FIT1047 - Week 9

Networks: Network and Transport layers

MONASH University

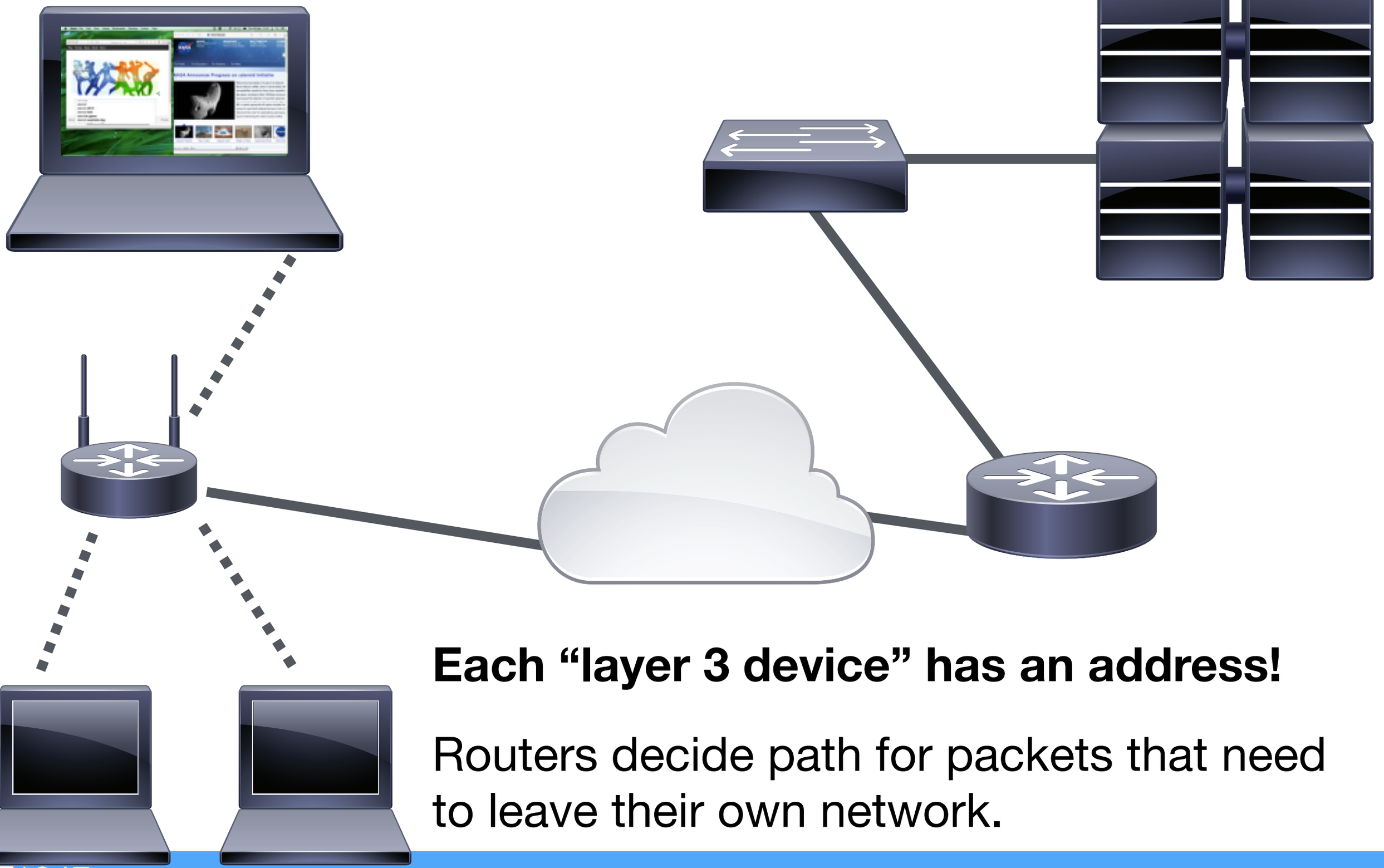
Goals for this week

- See how **routers connect different networks**
- Understand how the transport layer makes sure messages **arrive correctly** and at the **right process**
- Study the **structure of the Internet**

The Network Layer:

