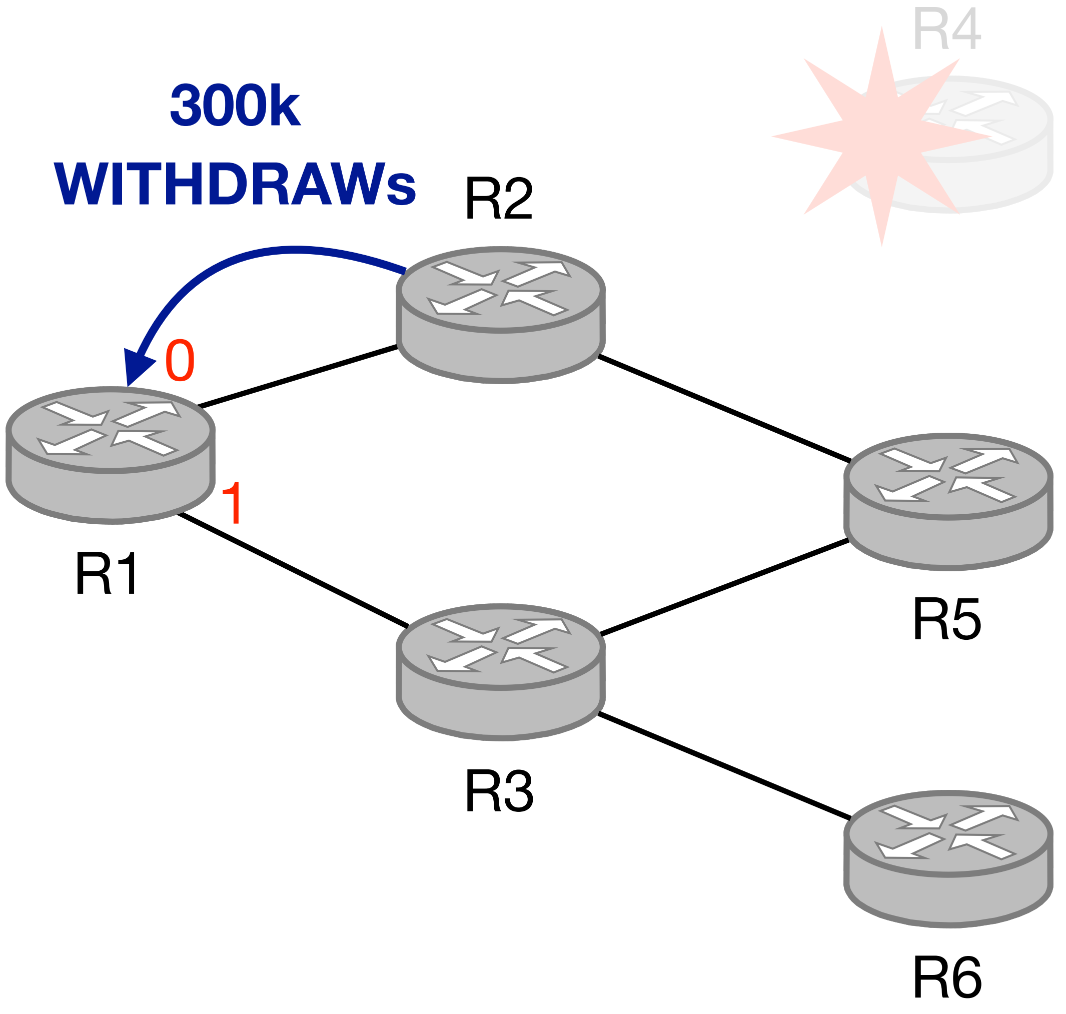
	IP prefix	Next-Hop
1	1.0.0.0/24	0
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



R2 sends a burst of 300k routing messages withdrawing the routes learned from R4

R1's Forwarding Table

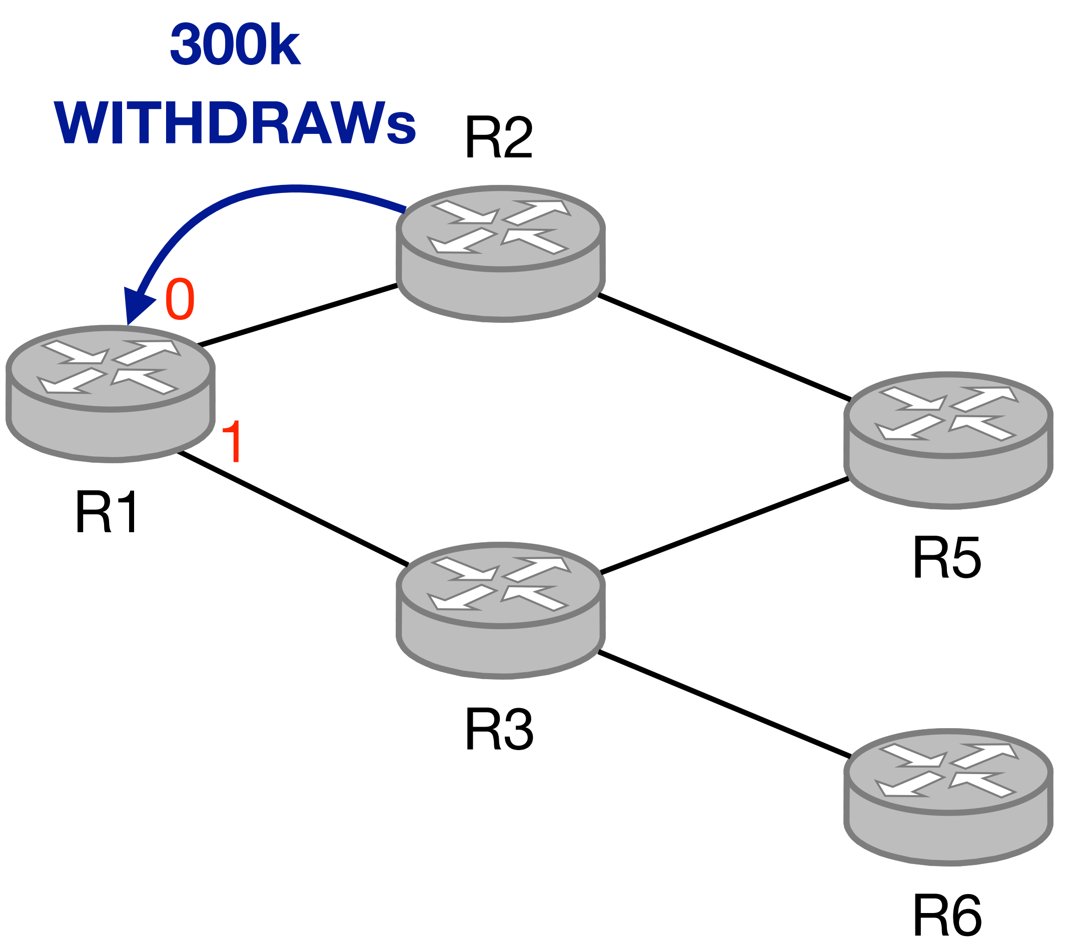
	IP prefix	Next-Hop
1	1.0.0.0/24	0
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



R1 receives the messages one-by-one and updates its forwarding table entry-by-entry

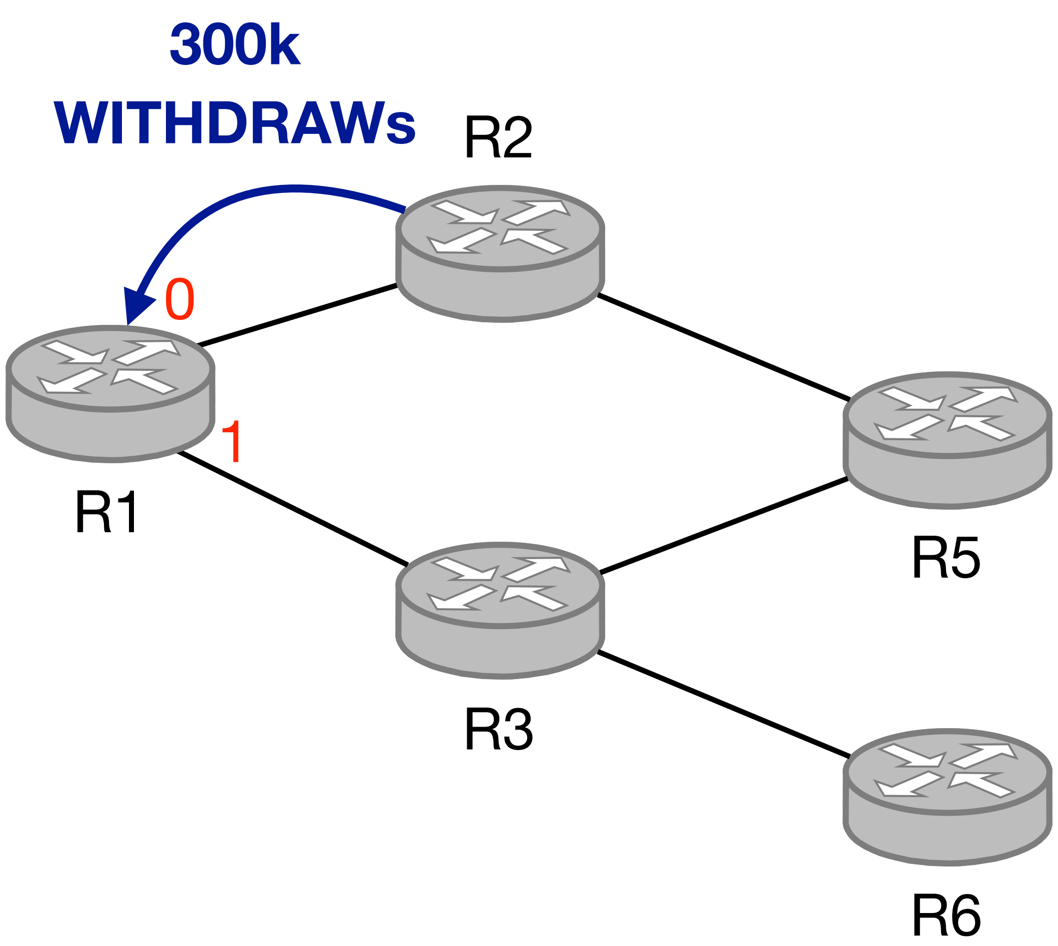
R1's Forwarding Table

	IP prefix	Next-Hop
1	1.0.0.0/24	0
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



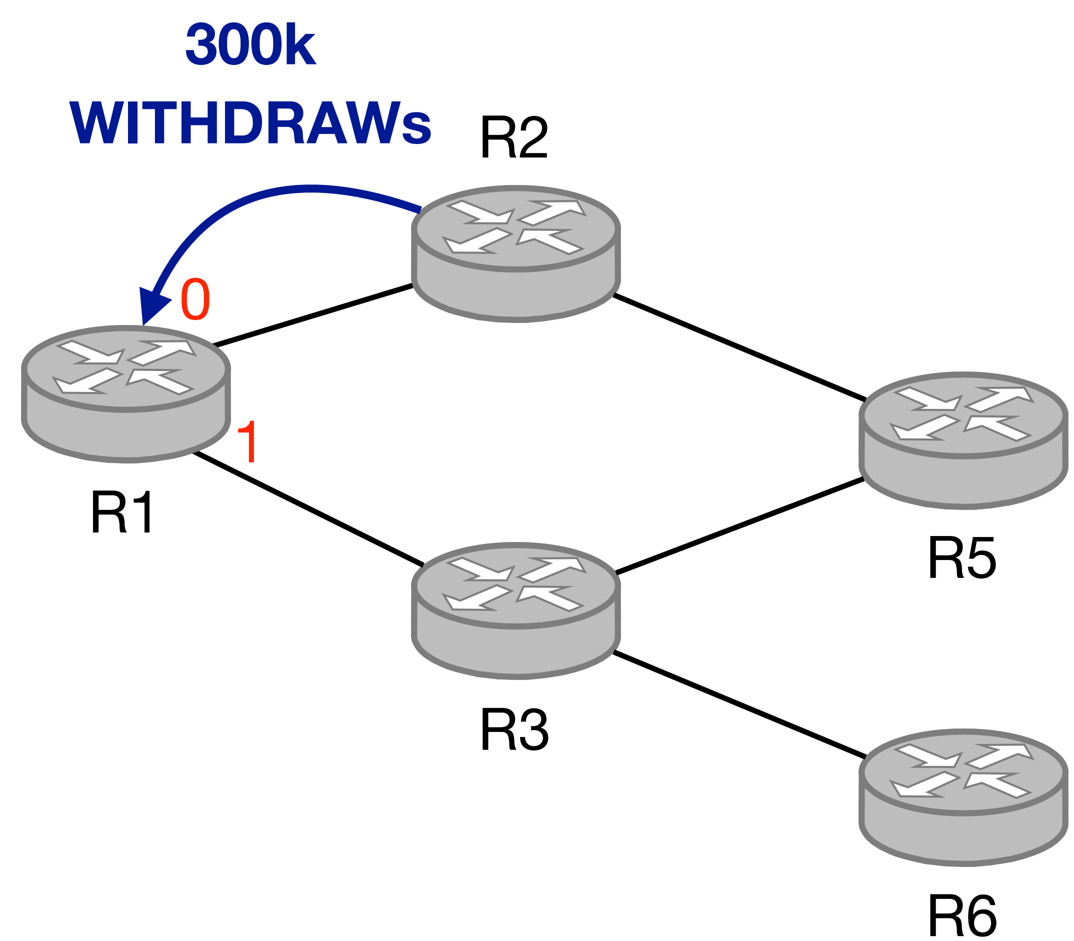
R1's Forwarding Table

	IP prefix	Next-Hop
1	1.0.0.0/24	1
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



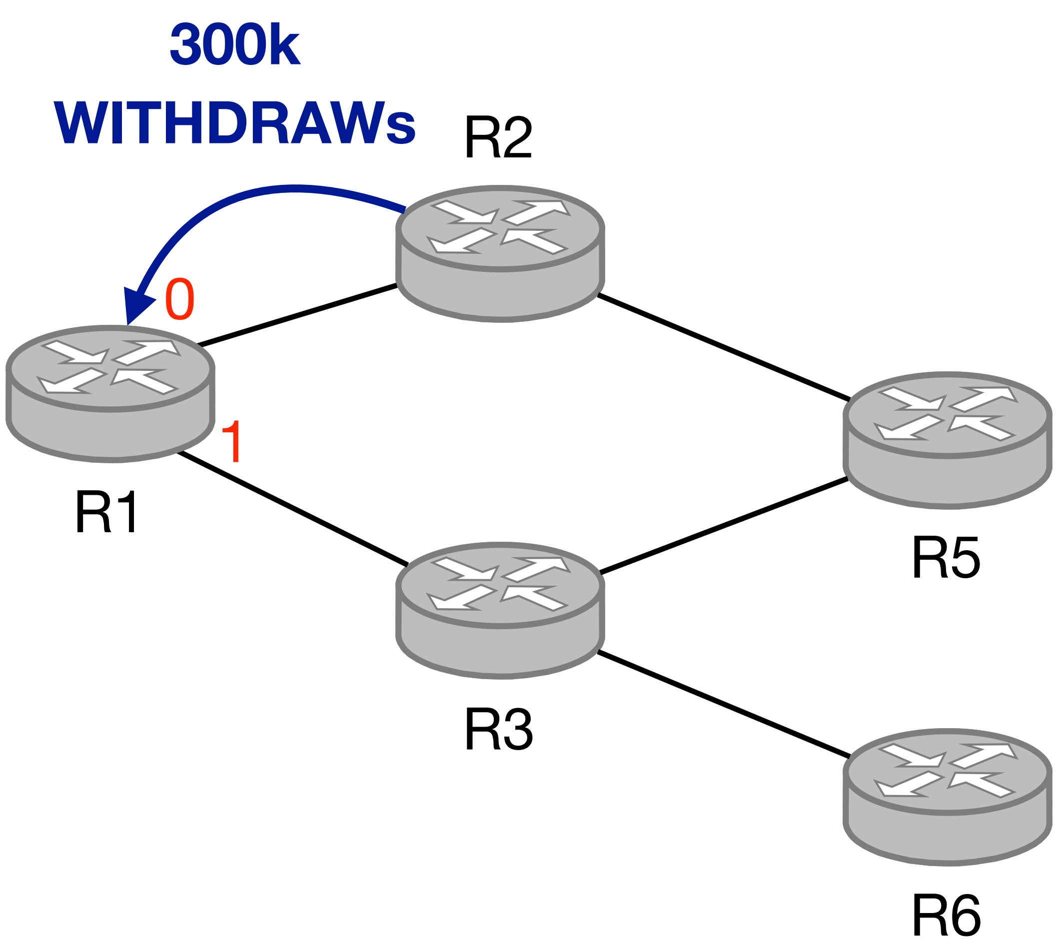
R1's Forwarding Table

	IP prefix	Next-Hop
1	1.0.0.0/24	1
2	1.0.1.0/16	1
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



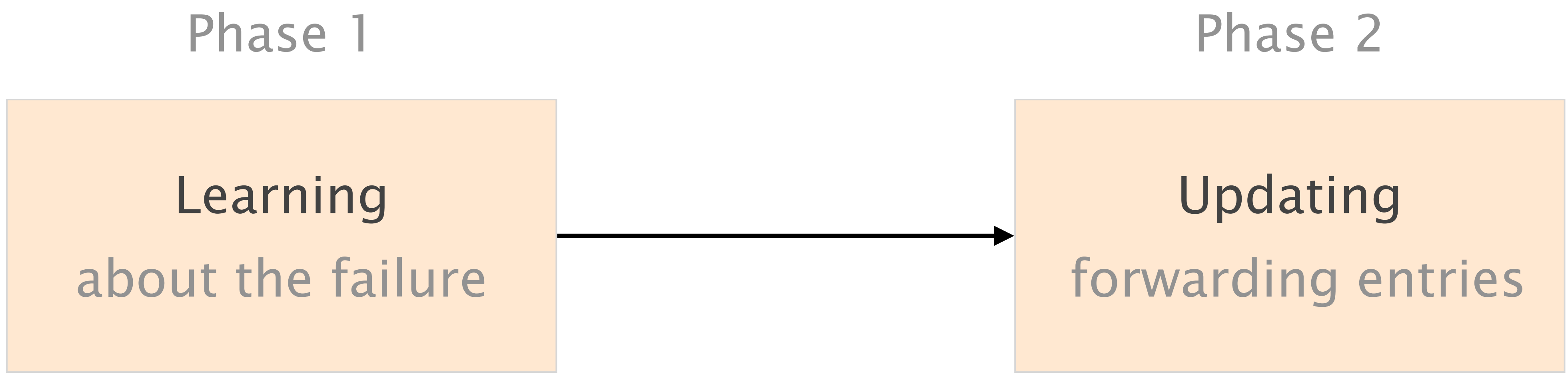
R1's Forwarding Table

	IP prefix	Next-Hop
1	1.0.0.0/24	1
2	1.0.1.0/16	1
…	…	…
300k	100.0.0.0/8	1
…	…	…
600k	200.99.0.0/24	0



Internet convergence

a two-phase process



Internet convergence

a two-phase process

both of which are terribly slow

Internet convergence

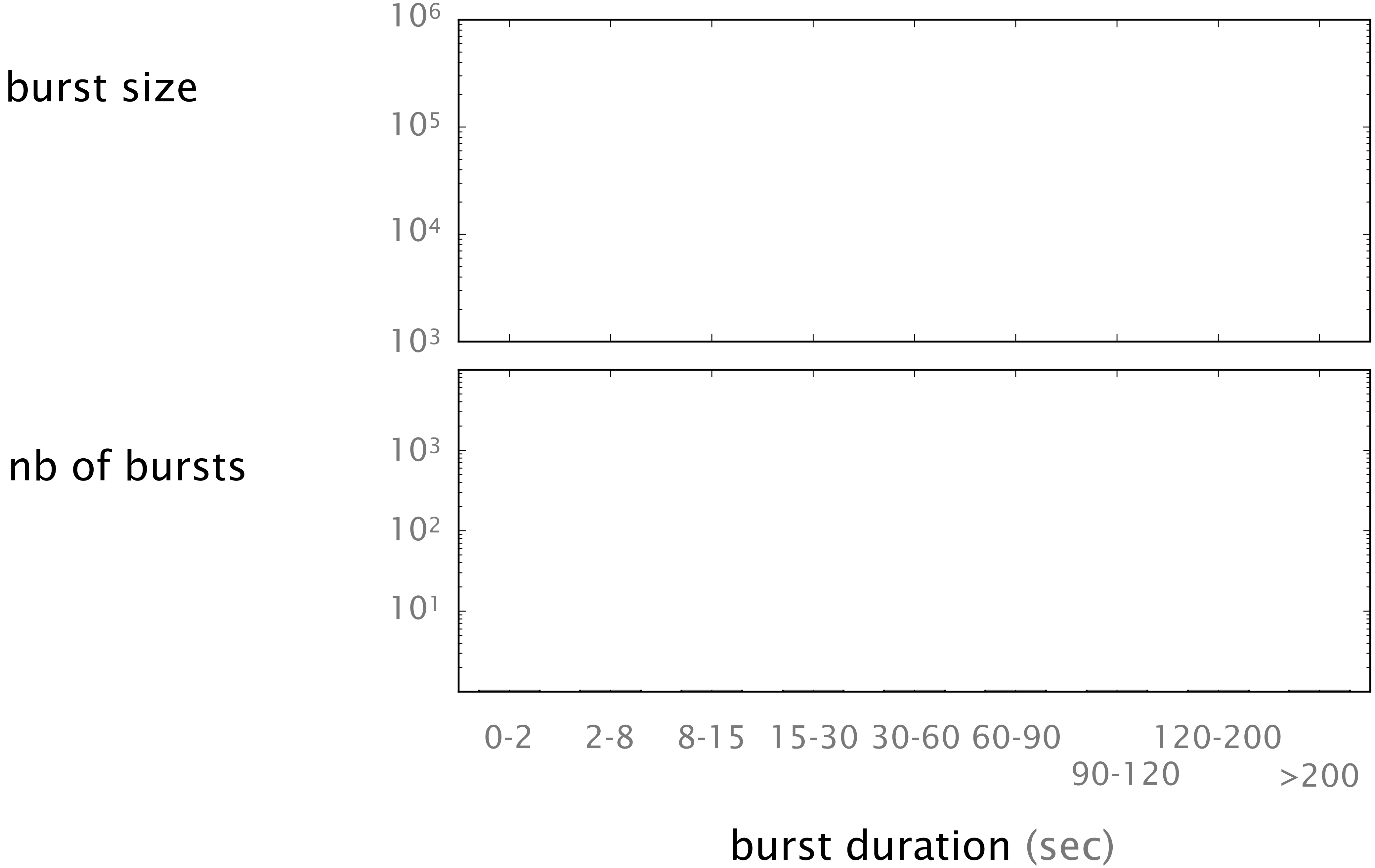
a two-phase process

Internet convergence

a two-phase process

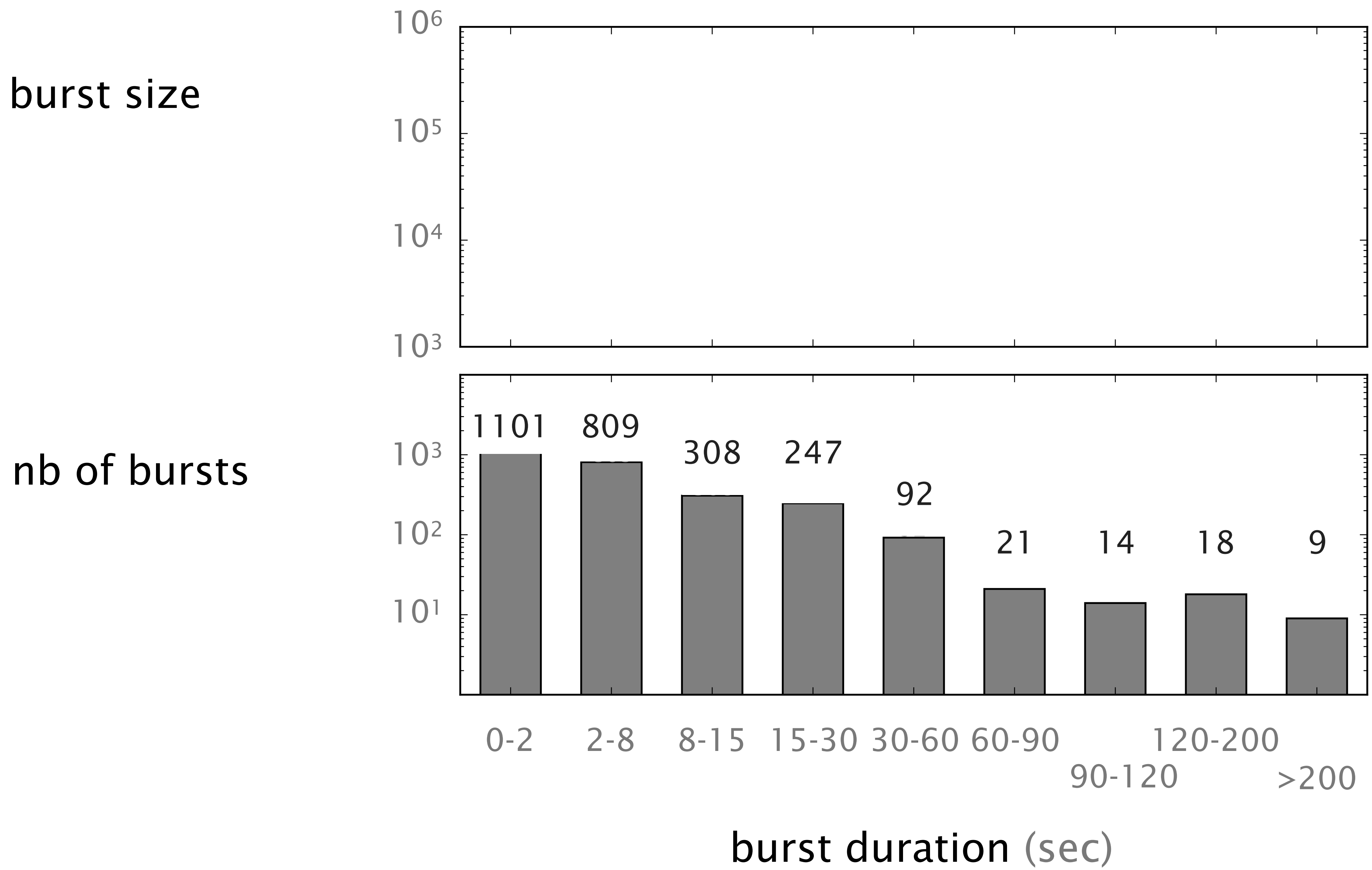
We measured how long it takes for large bursts of BGP updates to propagate in the Internet

dataset	a month (July'16) worth of Internet updates from ~200 routers scattered around the globe
methodology	detect the beginning and the end of a burst using a 10 sec sliding window



burst size
nb of bursts
10^6
10^5
10^4
10^3
10^3
10^2
10^1
0-2
2-8
8-15
15-30
30-60
60-90
90-120
120-200
>200
burst duration (sec)

