Communication Networks

Spring 2022

Q&A Session

Coralie Busse-Grawitz

Tobias Bühler

ETH Zürich

August 15 2022

Today’s Q&A session

Important concepts

Answering received questions

Individual questions

Today's Q&A session

Important concepts

Answering received questions

Individual questions

Important concepts

L2 vs. L3

BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)

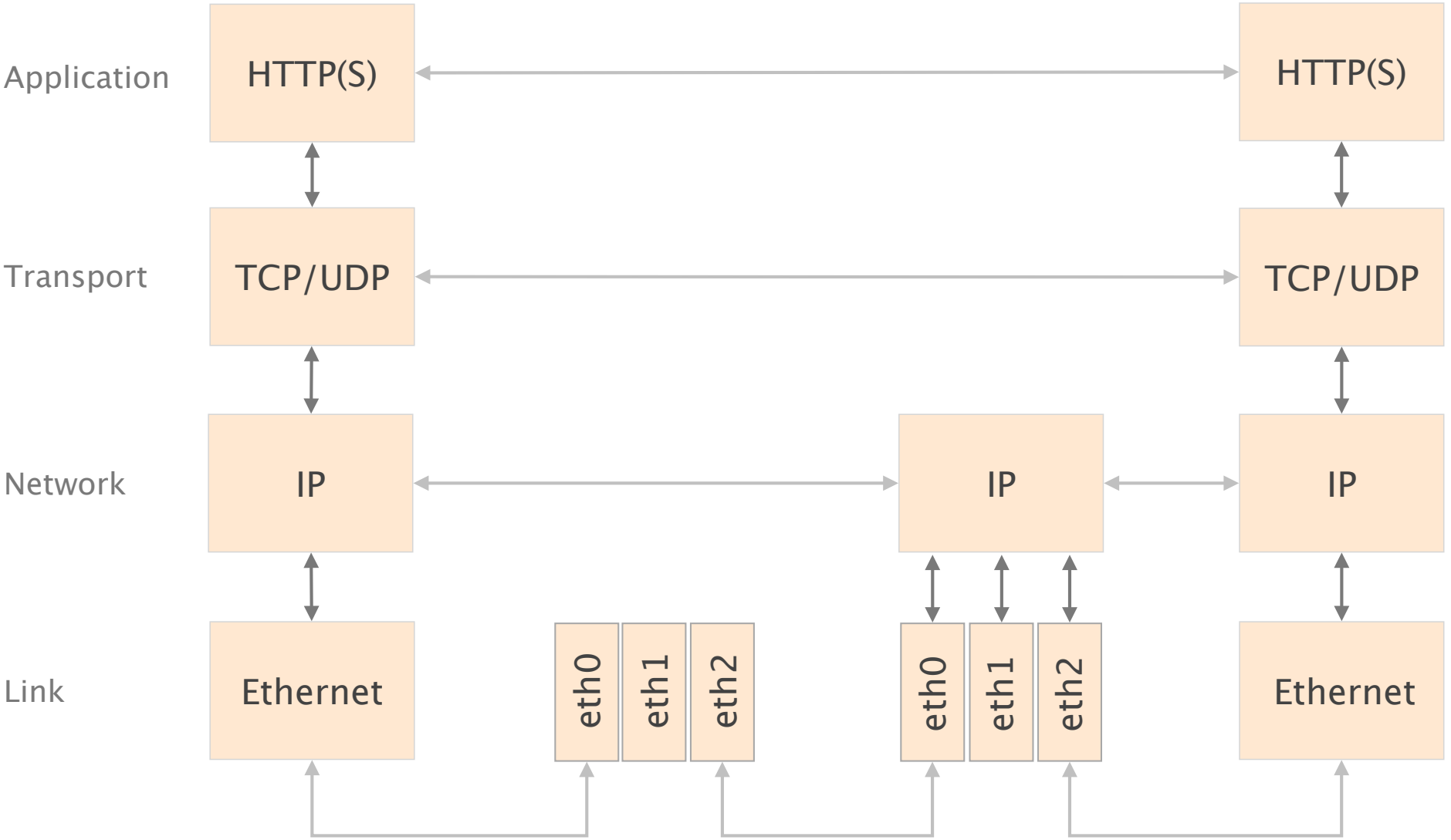
Important concepts

L2 vs. L3

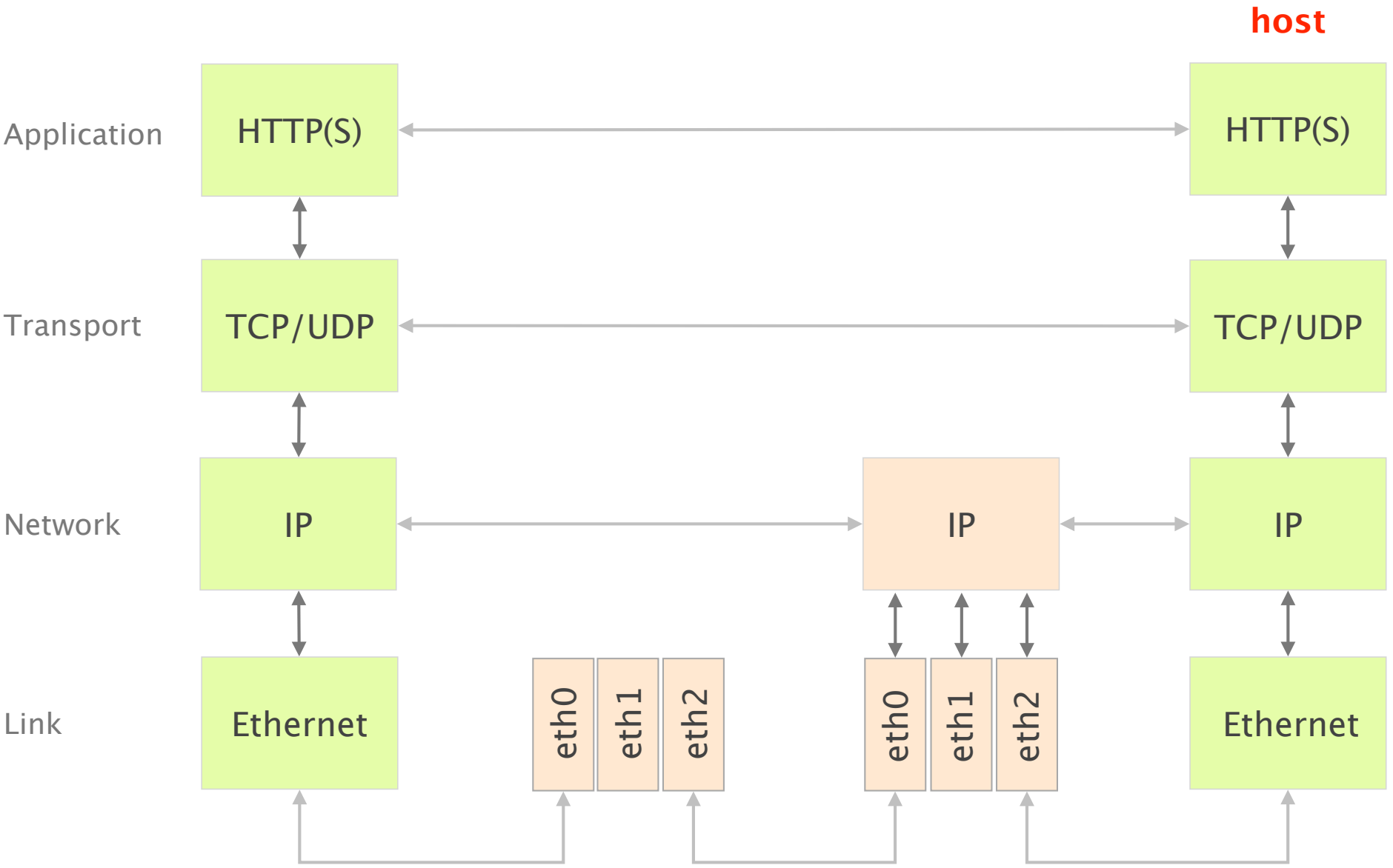
BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)

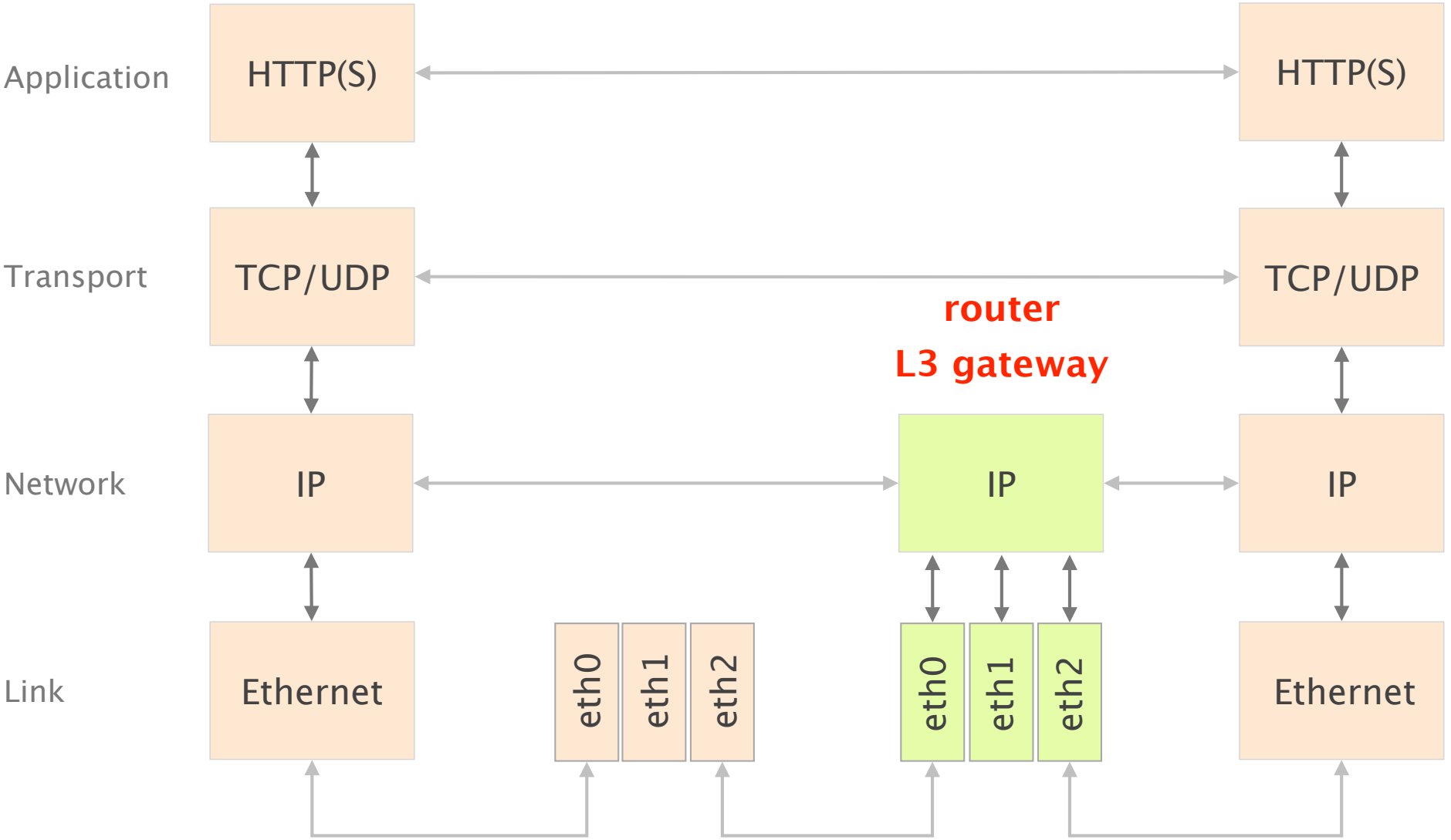

Application
Transport
Network
Link
HTTP(S)
HTTP(S)
TCP/UDP
TCP/UDP
IP
IP
IP
Ethernet
eth0
eth1
eth2
eth0
eth1
eth2
Ethernet

Since when bits arrive they must make it to the application, all the layers exist on a host



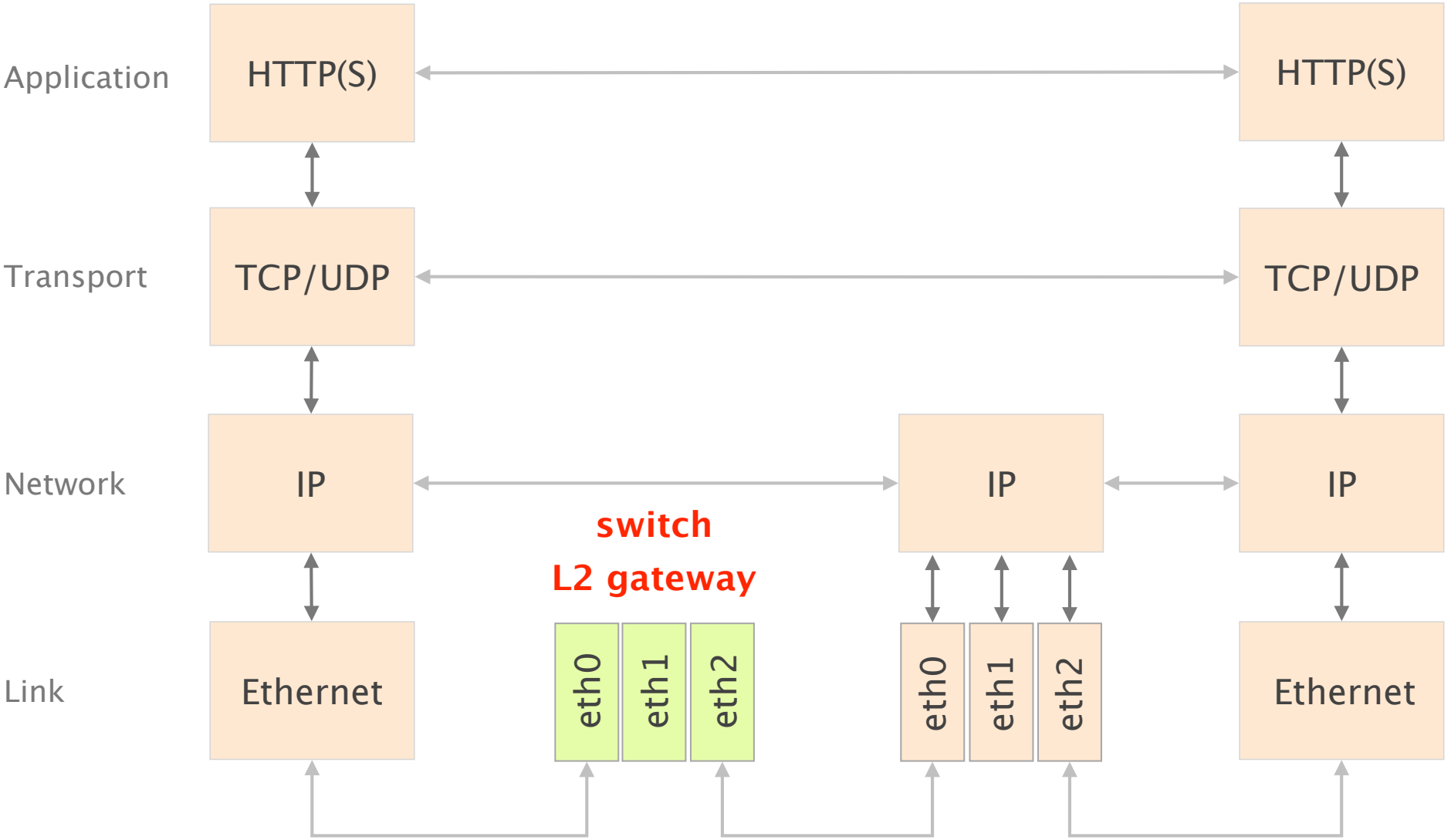
Routers act as L3 gateway

as such they implement L2 and L3



Switches act as L2 gateway

as such they only implement L2



Important concepts

L2 vs. L3

BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)

Discussed in the second part

A lot of your questions are related to BGP

Important concepts

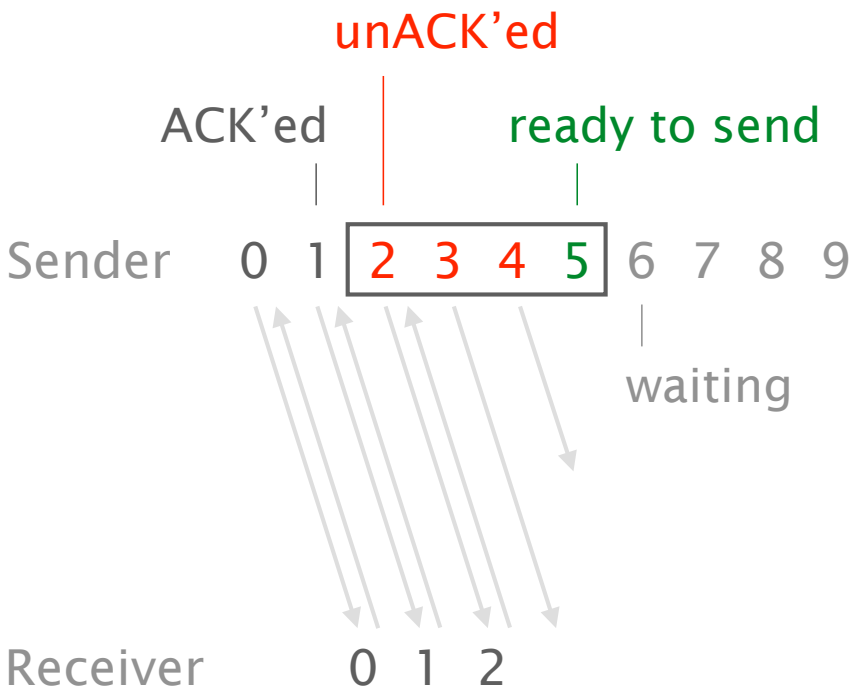
L2 vs. L3

BGP and related topics

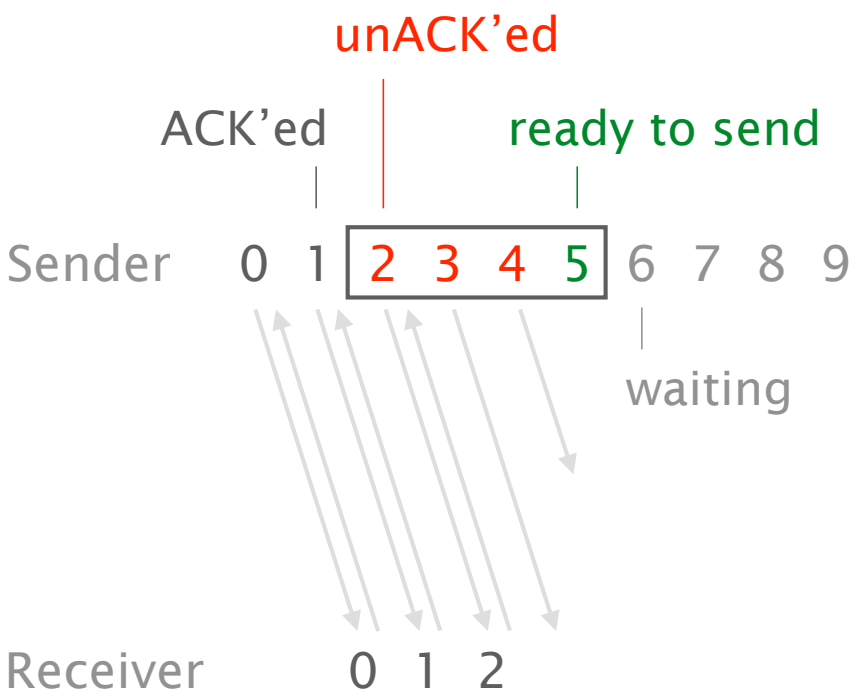
Reliable transport and congestion control