Adresses

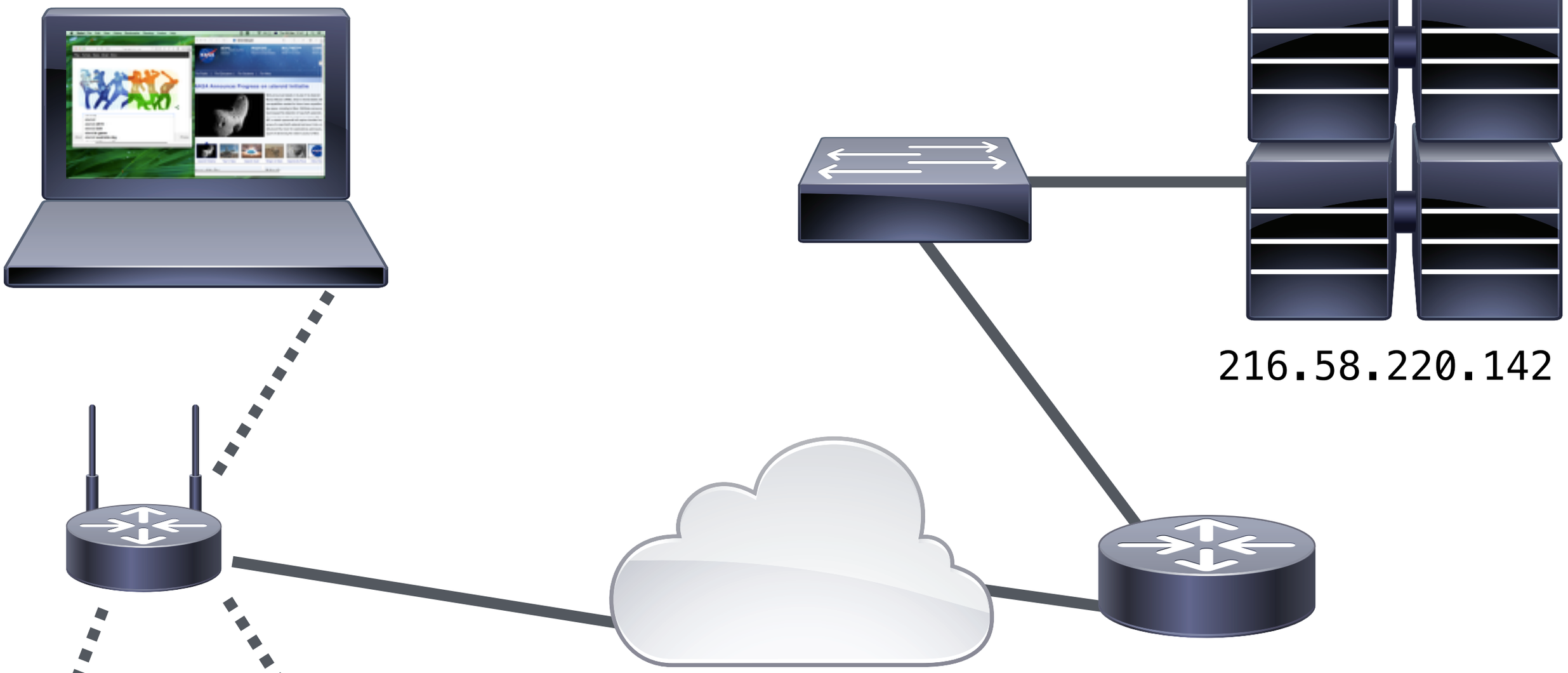
Addressing devices



Each “layer 3 device” has an address!

Routers decide path for packets that need to leave their own network.

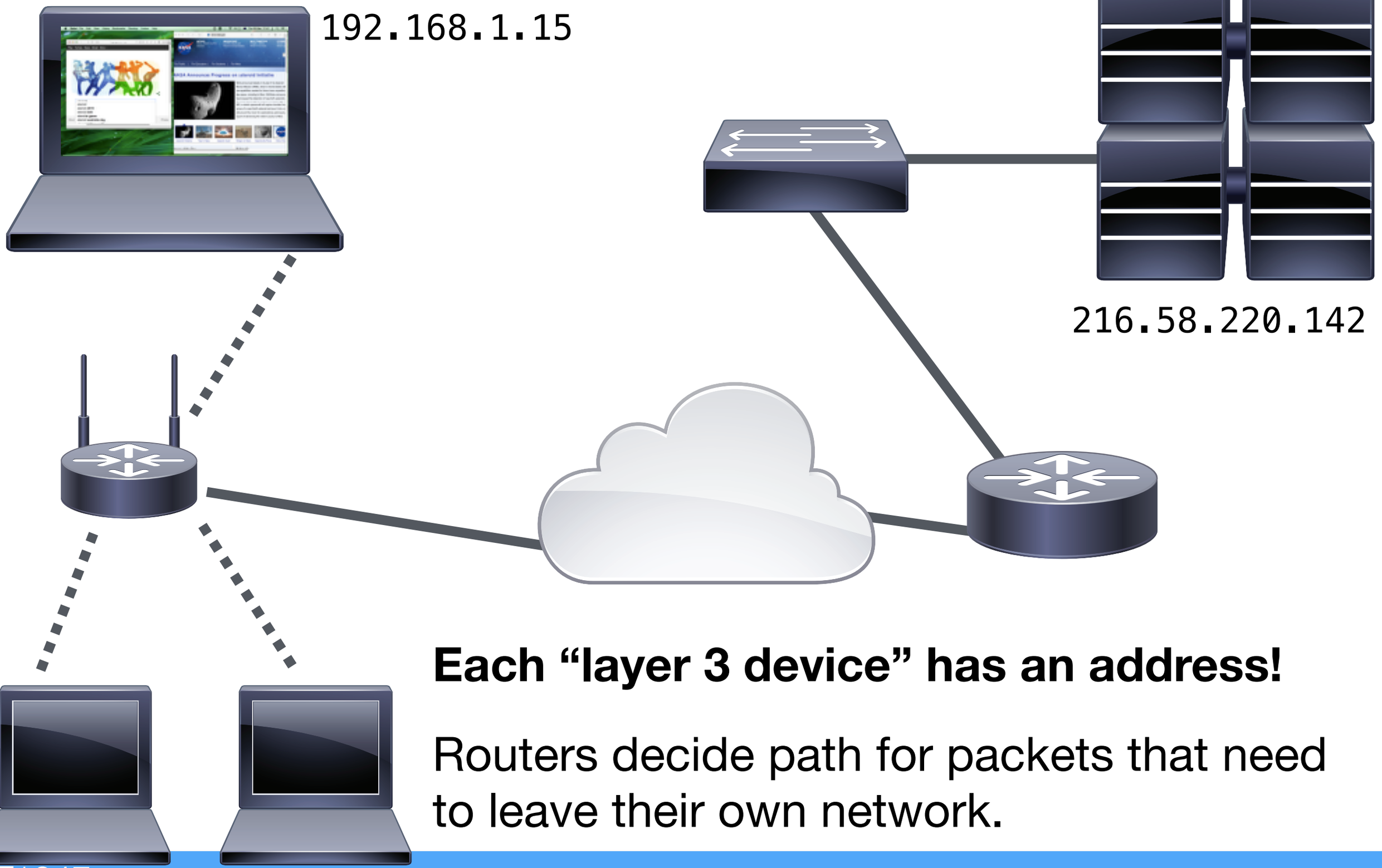
Addressing devices



Each “layer 3 device” has an address!

Routers decide path for packets that need to leave their own network.