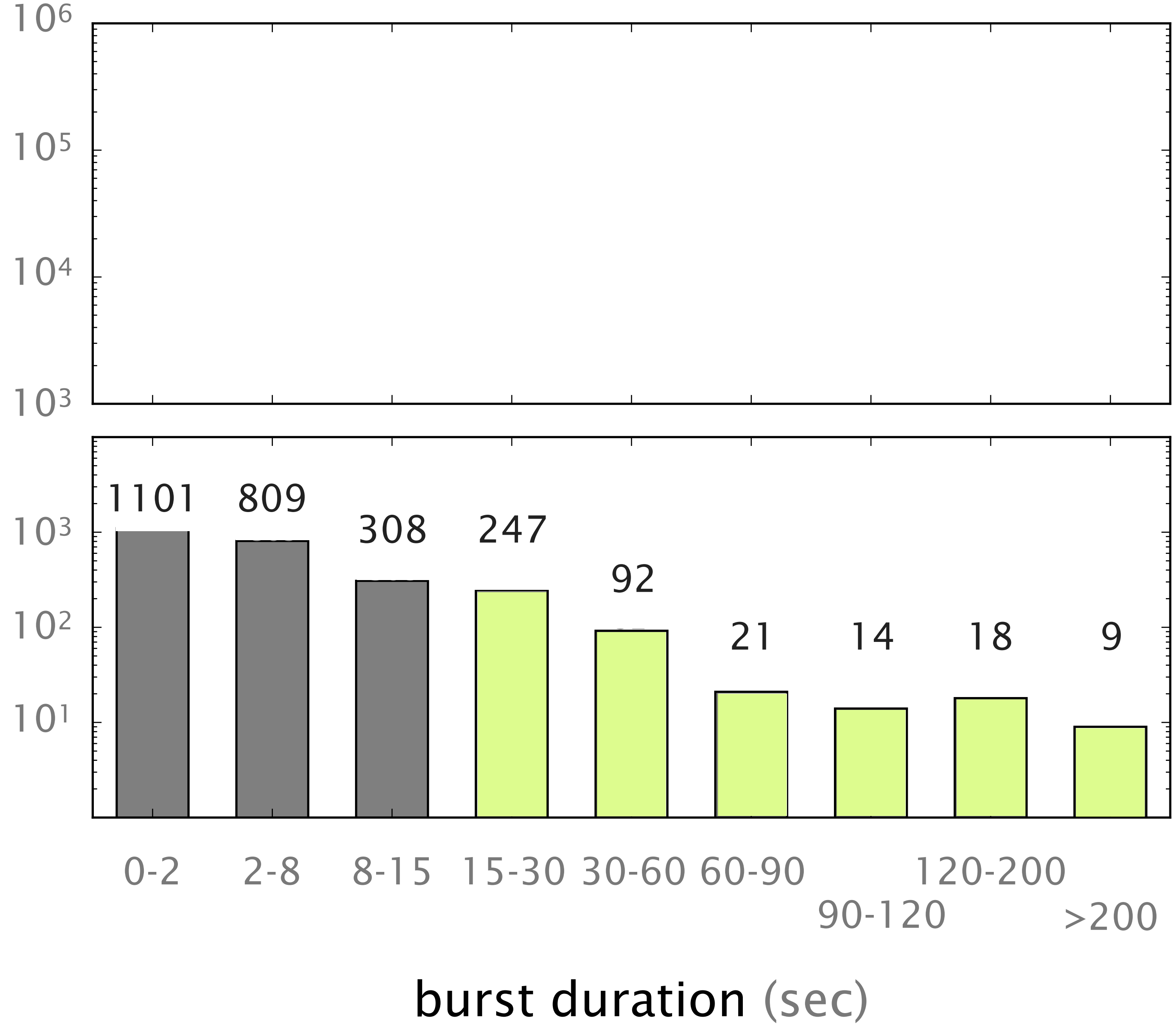
We found a total of 2619 bursts over the month



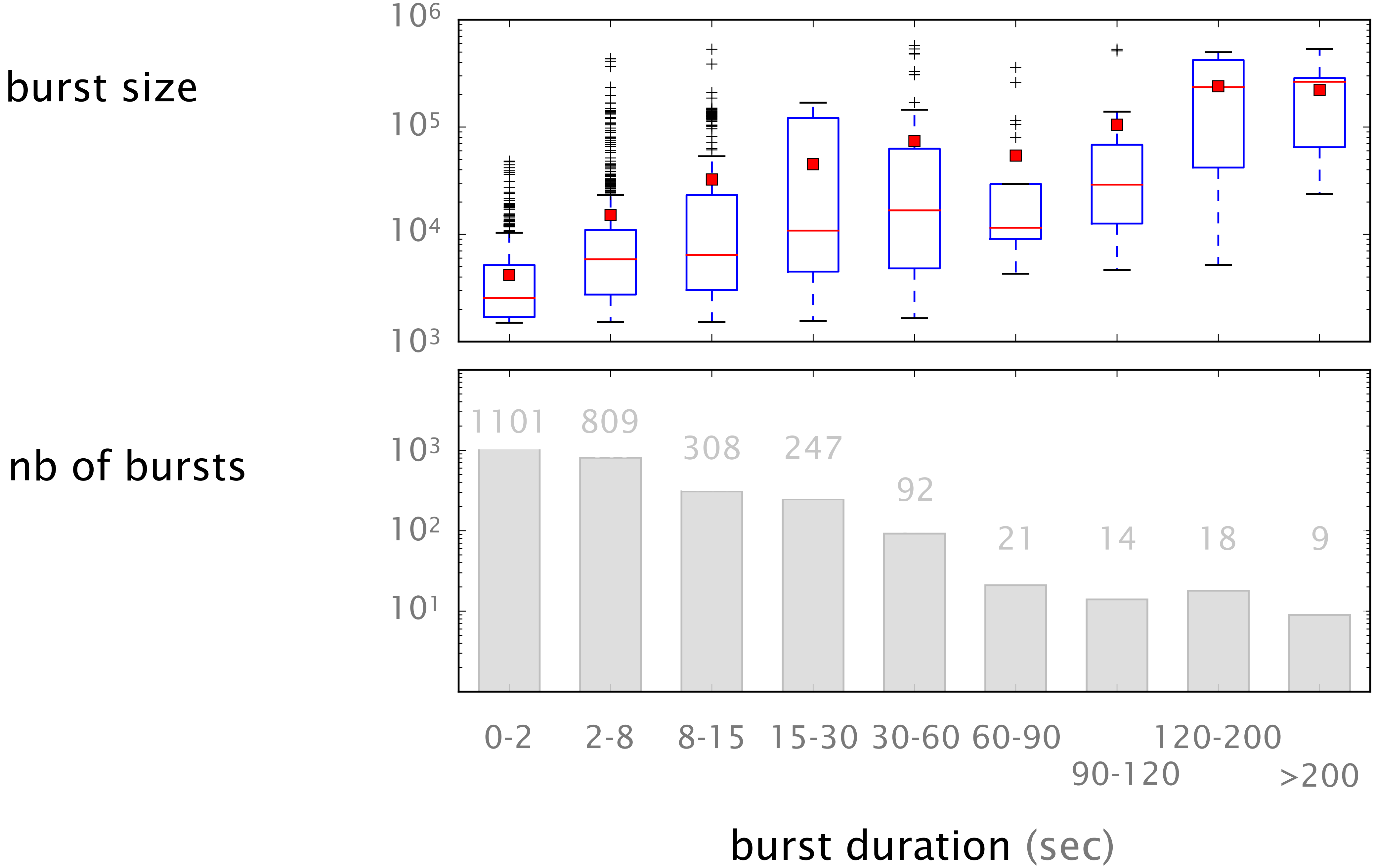
15% of the bursts takes more than 15s to be learned

burst size

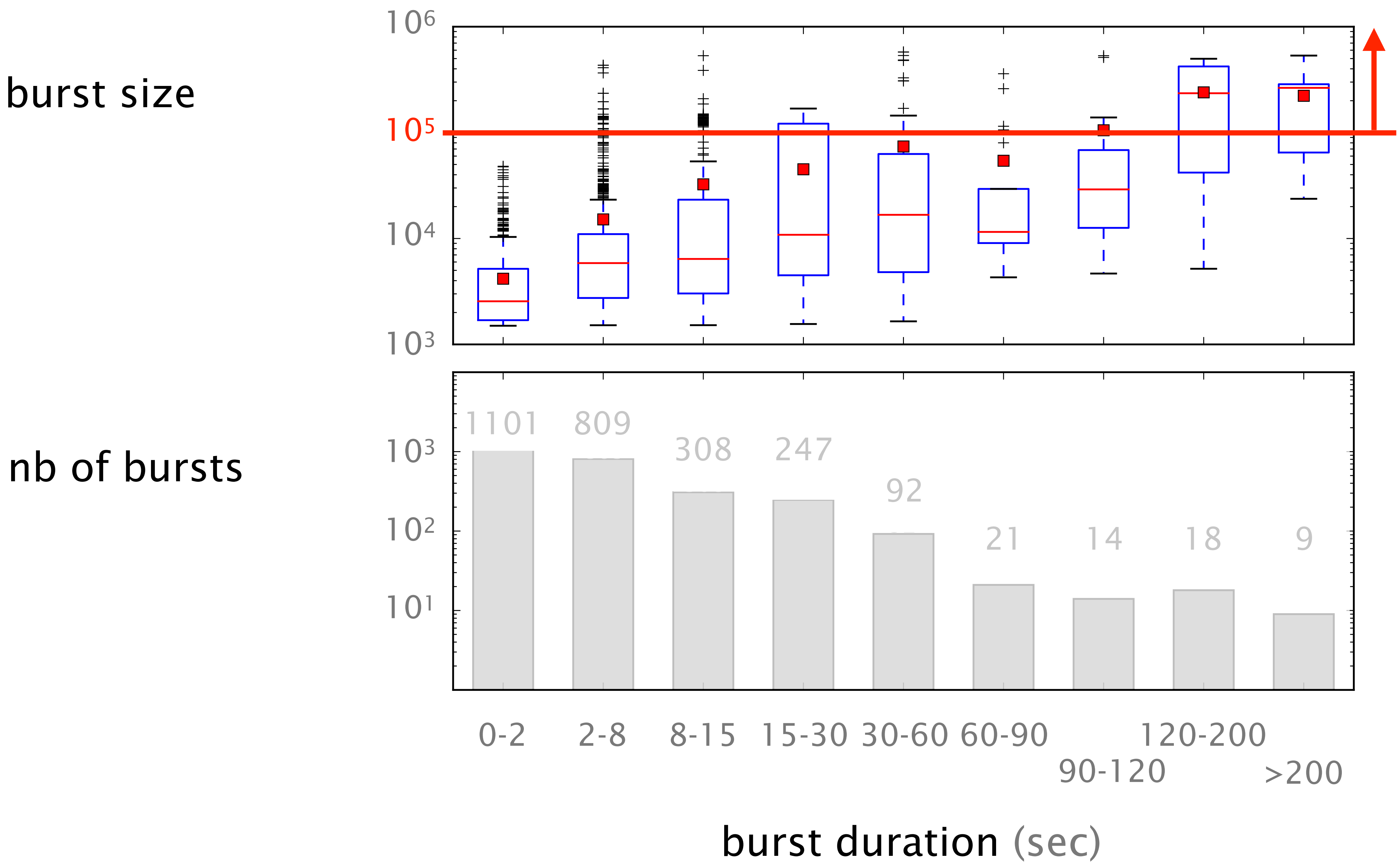
nb of bursts



Slow-to-learn bursts tend to be big



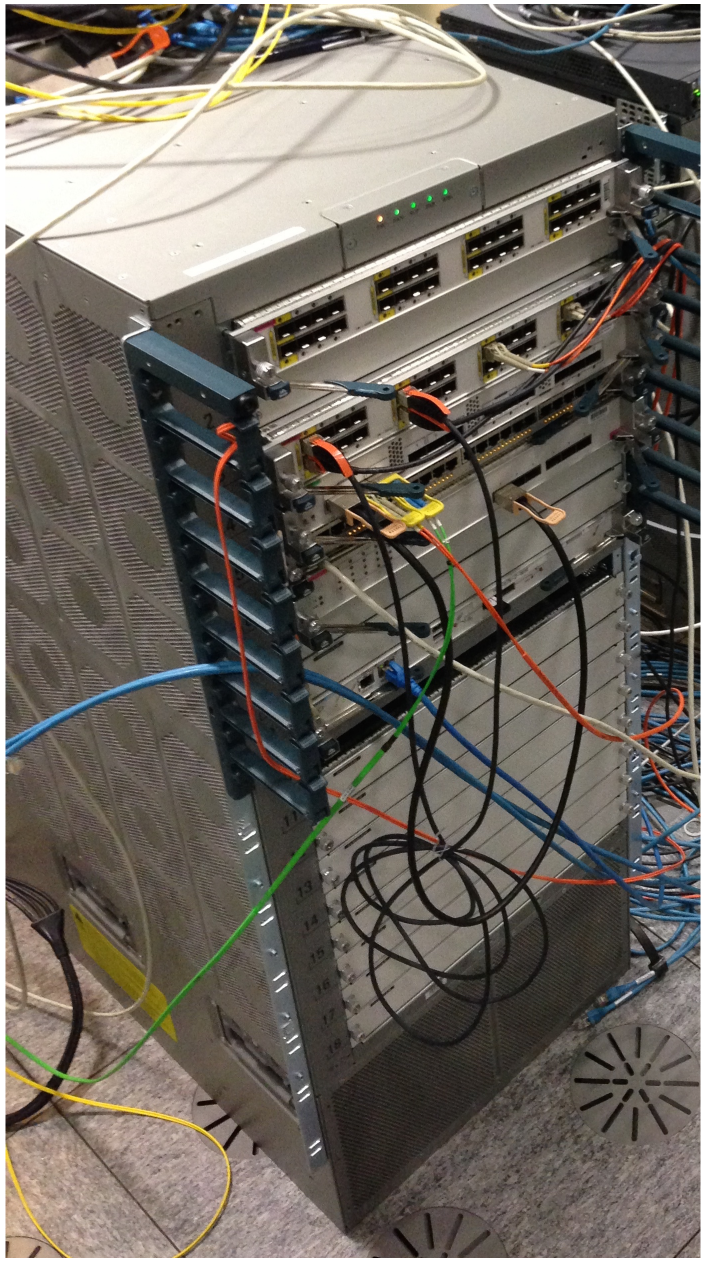
~10% of the bursts contained more than 100k prefixes



Internet convergence

a two-phase process

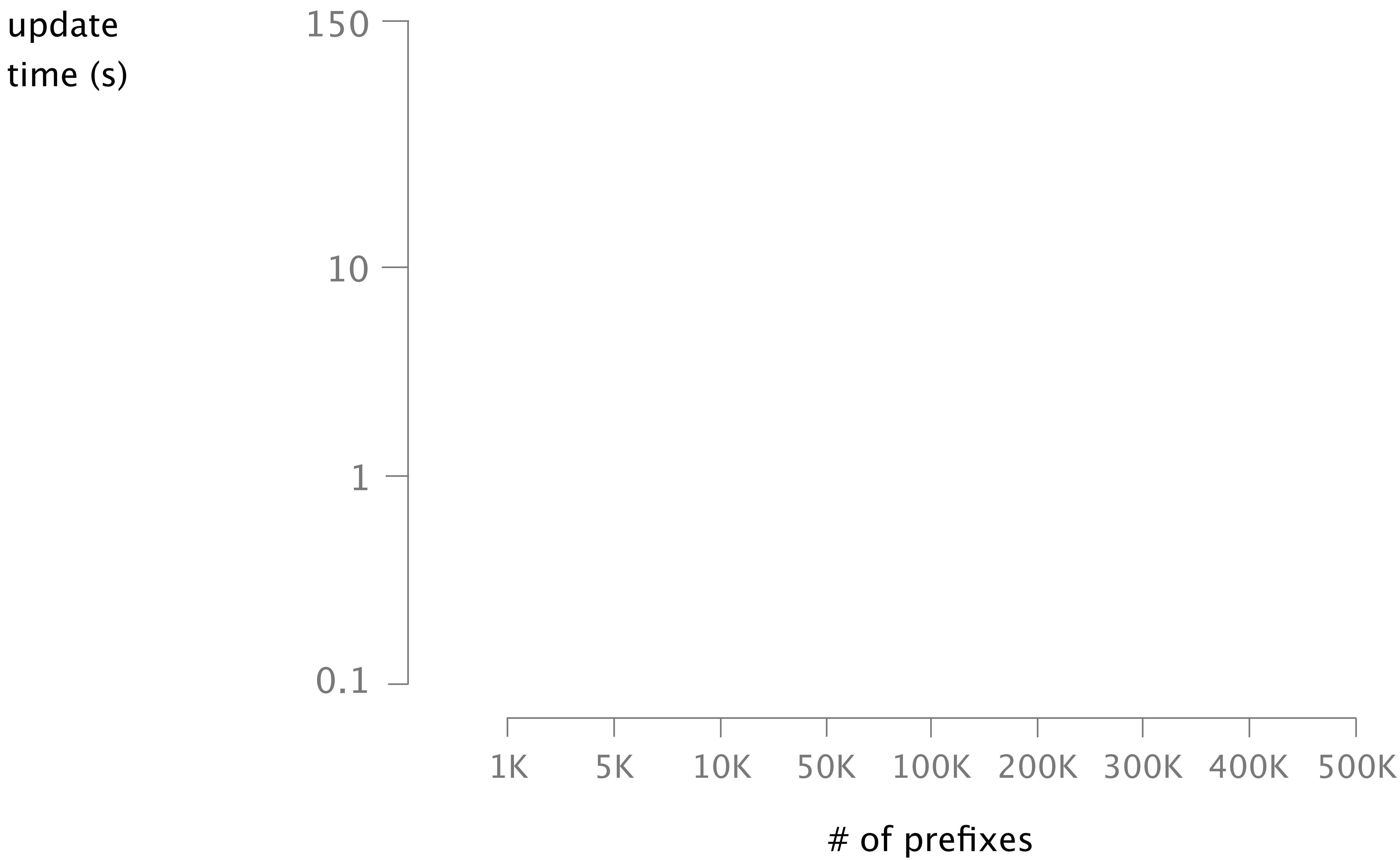
We measured how long it takes for recent routers to update a growing number of forwarding entries