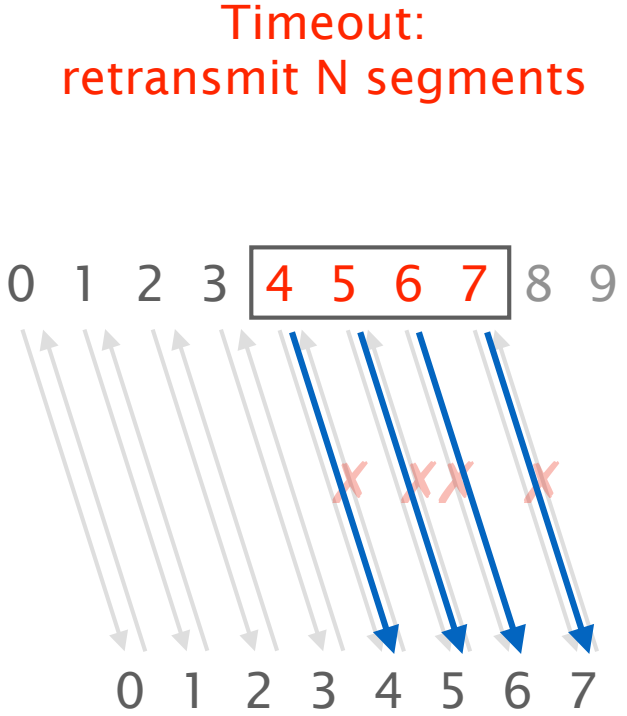
Traceroute (and ping)

Go-Back-N (GBN)


unACK'ed
ACK'ed
ready to send
Sender
0 1 2 3 4 5 6 7 8 9
waiting
Receiver
0 1 2

Go-Back-N (GBN)


unACK'ed
ACK'ed
ready to send
Sender
0 1 2 3 4 5 6 7 8 9
waiting
Receiver
0 1 2


Timeout:
retransmit N segments
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7

Different types of ACKs

Cumulative ACKs: acknowledges all previous data segments

- But no information which data segment triggered it
- TCP uses cumulative ACKs

Individual ACKs: no connection to previous data segments

- Over time sender can detect missing data segments
- Often used together with per-packet timers

Different types of ACKs - SACK

SACK (Selective ACKnowledgement): combines advantages

- Bigger ACKs and more complicated implementation
- Detailed view of the buffer at the receiver side

Important: acknowledged segments in the SACK header are **not** removed from the sender window/buffer

Important TCP features

TCP performs an initial handshake and session tear down

TCP is stream oriented (sends/receives a byte stream)

TCP retransmits packets after receiving duplicated ACKs

TCP performs flow control (to protect the receiver)

TCP performs congestion control (to protect the network)

TCP often sends and receives bytes at the same time

The TCP CC algorithm tries to solve three problems

It can **estimate** the available bandwidth

It can **adapt** the bandwidth usage of a flow

And it tries to share the available bandwidth **fairly** between flows

The algorithm discussed in the lecture uses AI**MD**

- Increase slowly, decrease very quickly

There exists a large number of different CWND algorithms

- All with their own benefits and problems

Important concepts

L2 vs. L3

BGP and related topics

Reliable transport and congestion control

Traceroute (and ping)

`traceroute` is a tool to reconstruct the path of traffic

`traceroute` exploits the time-to-live (TTL),

as routers return an “ICMP Time exceeded” when it is zero

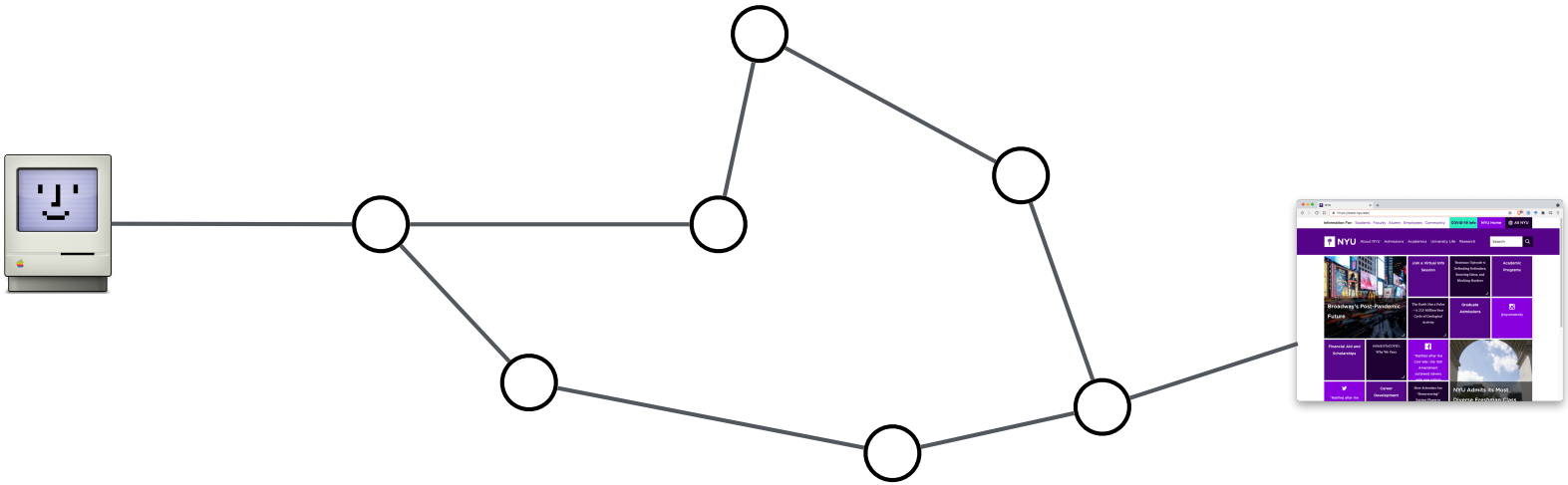
`traceroute` sends three packets with the same TTL

to explore multiple paths (hint: loadbalancing)

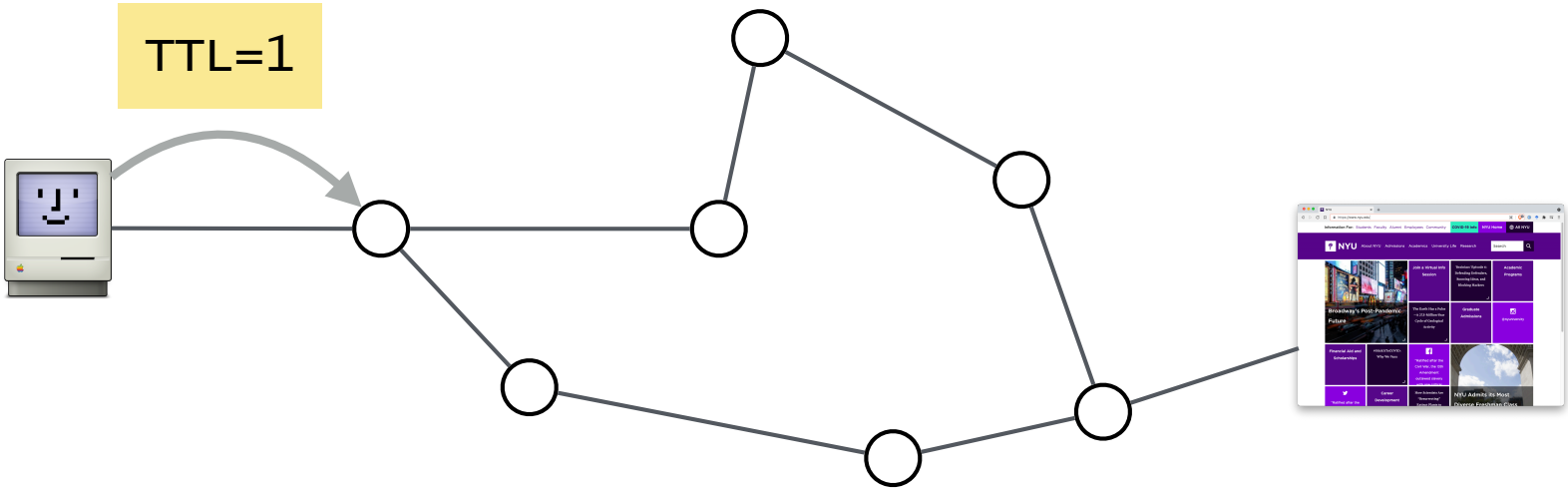
`traceroute` increases the TTL with every round