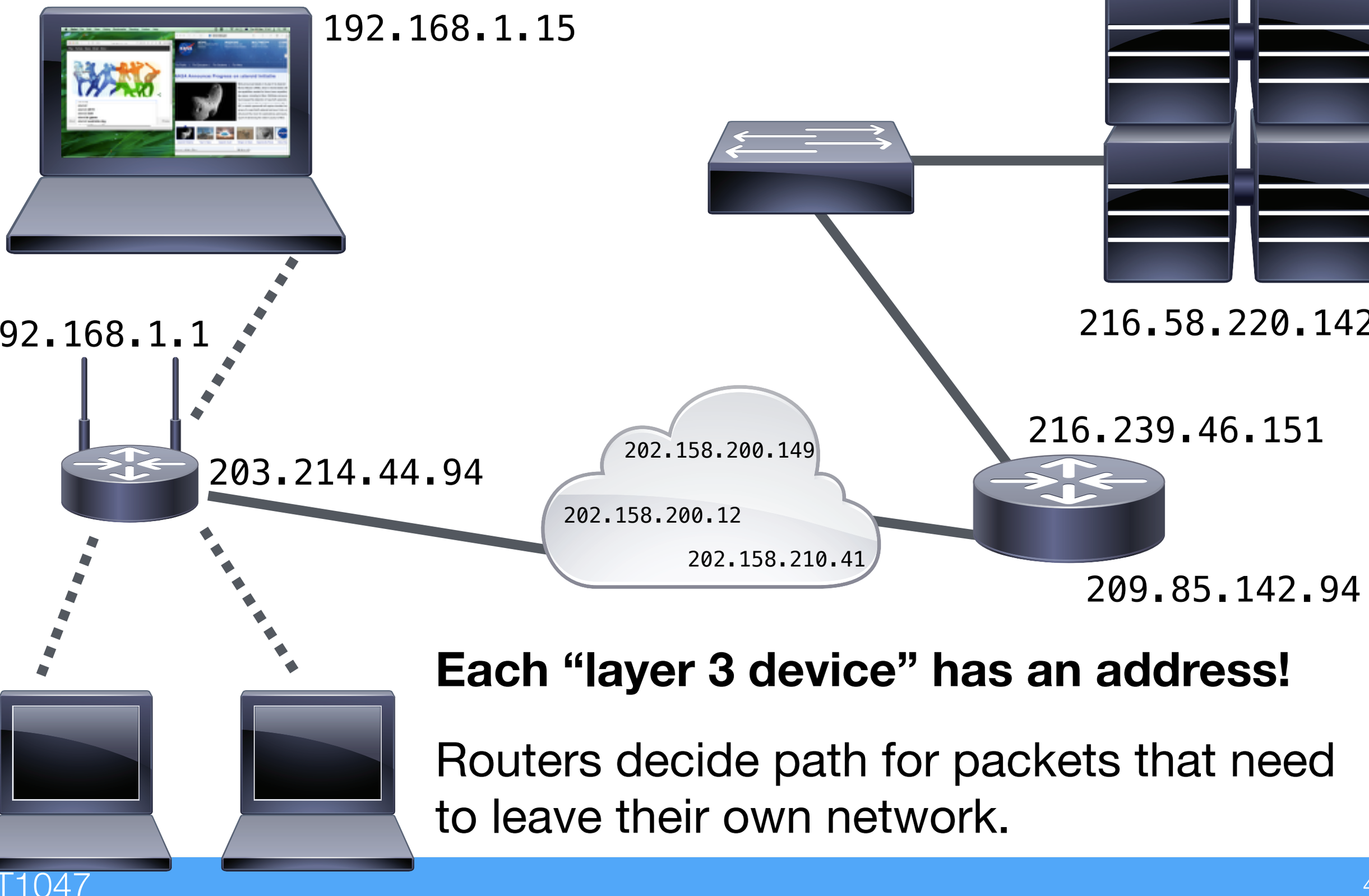
Addressing devices



Each “layer 3 device” has an address!

Routers decide path for packets that need to leave their own network.

Addressing devices



Each “layer 3 device” has an address!

Routers decide path for packets that need to leave their own network.

IP version 4 addresses

32 bit addresses

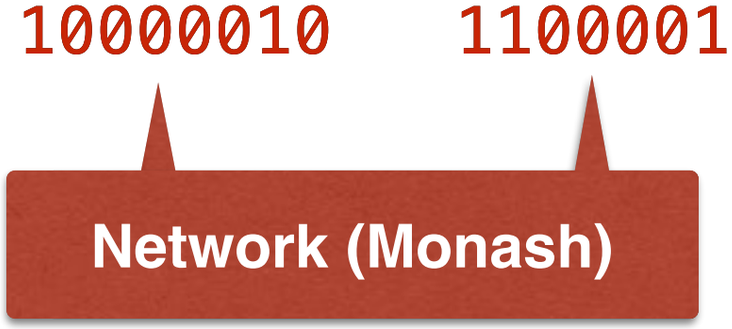
- Written using “dotted decimal” notation
- Example: 130.194.66.43

```
10000010    11000010    01000010    00101011
```

IP version 4 addresses

32 bit addresses

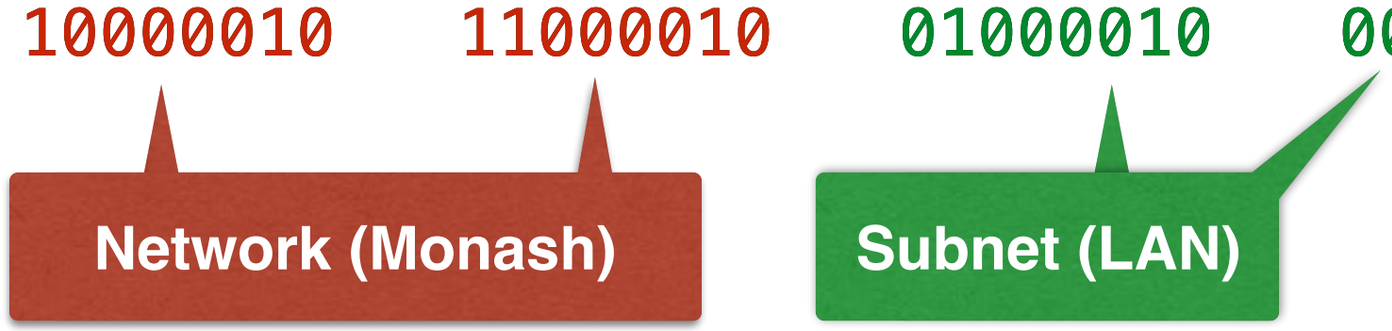
- Written using “dotted decimal” notation
- Example: 130.194.66.43



IP version 4 addresses

32 bit addresses

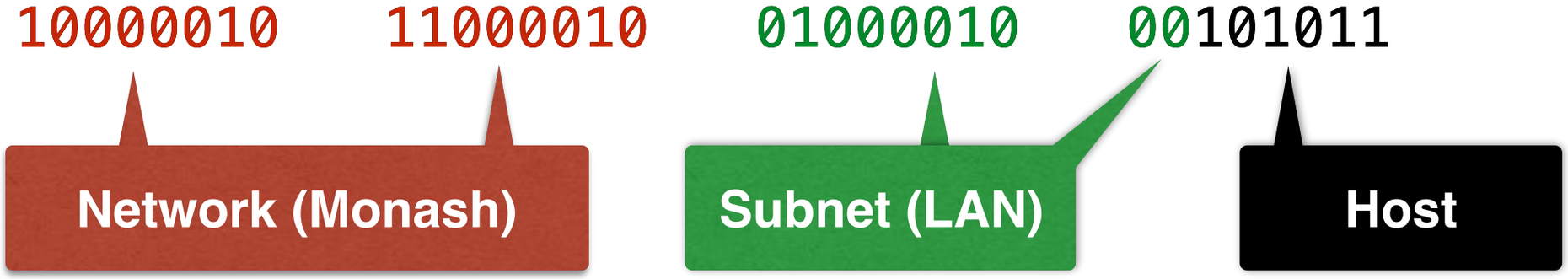
- Written using “dotted decimal” notation
- Example: 130.194.66.43