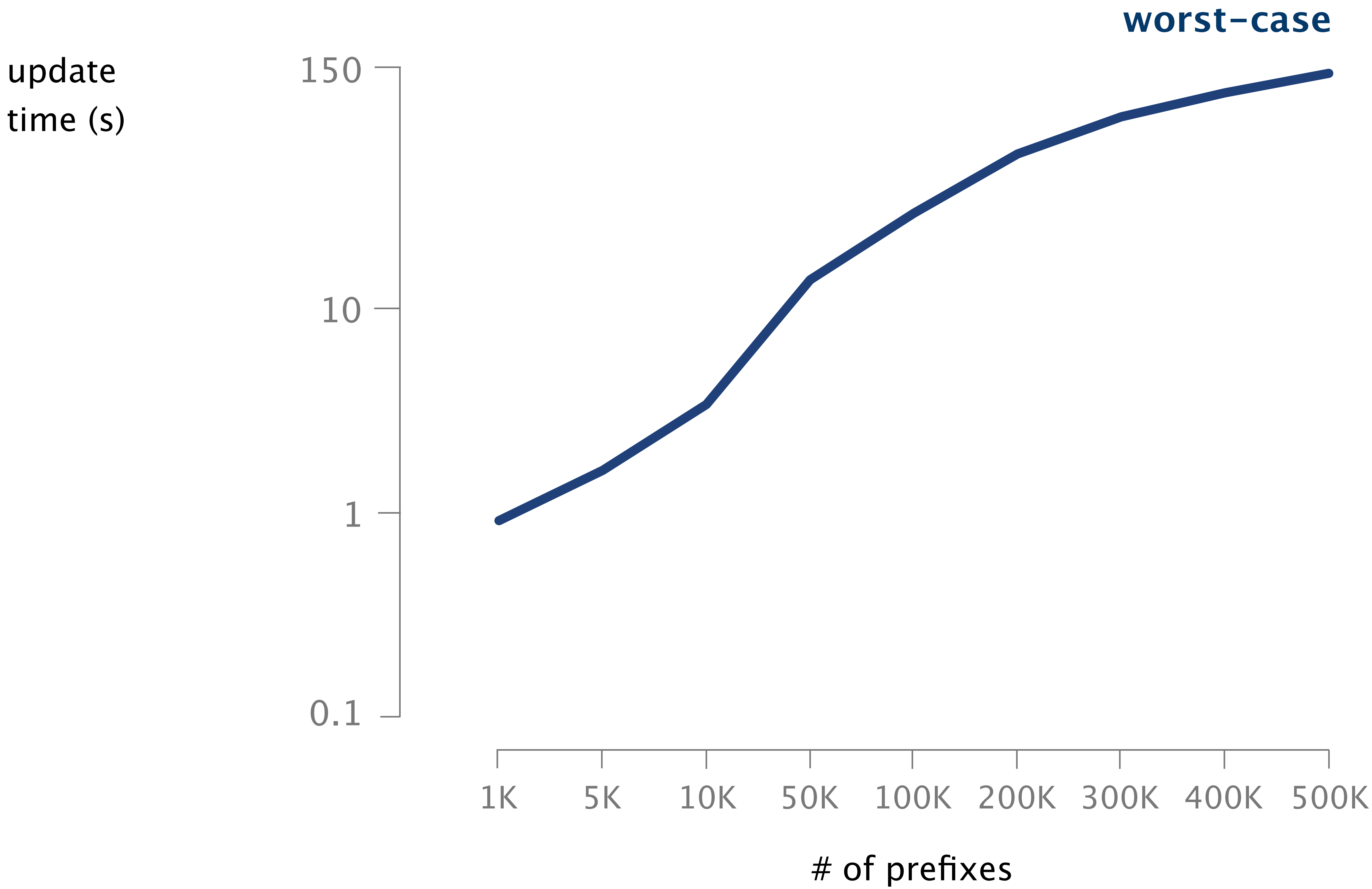


Cisco Nexus 7k

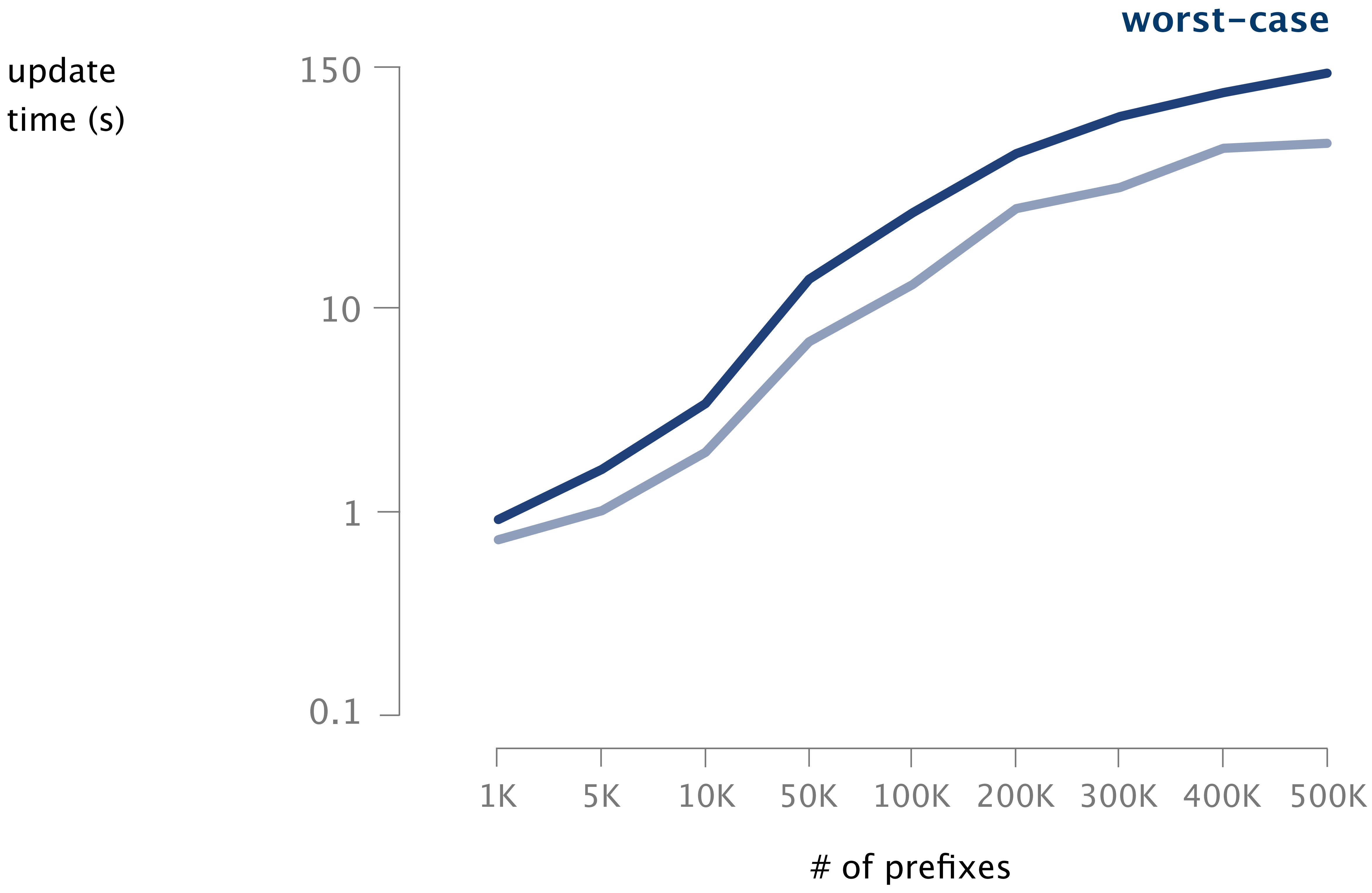
ETH recent routers

25 deployed

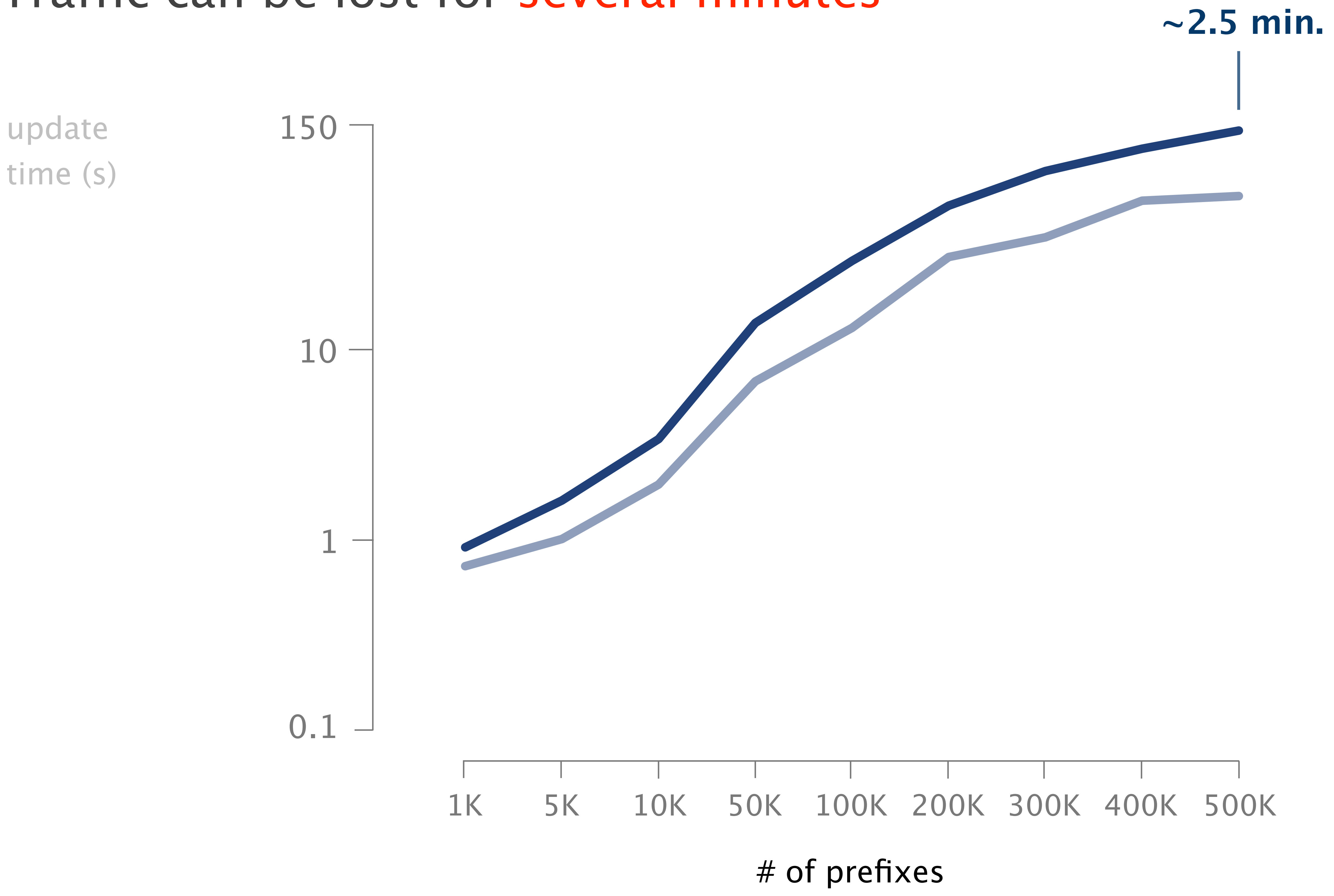

update
time (s)
150
10
1
0.1
1K
5K
10K
50K
100K
200K
300K
400K
500K
# of prefixes



worst-case
update
time (s)
150
10
1
0.1
1K
5K
10K
50K
100K
200K
300K
400K
500K
# of prefixes

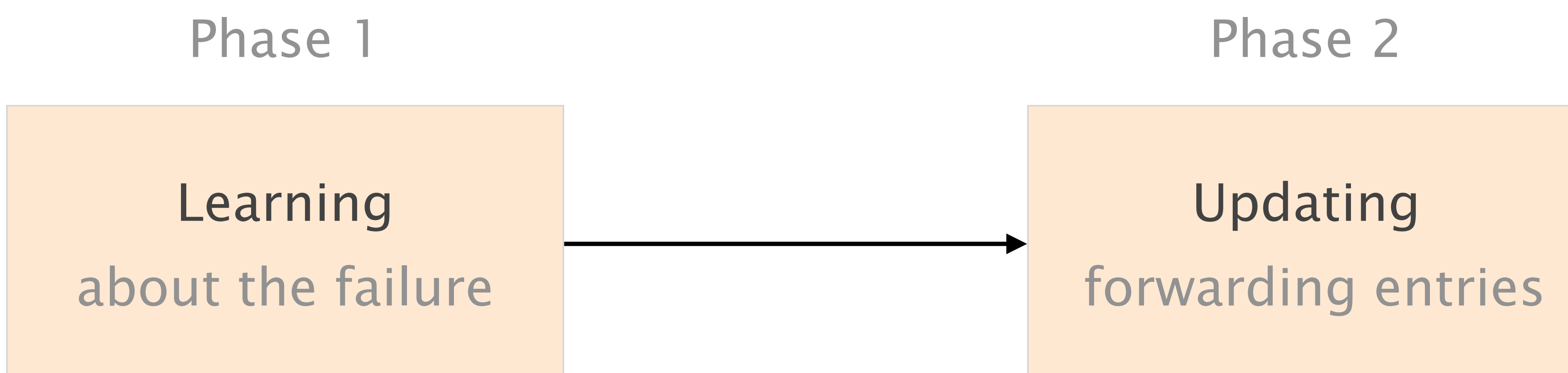

worst-case
update
time (s)
150
10
1
0.1
1K
5K
10K
50K
100K
200K
300K
400K
500K
# of prefixes

Traffic can be lost for several minutes



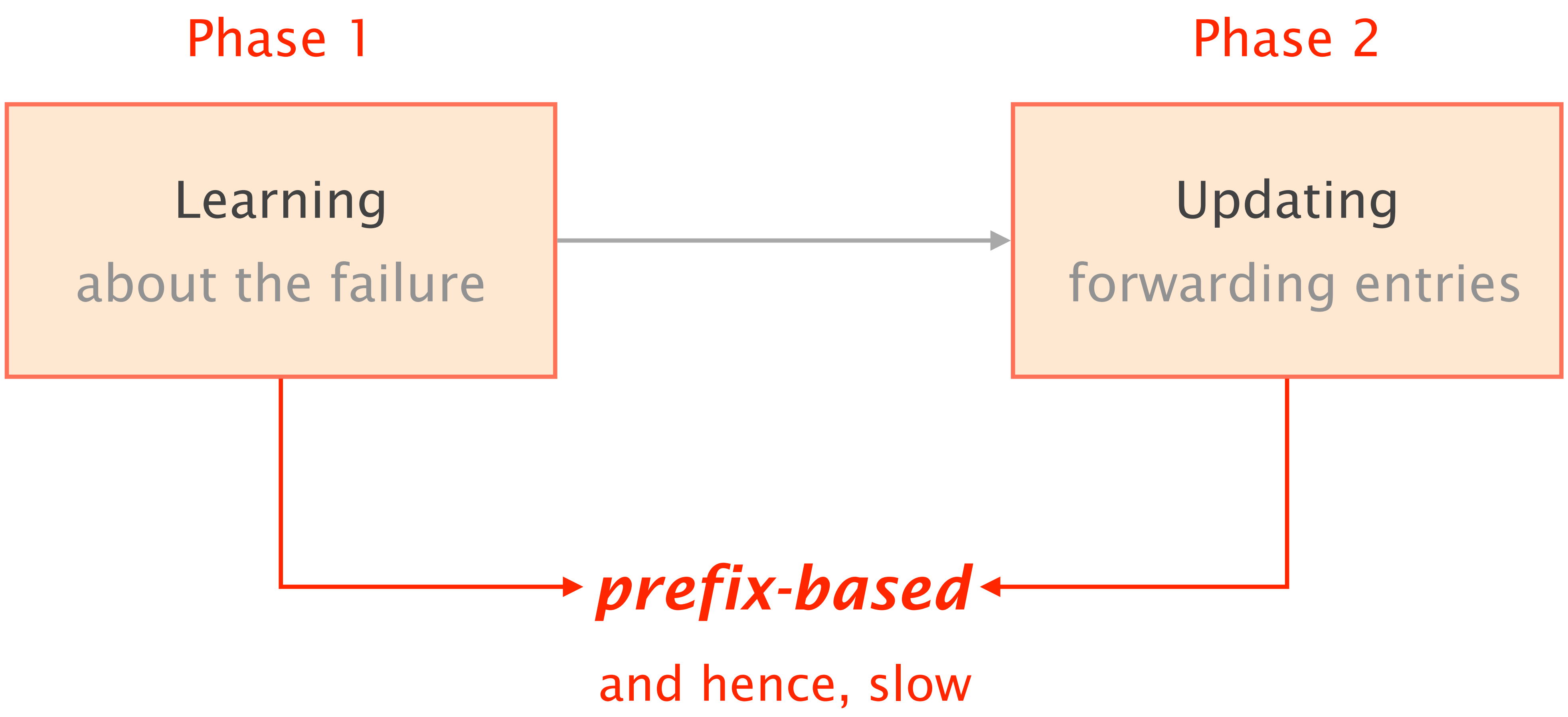
Internet convergence

a two-phase process

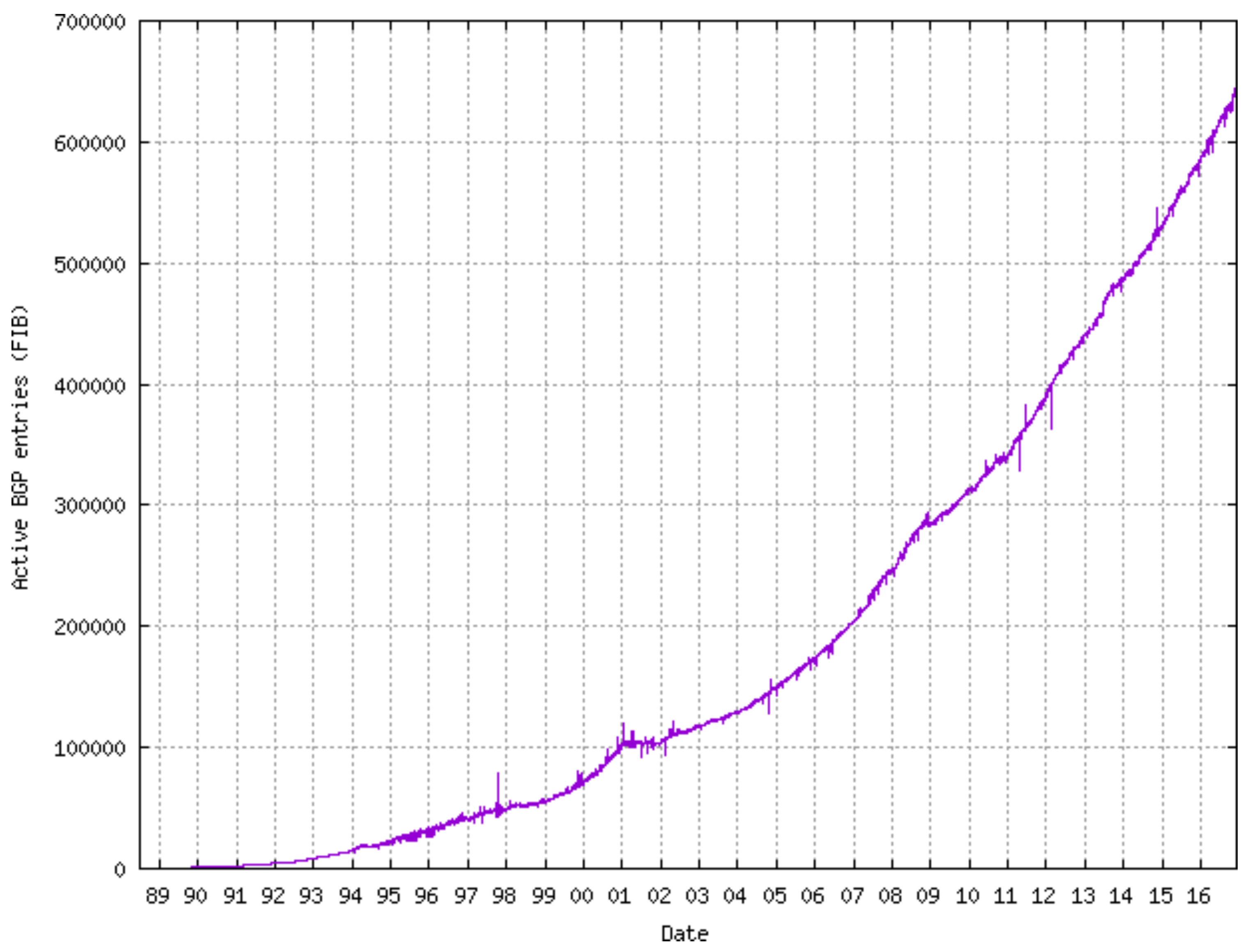


Internet convergence

a two-phase process



Convergence times are only gonna get worse as the Internet continues to grow (*)



(*) http://www.cidr-report.org/

SWIFT: Predictive Fast Rerouting

Joint work with: Thomas Holterbach, Alberto Dainotti, Stefano Vissicchio

SWIFT: Predictive Fast Rerouting

speed up...

learning
about the failure

updating
the data plane

SWIFT: Predictive Fast Rerouting

speed up... learning
about the failure

solution *predict* the extent
of a failure from
few messages

SWIFT: Predictive Fast Rerouting

speed up… learning
about the failure

solution *predict* the extent
of a failure from
few messages

challenge speed and precision

SWIFT: Predictive Fast Rerouting

speed up…	learning about the failure	updating the data plane
solution	*predict* the extent of a failure from few messages	
challenge	speed and precision	

SWIFT: Predictive Fast Rerouting

speed up…	learning about the failure	updating the data plane
solution	*predict* the extent of a failure from few messages	update *groups* of entries instead of individual ones
challenge	speed and precision	

SWIFT: Predictive Fast Rerouting

speed up…	learning about the failure	updating the data plane
solution	*predict* the extent of a failure from few messages	update *groups* of entries instead of individual ones
challenge	speed and precision	failure model

SWIFT: Predictive Fast Rerouting

1 **Predicting**
out of few messages

2 **Updating**
groups of entries

3 **Supercharging**
existing systems

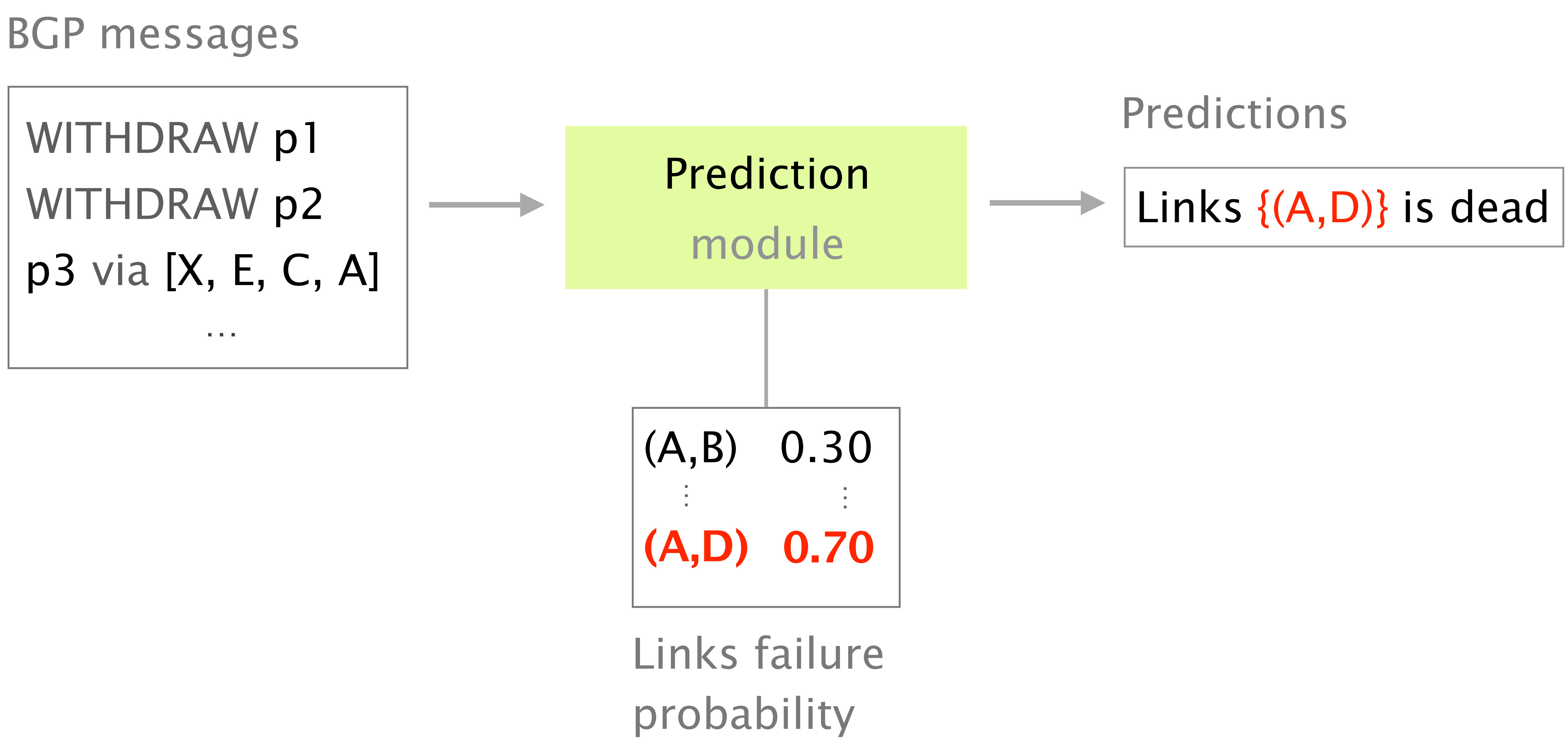
SWIFT: Predictive Fast Rerouting

1 **Predicting**
out of few messages

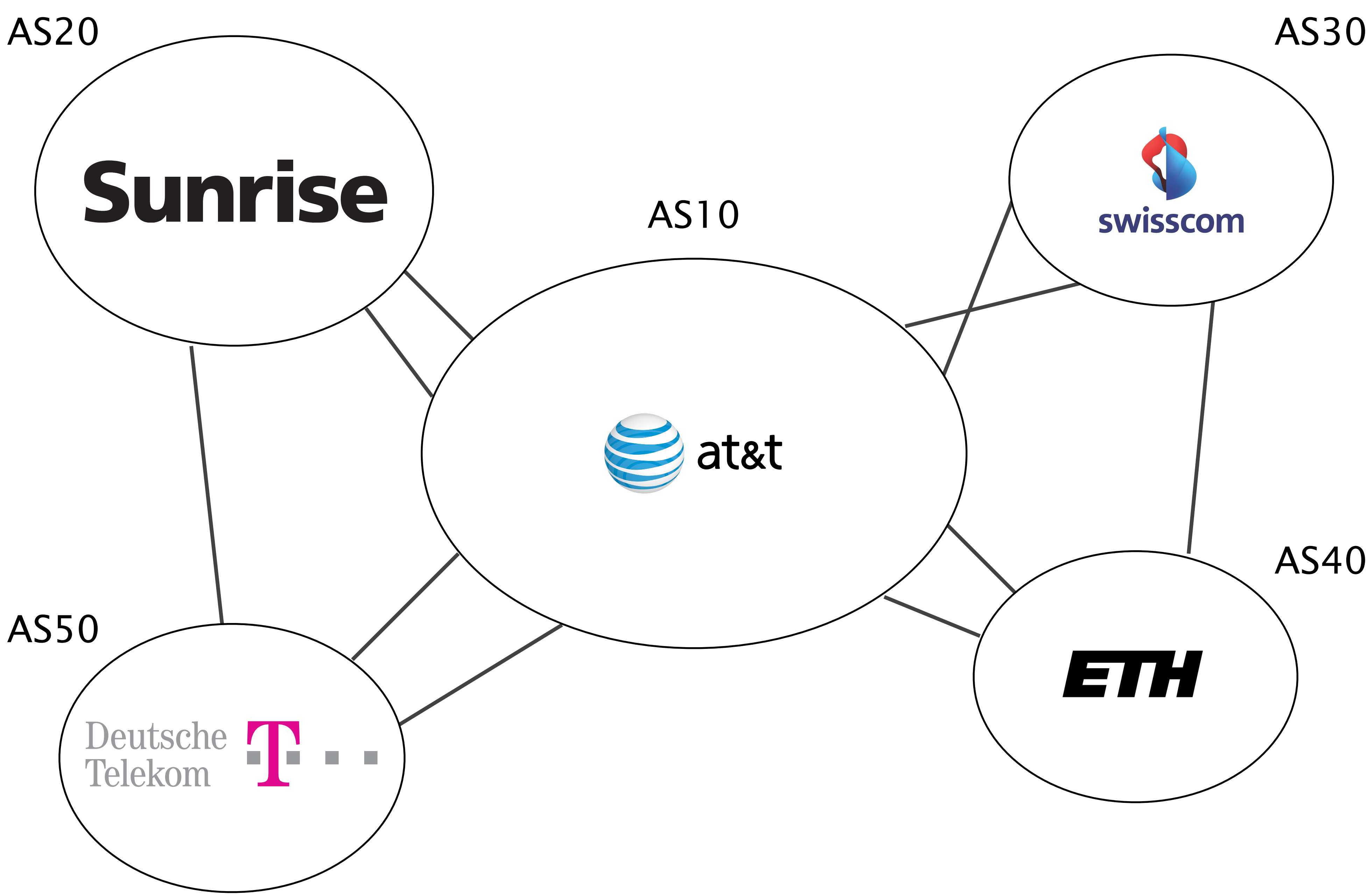
Updating
groups of entries

Supercharging
existing systems

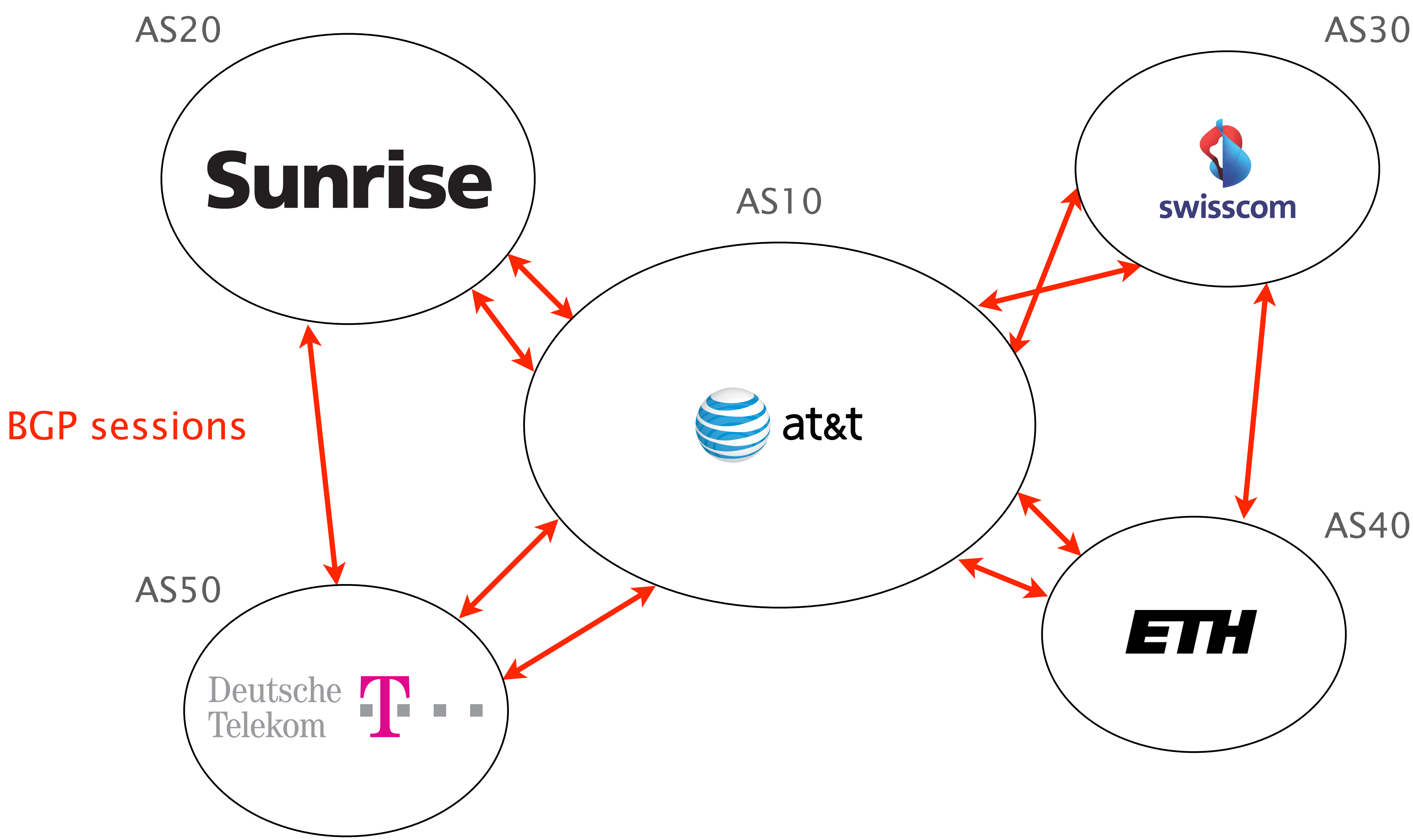
SWIFT leverages **redundant info** in routing messages to predict **which links** went down