to explore the path step-by-step

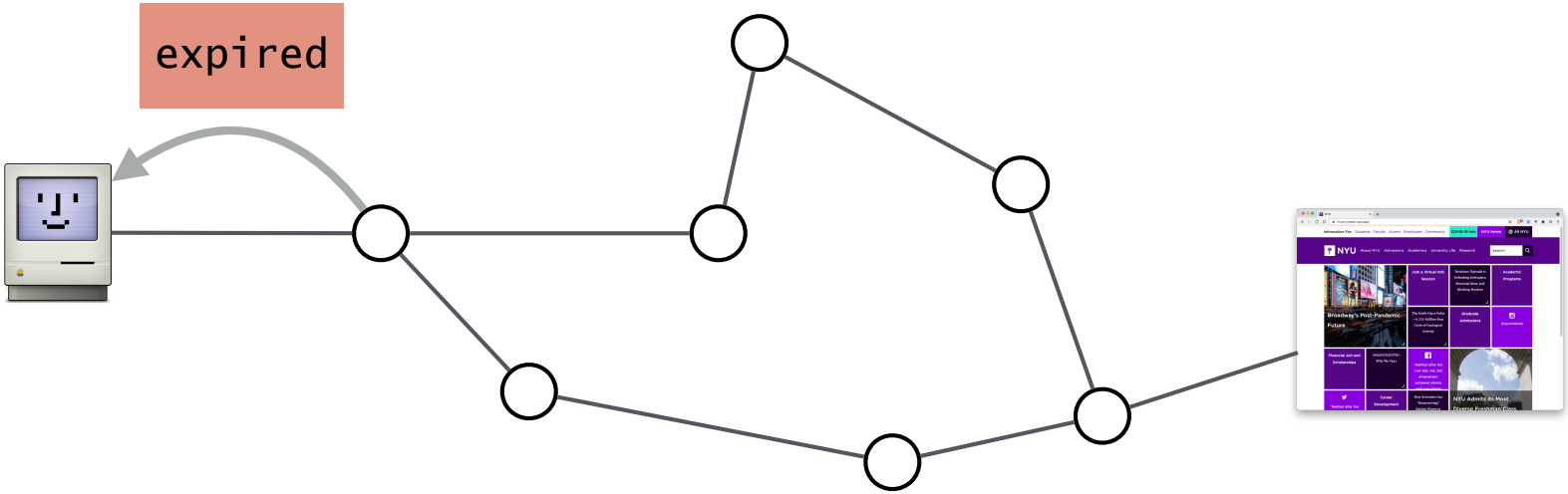
Simple illustration of `traceroute`



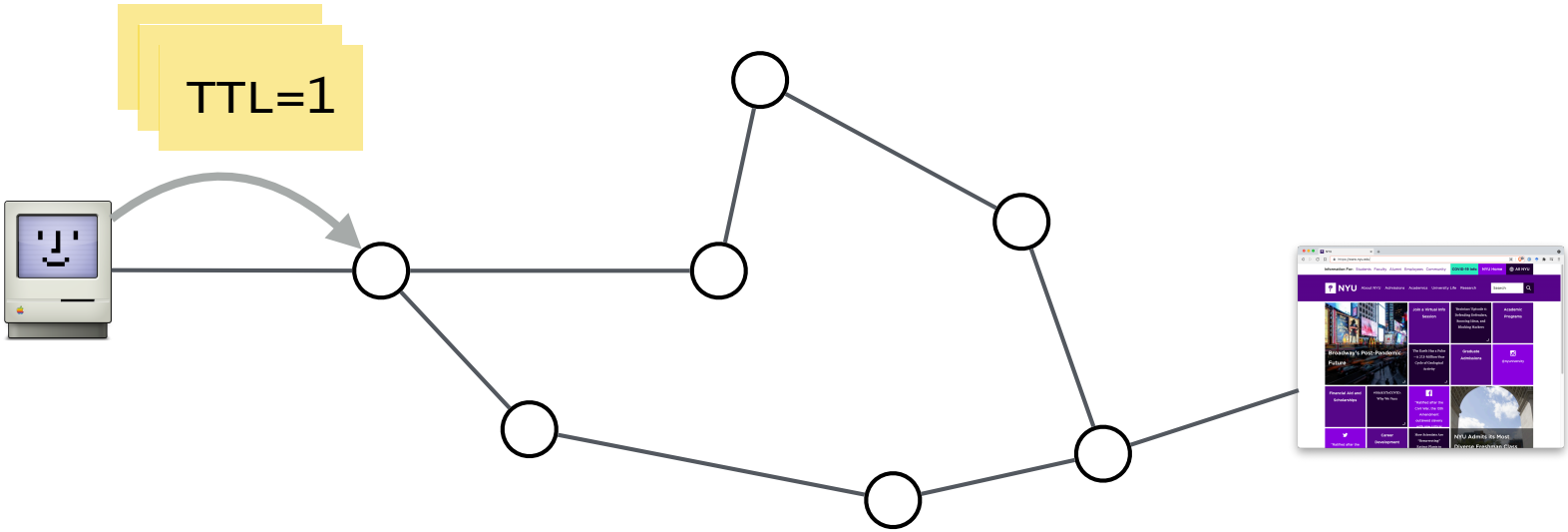
Simple illustration of `traceroute`



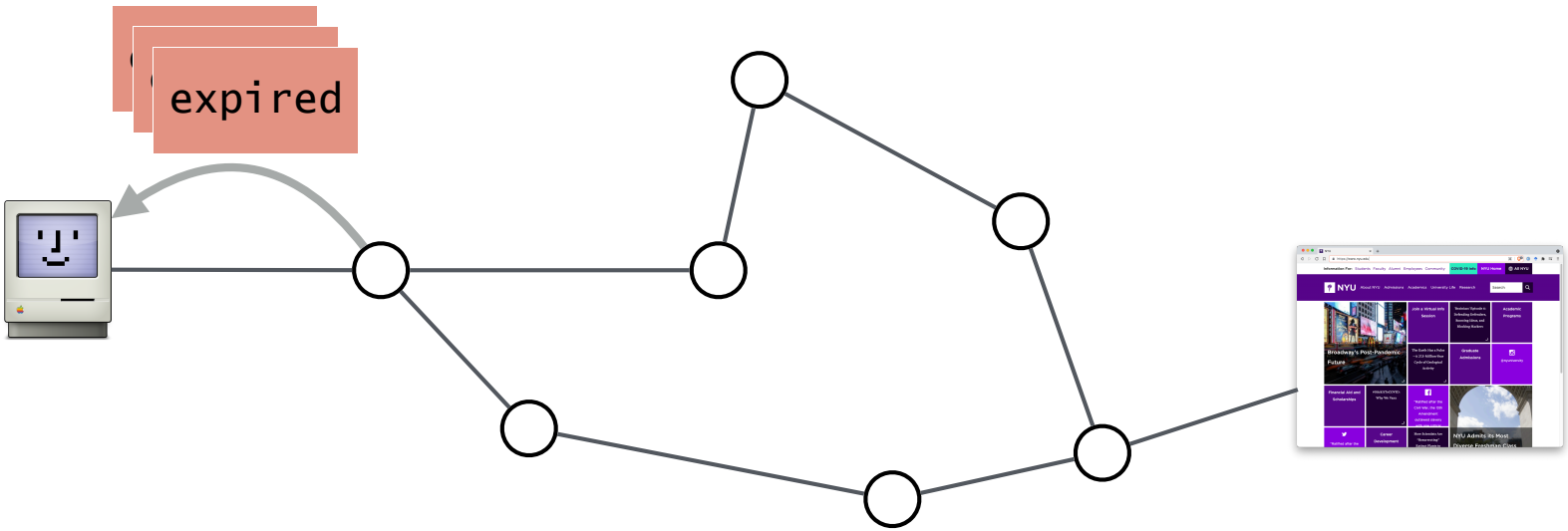
Simple illustration of `traceroute`



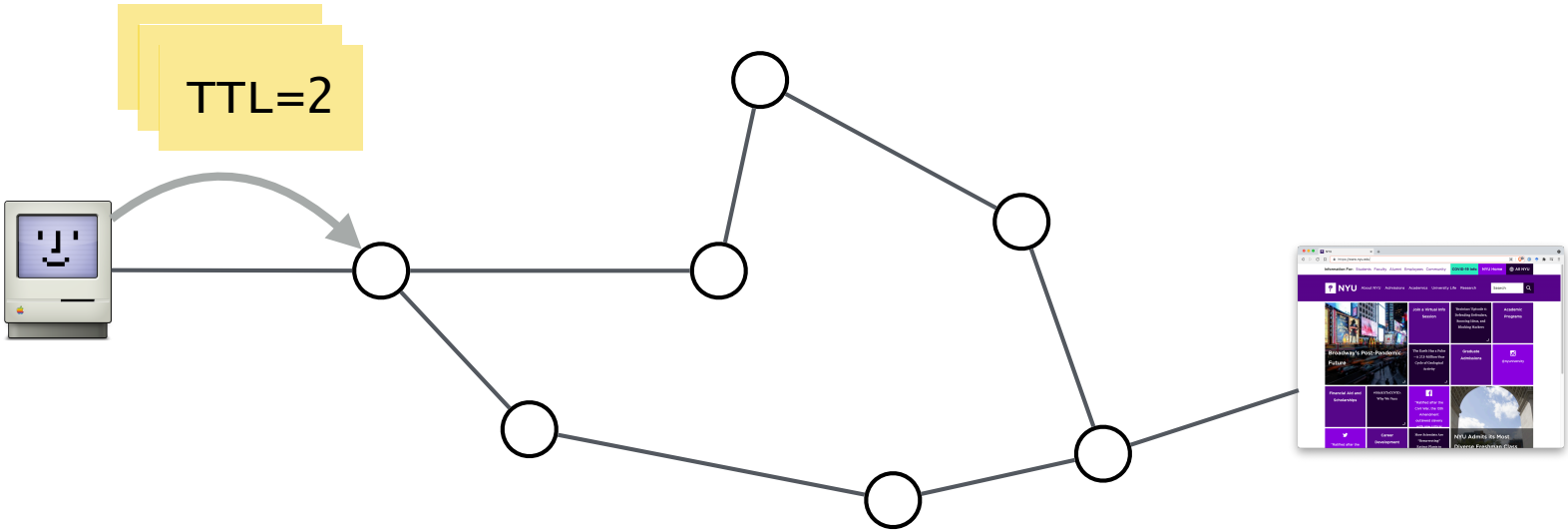
Simple illustration of `traceroute`



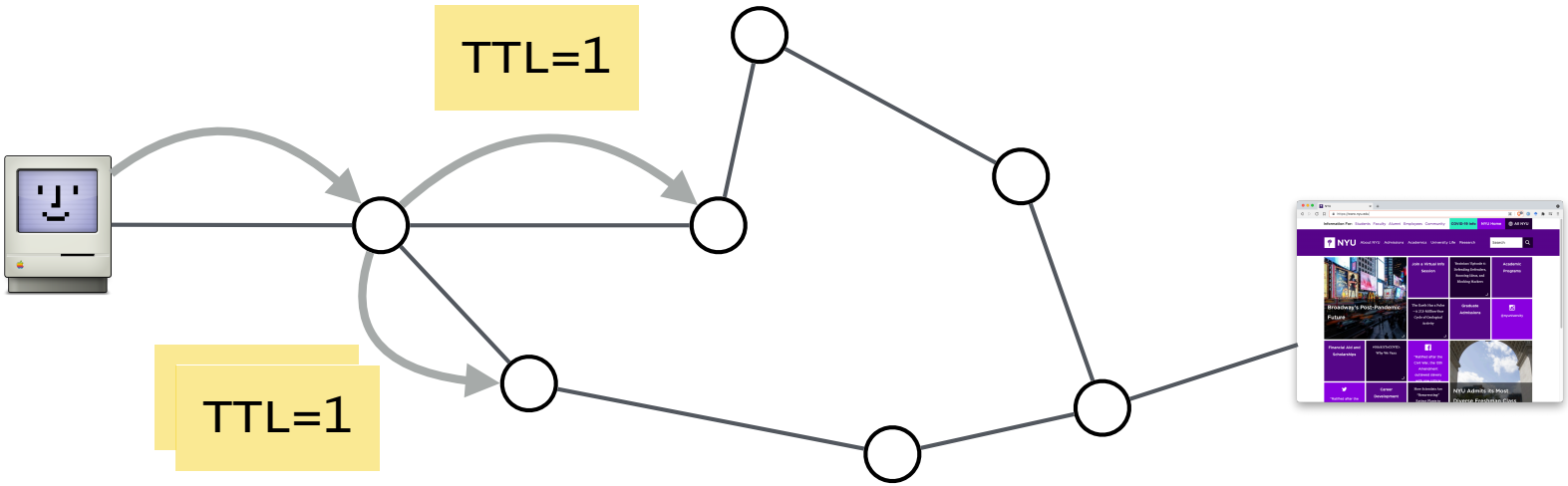
Simple illustration of `traceroute`



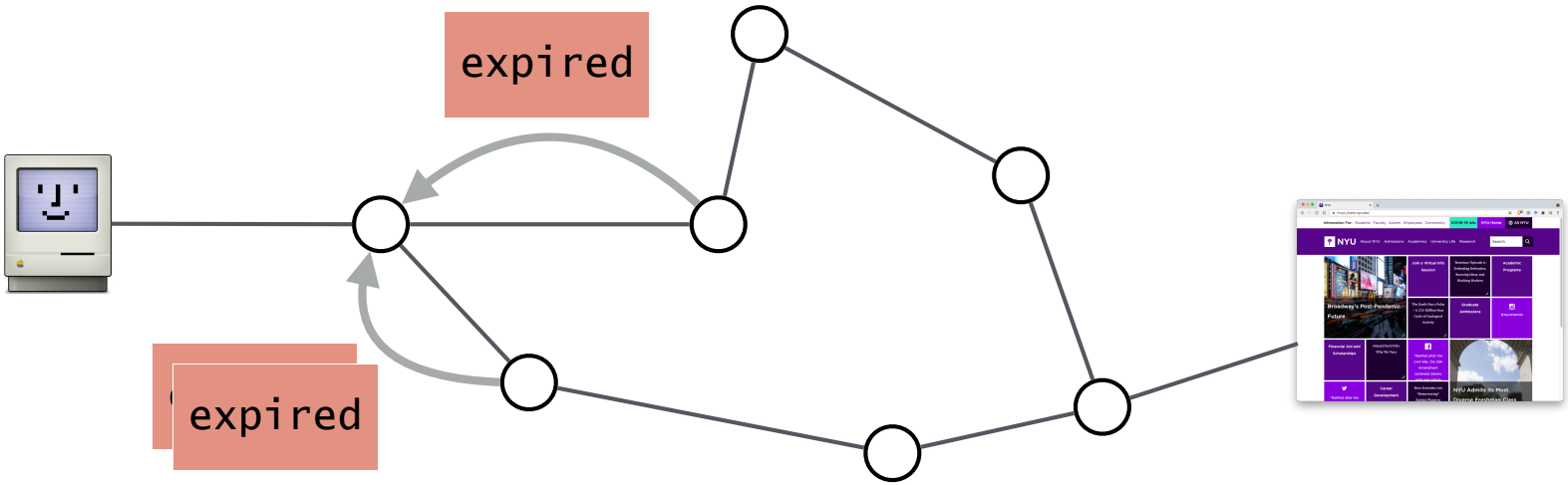
Simple illustration of `traceroute`



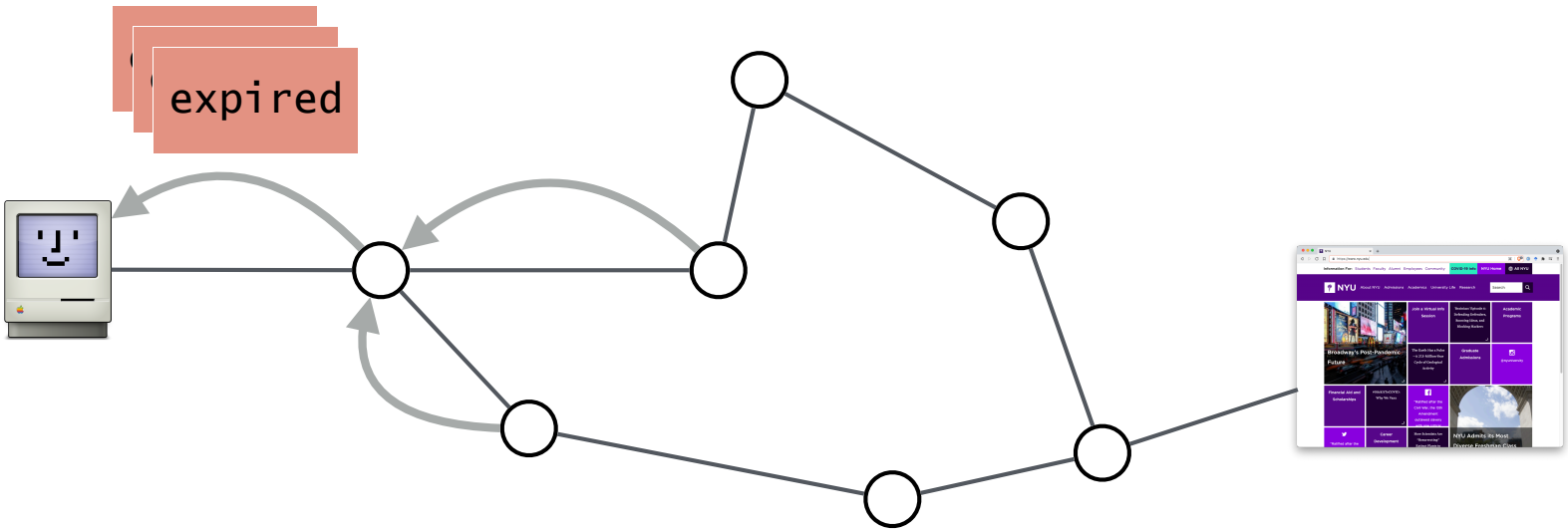
Simple illustration of `traceroute`



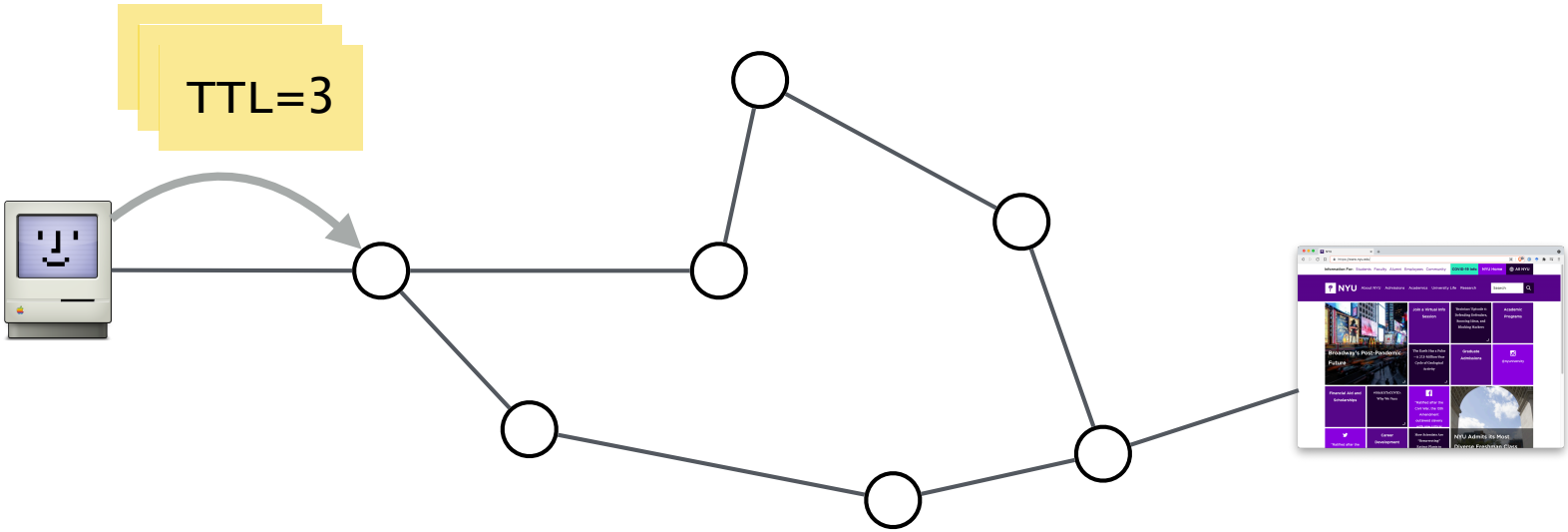
Simple illustration of `traceroute`



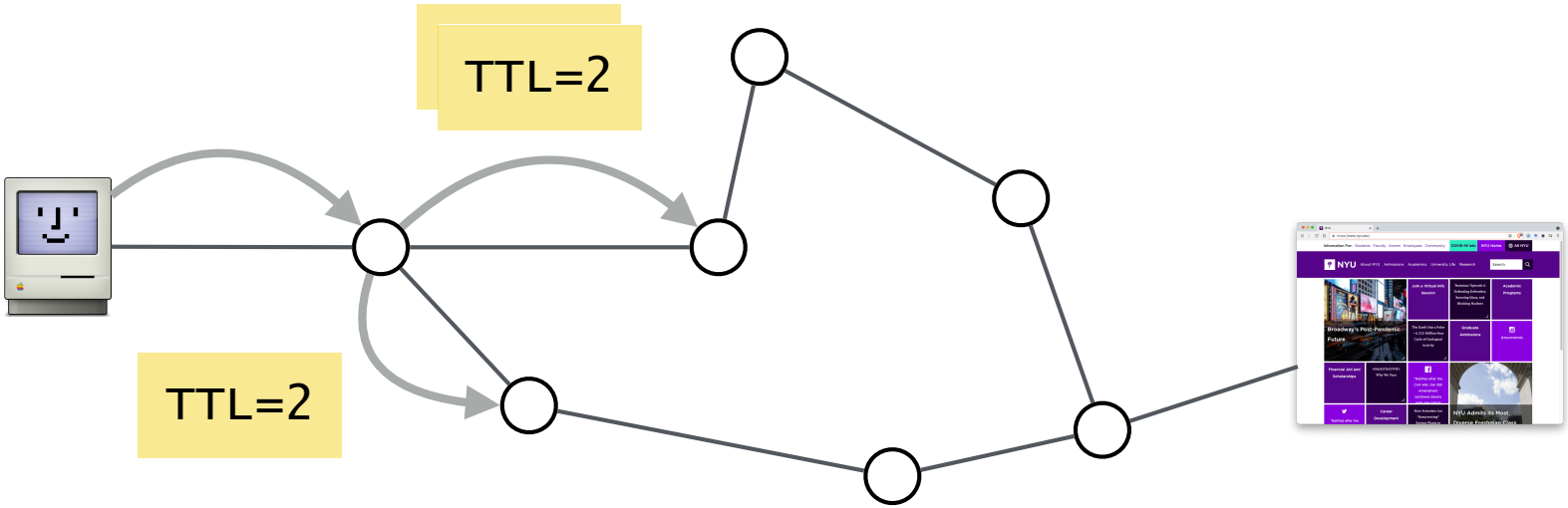
Simple illustration of `traceroute`



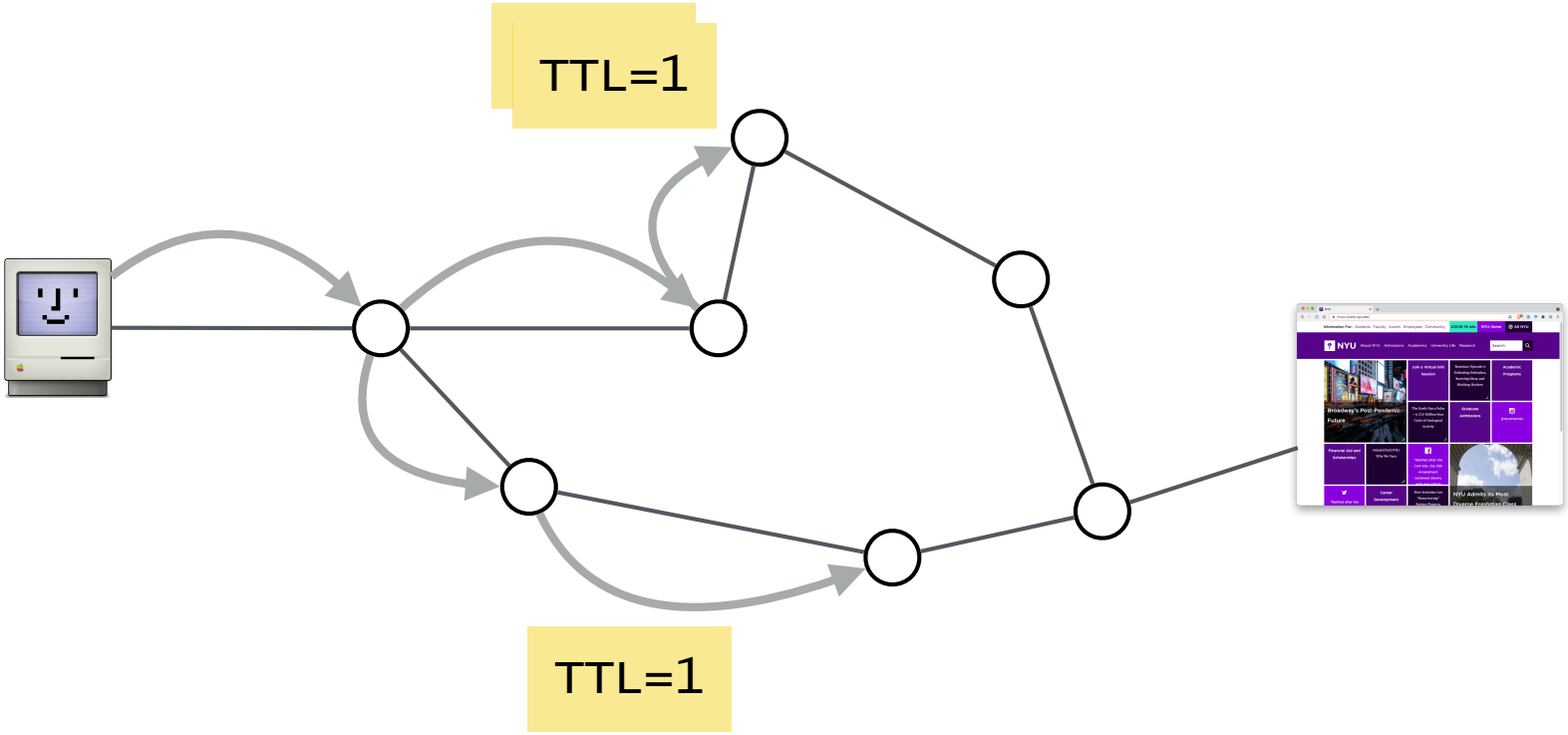
Simple illustration of `traceroute`



Simple illustration of `traceroute`



Simple illustration of `traceroute`



traceroute output

```
➜  ~ traceroute www.nyu.edu
traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
 1  internetbox (192.168.1.1)  5.690 ms  4.478 ms  5.014 ms
 2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1)  4.980 ms  10.336 ms  3.920 ms
 3  * * *
 4  * * *
 5  i79zhb-015-ae6.bb.ip-plus.net (138.187.129.155)  9.031 ms  4.717 ms  4.757 ms
 6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38)  4.986 ms  4.130 ms  4.925 ms
 7  ip4.gtt.net (212.115.128.45)  5.004 ms  4.504 ms  9.982 ms
 8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214)  90.148 ms  89.701 ms  90.097 ms
 9  ip4.gtt.net (209.120.137.218)  89.987 ms  89.607 ms  90.059 ms
10  * * *
11  nyugwa-ptp-dmzgwa-vl3081.net.nyu.edu (128.122.254.108)  89.754 ms  129.602 ms
    nyugwa-ptp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110)  89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116)  89.454 ms  89.948 ms  90.312 ms
13  * * *
14  wsqdcgwa-vl902.net.nyu.edu (128.122.1.38)  90.985 ms  89.724 ms  90.310 ms
15  * * *
16  * * *
```

traceroute output

```
➜  ~ traceroute www.nyu.edu
traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
 1  internetbox (192.168.1.1)  5.690 ms  4.478 ms  5.014 ms
 2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1)  4.980 ms  10.336 ms  3.920 ms
 3  * * *
 4  * * *
 5  i79zhb-015-ae6.bb.ip-plus.net (138.187.129.155)  9.031 ms  4.717 ms  4.757 ms
 6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38)  4.986 ms  4.130 ms  4.925 ms
 7  ip4.gtt.net (212.115.128.45)  5.004 ms  4.504 ms  9.982 ms
 8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214)  90.148 ms  89.701 ms  90.097 ms
 9  ip4.gtt.net (209.120.137.218)  89.987 ms  89.607 ms  90.059 ms
10  * * *
11  nyugwa-ptp-dmzgwa-vl3081.net.nyu.edu (128.122.254.108)  89.754 ms  129.602 ms
    nyugwa-ptp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110)  89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116)  89.454 ms  89.948 ms  90.312 ms
13  * * *
14  wsqdcgwa-vl902.net.nyu.edu (128.122.1.38)  90.985 ms  89.724 ms  90.310 ms
15  * * *
16  * * *
```

TTL values/hops

traceroute output

```
➜  ~ traceroute www.nyu.edu
traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
 1  internetbox (192.168.1.1)  5.690 ms  4.478 ms  5.014 ms
 2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1)  4.980 ms  10.336 ms  3.920 ms
 3  * * *
 4  * * *
 5  i79zhb-015-ae6.bb.ip-plus.net (138.187.129.155)  9.031 ms  4.717 ms  4.757 ms
 6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38)  4.986 ms  4.130 ms  4.925 ms
 7  ip4.gtt.net (212.115.128.45)  5.004 ms  4.504 ms  9.982 ms
 8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214)  90.148 ms  89.701 ms  90.097 ms
 9  ip4.gtt.net (209.120.137.218)  89.987 ms  89.607 ms  90.059 ms
10  * * *
11  nyugwa-ptp-dmzgwa-vl3081.net.nyu.edu (128.122.254.108)  89.754 ms  129.602 ms
    nyugwa-ptp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110)  89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116)  89.454 ms  89.948 ms  90.312 ms
13  * * *
14  wsqdcgwa-vl902.net.nyu.edu (128.122.1.38)  90.985 ms  89.724 ms  90.310 ms
15  * * *
16  * * *
```

name and IP address of the hop

traceroute output

```
➜  ~ traceroute www.nyu.edu
traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
 1  internetbox (192.168.1.1)  5.690 ms  4.478 ms  5.014 ms
 2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1)  4.980 ms  10.336 ms  3.920 ms
 3  * * *
 4  * * *
 5  i79zhb-015-ae6.bb.ip-plus.net (138.187.129.155)  9.031 ms  4.717 ms  4.757 ms
 6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38)  4.986 ms  4.130 ms  4.925 ms
 7  ip4.gtt.net (212.115.128.45)  5.004 ms  4.504 ms  9.982 ms
 8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214)  90.148 ms  89.701 ms  90.097 ms
 9  ip4.gtt.net (209.120.137.218)  89.987 ms  89.607 ms  90.059 ms
10  * * *
11  nyugwa-ptp-dmzgwa-vl3081.net.nyu.edu (128.122.254.108)  89.754 ms  129.602 ms
    nyugwa-ptp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110)  89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116)  89.454 ms  89.948 ms  90.312 ms
13  * * *
14  wsqdcgwa-vl902.net.nyu.edu (128.122.1.38)  90.985 ms  89.724 ms  90.310 ms
15  * * *
16  * * *
```

not all hops reply

traceroute output

```
➜  ~ traceroute www.nyu.edu
traceroute to web.gslb.nyu.edu (216.165.47.12), 64 hops max, 52 byte packets
 1  internetbox (192.168.1.1)  5.690 ms  4.478 ms  5.014 ms
 2  1.40.79.83.dynamic.wline.res.cust.swisscom.ch (83.79.40.1)  4.980 ms  10.336 ms  3.920 ms
 3  * * *
 4  * * *
 5  i79zhb-015-ae6.bb.ip-plus.net (138.187.129.155)  9.031 ms  4.717 ms  4.757 ms
 6  i79tix-025-ae11.bb.ip-plus.net (138.187.130.38)  4.986 ms  4.130 ms  4.925 ms
 7  ip4.gtt.net (212.115.128.45)  5.004 ms  4.504 ms  9.982 ms
 8  ae4.cr2-nyc2.ip4.gtt.net (89.149.129.214)  90.148 ms  89.701 ms  90.097 ms
 9  ip4.gtt.net (209.120.137.218)  89.987 ms  89.607 ms  90.059 ms
10  * * *
11  nyugwa-ptp-dmzgwa-vl3081.net.nyu.edu (128.122.254.108)  89.754 ms  129.602 ms
    nyugwa-ptp-dmzgwb-vl3082.net.nyu.edu (128.122.254.110)  89.915 ms
12  nyufw-outside-ngfw-vl3080.net.nyu.edu (128.122.254.116)  89.454 ms  89.948 ms  90.312 ms
13  * * *
14  wsqdcgwa-vl902.net.nyu.edu (128.122.1.38)  90.985 ms  89.724 ms  90.310 ms
15  * * *
16  * * *
```