IP version 4 addresses

32 bit addresses

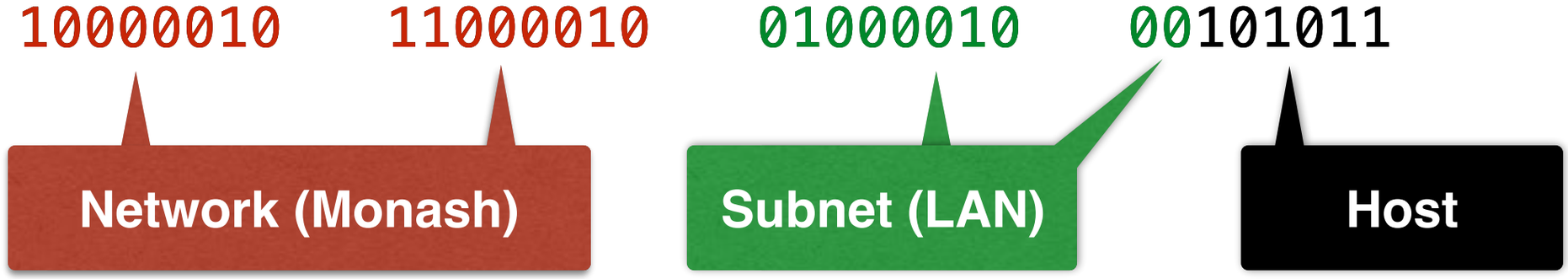
- Written using “dotted decimal” notation
- Example: 130.194.66.43



IP version 4 addresses

32 bit addresses

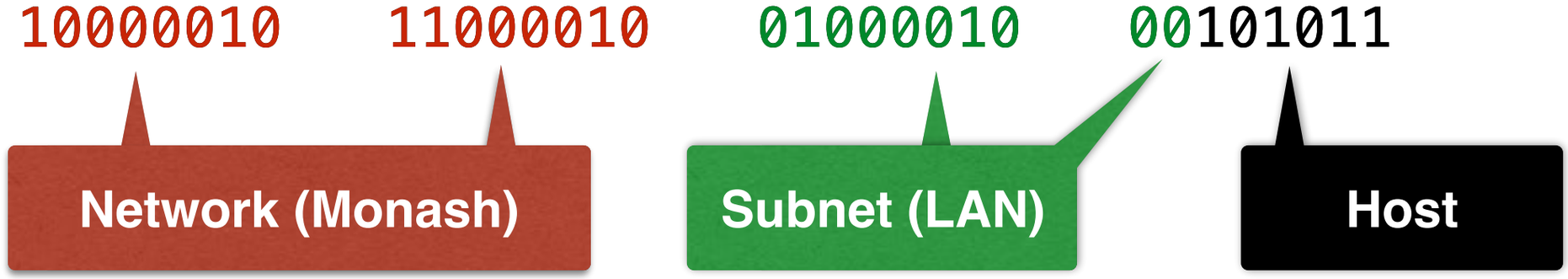
- Written using “dotted decimal” notation
- Example: 130.194.66.43



IP version 4 addresses

32 bit addresses

- Written using “dotted decimal” notation
- Example: 130.194.66.43



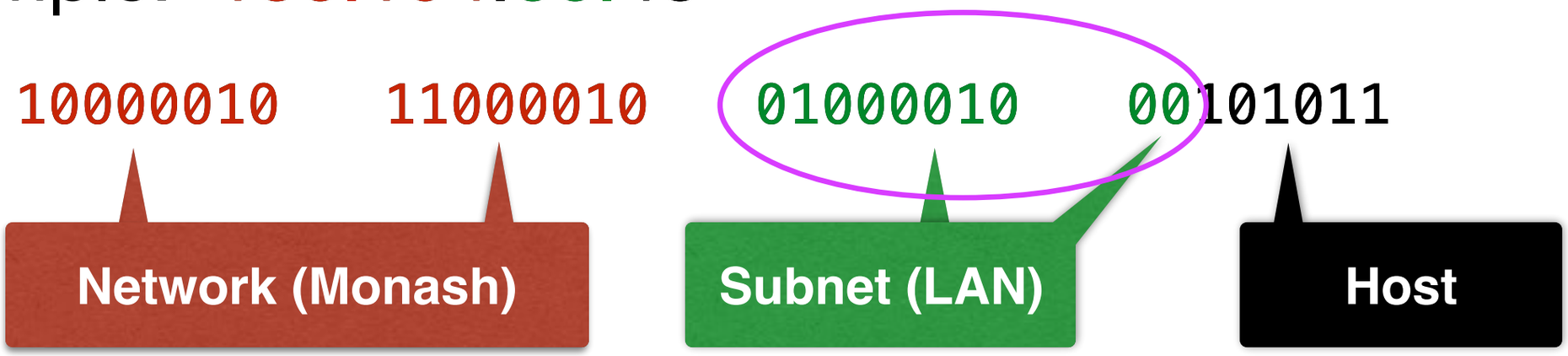
Hierarchy used for routing

- You can immediately see if a destination address is in the same subnet!
- Subnet **mask**: **255.255.192.0 or /22**

IP version 4 addresses

32 bit addresses

- Written using “dotted decimal” notation
- Example: 130.194.66.43



Hierarchy used for routing

- You can immediately see if a destination address is in the same subnet!
- Subnet **mask**: **255.255.192.0 or /22**

no. of bits in network + subnet

Network Classes

Subnet masks often expressed using dotted notation

- e.g. 255.255.0.0 meaning /16

Network Classes

Subnet masks often expressed using dotted notation

- e.g. 255.255.0.0 meaning /16

Previously used hierarchy:

- Class A: /8 (e.g. IBM, MIT, AT&T, Apple, …)
- Class B: /16 (e.g. Monash 130.194.0.0/16)
- Class C: /24