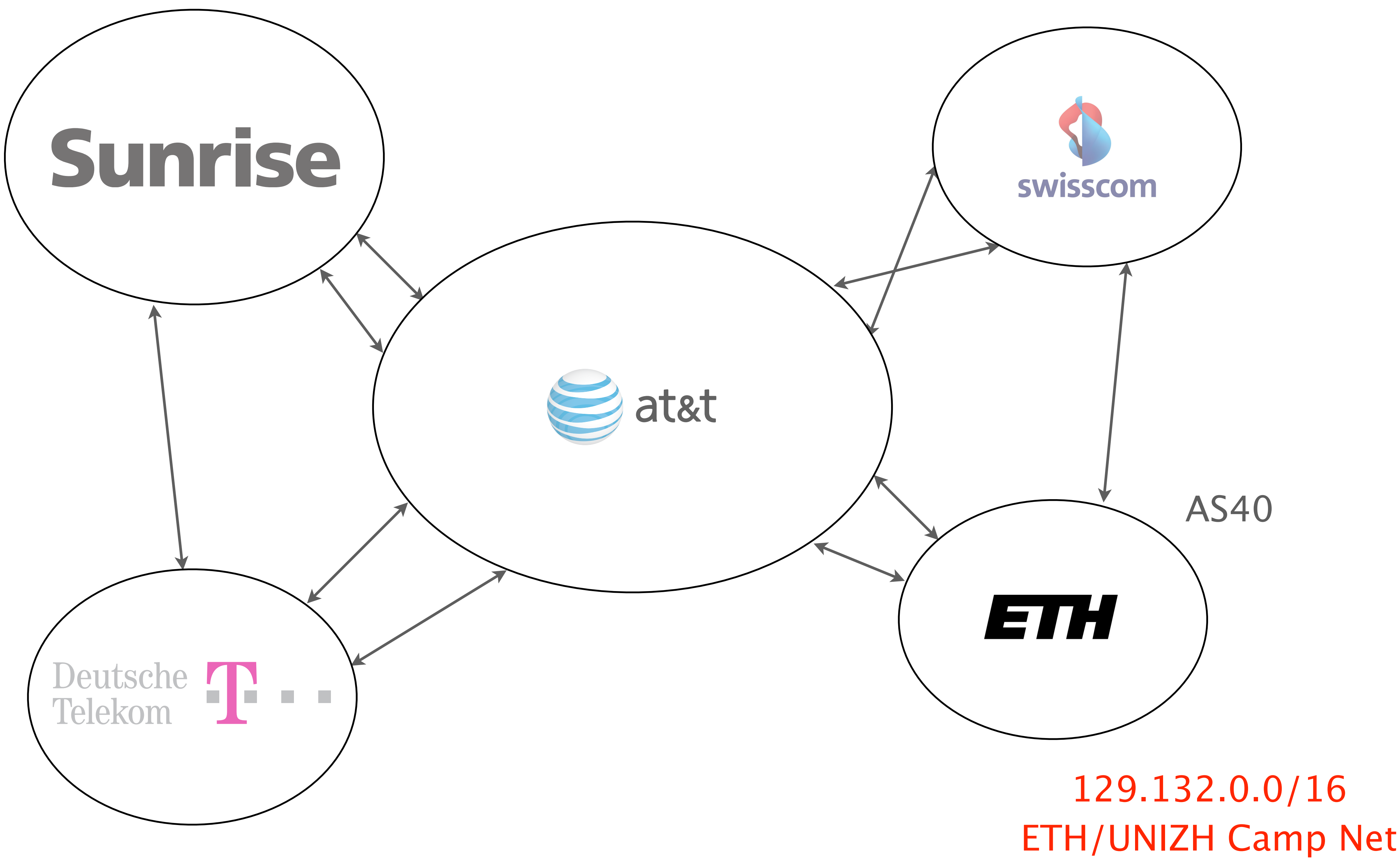
The Internet is a network of >50,000 networks, referred to as Autonomous Systems (AS)



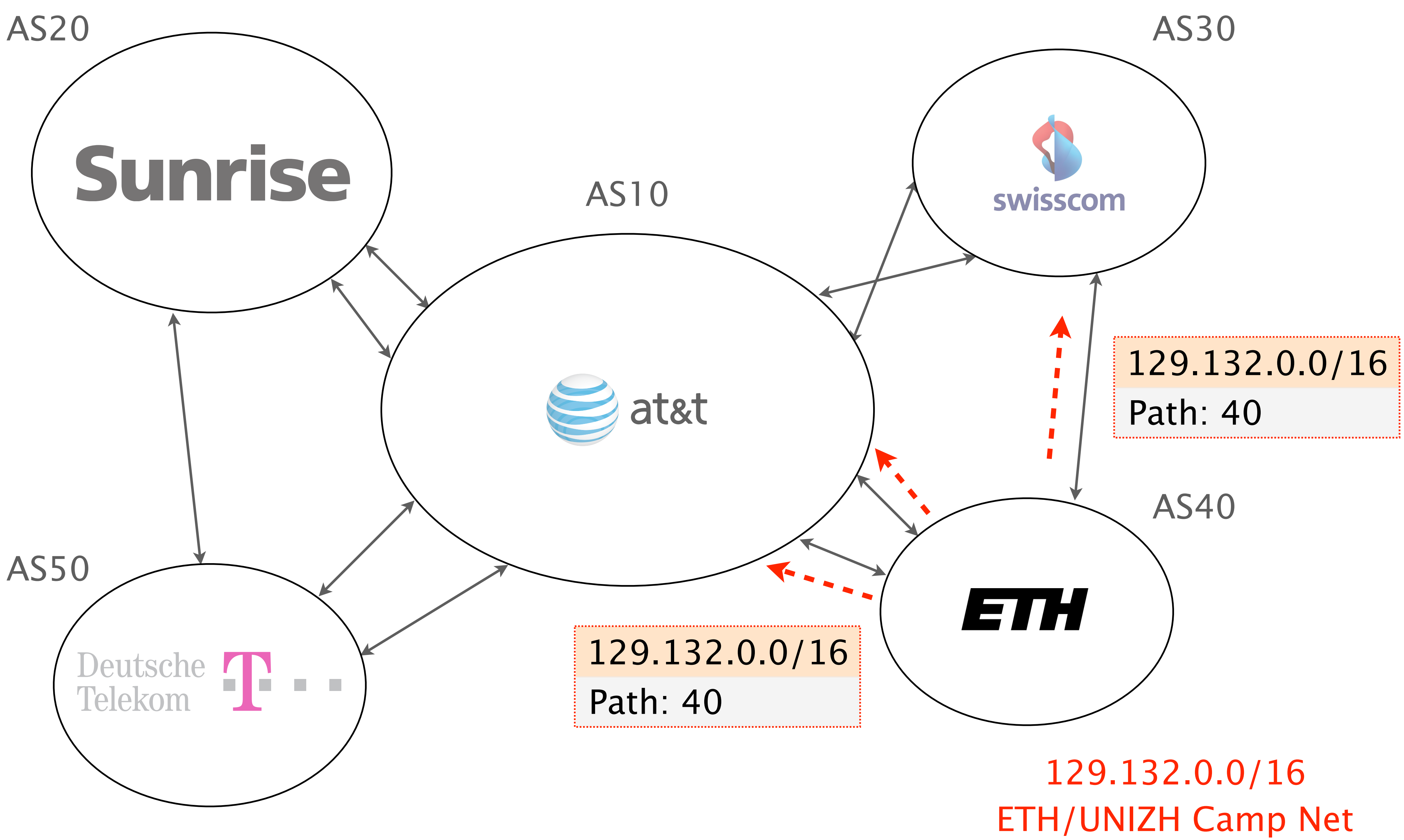
BGP is the routing protocol "glueing" the Internet together



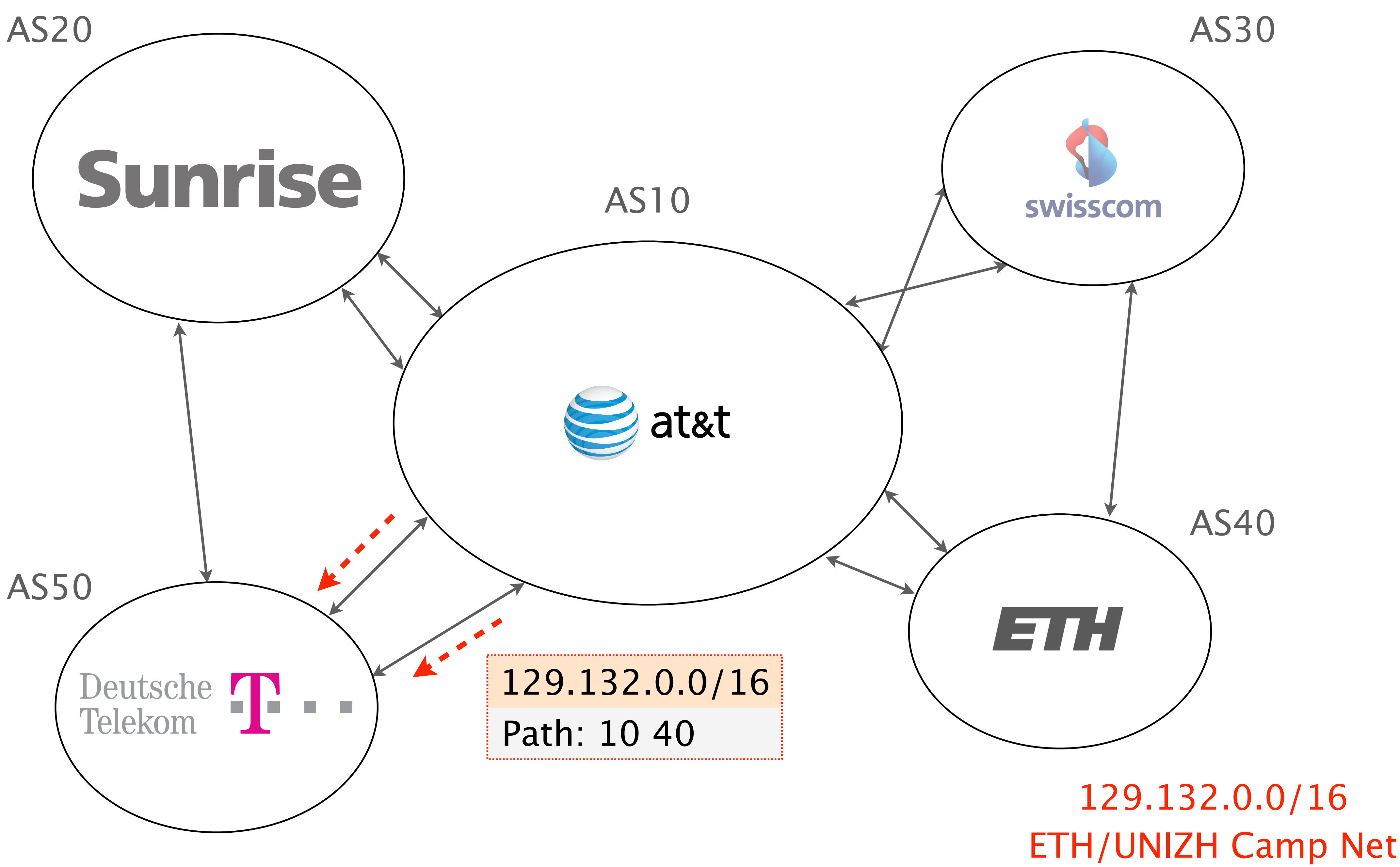
ASes exchange information about the destinations (IP addresses) they can reach



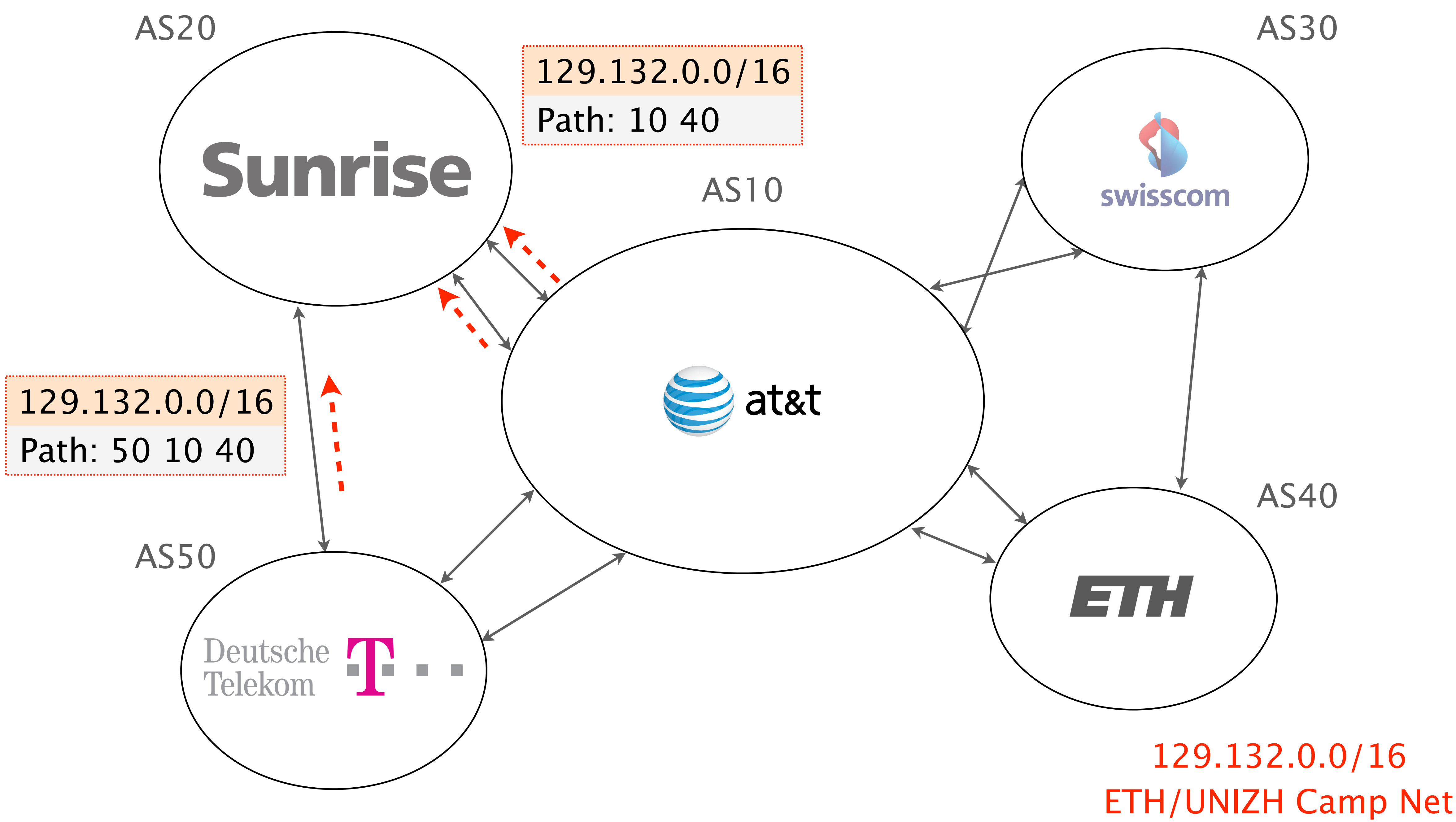
ASes exchange information about the destinations (IP addresses) they can reach

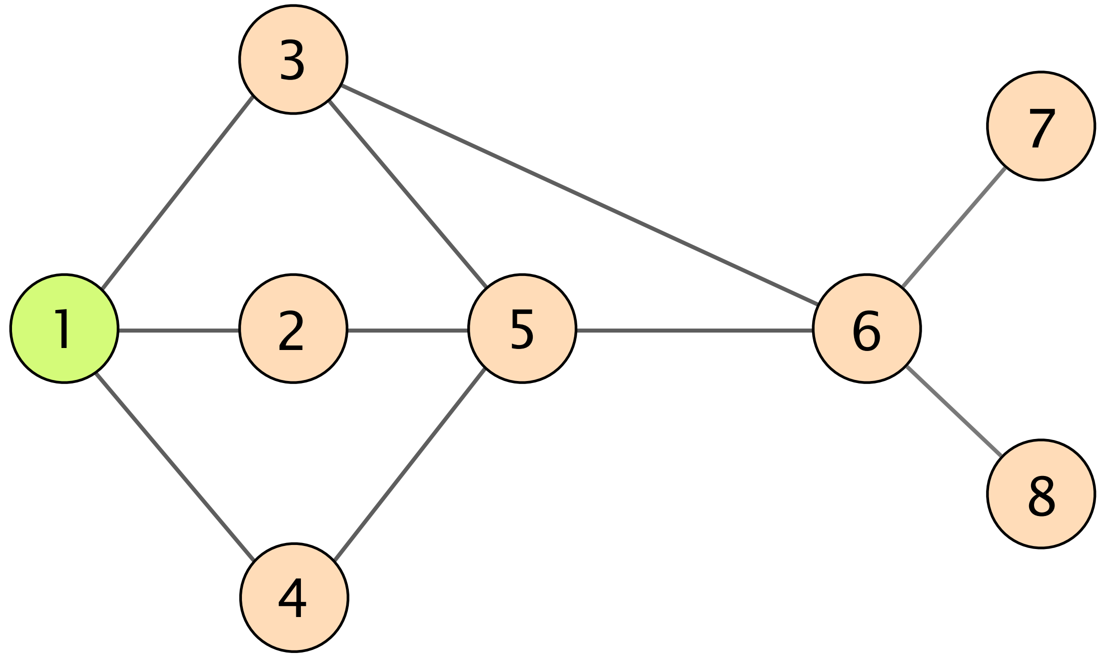


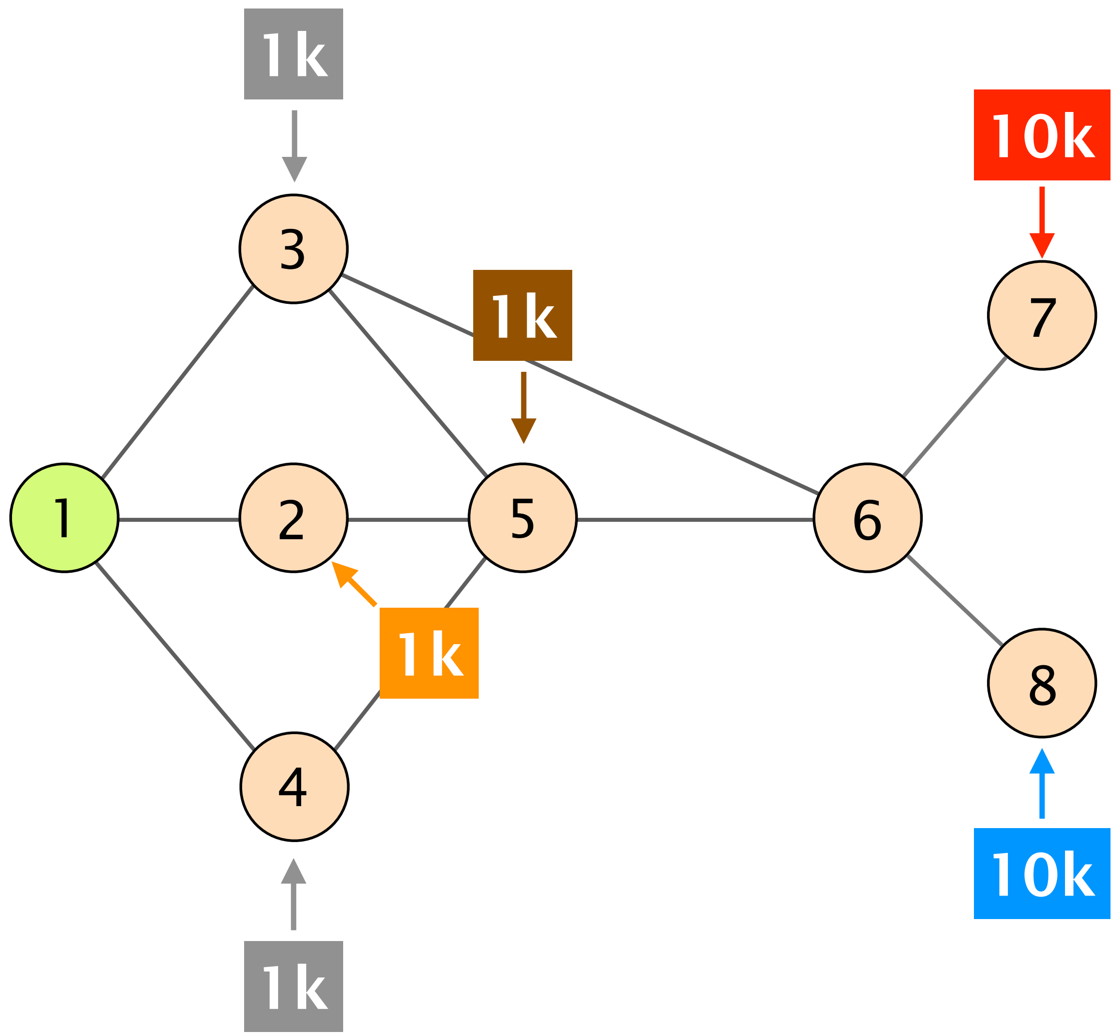
Reachability information is propagated hop-by-hop