RTT for all three probes

`ping` is a tool to test reachability of hosts

`ping` relies on ICMP echo request messages

and the destination sends ICMP echo reply messages back

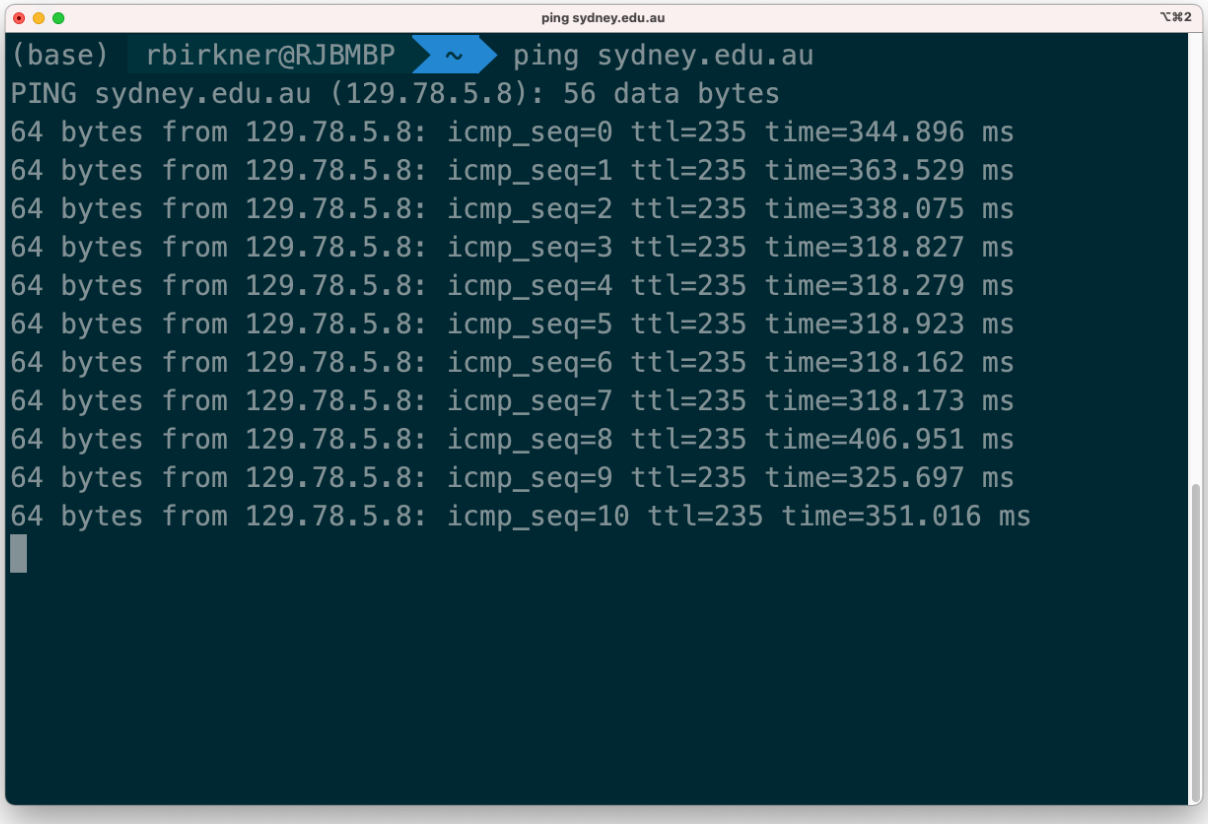
`ping` measures the RTT from source to destination,

reports packet loss and provides a statistical summary

test #1

…

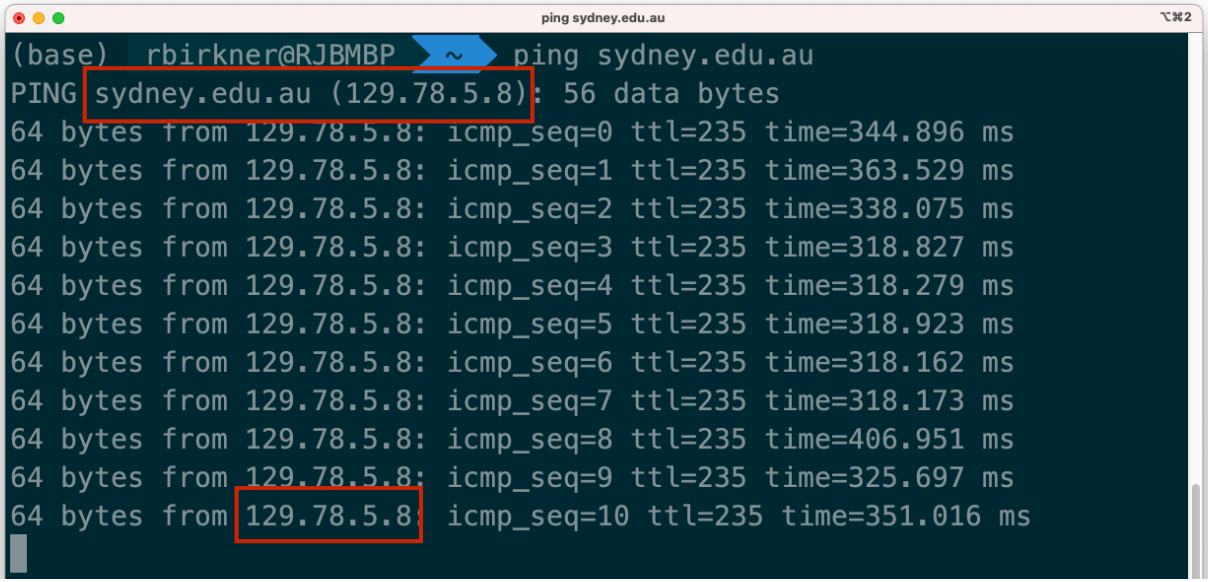
test #11



test #1

…

test #11

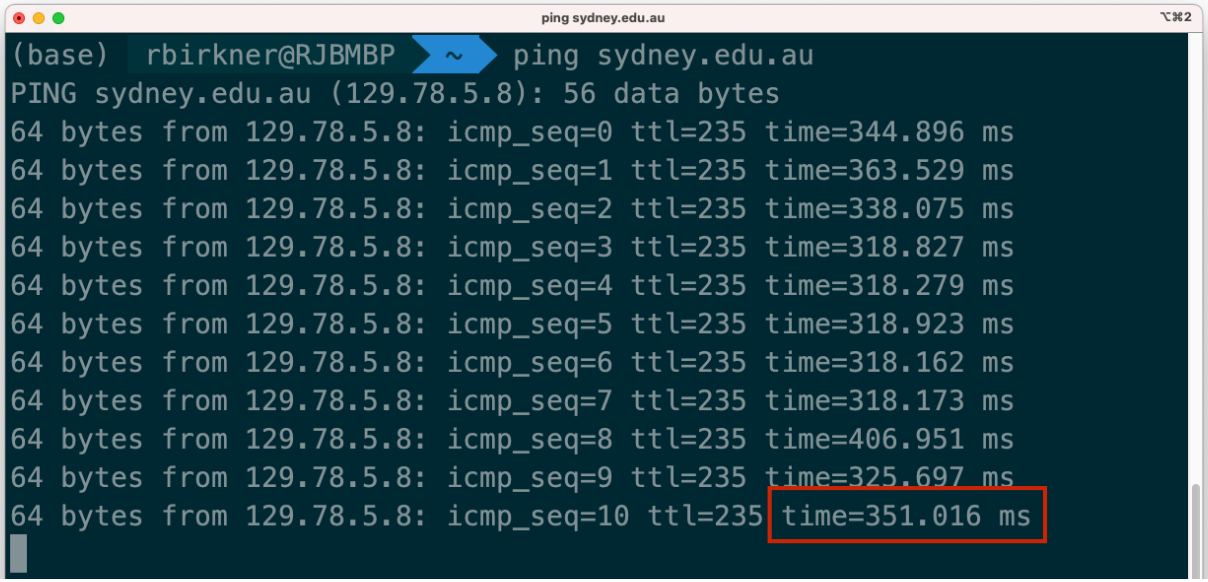


Destination address

test #1

…

test #11



RTT measurements

Round Trip Time

Both directions!

Today's Q&A session

Important concepts

Answering received questions

Individual questions

How does BGP update when the network changes?

Which messages are exchanged?

E.g., due to a link failure

On the wire, BGP is a rather simple protocol composed of four basic messages

type	used to…
OPEN	establish TCP-based BGP sessions
NOTIFICATION	report unusual conditions
UPDATE	inform neighbor of a new best route a change in the best route the removal of the best route
KEEPALIVE	inform neighbor that the connection is alive

On the wire, BGP is a rather simple protocol composed of four basic messages

type	used to…
OPEN	establish TCP-based BGP sessions
NOTIFICATION	report unusual conditions
UPDATE	inform neighbor of a new best route a change in the best route the removal of the best route
KEEPALIVE	inform neighbor that the connection is alive

On the wire, BGP is a rather simple protocol composed of four basic messages

type	used to…
OPEN	establish TCP-based BGP sessions
NOTIFICATION	report unusual conditions
UPDATE	inform neighbor of a new best route a change in the best route the removal of the best route
KEEPALIVE	inform neighbor that the connection is alive

There are two types of UPDATEs

UPDATE

inform neighbor of

a new best route

a change in the best route

the removal of the best route

There are two types of UPDATEs

UPDATE

inform neighbor of

Advertisements

- a new best route
- a change in the best route

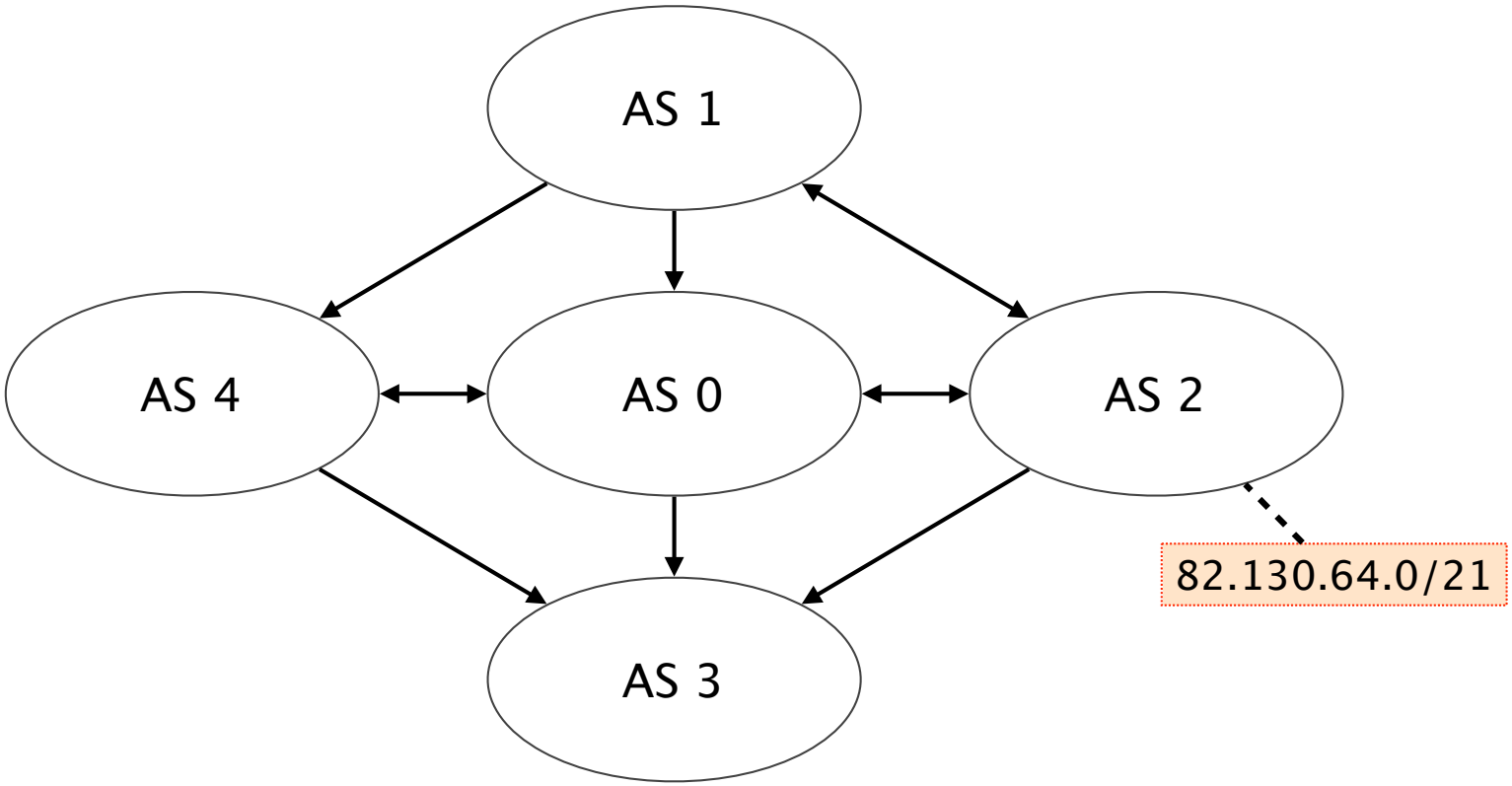
the removal of the best route

There are two types of UPDATEs

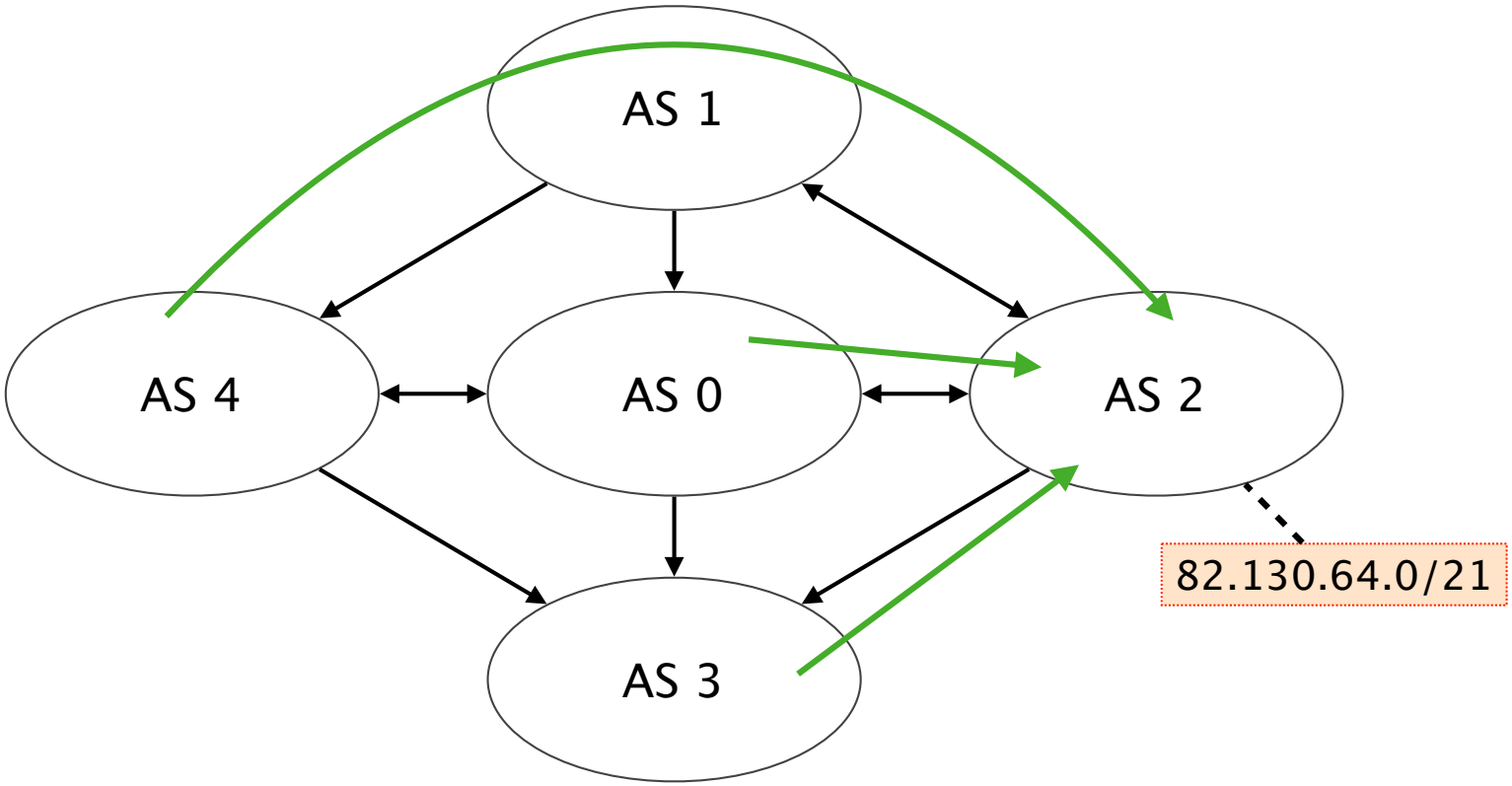
UPDATE — inform neighbor of

- Advertisements
  - a new best route
  - a change in the best route
- Withdrawal
  - the removal of the best route

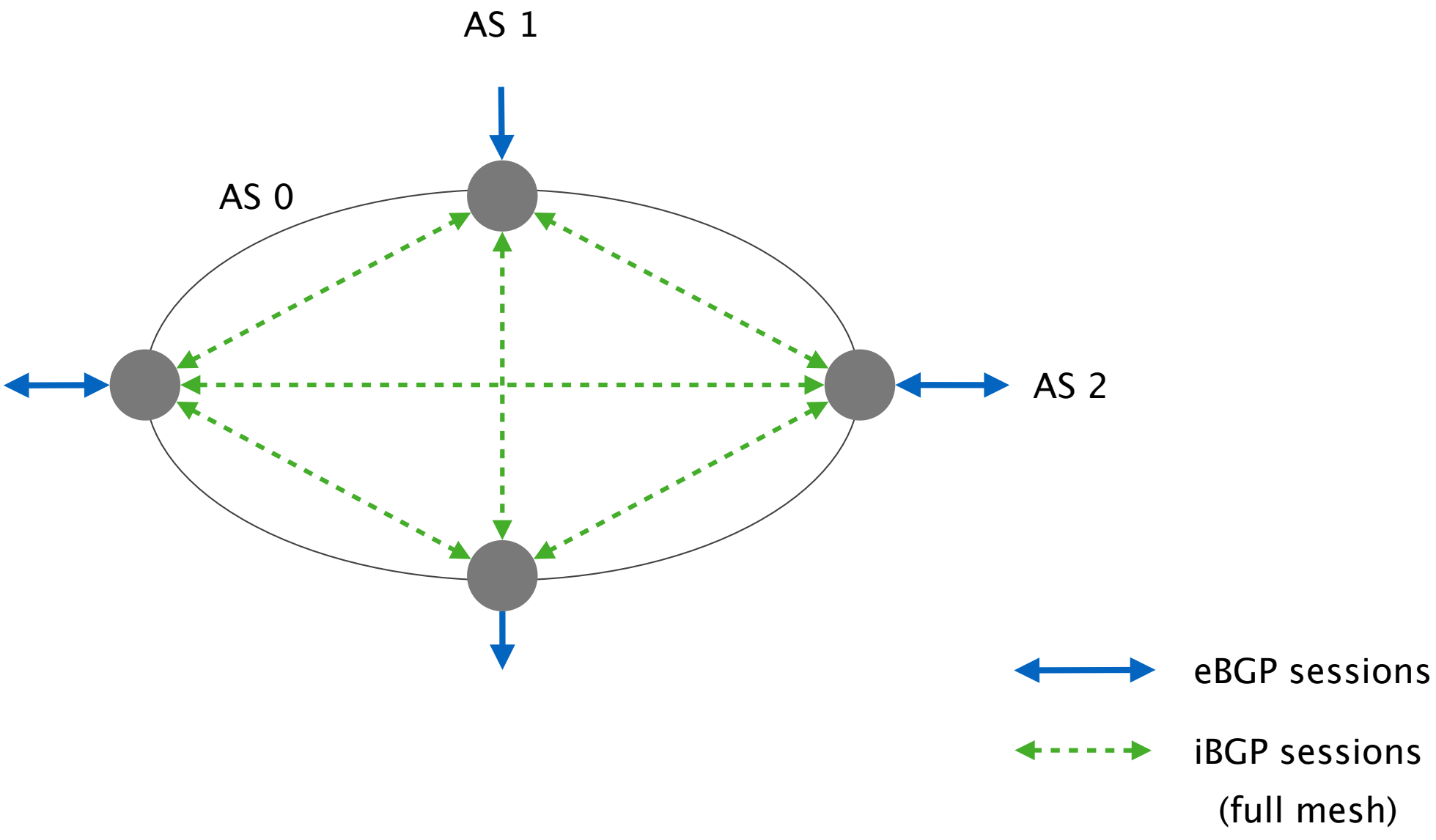
Let's look at an example (adapted from exercise 7.2)