Network Classes

Subnet masks often expressed using dotted notation

- e.g. 255.255.0.0 meaning /16

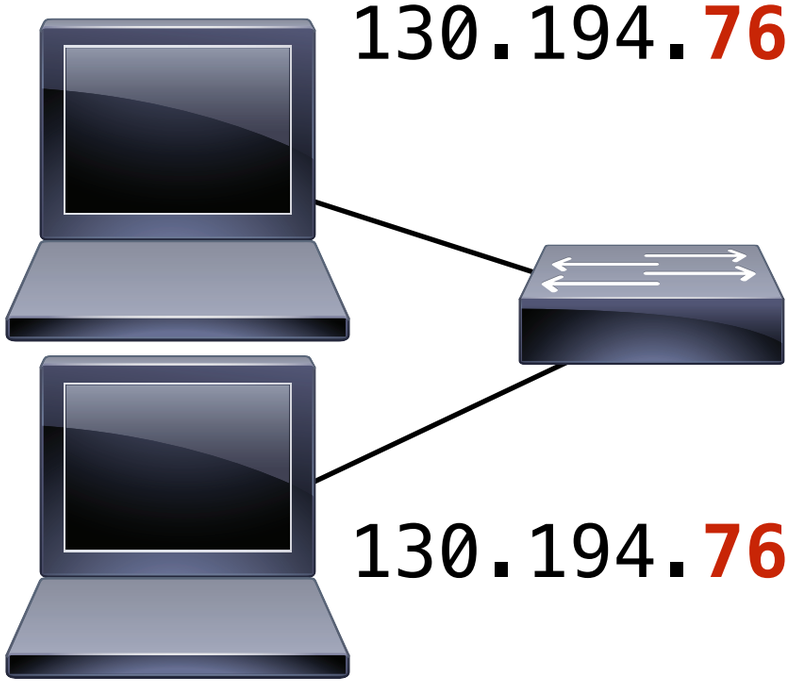
Previously used hierarchy:

- Class A: /8 (e.g. IBM, MIT, AT&T, Apple, …)
- Class B: /16 (e.g. Monash 130.194.0.0/16)
- Class C: /24

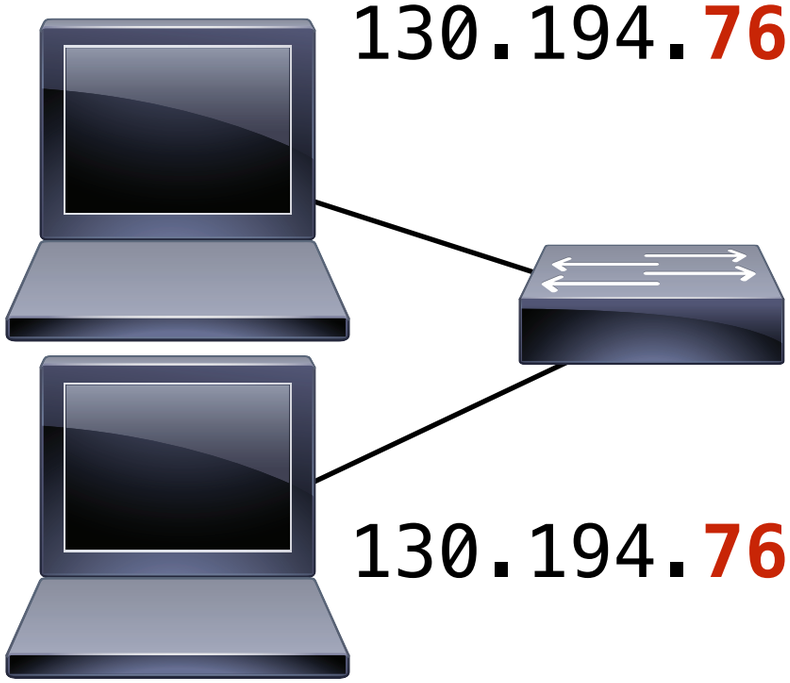
Now: classless

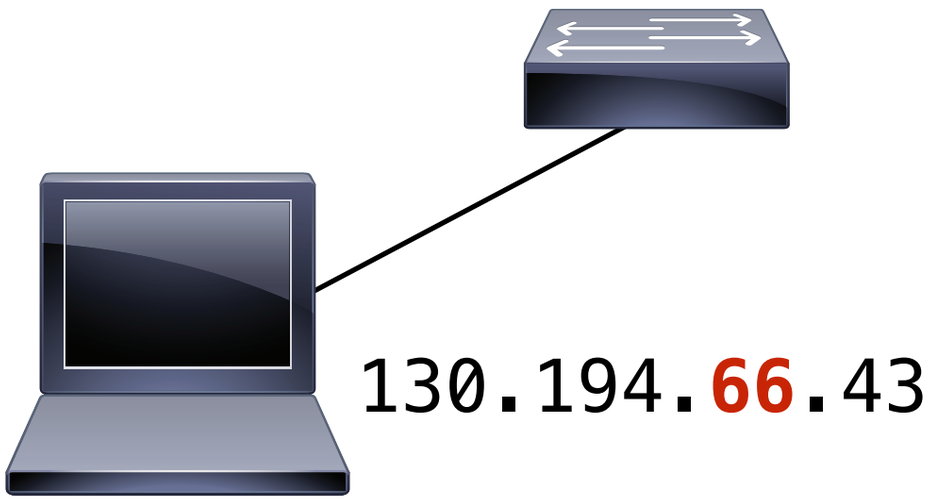
- e.g. /22, which can also be written as 255.255.252.0