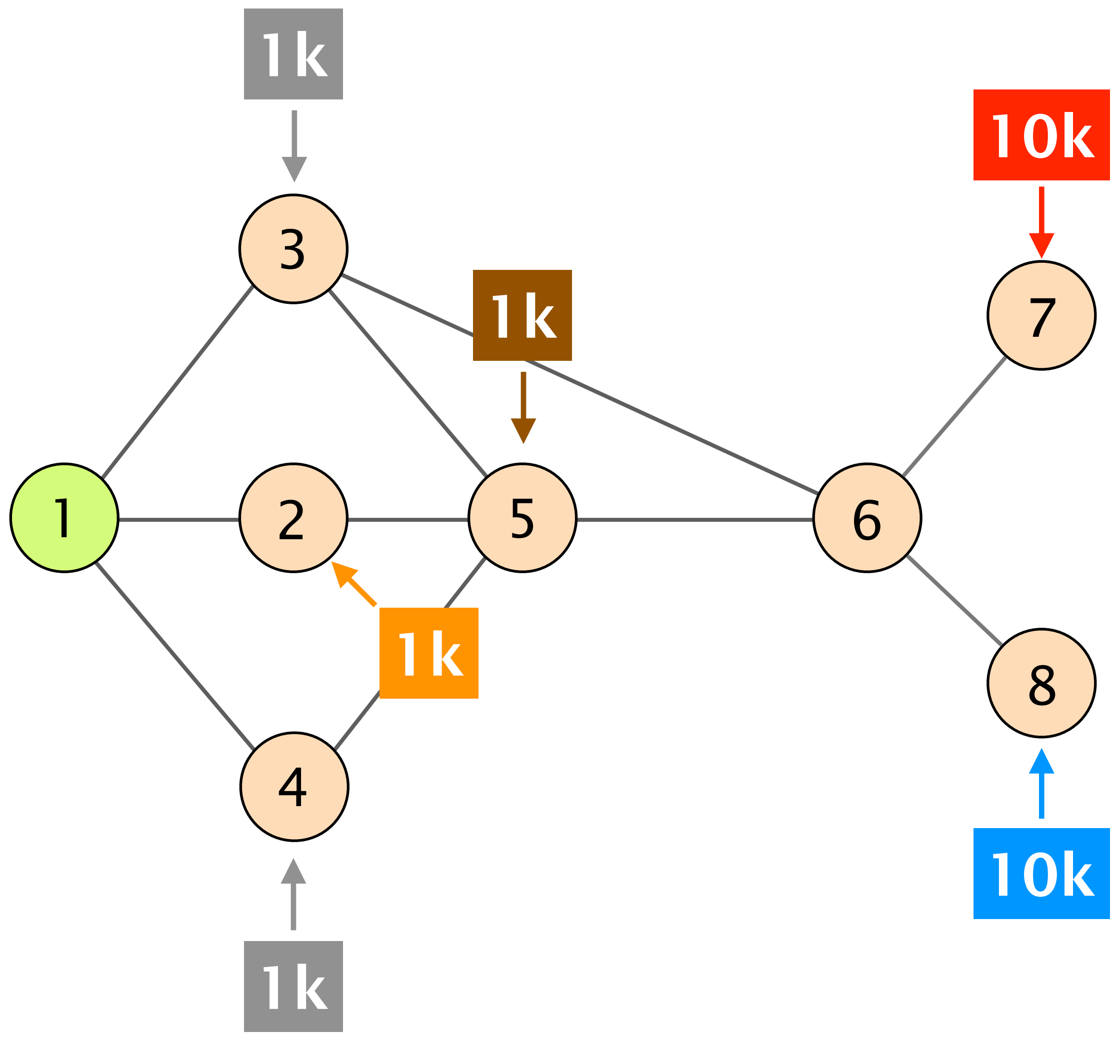
Reachability information is propagated hop-by-hop

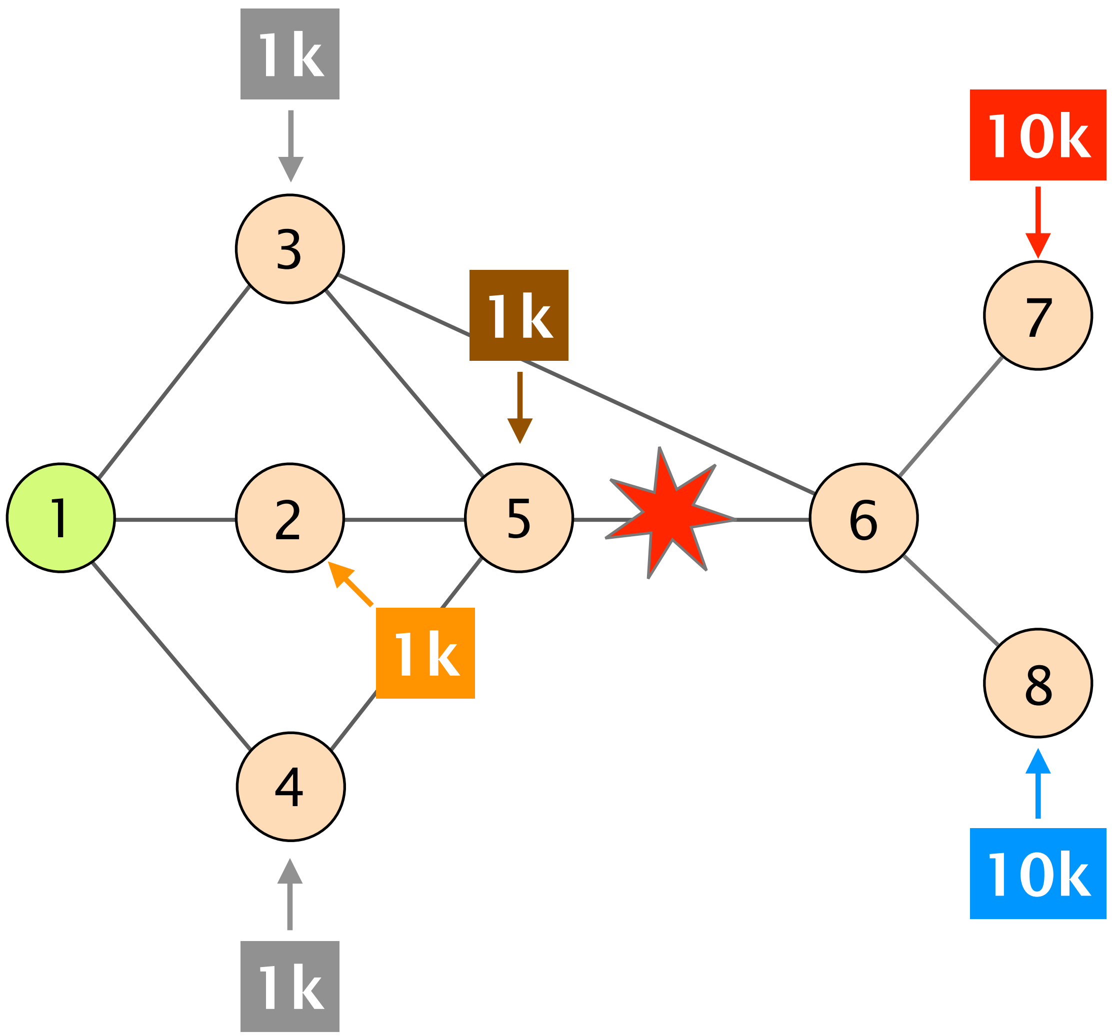


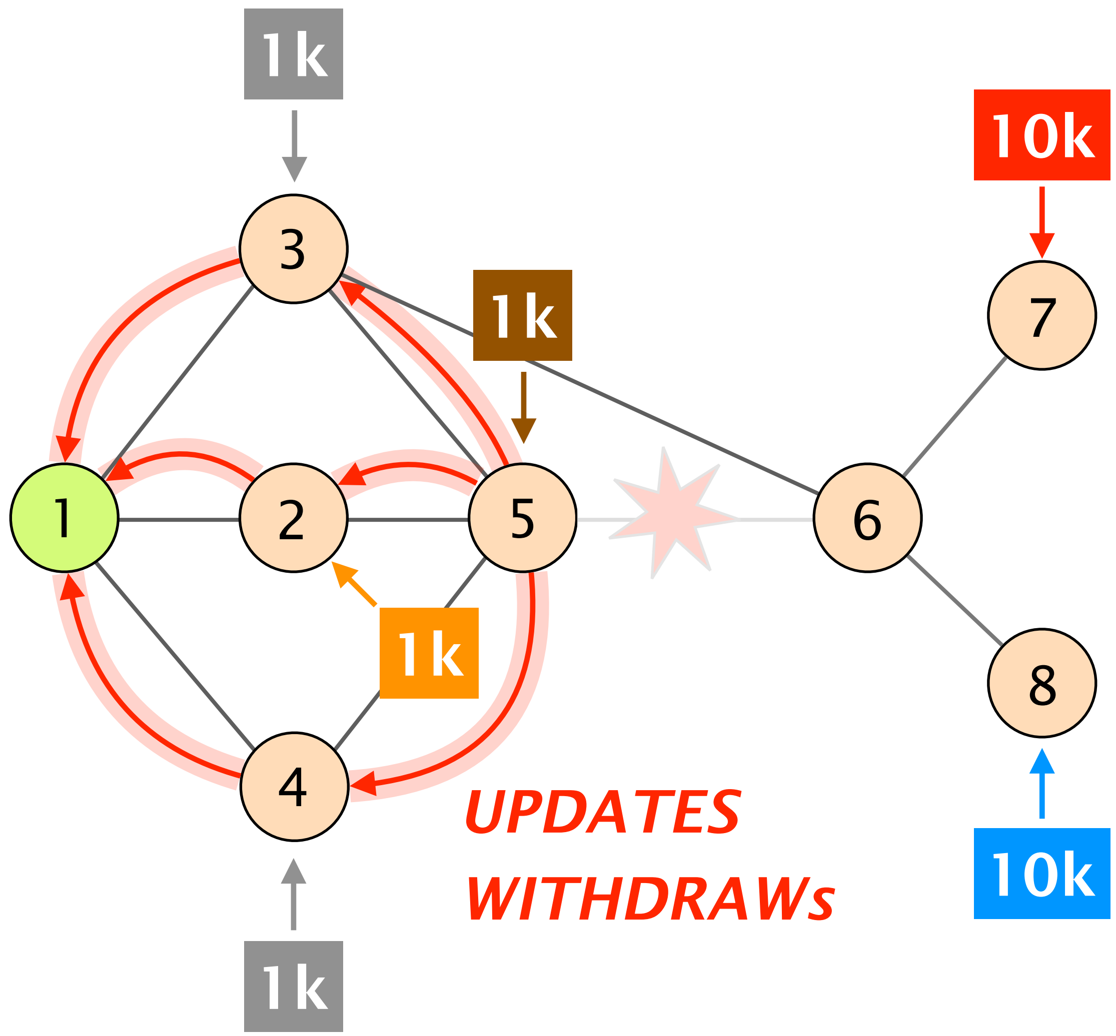

1
2
3
4
5
6
7
8



1k
10k
3
1k
7
1
2
5
6
1k
8
4
10k
1k


1k
10k
3
1k
7
1
2
5
6
1k
8
4
10k
1k


1k
10k
3
1k
7
1
2
5
6
1k
8
4
10k
1k



1k
10k
3
1k
7
1
2
5
6
1k
8
4
UPDATES
WITHDRAWs
10k
1k

The stream of messages following a disruption contain **redundant information** about the failed resource

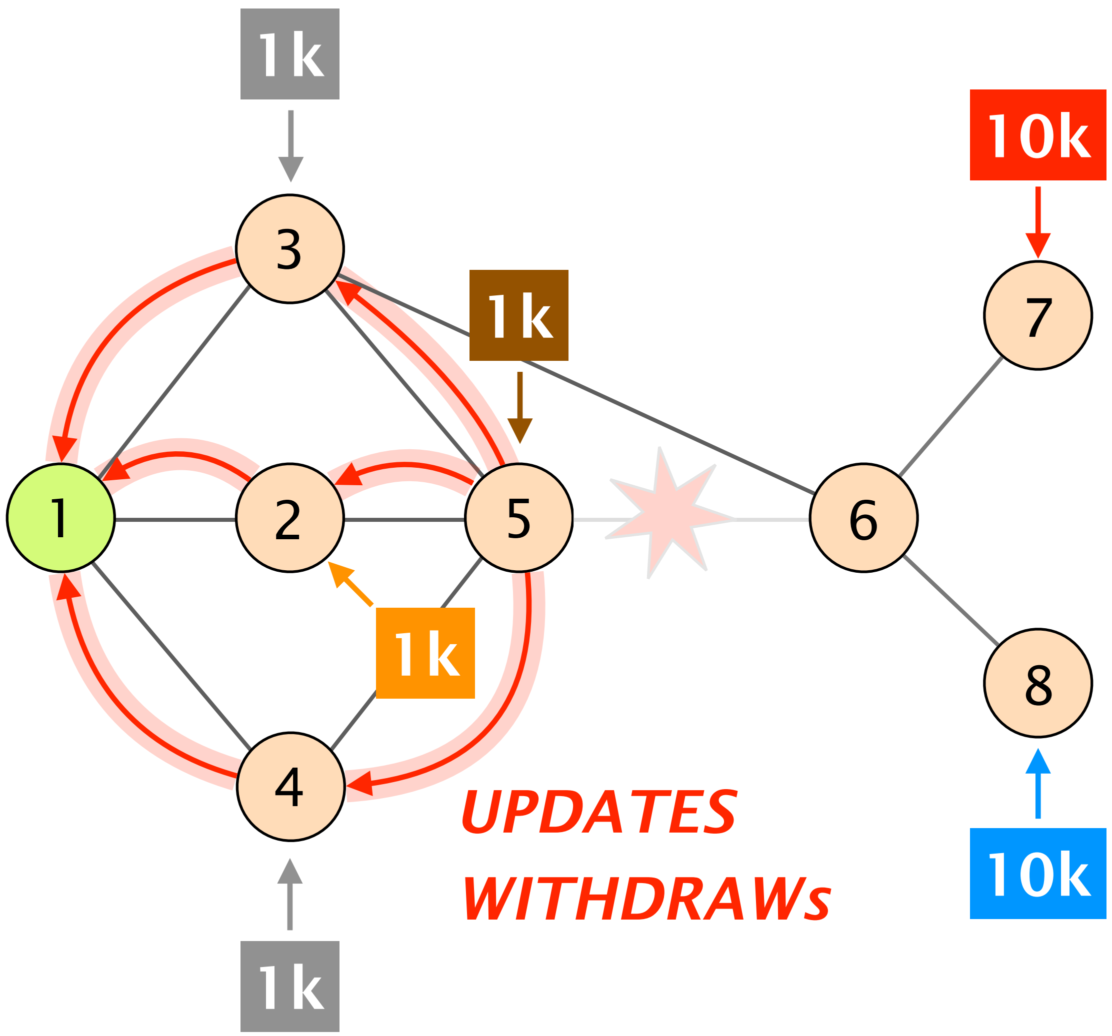
The stream of messages following a disruption contain redundant information about the failed resource

enables prediction

Feedback comes in two forms:

positive or *negative*

positive	unaffected prefixes are routed on paths which do not contain the failed link
negative	affected prefixes were routed on paths which did contain the failed link



affected prefixes:

(1 2 5 6 7) 10k

(1 2 5 6 8) 10k

unaffected prefixes:

(1 2) 1k

(1 2 5) 1k

affected prefixes:

(1 2 5 6 7) 10k

(1 2 5 6 8) 10k

order

unaffected prefixes:

(1 2) 1k

(1 2 5) 1k

After receiving messages pertaining to one path, the failure can be located anywhere alongside it

candidates set:

(1,2); (2,5); (5,6); (6,7)

affected prefixes:

(1 2 5 6 7)	10k
(1 2 5 6 8)	10k

unaffected prefixes:

(1 2)	1k
(1 2 5)	1k

After receiving messages pertaining to multiple paths, the failure is likely located in their intersections

candidates set:

(1,2); (2,5); (5,6); (6,7)

(1,2); (2,5); (5,6);

affected prefixes:

(1 2 5 6 7) 10k

(1 2 5 6 8) 10k

unaffected prefixes:

(1 2) 1k

(1 2 5) 1k

Not receiving messages pertaining to given links is a strong indication that they are still up

candidates set:	affected prefixes:	
(1,2); (2,5); (5,6); (6,7)	(1 2 5 6 7)	10k
(1,2); (2,5); (5,6);	(1 2 5 6 8)	10k

	unaffected prefixes:	
(2,5); (5,6);	(1 2)	1k
(5,6)	(1 2 5)	1k

candidates set:

(1,2); (2,5); (5,6); (6,7)

(1,2); (2,5); (5,6);

(2,5); (5,6);

prediction (5,6)

affected prefixes:

(1 2 5 6 7) 10k

(1 2 5 6 8) 10k

unaffected prefixes:

(1 2) 1k

(1 2 5) 1k

SWIFT predicts link failures in two stages:

burst detection and link identification

Step 1

burst detection

Step 1
burst detection

Whenever the frequency of WITHDRAWALs is higher than a threshold (e.g., >99th percentile)

Step 1
burst detection

Whenever the frequency of WITHDRAWALs is higher than a threshold (e.g., >99th percentile)

Step 2
link identification

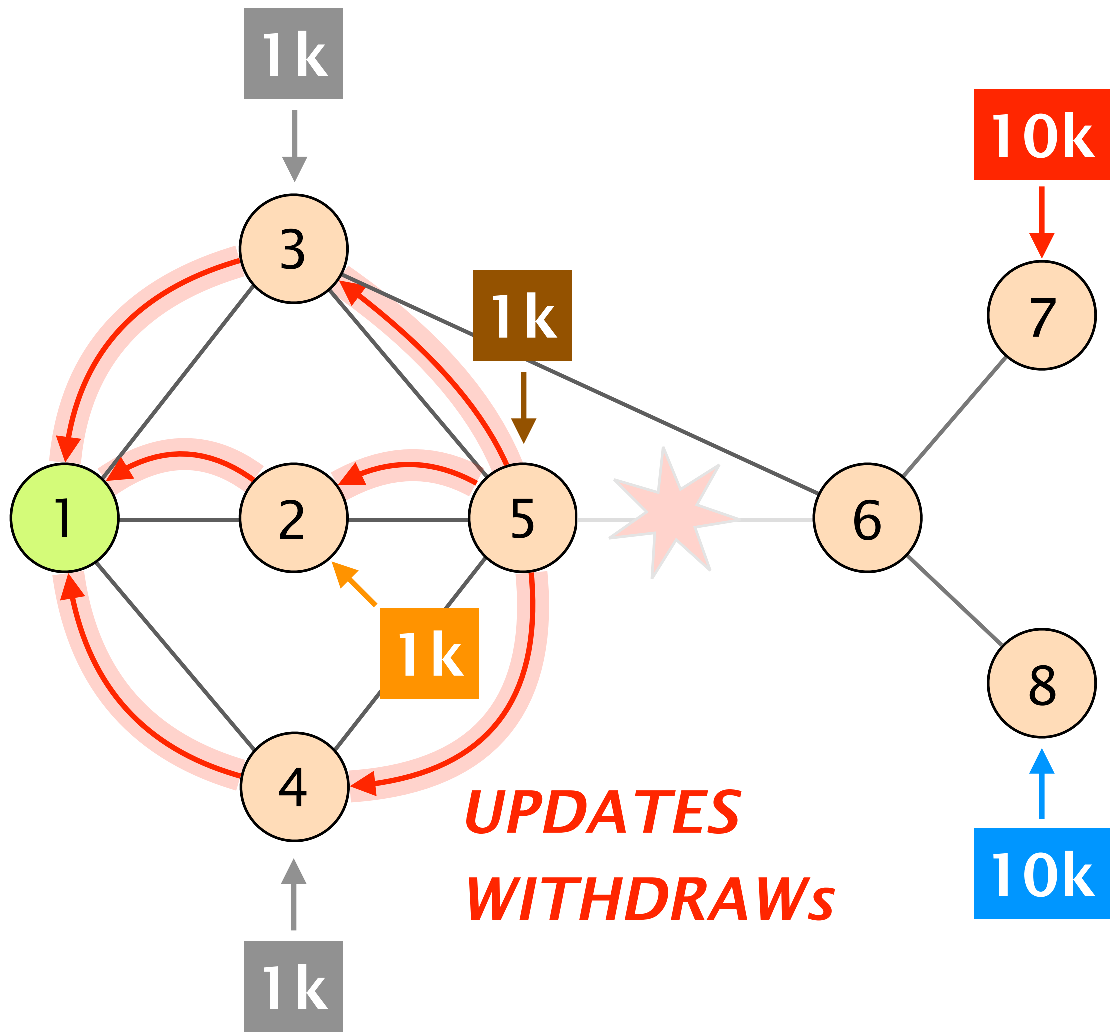
Step 1
burst detection

Whenever the frequency of WITHDRAWALs is higher than a threshold (e.g., >99th percentile)

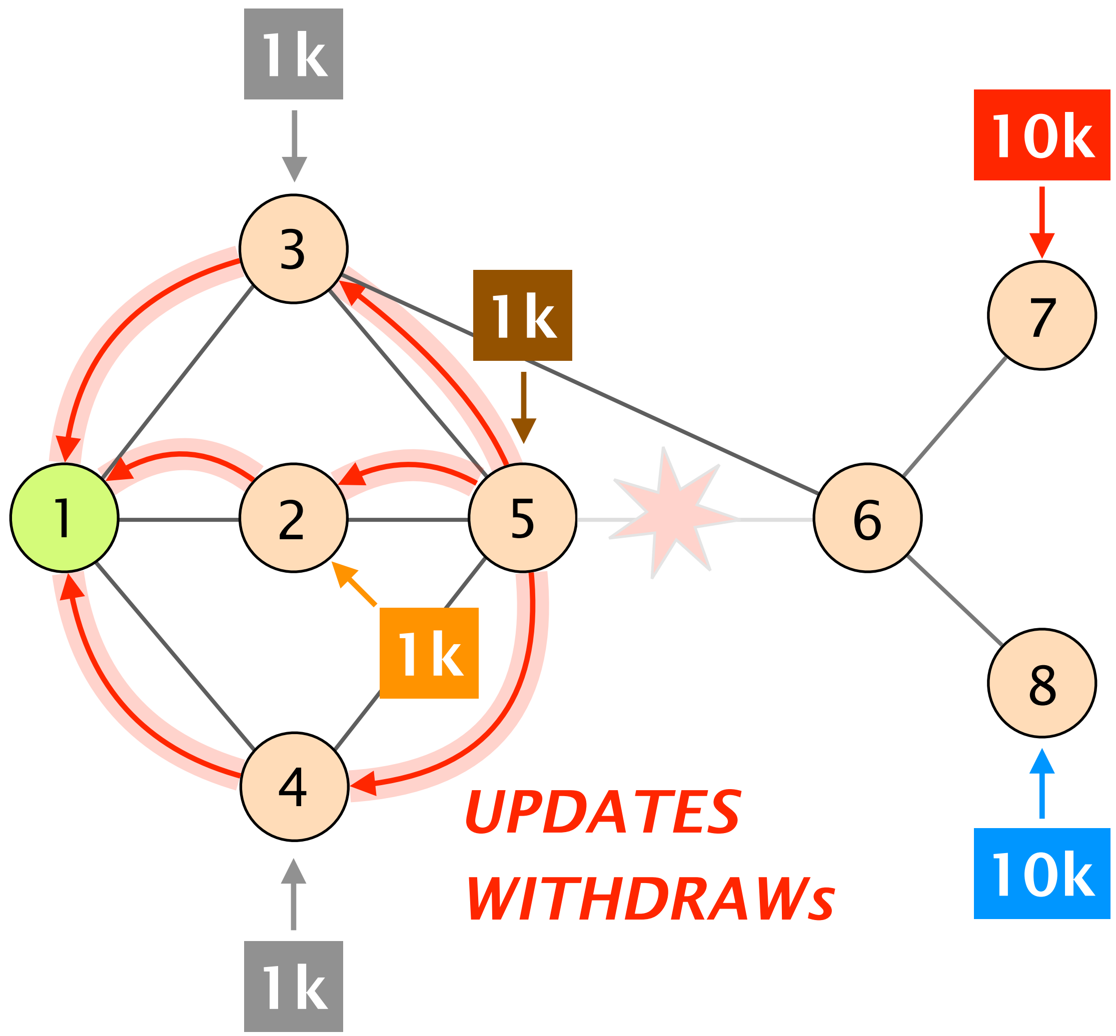
Step 2
link identification

Return the link(s) that maximizes the weighted geometric mean between:

Withdrawal share $WS(l,t)$	Path share $PS(l,t)$
fraction of withdraws crossing link *l*	proportion of prefixes withdrawn on link *l*