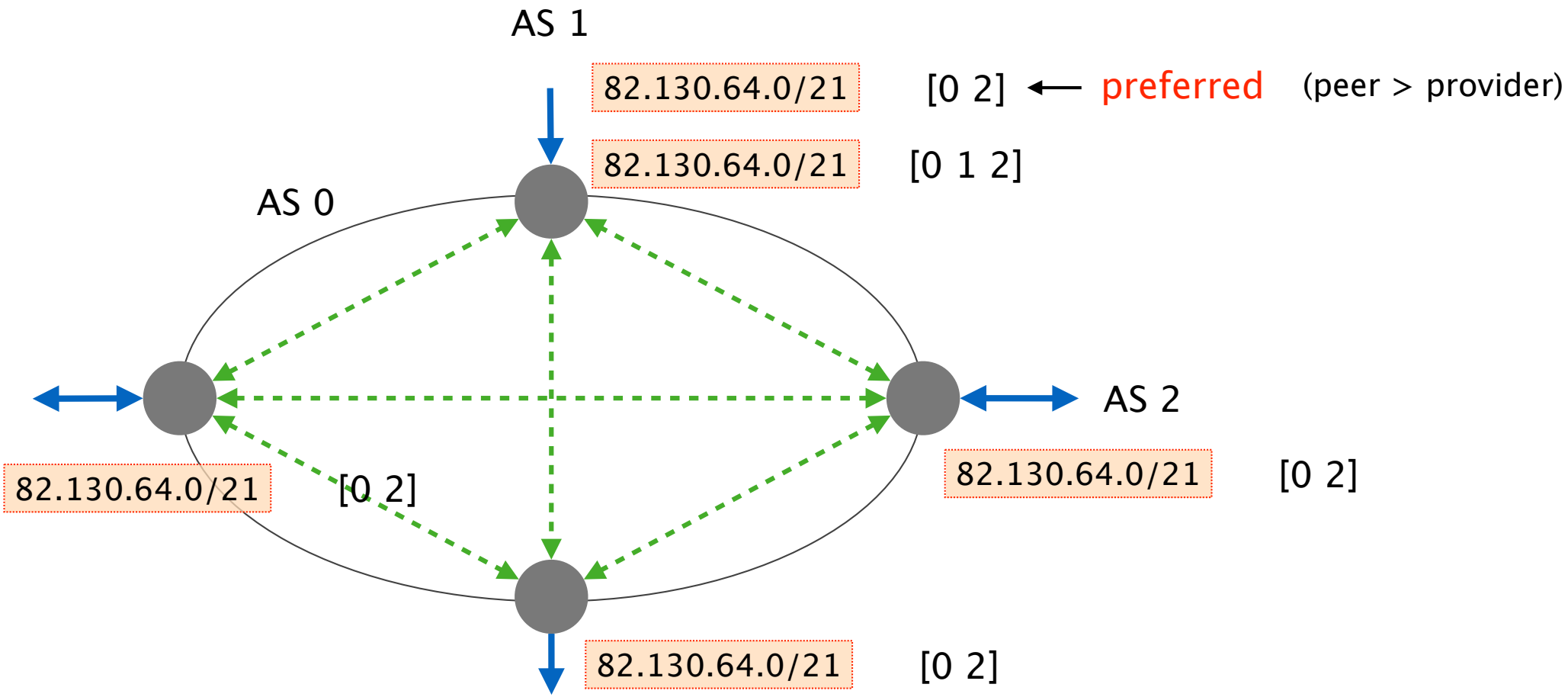
Forwarding paths without any failures



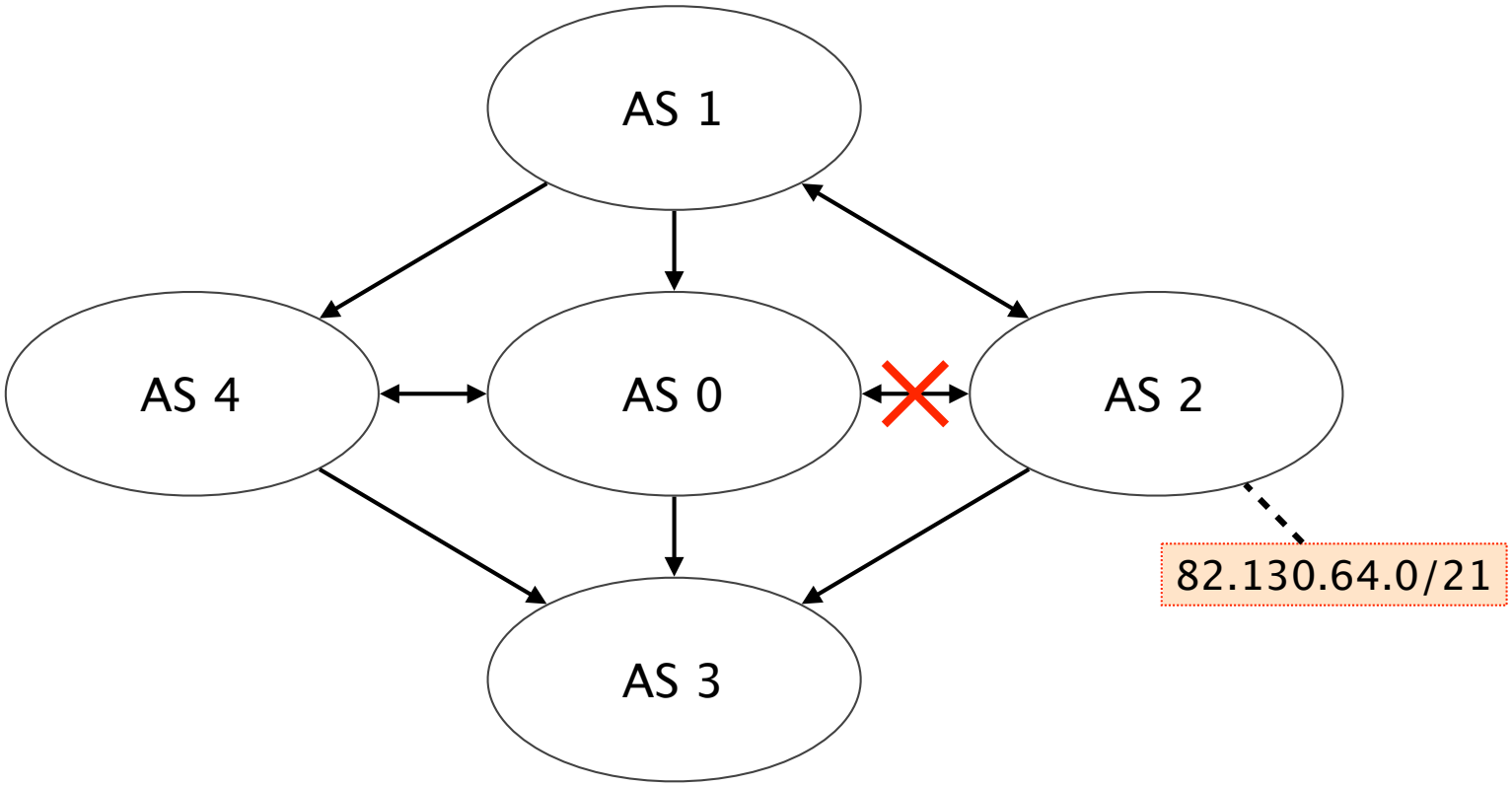
Let's focus on AS 0, we assume it has 4 routers



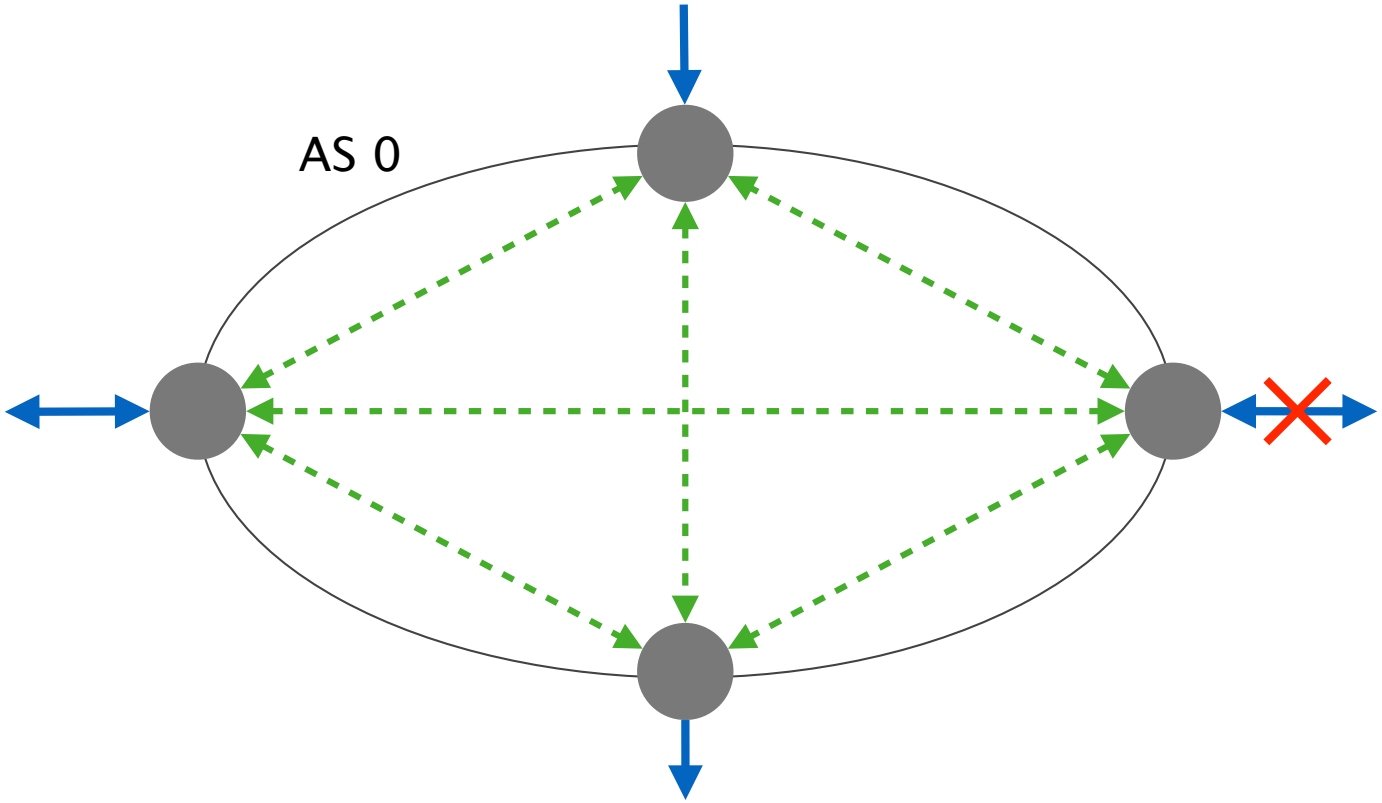
Most routers know one route towards 82.130.64.0/21, the router connected to AS 1 knows two routes



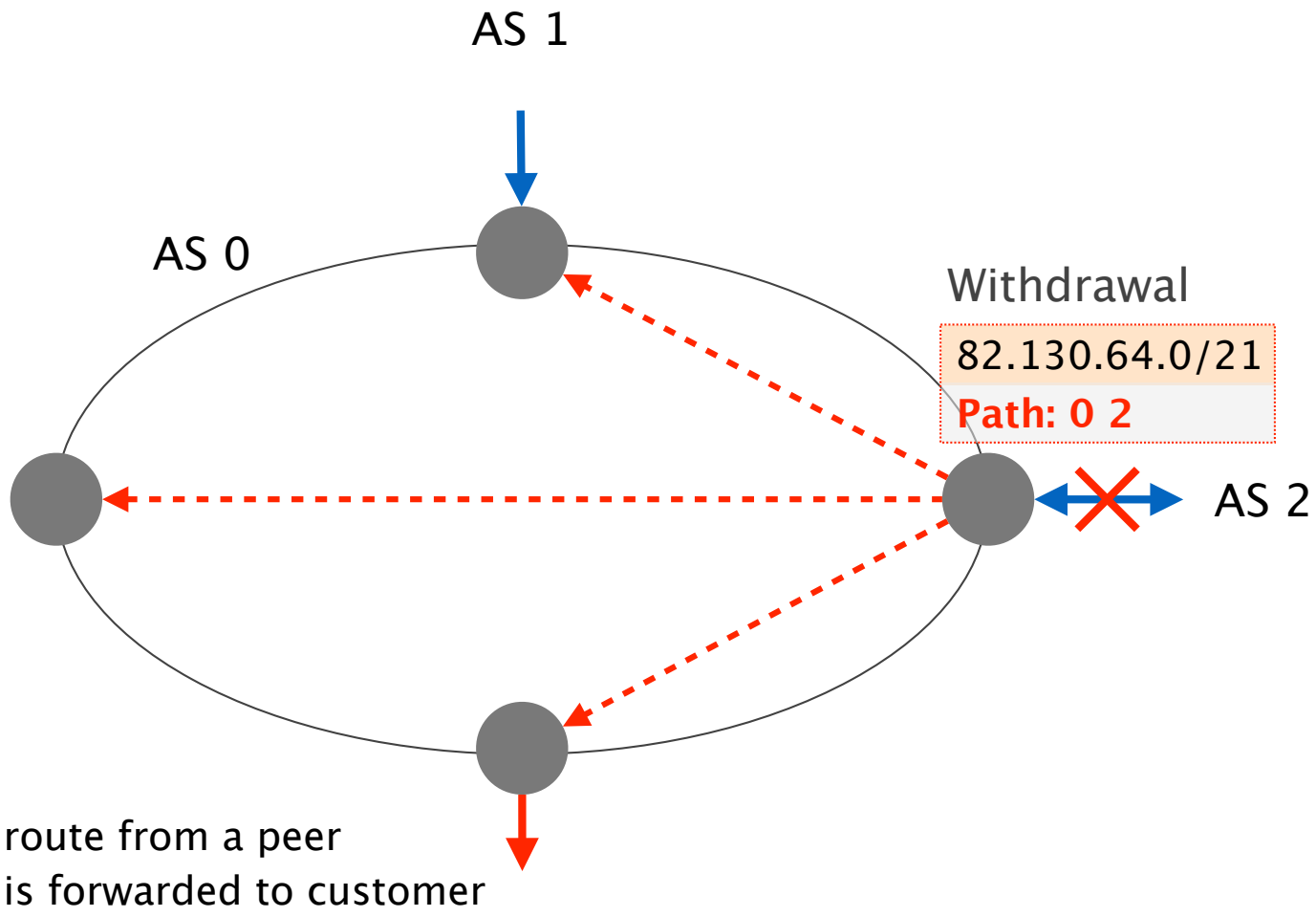
Now the link between AS 0 and AS 2 fails



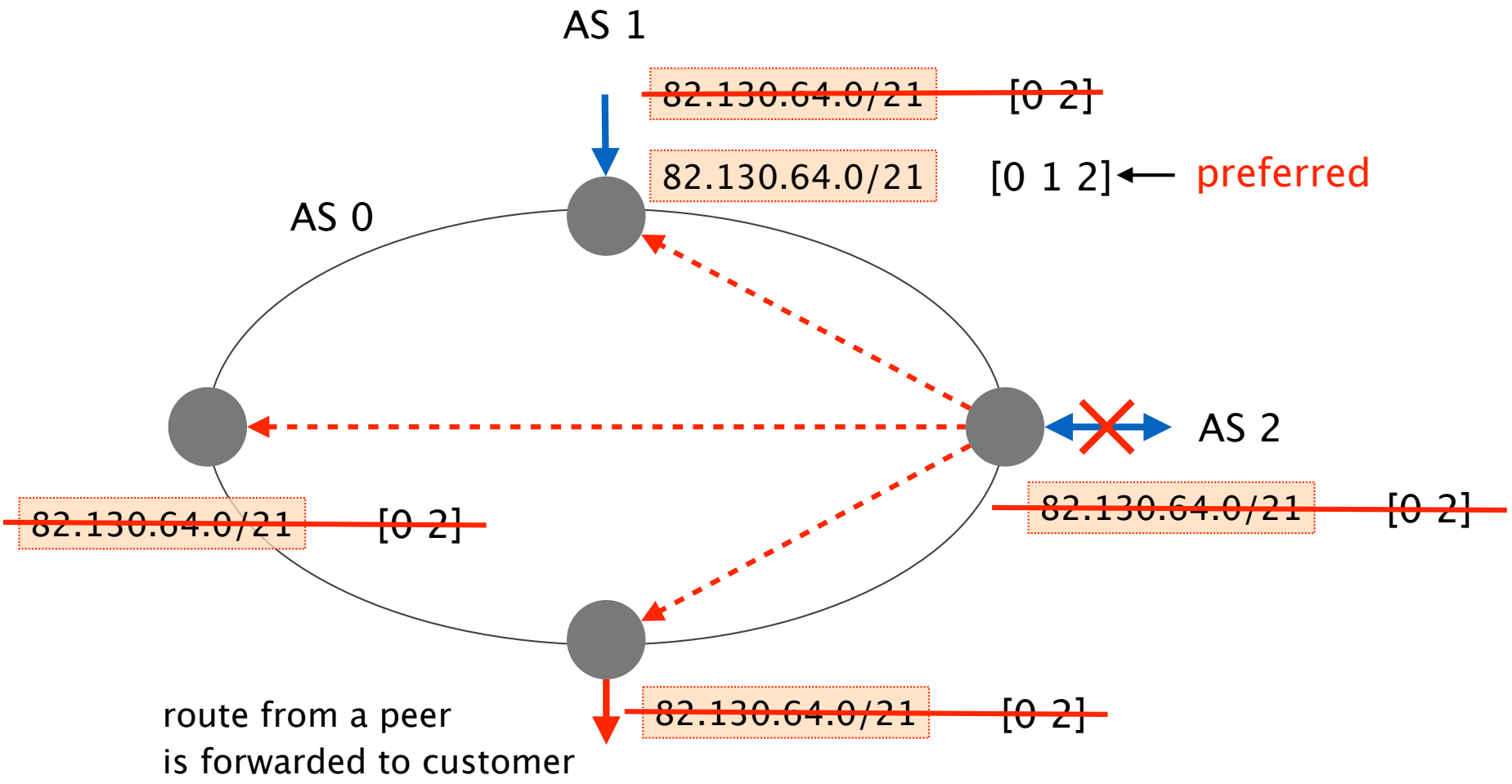
The directly connected routers detect the failure first