Subnets

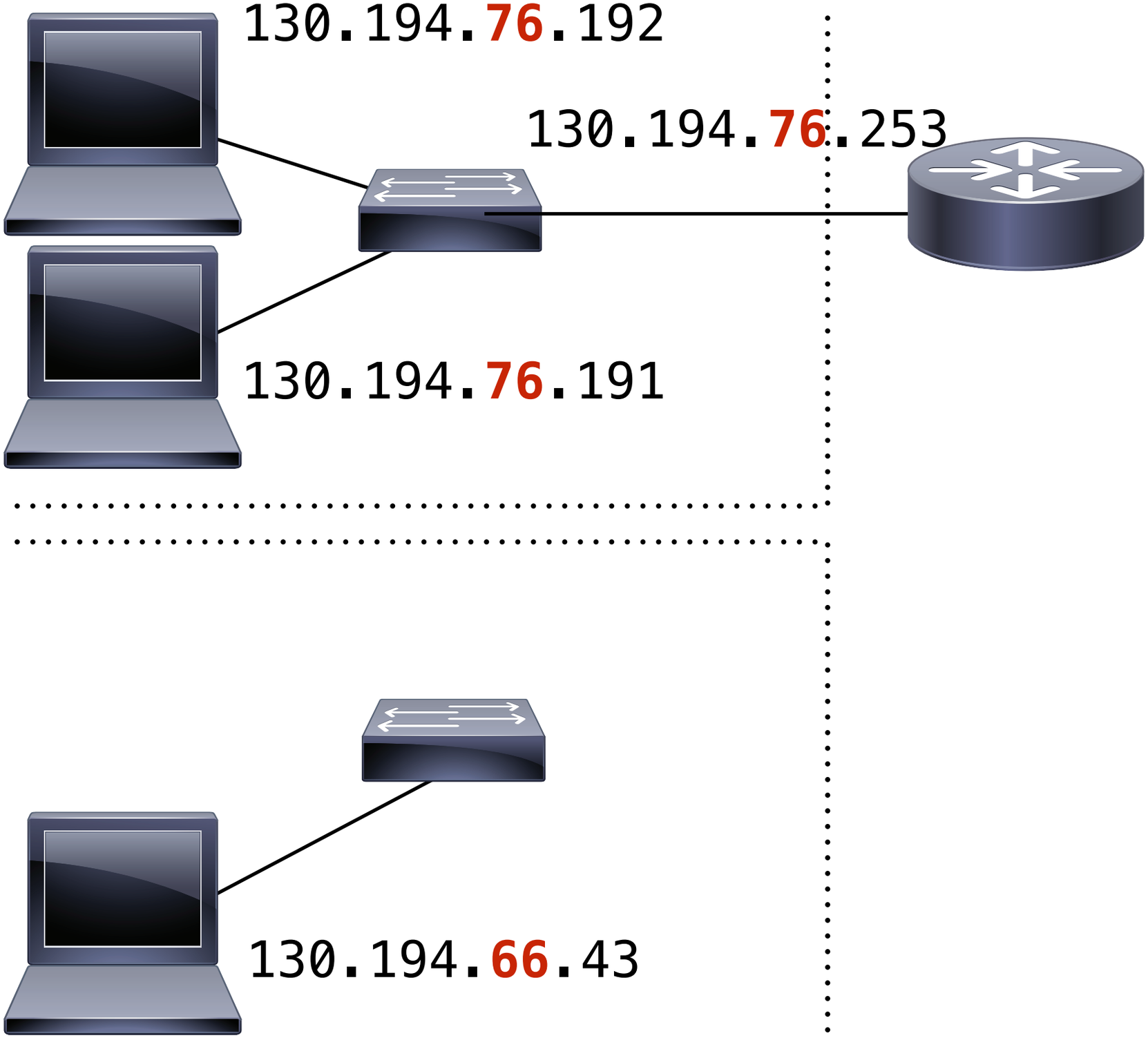


Subnets

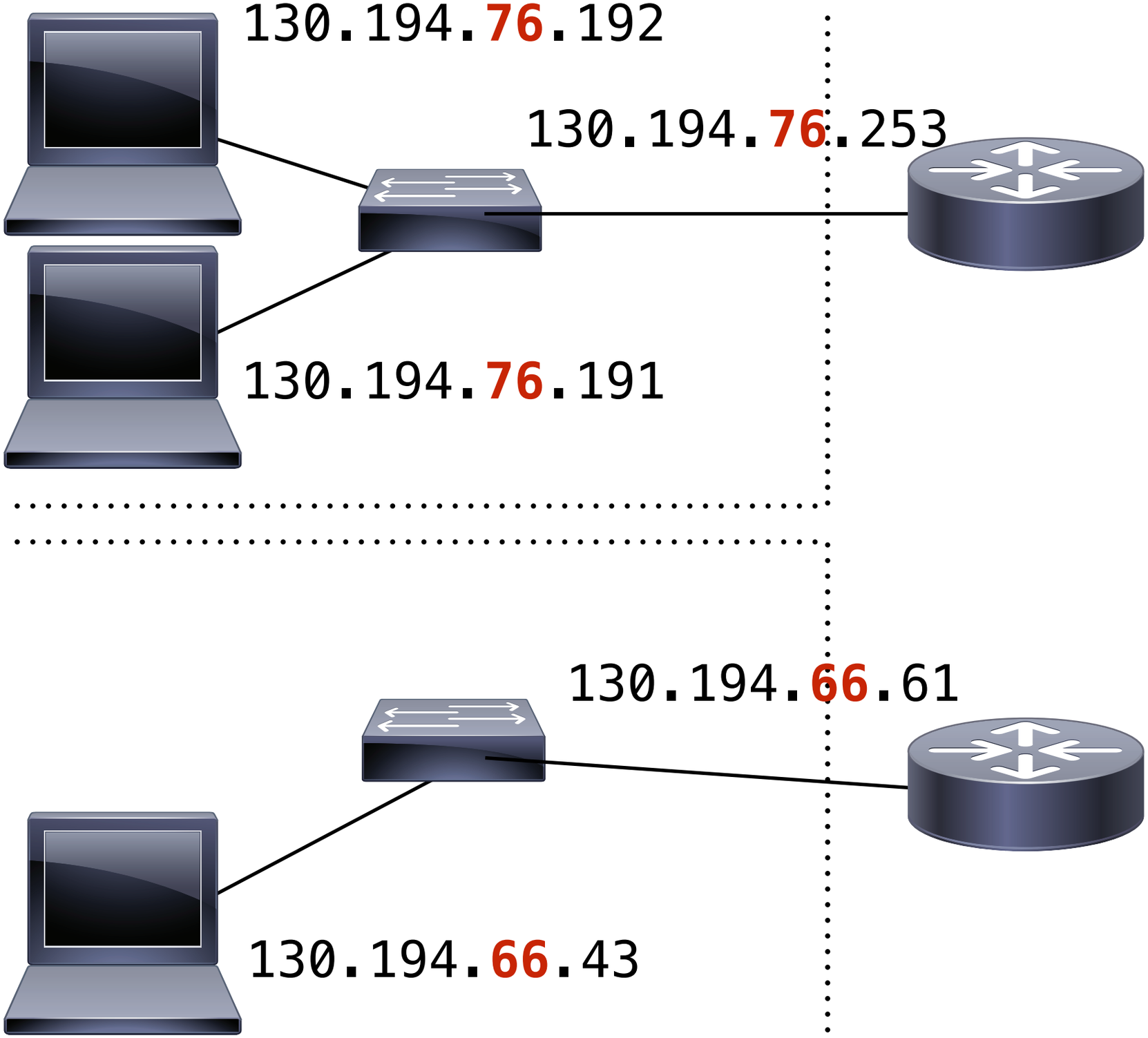




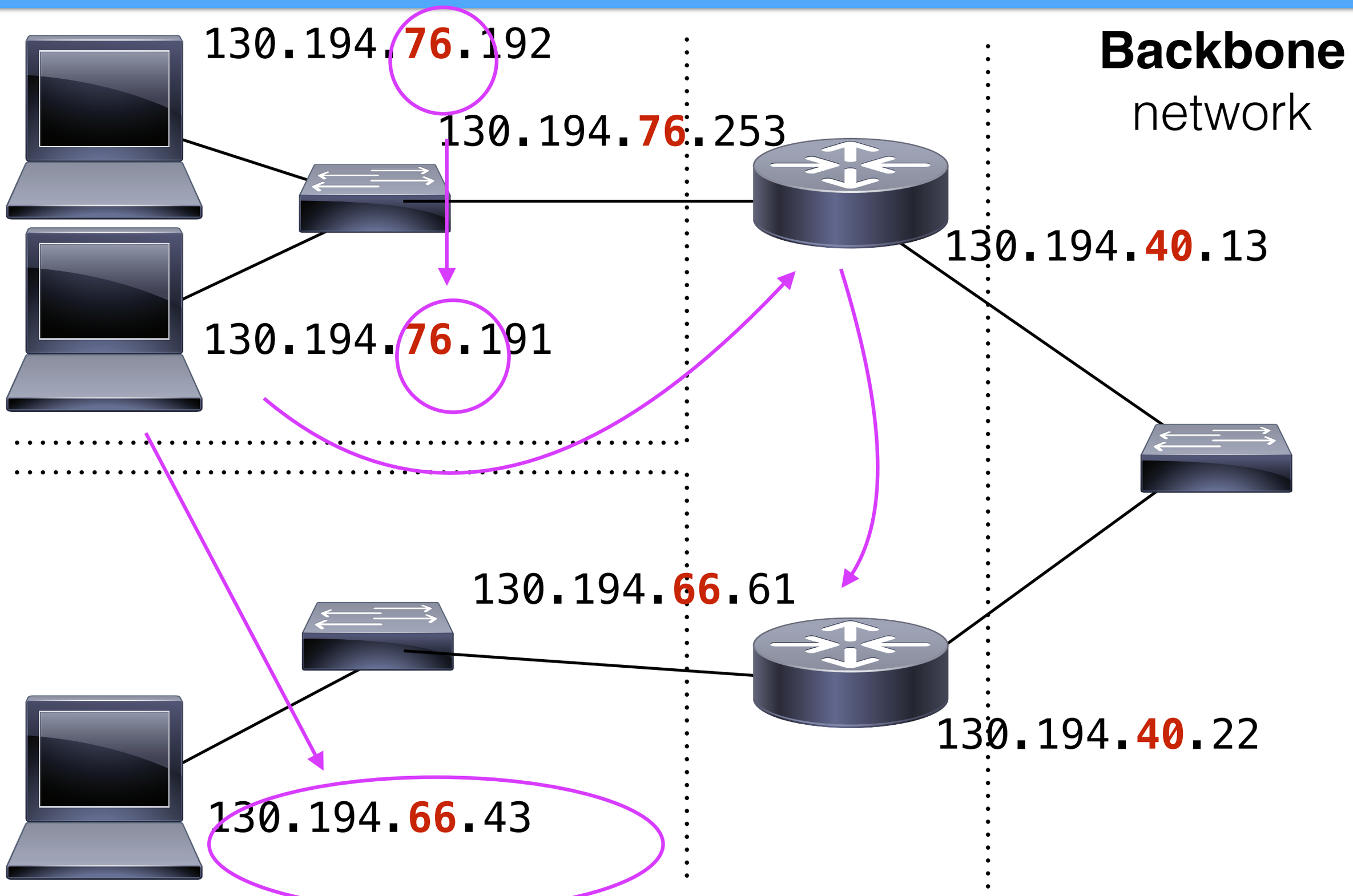
Subnets



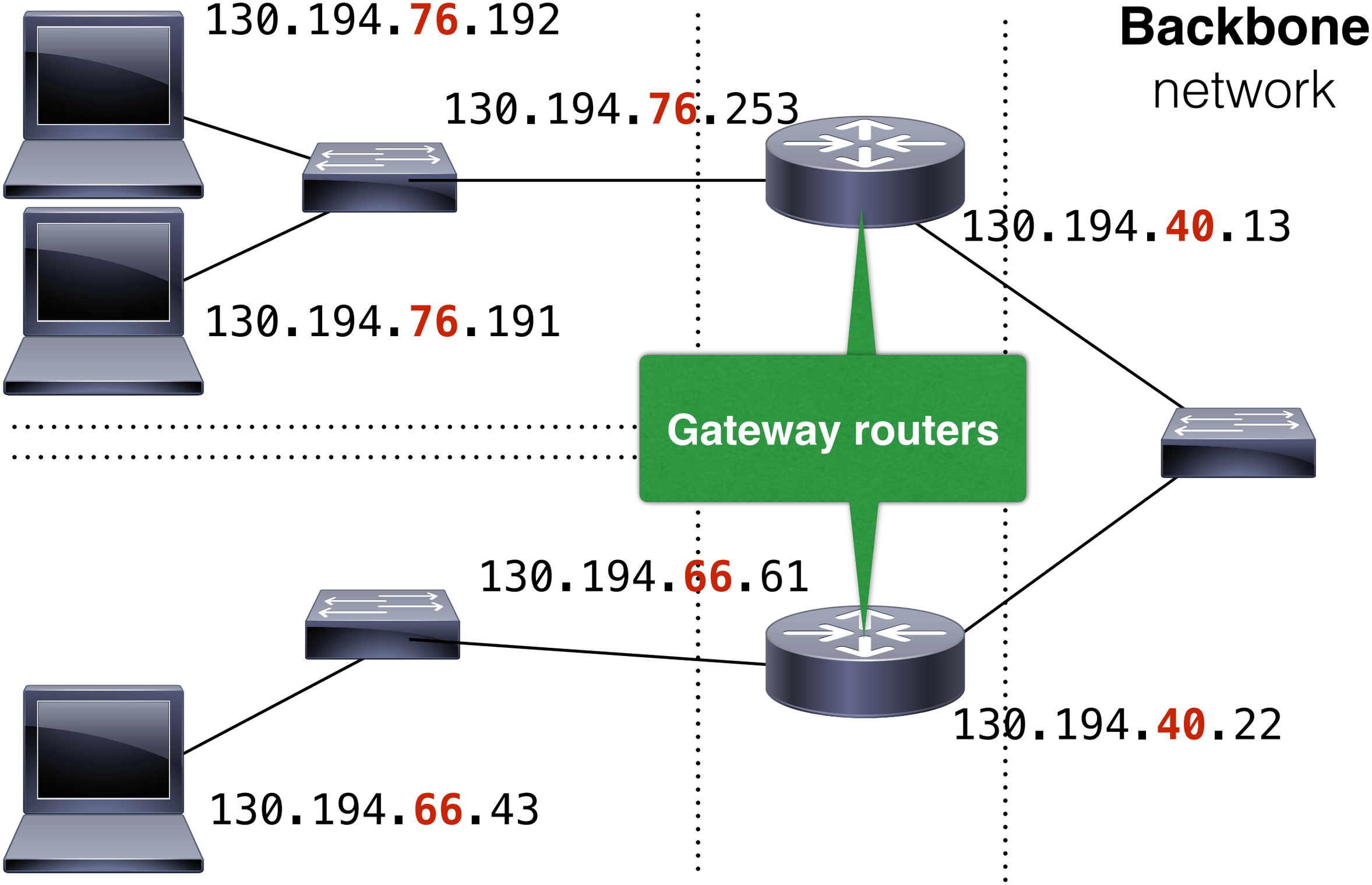
Subnets


130.194.76.192
130.194.76.253
130.194.76.191
130.194.66.61
130.194.66.43

Subnets


130.194.76.192
130.194.76.253
Backbone
network
130.194.40.13
130.194.76.191
130.194.66.61
130.194.40.22
130.194.66.43

Subnets


130.194.76.192
130.194.76.253
Backbone
network
130.194.40.13
130.194.76.191
Gateway routers
130.194.66.61
130.194.40.22
130.194.66.43

IP version 6

IPv4 addresses: 32 bits

- In theory 4,294,467,295 addresses