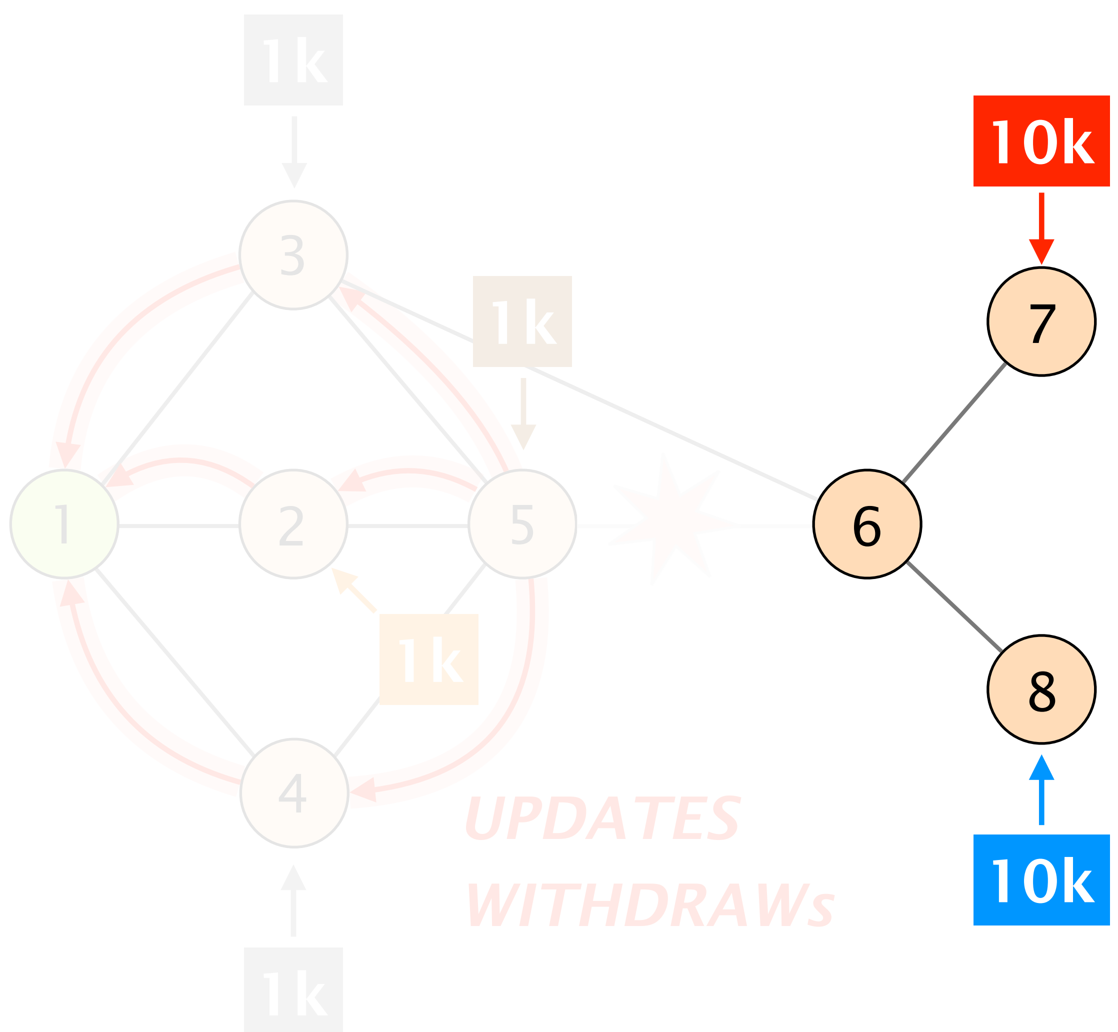


link	WS	PS	FS
(1,2)			
(2,5)			
(5,6)			
(6,7)			
(6,8)			
other			



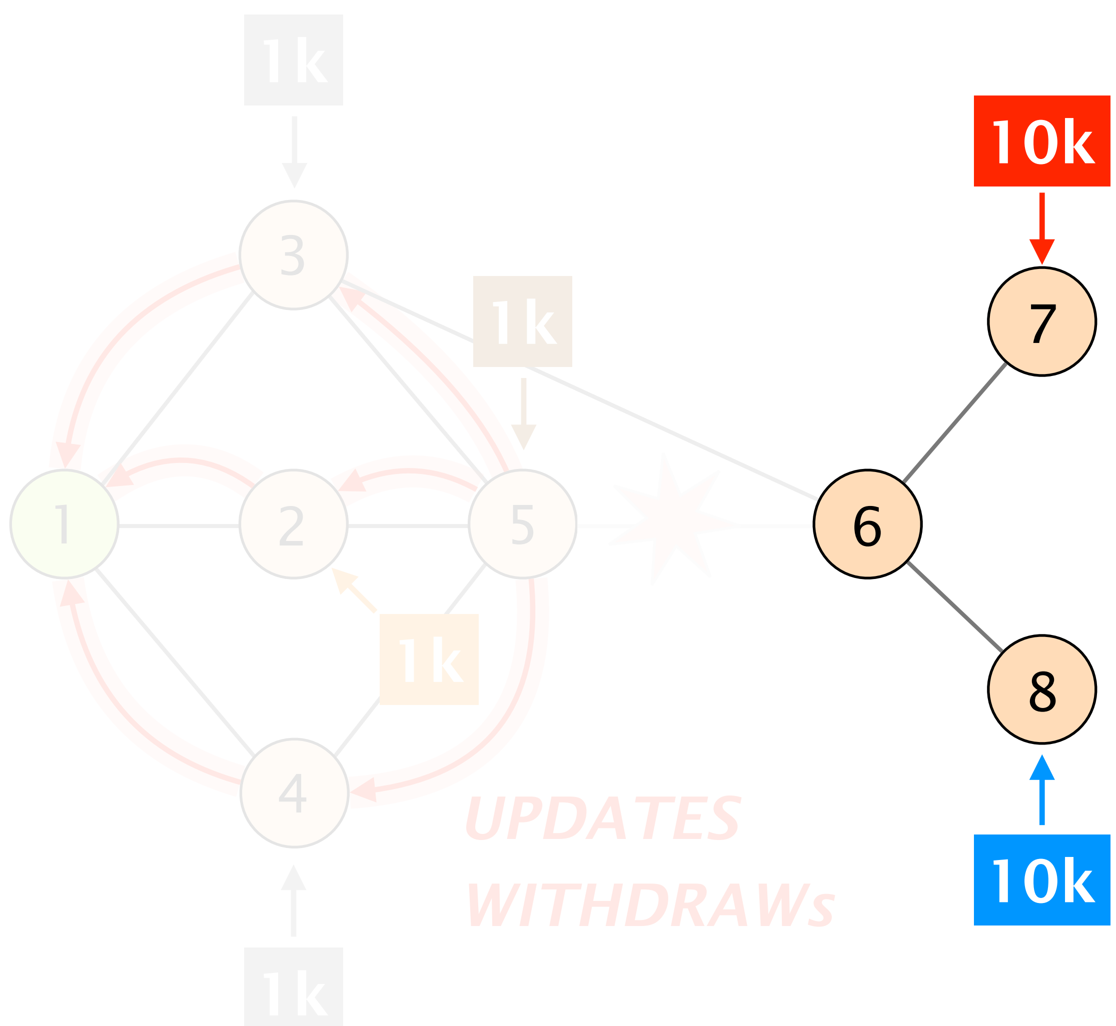
link	WS	PS	FS
(1,2)	1	.91	.95
(2,5)	1	.95	.97
(5,6)	1	1	1
(6,7)	.5	1	.7
(6,8)	.5	1	.7
other	0	0	0

From 1's perspective, 50% of the prefixes appearing in the messages were going over (6,7) and (6,8)



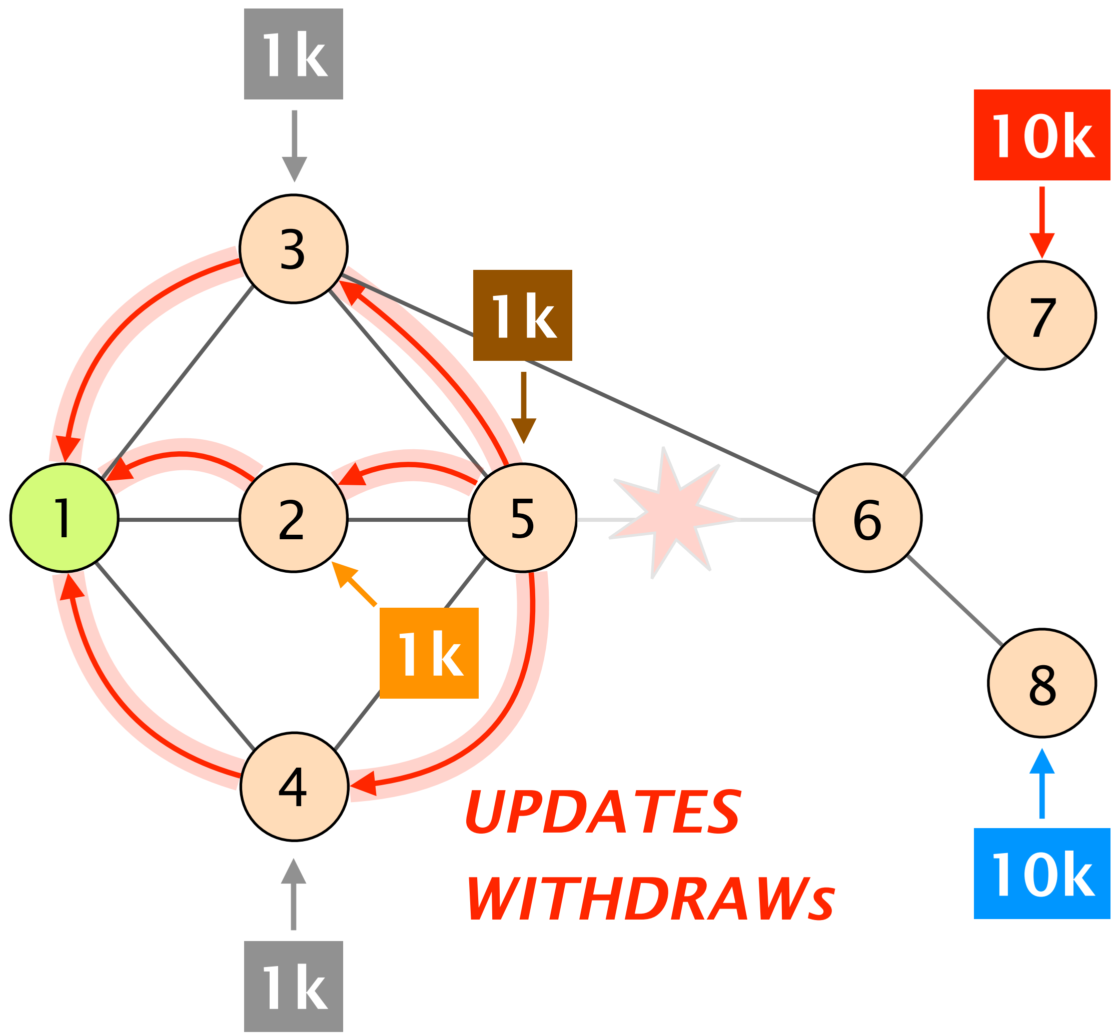
link	WS	PS	FS
(1,2)	1	.91	.95
(2,5)	1	.95	.97
(5,6)	1	1	1
(6,7)	.5	1	.7
(6,8)	.5	1	.7
other	0	0	0

From 1's perspective, 100% of the prefixes going over (6,7) and (6,8) are down



link	WS	PS	FS
(1,2)	1	.91	.95
(2,5)	1	.95	.97
(5,6)	1	1	1
(6,7)	.5	1	.7
(6,8)	.5	1	.7
other	0	0	0

Only the failed link has 1 for both metric



link	WS	PS	FS
(1,2)	1	.91	.95
(2,5)	1	.95	.97
(5,6)	1	1	1
(6,7)	.5	1	.7
(6,8)	.5	1	.7
other	0	0	0

When run on the full burst, *SWIFT* is **guaranteed** to find the right link

Theorem

If all ASes inject at least one prefix, *SWIFT* will always pinpoint the failed link

When run on the full burst

not that helpful…

Yet, *SWIFT* work well in realistic scenarios

Intuition

Messages tend to be interleaved

providing diverse path information early on

SWIFT compensates for lack of information,
by being **overly cautious** and **iterative**

Returns set of links failures

all links with high fit score

Runs multiple times sequentially

after 2.5k, 5k, 7.5k, 10k,… messages

Returns set of links failures

all links with high fit score

Runs multiple times sequentially

after 2.5k, 5k, 7.5k, 10k,... messages

Increase the number of **false positives**

the amount of wrongly rerouted traffic

Good news

False positives are **not** an issue!

26 seconds *vs* 129 600 seconds

allowed downtime
for 99.999%

allowed free-riding
on a peering link

SWIFT prediction are accurate early on in the burst

dataset

2619 large bursts identified in July 2016

acts as “ground truth”

methodology

iteratively run *SWIFT* prediction

on a growing subset of each burst

compare *SWIFT* prediction

with the actual WITHDRAWs

SWIFT predicts ~90% of the withdrawn prefixes based on only 2.5k messages

	50th	75th	90th
2.5K	**87.50%**	99.10%	99.99%
5.0K	89.70%	98.80%	98.99%
7.5K	92.99%	99.10%	99.99%
10K	95.40%	99.60%	99.99%

SWIFT reroutes few non-disrupted prefixes

despite not being optimized for it

	50th	75th	90th
2.5K	**0.2x**	1.4x	8.9x
5.0K	0.2x	1.6x	7.2x
7.5K	0.2x	1.8x	7.8x
10K	0.4x	2.8x	9.6x

SWIFT: Predictive Fast Rerouting

Predicting

out of few messages

2 Updating

groups of entries

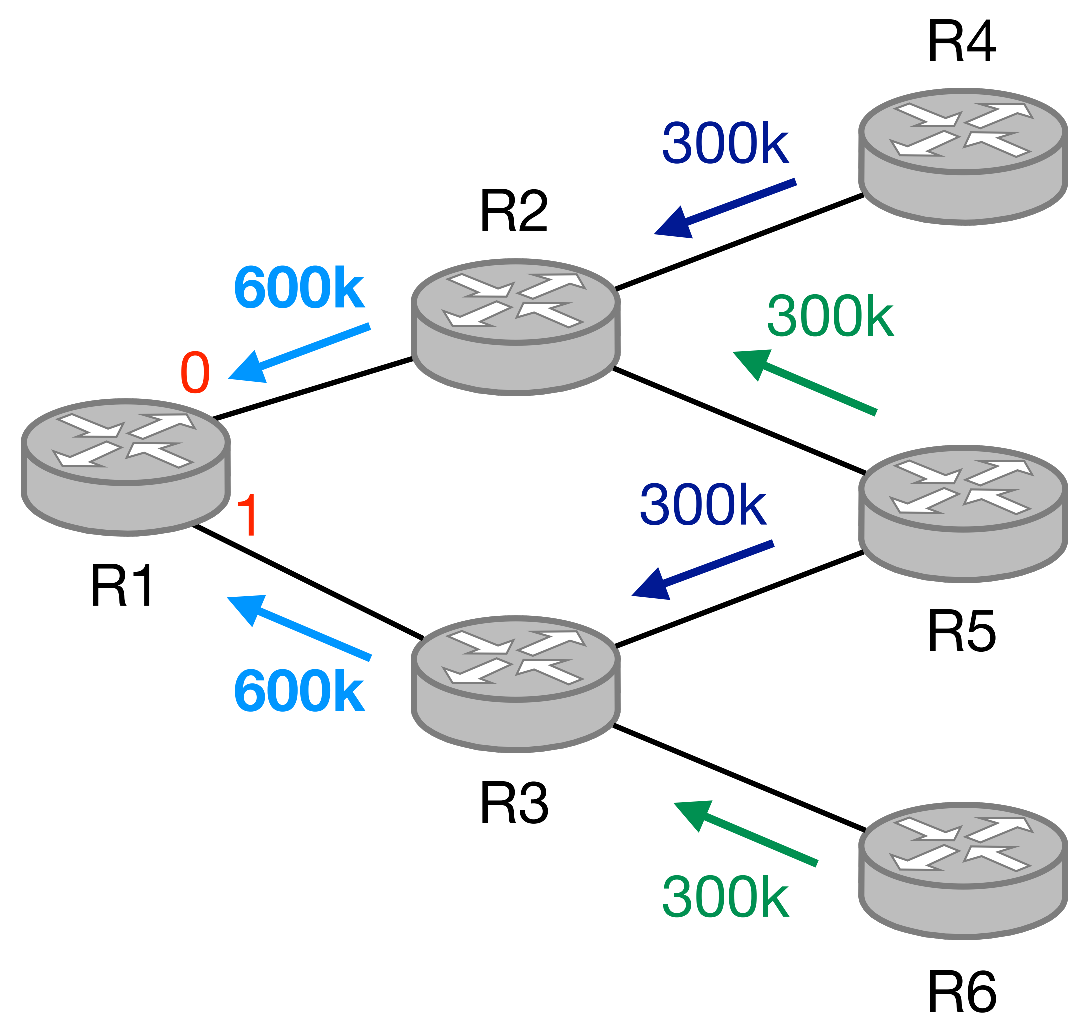
Supercharging

existing systems

Upon a prediction, *SWIFT* needs to update the corresponding forwarding entries

R1's Forwarding Table

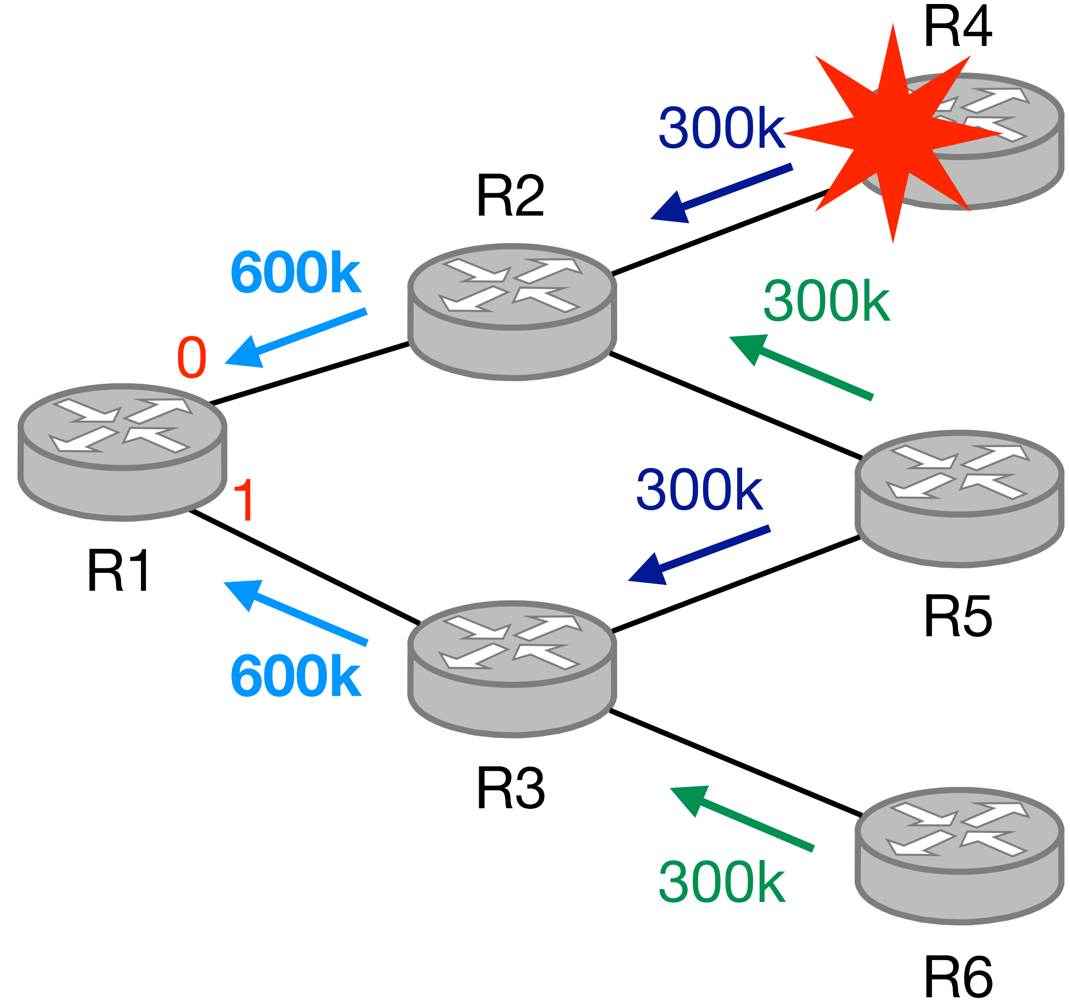
1	1.0.0.0/24	0
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



Upon the prediction of (R2, R4) going down, 300k entries have to be updated to 1

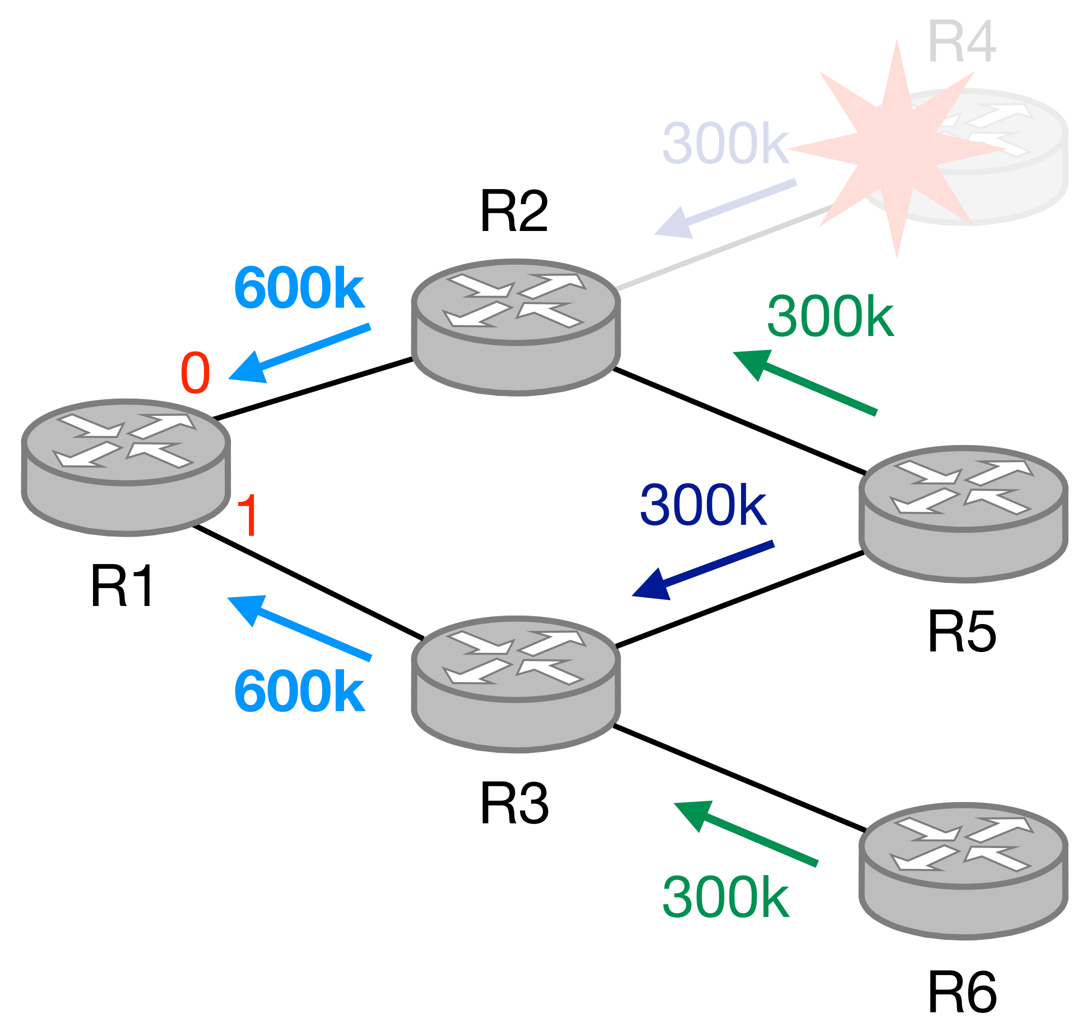
R1's Forwarding Table

1	1.0.0.0/24	0
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



forwarding entries
going over the failed link

1	1.0.0.0/24	0
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0



The problem is that forwarding tables are flat

Entries do not share *any* information

even if they are identical

Upon failure, all of them have to be updated

inefficient, but also unnecessary

The problem is that forwarding tables are flat

Entries do not share *any* information

even if they are identical

Upon failure, all of them have to be updated

inefficient, but also unnecessary

Solution: introduce a hierarchy

as with any problem in CS…

Replace this…

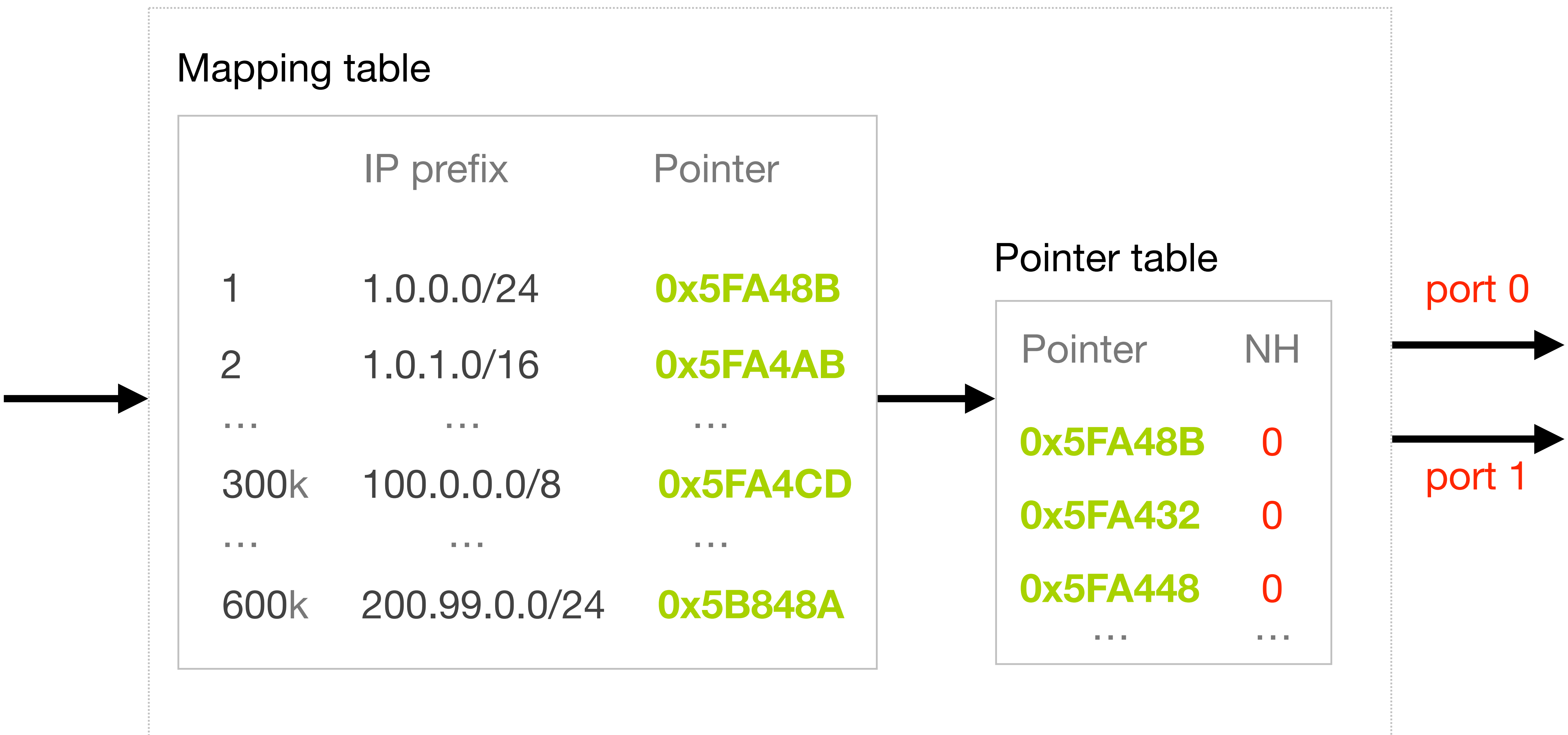
R1’s Forwarding Table

	IP prefix	Next-Hop
1	1.0.0.0/24	0
2	1.0.1.0/16	0
…	…	…
300k	100.0.0.0/8	0
…	…	…
600k	200.99.0.0/24	0