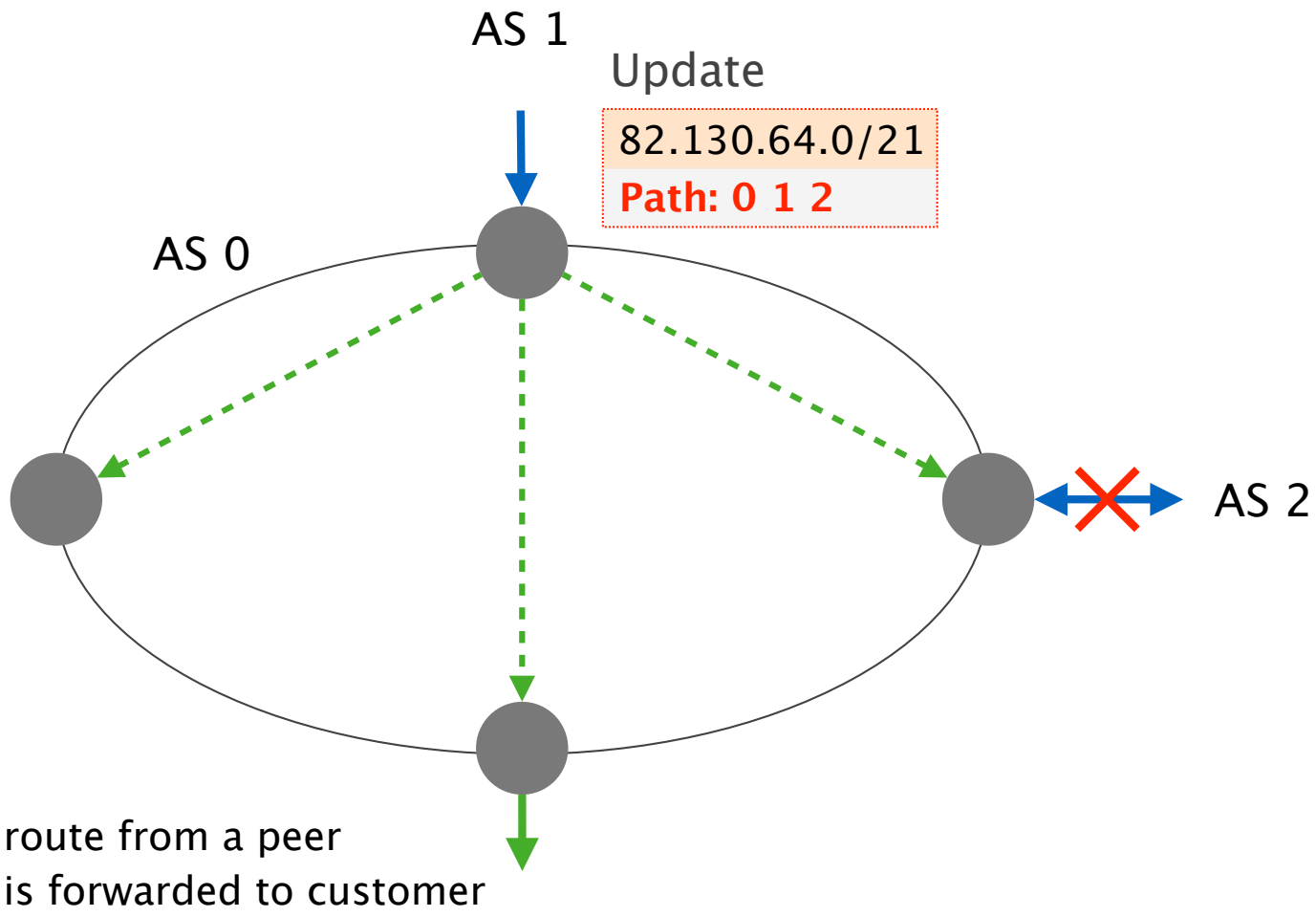
The router closest to the failure withdraws its current best route



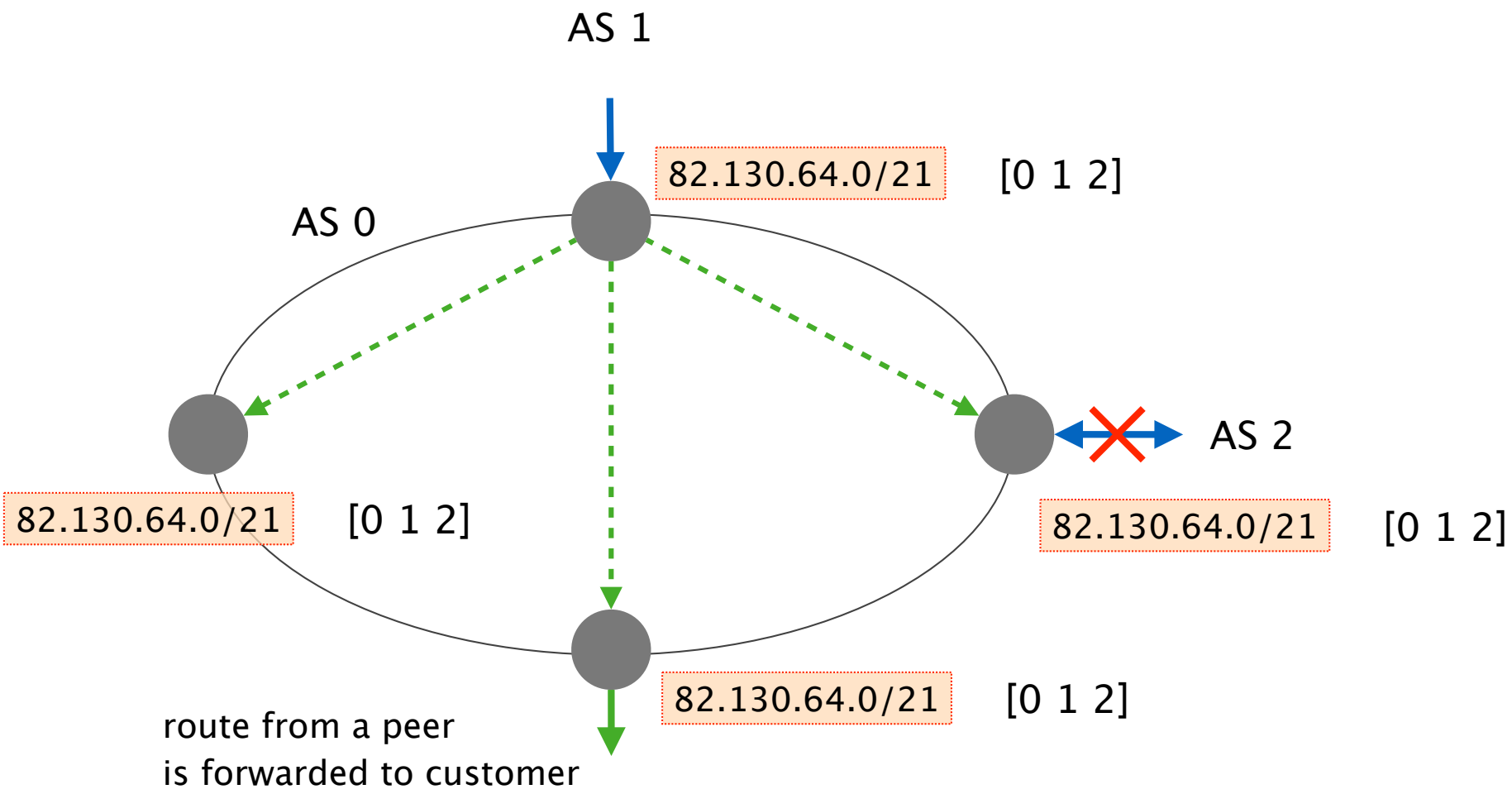
The router closest to the failure withdraws its current best route



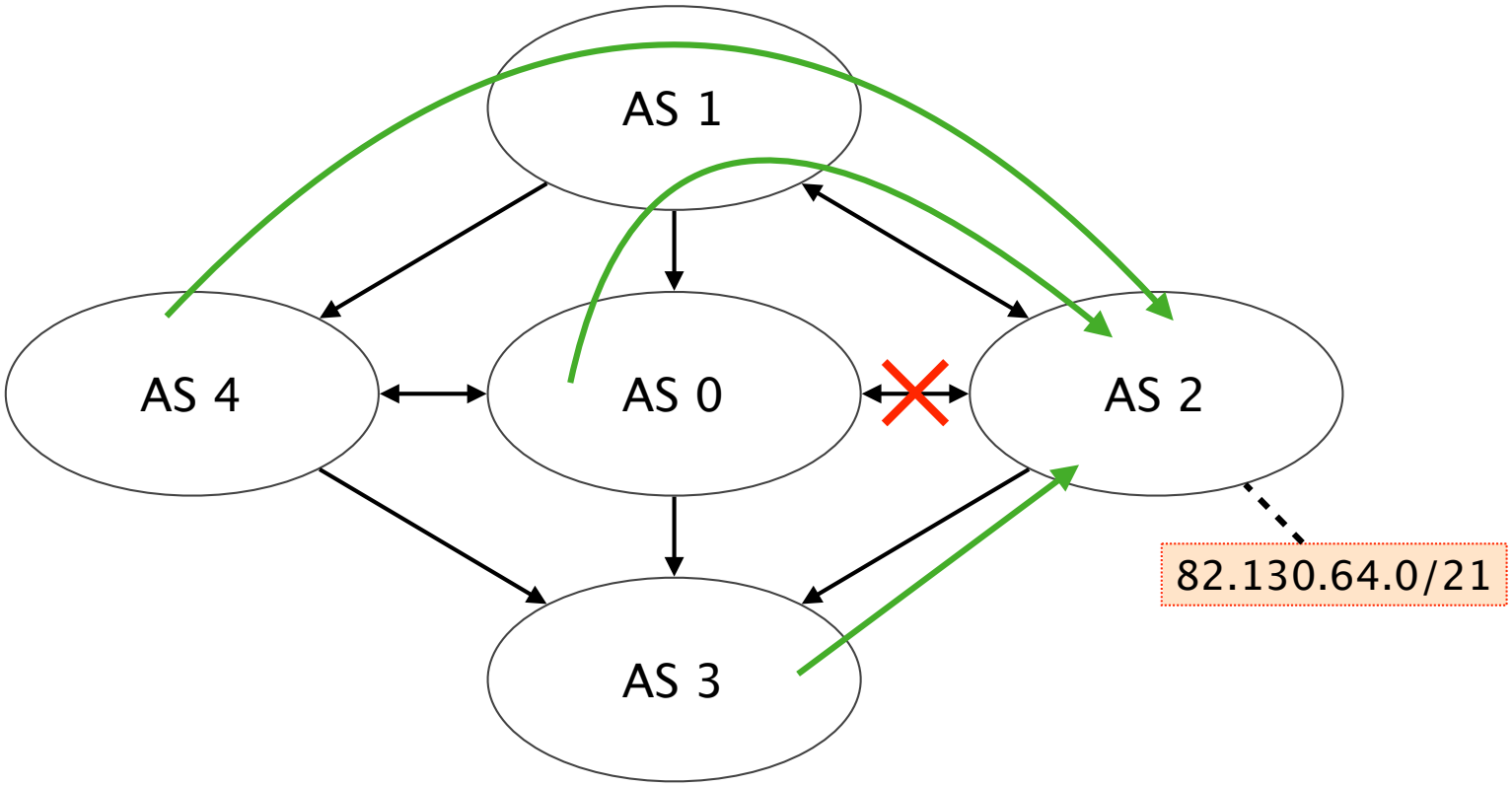
The router connected to AS 1 announces its new best route



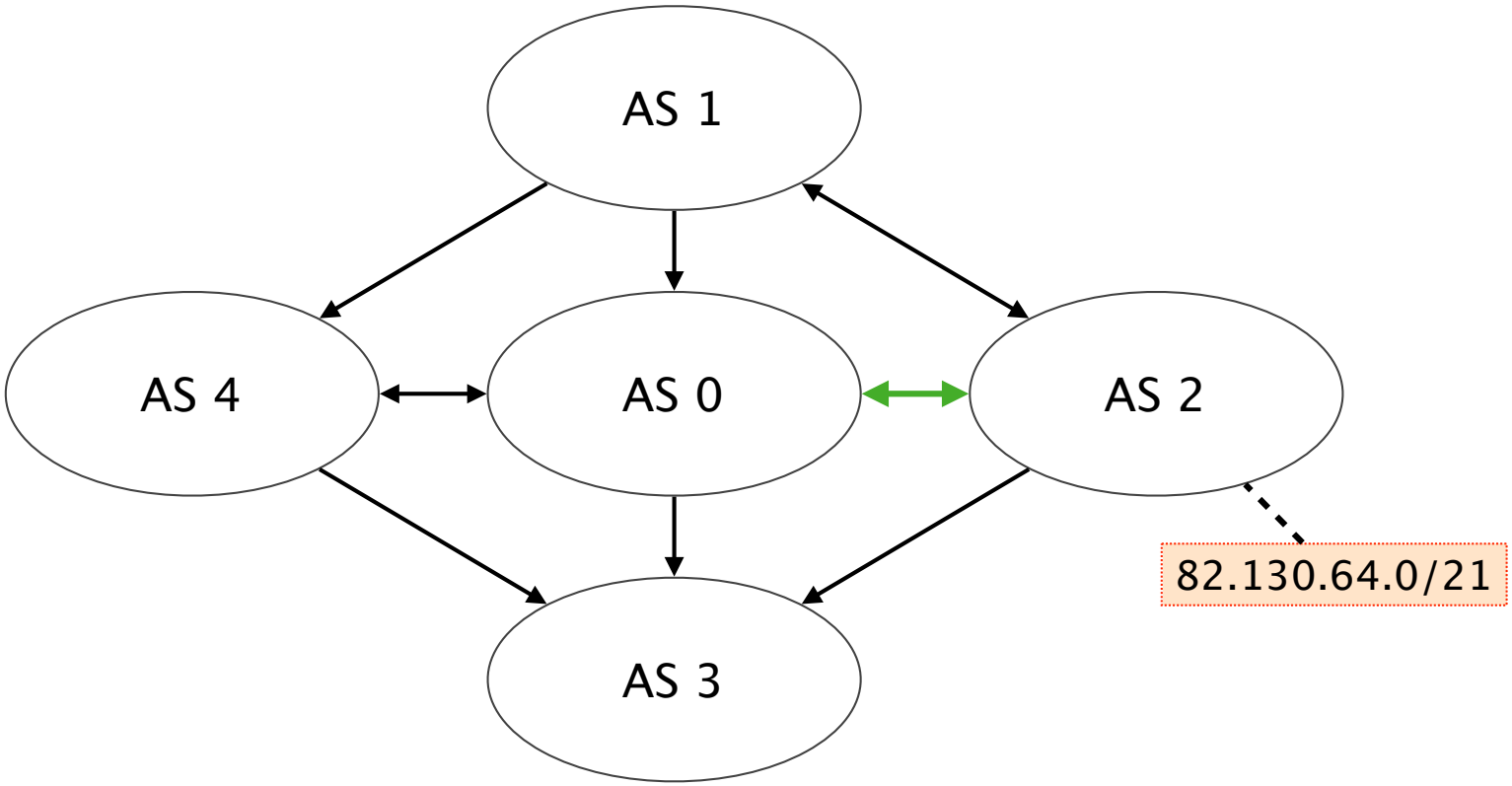
Every router is once again able to reach the prefix



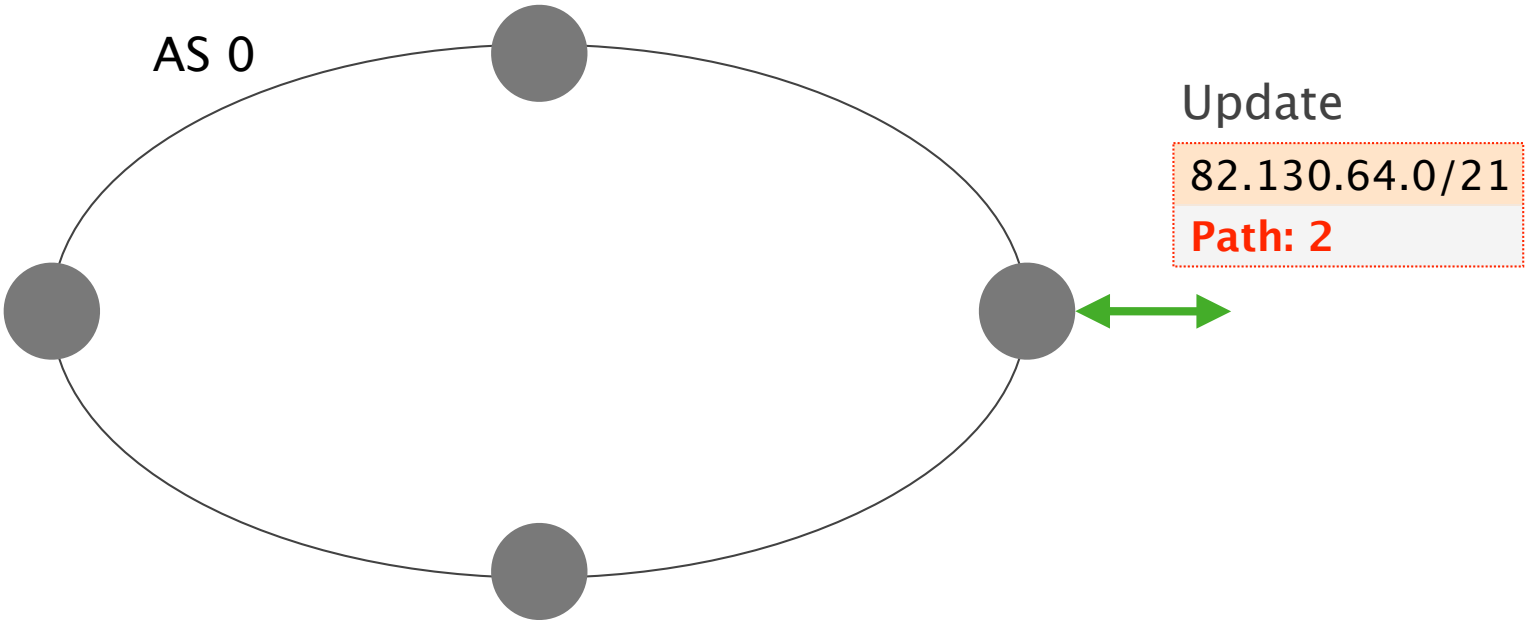
Updated forwarding paths



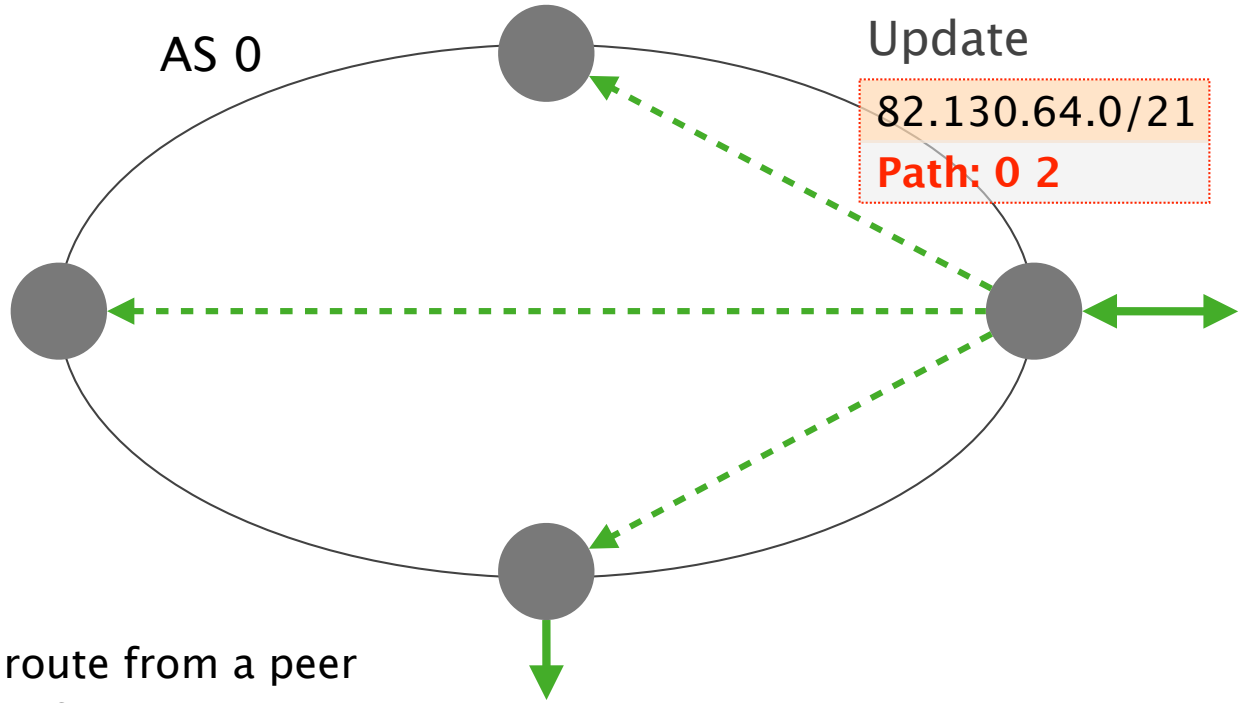
Now the link between AS 0 and AS 2 is working again