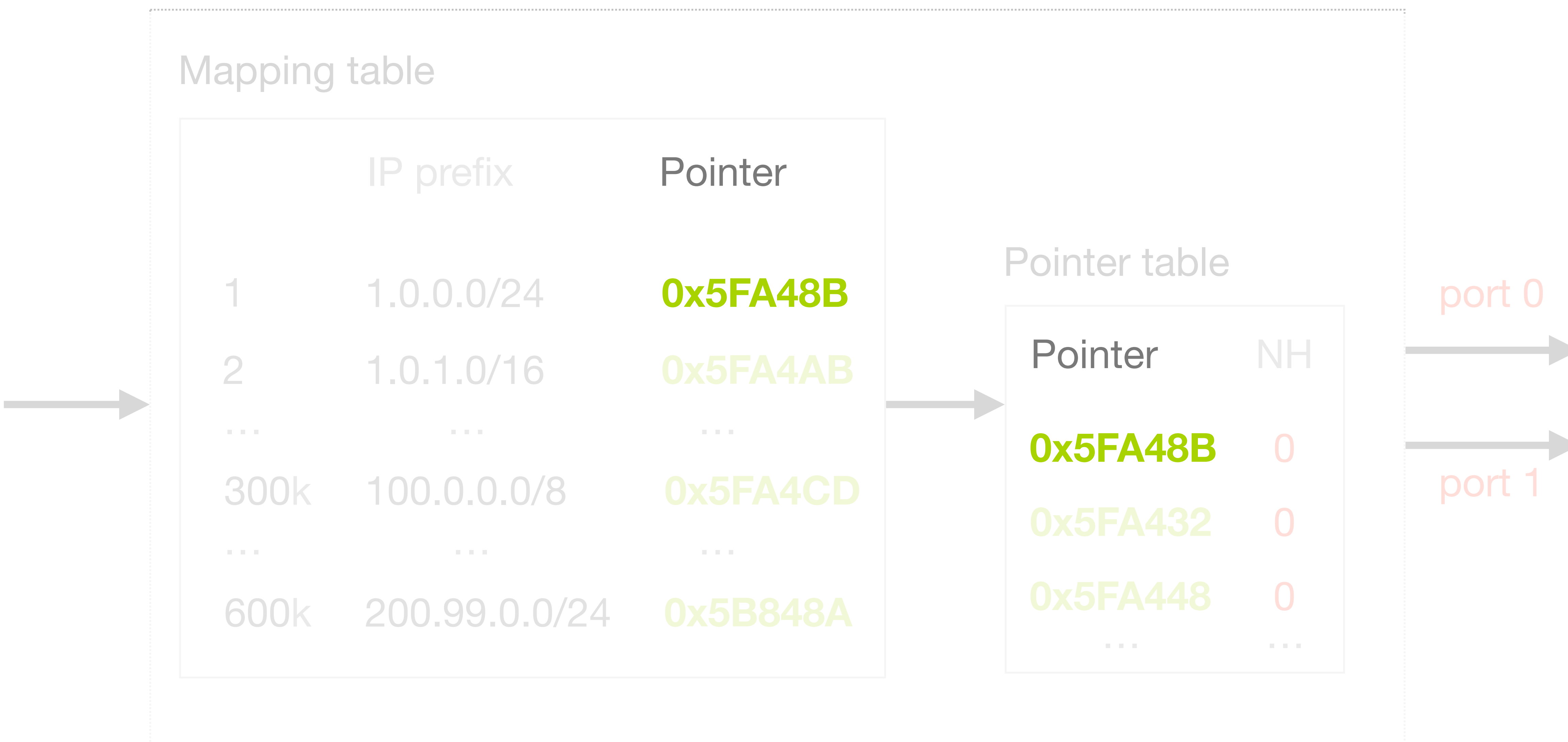
port 0

port 1

… with this


Mapping table
IP prefix
Pointer
1
1.0.0.0/24
0x5FA48B
2
1.0.1.0/16
0x5FA4AB
…
300k
100.0.0.0/8
0x5FA4CD
…
600k
200.99.0.0/24
0x5B848A
Pointer table
Pointer
NH
0x5FA48B
0
0x5FA432
0
0x5FA448
0
…
port 0
port 1

SWIFT computes the pointer attached to each prefix in function of the links it crosses in the Internet



pointer 0x5FA48B

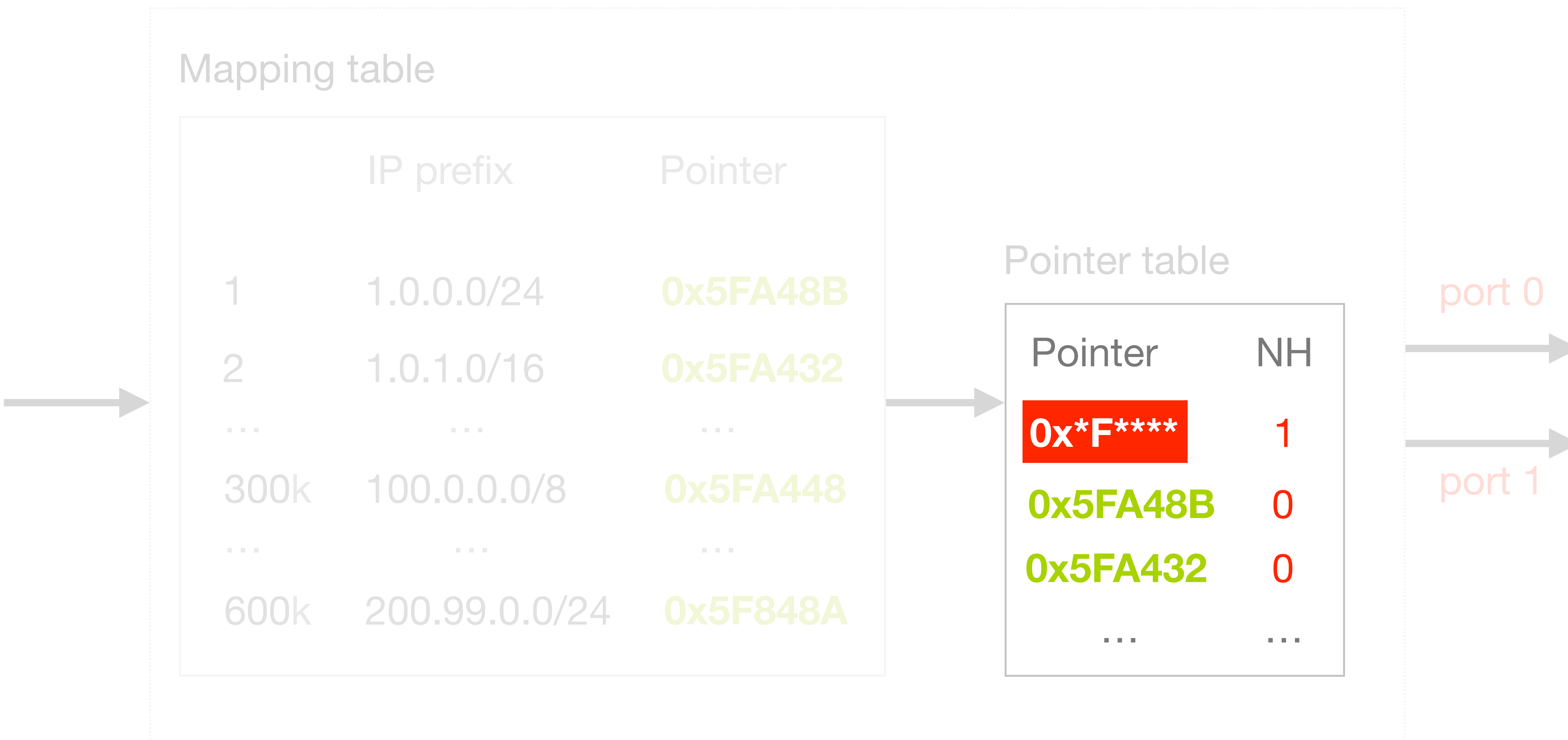
pointer 0x5FA48B

5***** indicates a prefix
crossing the link (1,2)

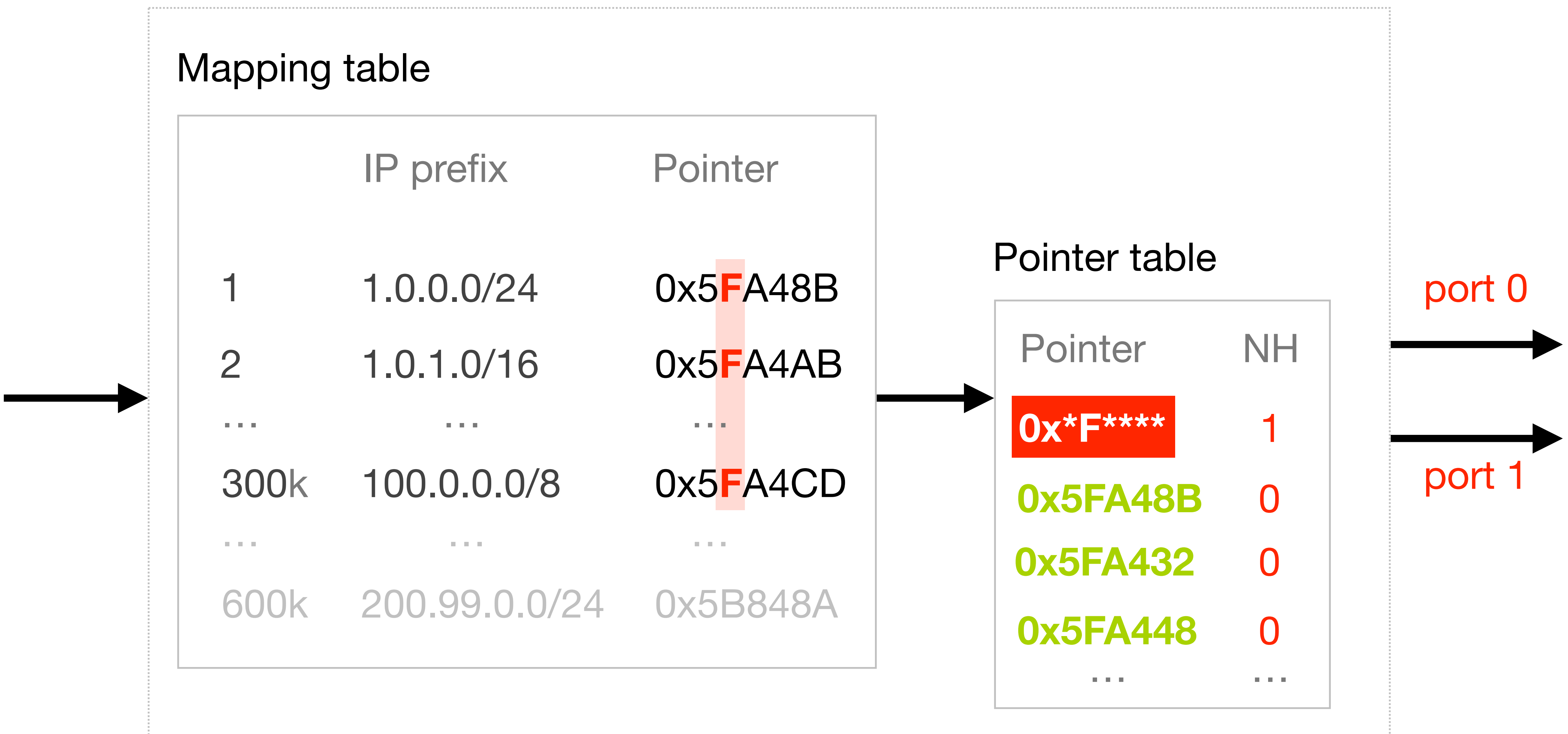
pointer 0x5**F**A48B

*F**** indicates a prefix
crossing the link (2,4)

Upon the predicted failure of link (2,4),
SWIFT add a rerouting rules for 0x*F**** to port 1



This rule automatically reroutes 300k prefixes



Pointers are encoded on 48 bits, *SWIFT* therefore reduce the graph before encoding the links

Recipe #1

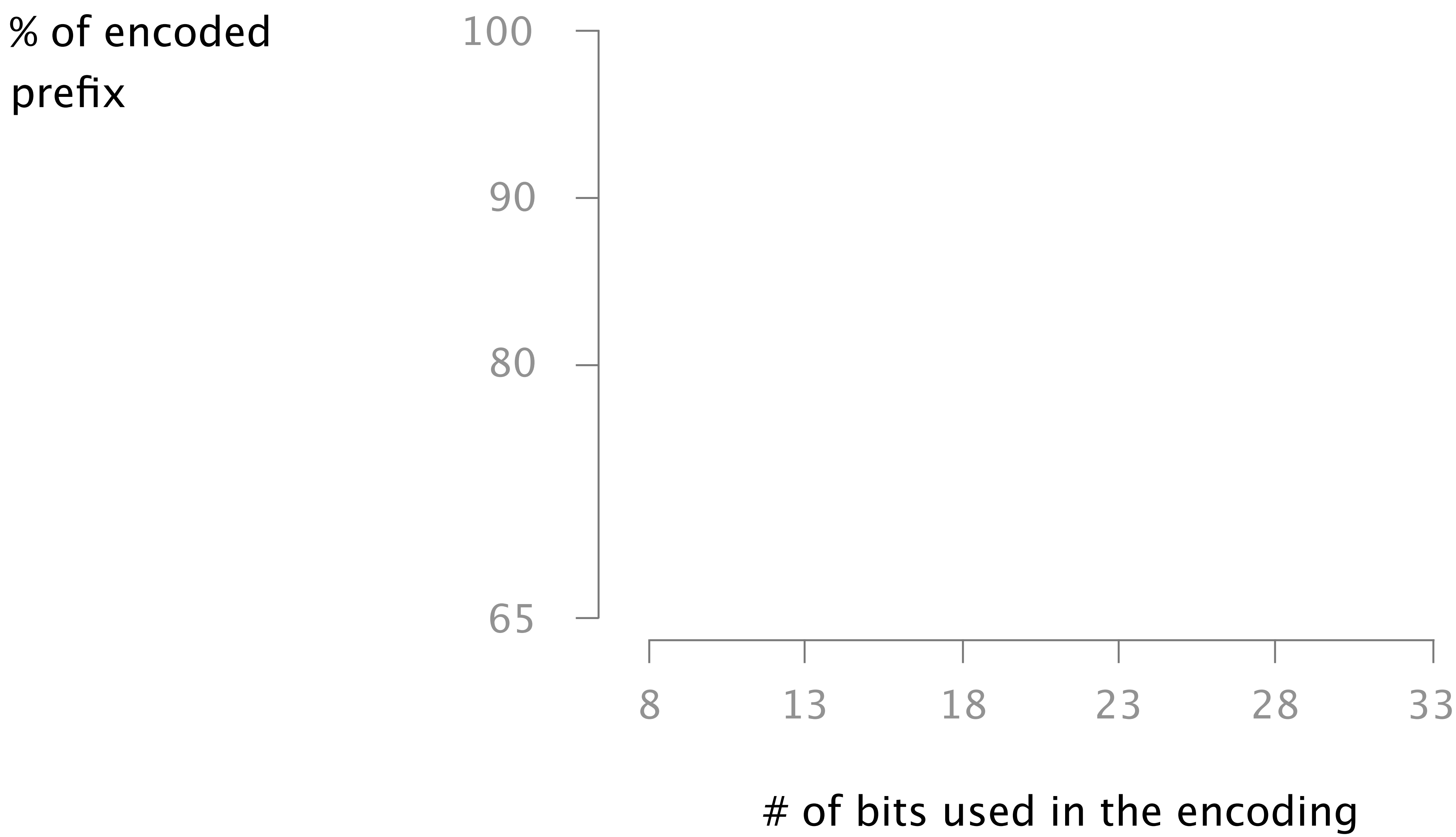
Ignore any link seeing less than 1.5k pfxes

anything less converges fast enough already

Recipe #2

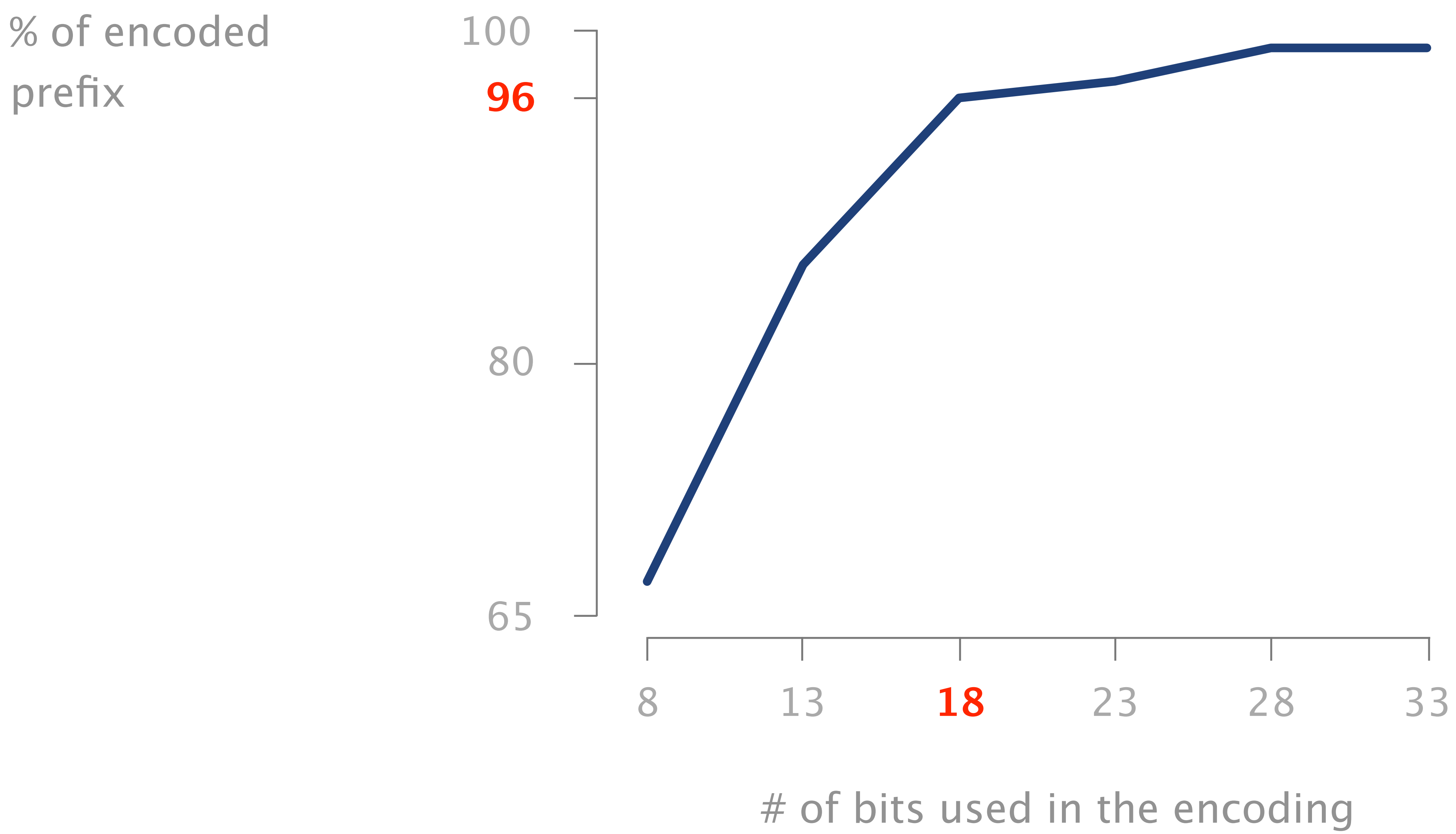
Ignore link far away from the SWIFTed node

less likely to create large bursts of WITHDRAWs


% of encoded
prefix
100
90
80
65
8
13
18
23
28
33
# of bits used in the encoding

SWIFT encoding enables to reroute

96% of the prefixes with only 18 bits



SWIFT: Predictive Fast Rerouting

Predicting

out of few messages

Updating

groups of entries

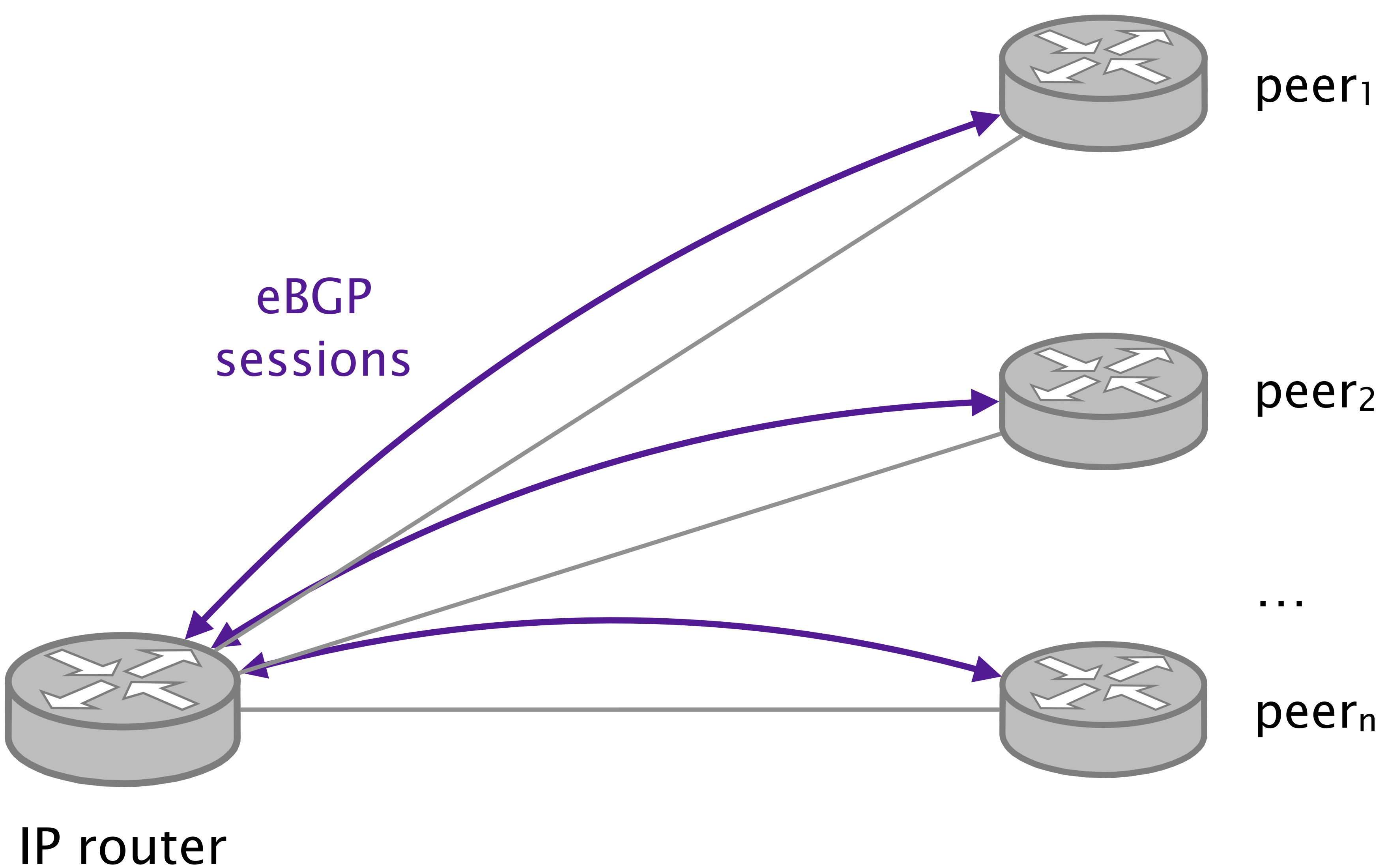
3 Supercharging

existing systems

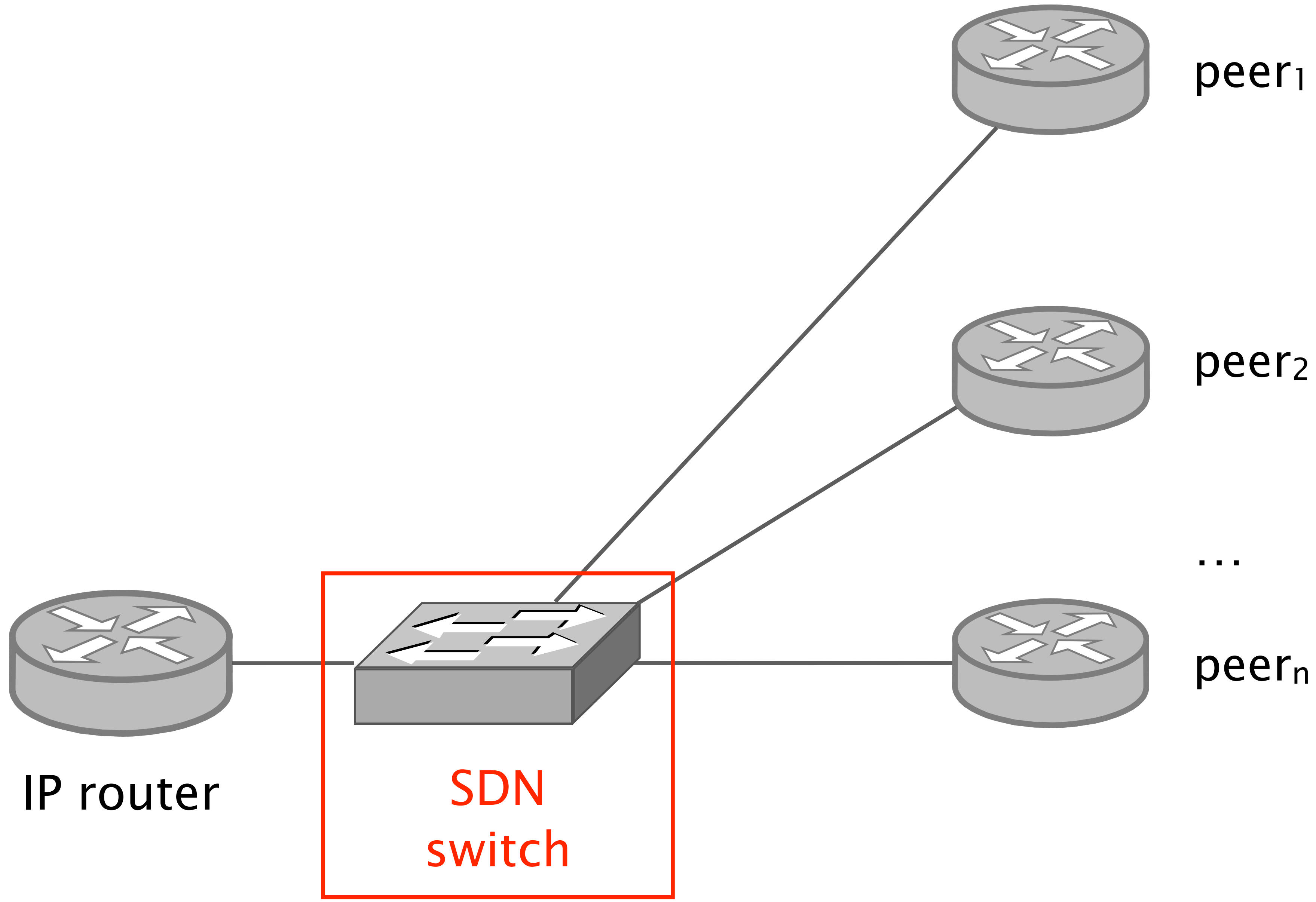
We implemented a full *SWIFT* prototype
and boosted existing routers' convergence

To boost the performance of a router,

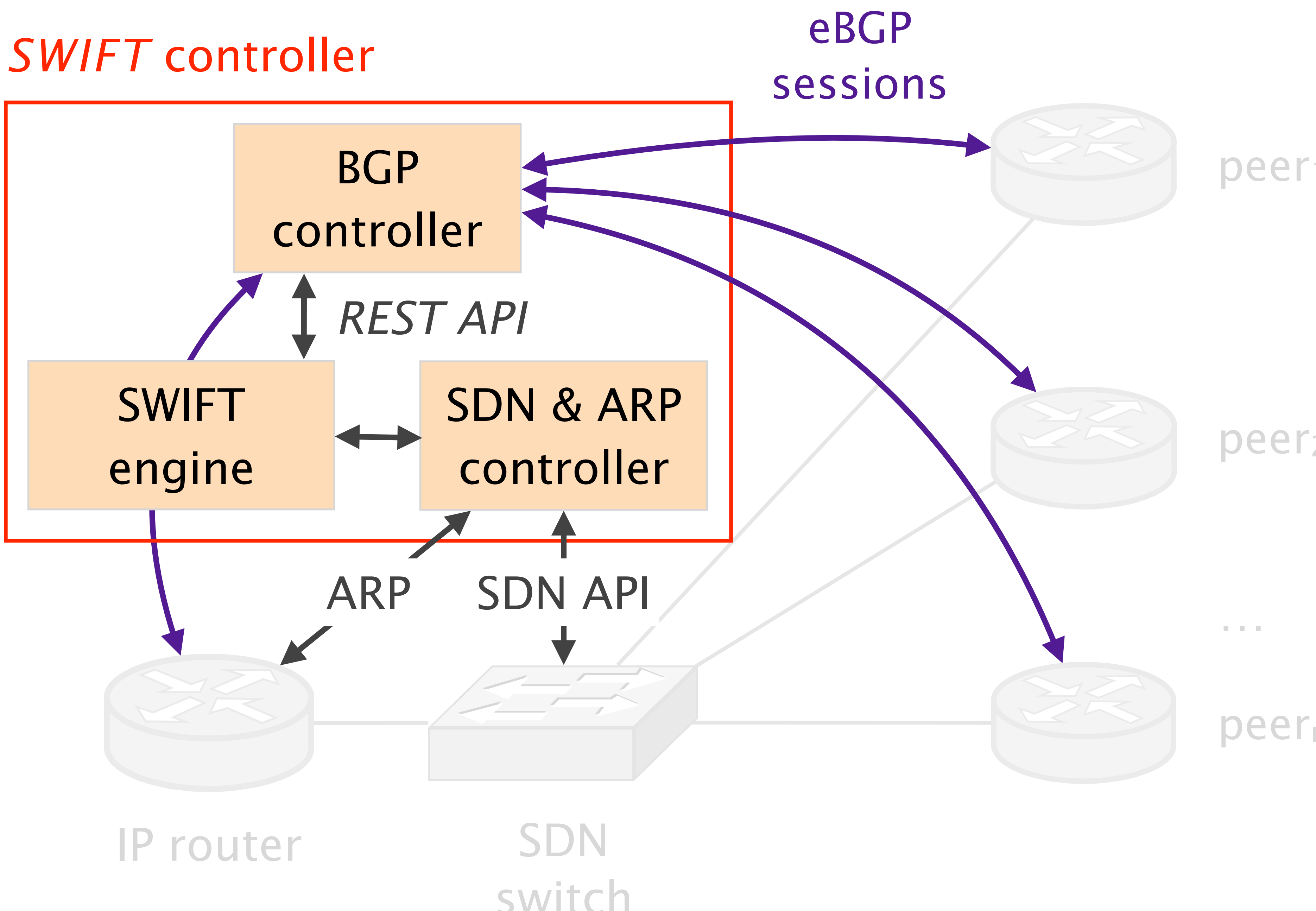
we augmented its control-plane and data-plane

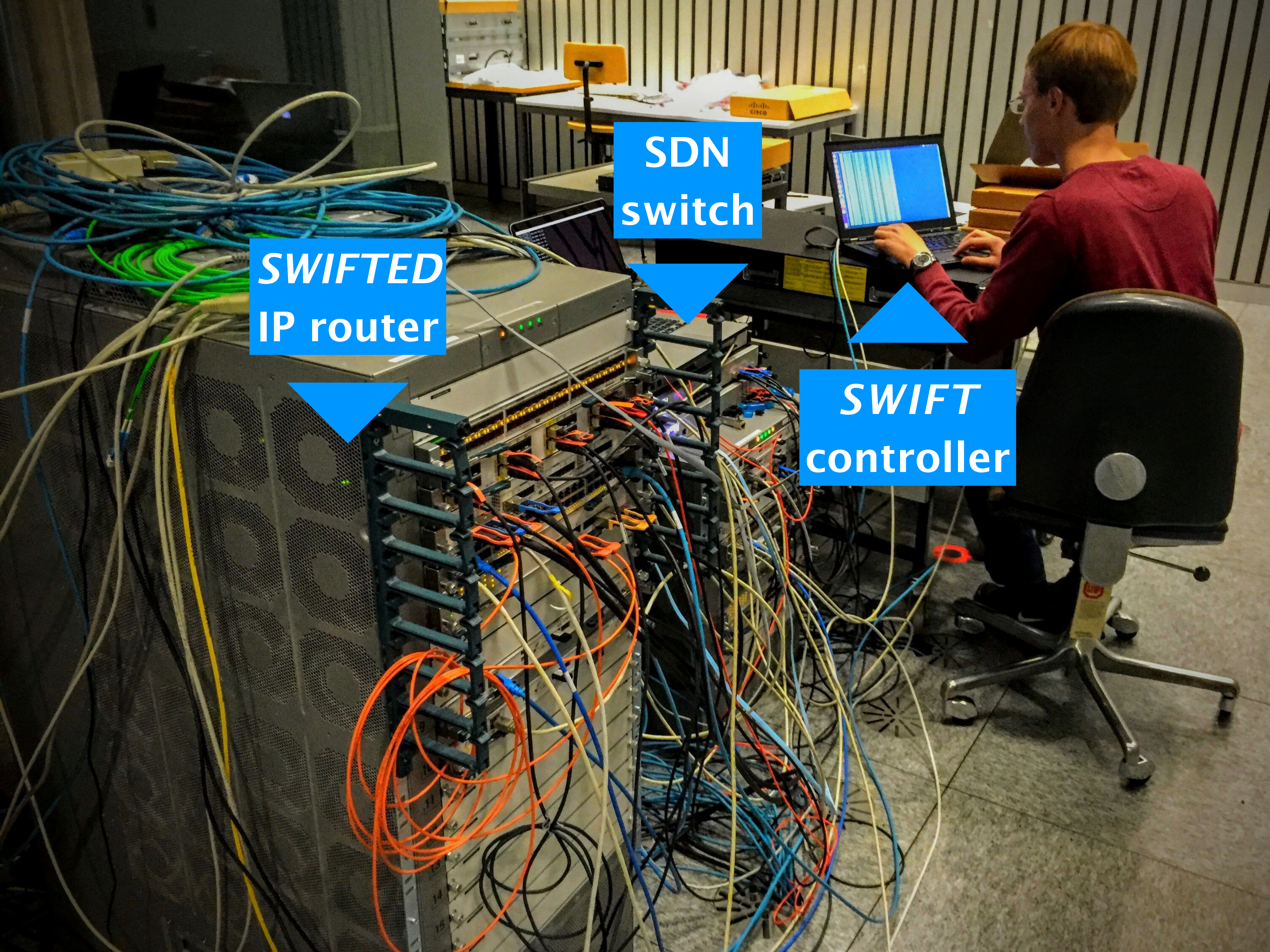


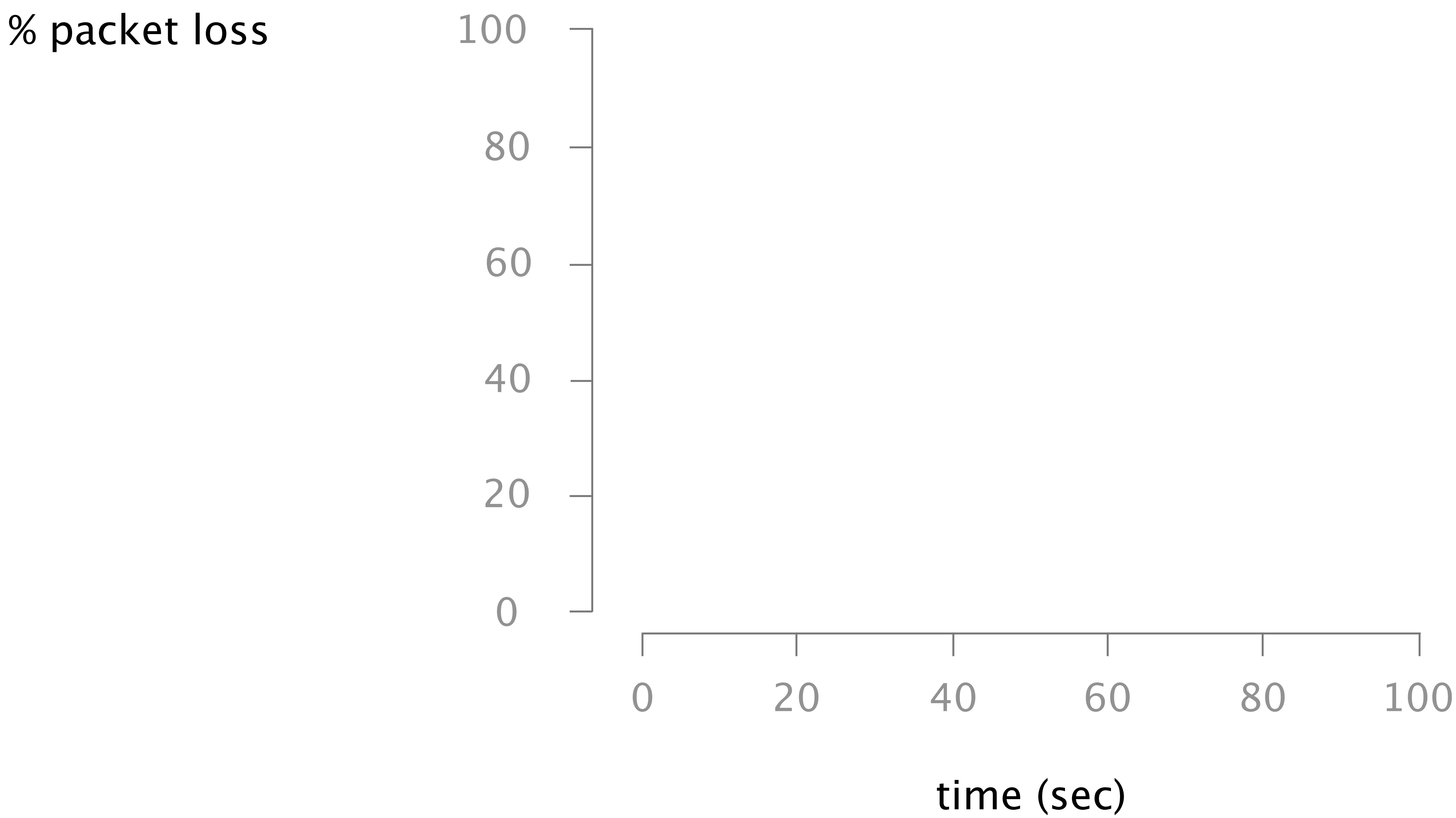
In the data-plane, we insert a SDN switch to implement the pointer table

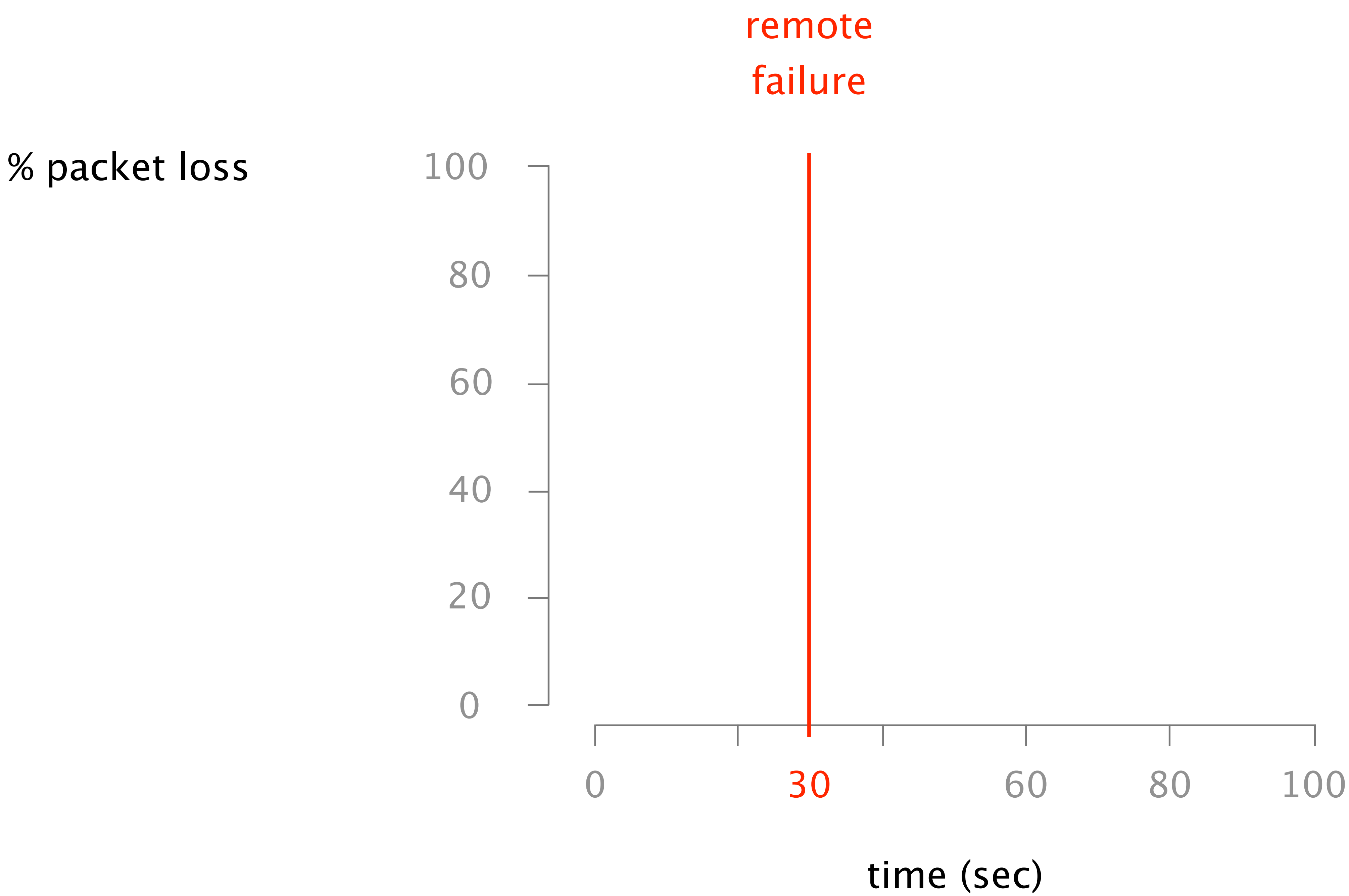


In the control-plane, we insert a controller to maintain BGP state, predicts failures and drive the switch



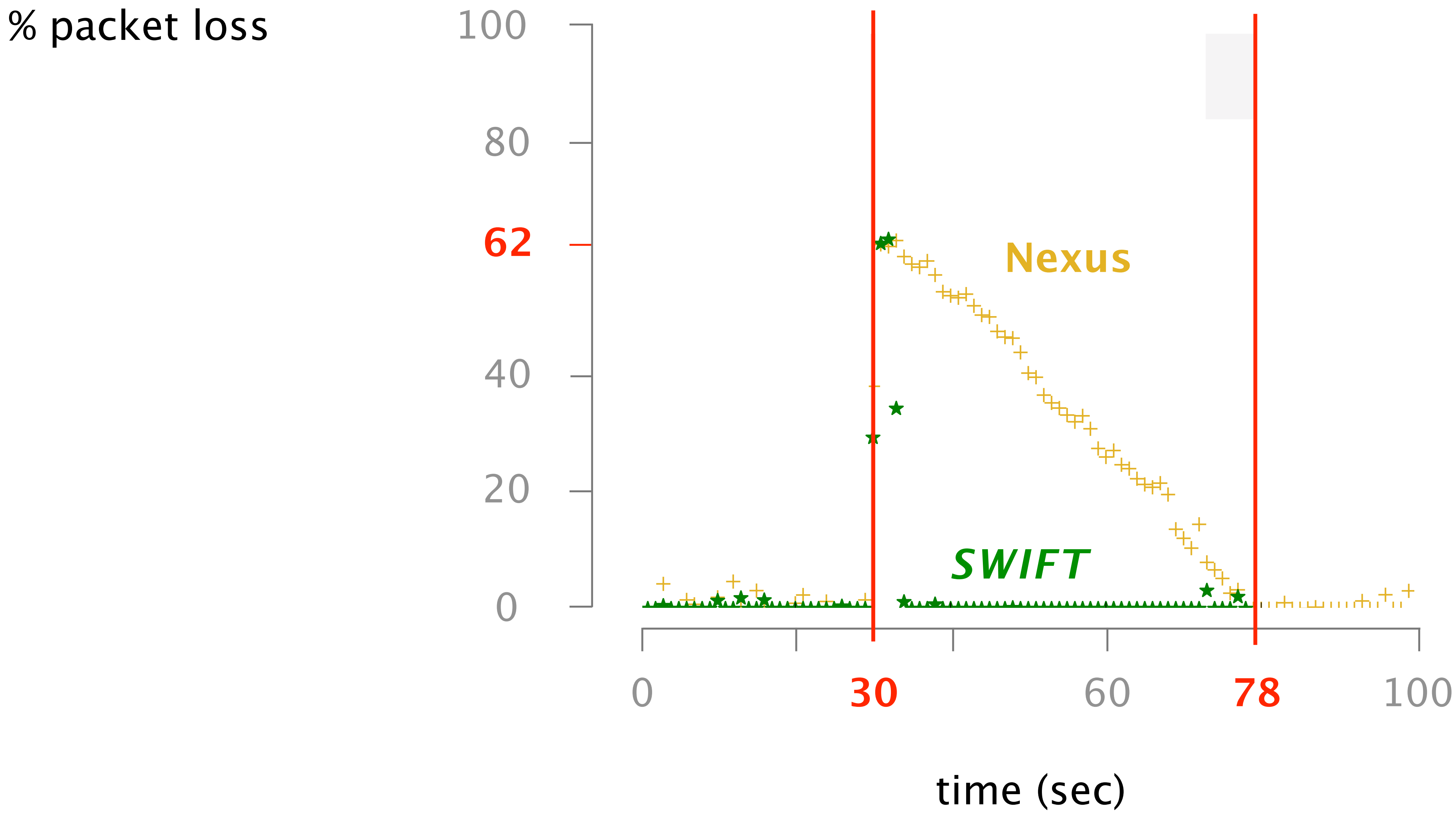

SDN
switch
SWIFTED
IP router
SWIFT
controller


% packet loss
100
80
60
40
20
0
0
20
40
60
80
100
time (sec)


remote
failure
% packet loss
100
80
60
40
20
0
0
30
60
80
100
time (sec)

SWIFT reduces the convergence time of the Cisco router

from ~50s to max 3s



SWIFT: Predictive Fast Rerouting

Predicting

out of few messages

Updating

groups of entries

Supercharging

existing systems

SWIFT significantly speeds up router convergence upon frequent remote Internet failure

SWIFT predictions are quick and accurate

close to 90% accuracy early on in the burst

SWIFT encoding is efficient

fast converge 95% of the predicted prefixes

SWIFT works in practice

95% speed-up compared to Cisco routers