AS 0 receives an update from AS 2

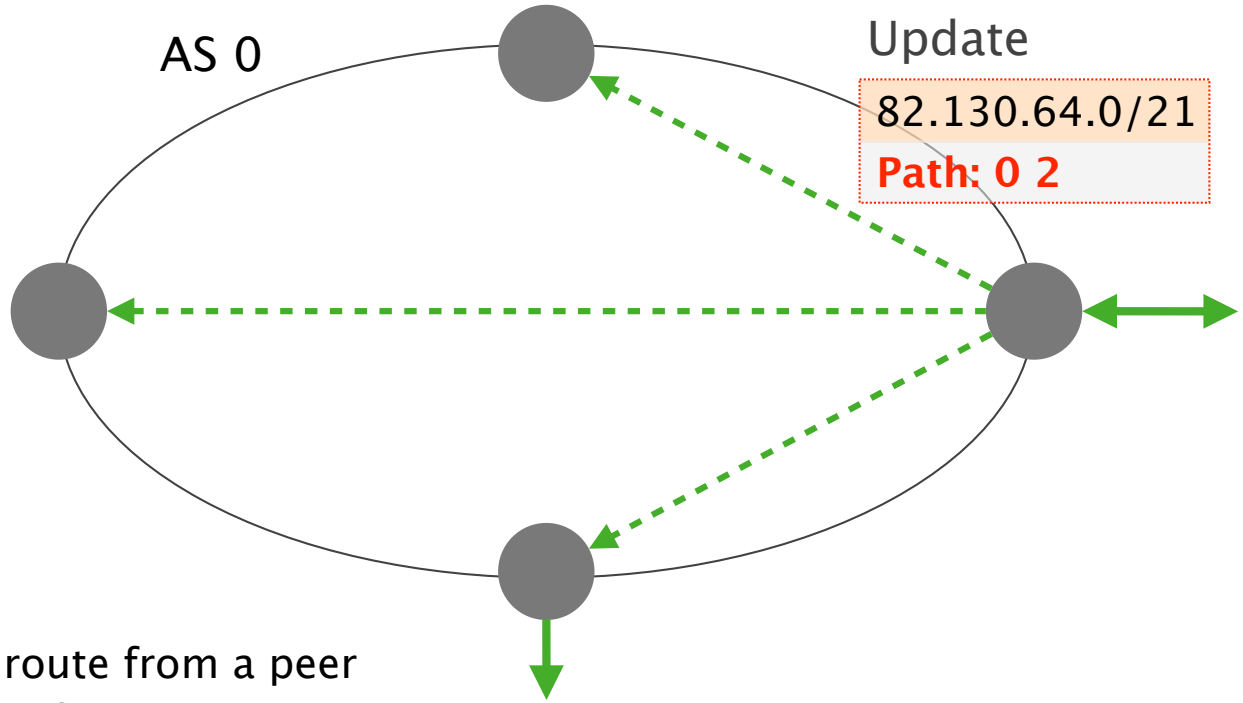


The router connected to AS 2 propagates this route as it is preferred



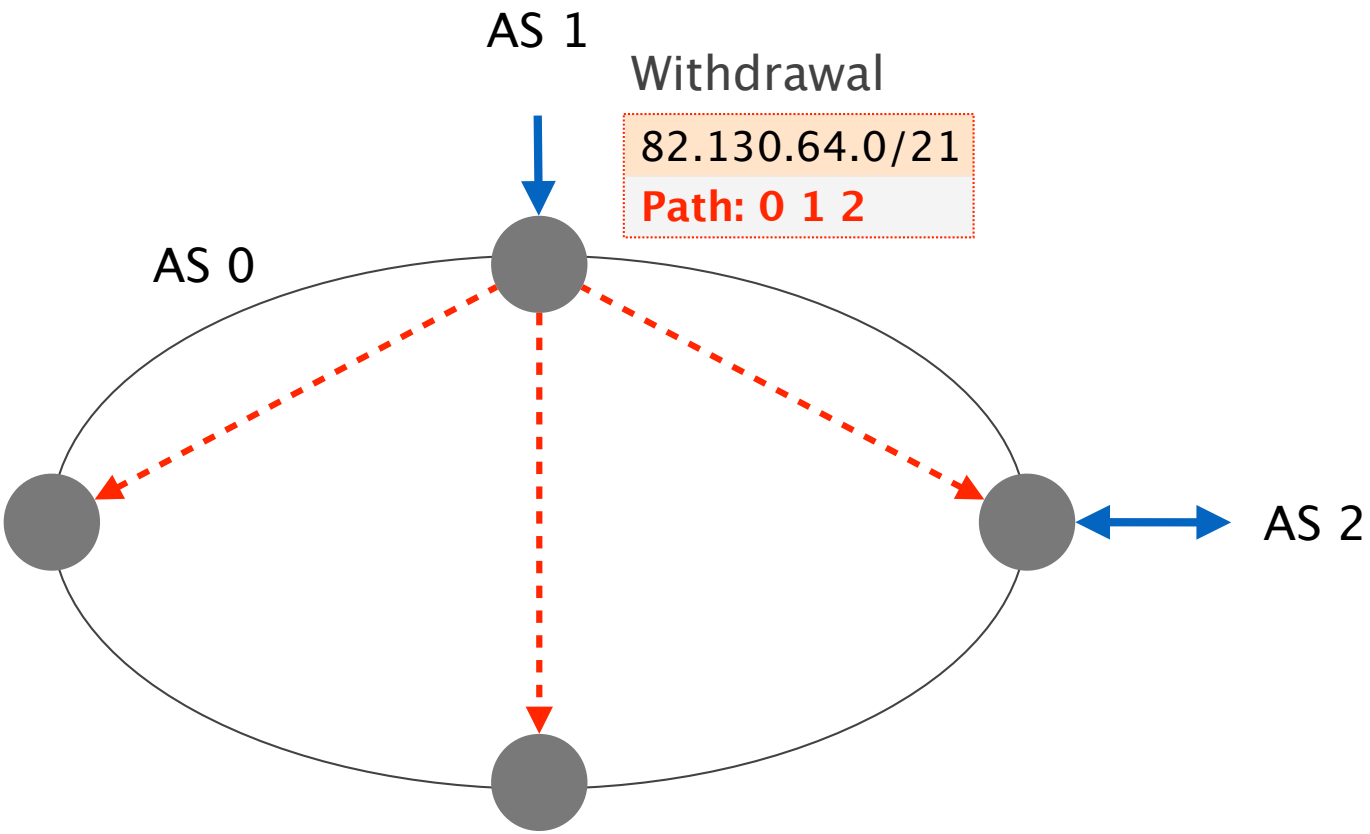
	AS path	
82.130.64.0/21	[0 2] ←	preferred
82.130.64.0/21	[0 1 2]	

At this point, every router knows about both routes



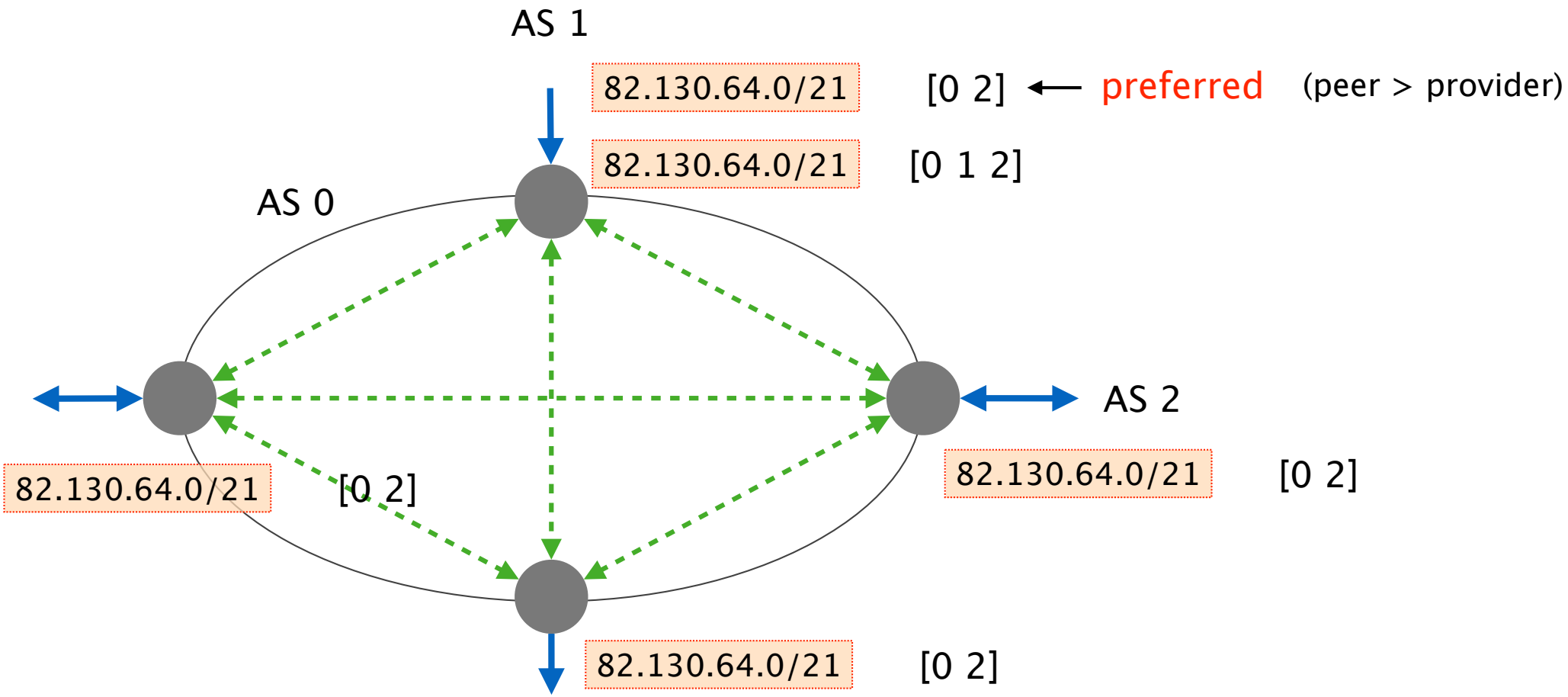
	AS path	
82.130.64.0/21	[0 2]	← preferred
82.130.64.0/21	[0 1 2]	

Finally, the router connected to AS 1 withdraws its second, non-best route



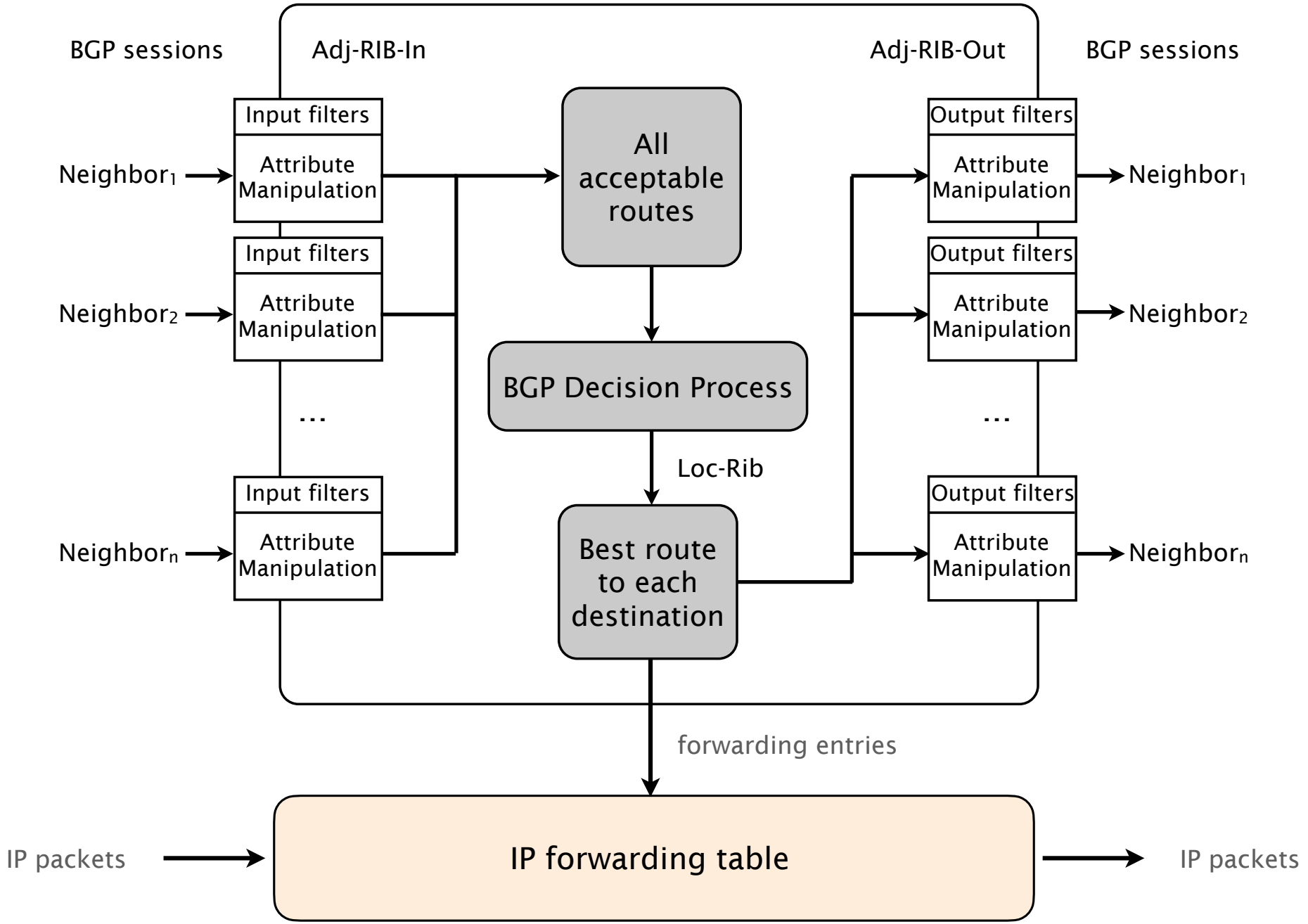
This withdrawal does not need to be forwarded to the peer. The previous update implicitly removed the route

We are once again in the initial state



High-level recap of the inner workings of the BGP protocol

How does the decision process work?


BGP sessions
Adj-RIB-In
Adj-RIB-Out
BGP sessions
Input filters
Attribute Manipulation
Neighbor1
Input filters
Attribute Manipulation
Neighbor2
...
Input filters
Attribute Manipulation
Neighborn
All acceptable routes
BGP Decision Process
Loc-Rib
Best route to each destination
Output filters
Attribute Manipulation
Neighbor1
Output filters
Attribute Manipulation
Neighbor2
...
Output filters
Attribute Manipulation
Neighborn
forwarding entries
IP packets
IP forwarding table
IP packets

Given the set of all acceptable routes for each prefix, the BGP Decision process elects a single route

BGP is often referred to as a single path protocol

Prefer routes…

with higher LOCAL-PREF

with shorter AS-PATH length

with lower MED

learned via eBGP instead of iBGP

with lower IGP metric to the next-hop

with smaller egress IP address (tie-break)

An AS influences the traffic
by modifying route attributes

Attributes	Usage
NEXT-HOP	egress point identification
AS-PATH	loop avoidance outbound traffic control inbound traffic control
LOCAL-PREF	outbound traffic control
MED	inbound traffic control

How relevant are the group projects for the exam?

How relevant are the group projects for the exam?

We will not ask you to write correct FRRouting configs

But you should be able to describe something in pseudo code

We could also show you a simple FRRouting config snippet

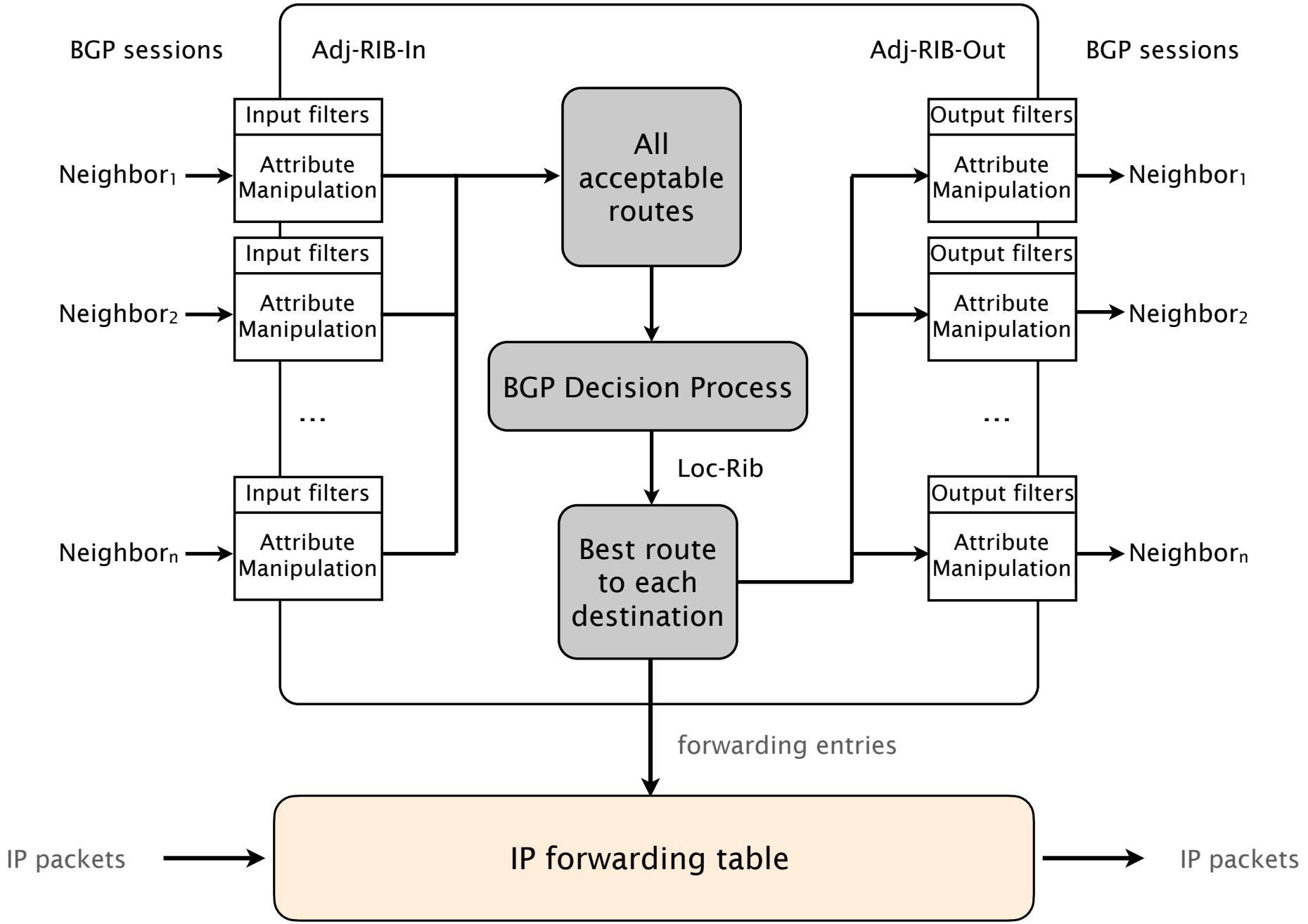
You should be able to understand what is happening

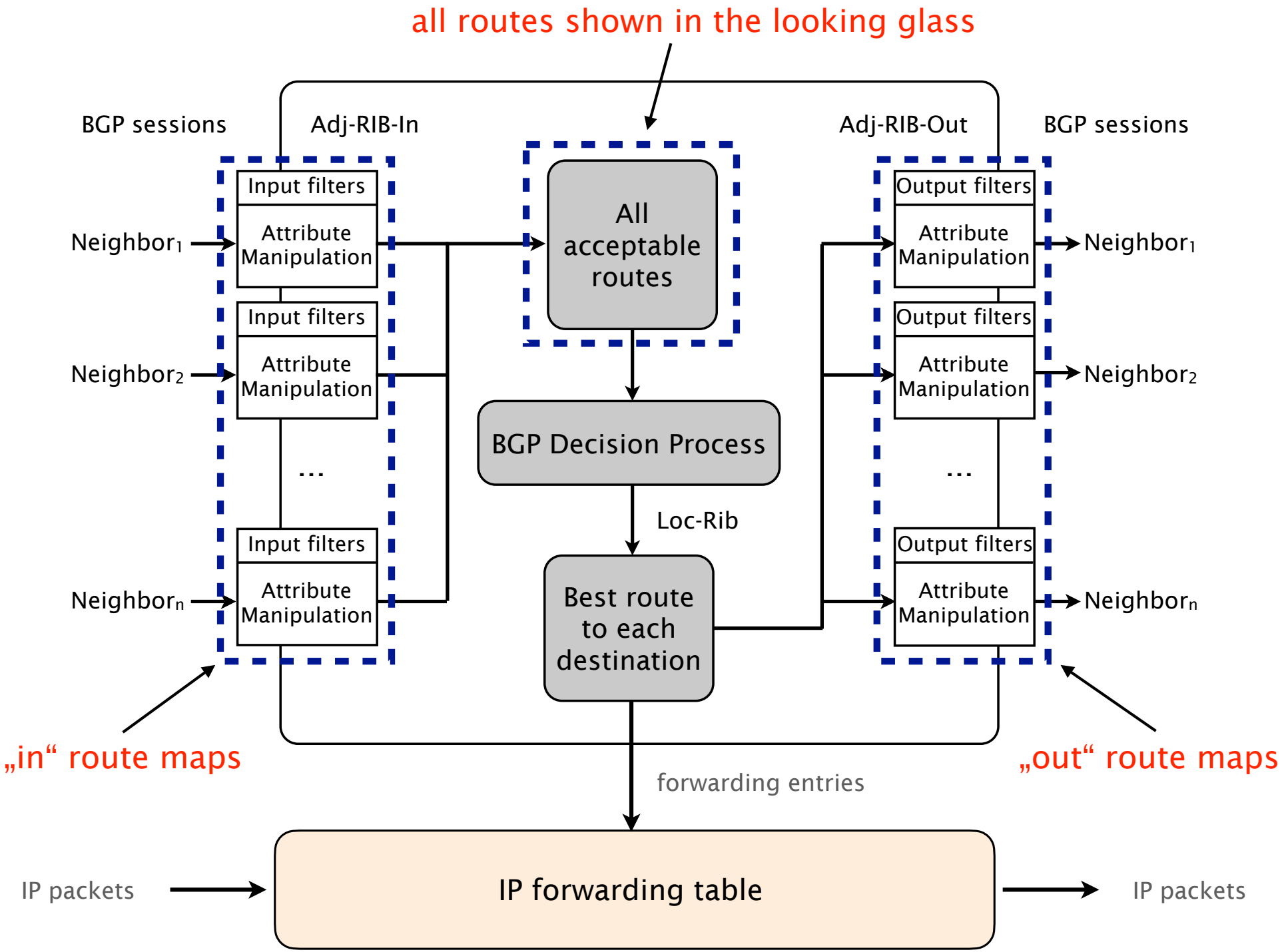
You do not need to write Python program code

Again, we could ask to explain something in pseudo code

How does the routing project relate to what we just saw?

Where do the "route-maps" come into play?


BGP sessions
Adj-RIB-In
Adj-RIB-Out
BGP sessions
Input filters
Attribute Manipulation
Neighbor1
Input filters
Attribute Manipulation
Neighbor2
...
Input filters
Attribute Manipulation
Neighborn
All acceptable routes
BGP Decision Process
Loc-Rib
Best route to each destination
Output filters
Attribute Manipulation
Neighbor1
Output filters
Attribute Manipulation
Neighbor2
...
Output filters
Attribute Manipulation
Neighborn
forwarding entries
IP packets
IP forwarding table
IP packets


all routes shown in the looking glass
BGP sessions
Adj-RIB-In
Adj-RIB-Out
BGP sessions
Input filters
Attribute Manipulation
Neighbor1
Input filters
Attribute Manipulation
Neighbor2
...
Input filters
Attribute Manipulation
Neighborn
All acceptable routes
BGP Decision Process
Loc-Rib
Best route to each destination
Output filters
Attribute Manipulation
Neighbor1
Output filters
Attribute Manipulation
Neighbor2
...
Output filters
Attribute Manipulation
Neighborn
„in“ route maps
„out“ route maps
forwarding entries
IP packets
IP forwarding table
IP packets

Let’s look at some examples

The shown examples are not correct FRRouting configs!

But they roughly show the level we would expect at the exam

Policy enforcement with local pref and BGP communities

IN route-map
connected to a **customer**

OUT route-map
connected to a **provider**

Policy enforcement with local pref and BGP communities

IN route-map
connected to a **customer**

```
for all routes:

  -> set local pref to 1000

  -> add community 10:500
```

OUT route-map
connected to a **provider**

Policy enforcement with local pref and BGP communities

IN route-map
connected to a **customer**

```
for all routes:

  -> set local pref to 1000

  -> add community 10:500
```

OUT route-map
connected to a **provider**

```
if route has community 10:500:

  -> send out

if route has community 10:100 or 10:200:

  -> drop / do not send out

if route is one of our prefixes:

  -> send out
```

Policy enforcement with local pref and BGP communities

IN route-map
connected to a **customer**

```
for all routes:
  -> set local pref to 1000
  -> add community 10:500
```

influences BGP
decision process

as tags for other
routers in the
same network

OUT route-map
connected to a **provider**

```
if route has community 10:500:
  -> send out
if route has community 10:100 or 10:200:
  -> drop / do not send out
if route is one of our prefixes:
  -> send out
```

our own prefixes are
always advertised

drop routes from
peers and providers

Inbound traffic engineering (option 1), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

OUT route-map
connected to **same** peer in BERN

Inbound traffic engineering (option 1), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> set MED to 10
```

OUT route-map
connected to **same** peer in BERN

Inbound traffic engineering (option 1), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> set MED to 10
```

OUT route-map
connected to **same** peer in BERN

```
for our prefix:

  -> set MED to 200
```

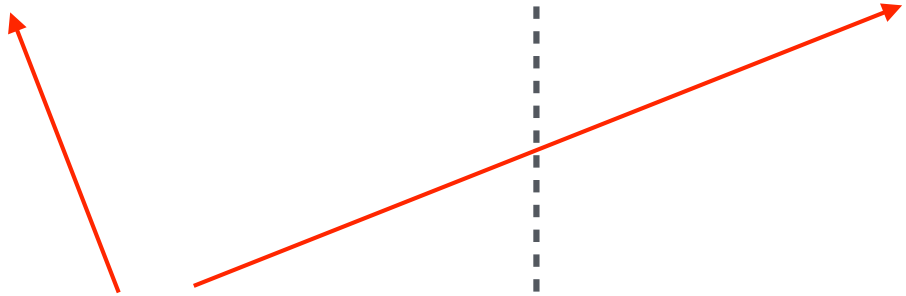
Inbound traffic engineering (option 1), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> set MED to 10
```

OUT route-map
connected to **same** peer in BERN

```
for our prefix:

  -> set MED to 200
```



We try to influence
the BGP decision process
of the peer

Remember, lower is better!

Inbound traffic engineering (option 2), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> send out unmodified
```

OUT route-map
connected to **same** peer in BERN

Inbound traffic engineering (option 2), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> send out unmodified
```

OUT route-map
connected to **same** peer in BERN

```
for our prefix:

  -> add our AS number 5 times to AS PATH
```

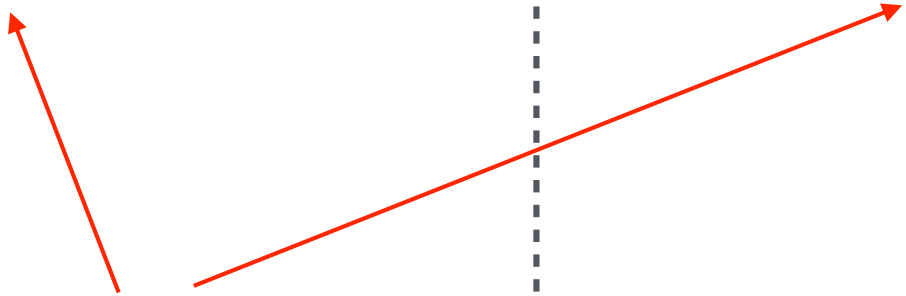
Inbound traffic engineering (option 2), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> send out unmodified
```

OUT route-map
connected to **same** peer in BERN

```
for our prefix:

  -> add our AS number 5 times to AS PATH
```



This time we influence
the AS PATH length

Could still be overwritten
by peer (e.g., local pref)

Inbound traffic engineering (option 3), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> send out unmodified
```

OUT route-map
connected to **same** peer in BERN

Inbound traffic engineering (option 3), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> send out unmodified
```

OUT route-map
connected to **same** peer in BERN

```
for our prefix:

  -> drop / do not send out
```

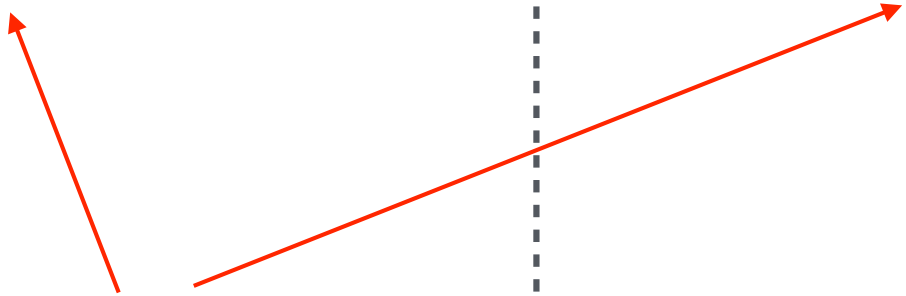
Inbound traffic engineering (option 3), traffic to our prefix should enter via ZURI

OUT route-map
connected to **peer** in ZURI

```
for our prefix:

  -> send out unmodified
```

OUT route-map
connected to **same** peer in BERN

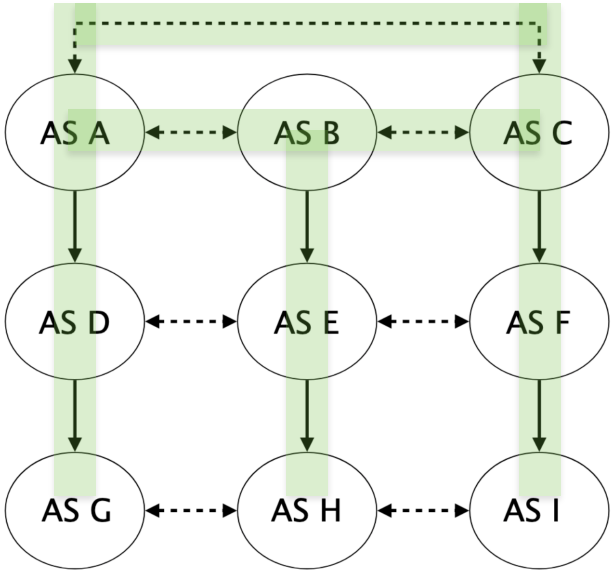
```
for our prefix:

  -> drop / do not send out
```



We should really only receive traffic via ZURI

But what if this connection fails? We lose all traffic

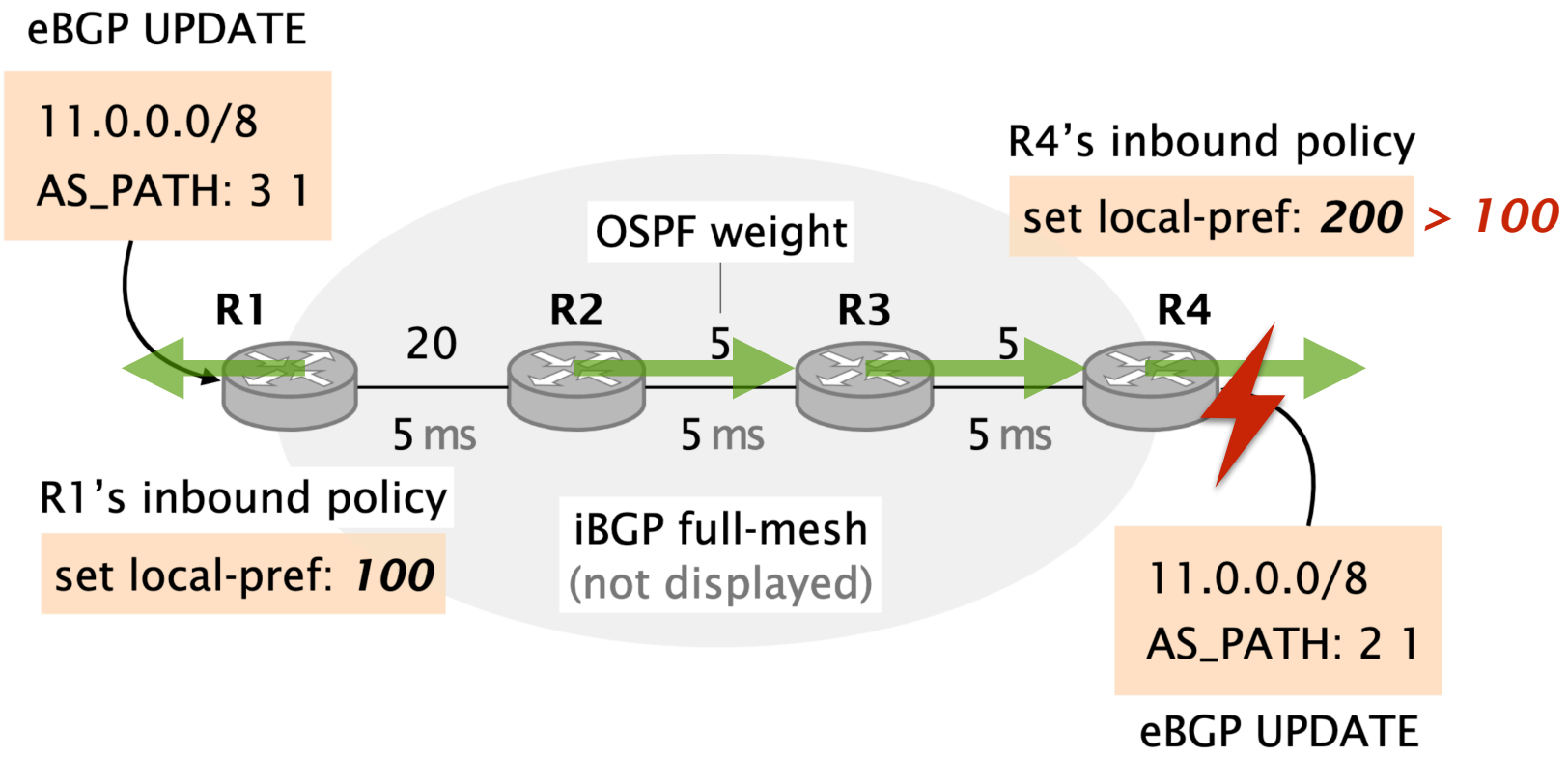
Exercise 7, task 3c)



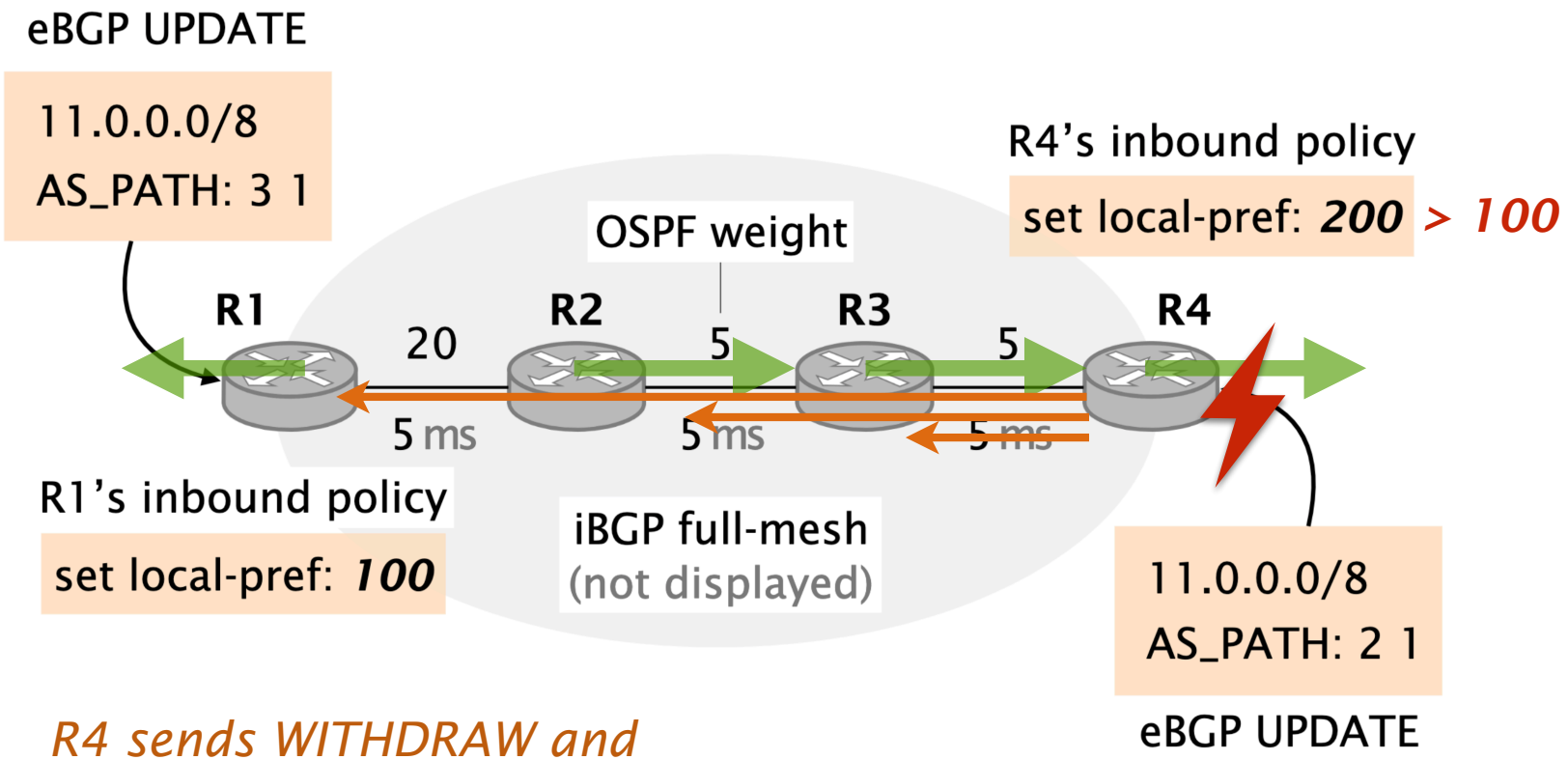
given A's and C's dumps,
draw AS-level topology.

AS relationships,
e.g. for A and B?

Exercise 7, task 4)



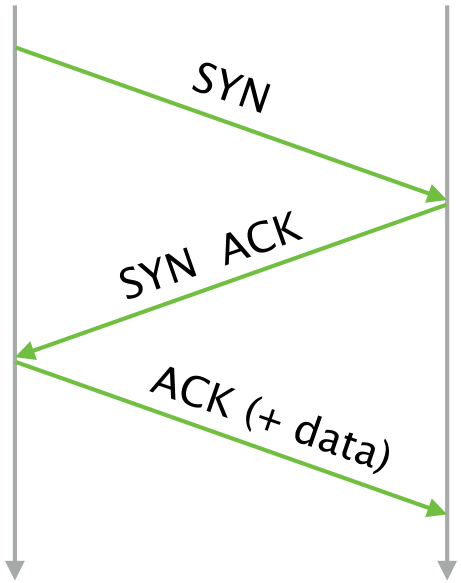
Exercise 7, task 4)



R4 sends WITHDRAW and
uses route over R1 instead

how do R3 & R4 know the path over R1?

How many RTT is a TCP handshake?



1 RTT is enough since one can already send data with the last ACK

Exam 2016 4.c)ii): what changes to the SACK algo?

reminder:

we do not answer any questions related to the old exams

Today's Q&A session

Important concepts

Answering received questions

Individual questions

Important information

Question deadline this Saturday 20.08.2022

Exam: 27.08.2022 – HIL G 41