- In practice probably only half are usable
- We've run out of new blocks of IPv4 addresses

IP version 6

IPv4 addresses: 32 bits

- In theory 4,294,467,295 addresses
- In practice probably only half are usable
- We've run out of new blocks of IPv4 addresses

IPv6

- "new" version of the IP protocol (from 1998…)
- 128 bits per address
- so that's four times more, right?

IP version 6

$2^{128}=$

IP version 6

$2^{128}=$

340,282,366,920,938,463,463,374,607,431,768,211,456

(340 undecillion)

IPv6 addresses

IP version 6

340,282,366,920,938,463,463,374,607,431,768,211,456

A bit excessive?

- At least 7 addresses for every atom of every person on earth
- 665,570,793,348,866,943,898,599 addresses per square meter of the surface of the earth

IP version 6

340,282,366,920,938,463,463,374,607,431,768,211,456

A bit excessive?

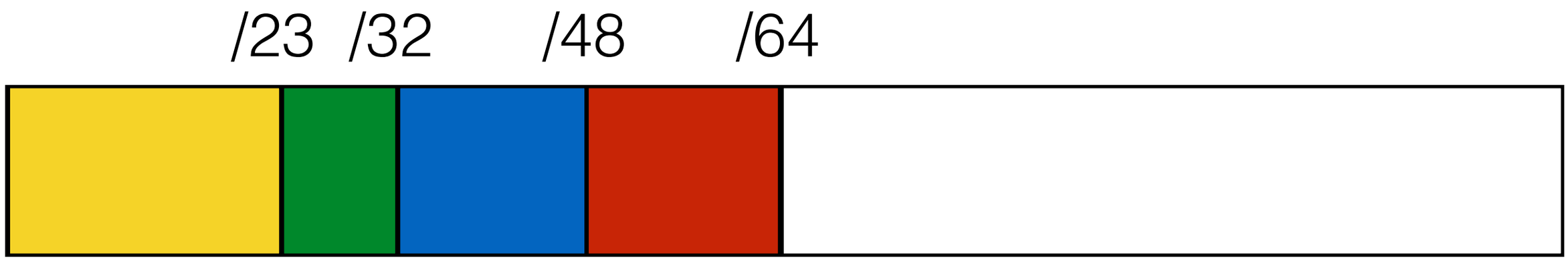
- At least 7 addresses for every atom of every person on earth
- 665,570,793,348,866,943,898,599 addresses per square meter of the surface of the earth

Required!

- The huge space is used to create **hierarchies**
- This makes it easy to assign whole subnets

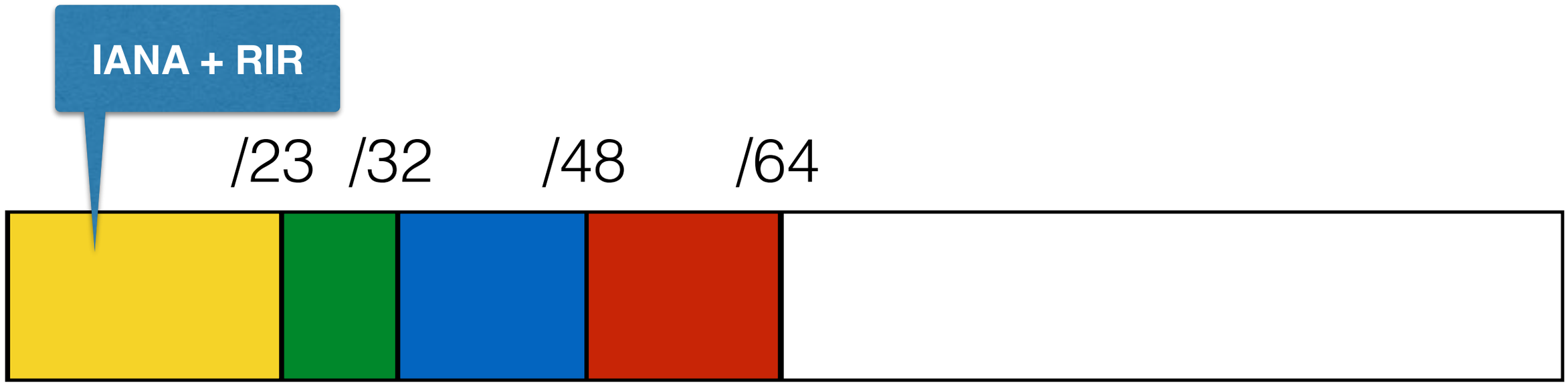
IP version 6 address space

Typical allocation:



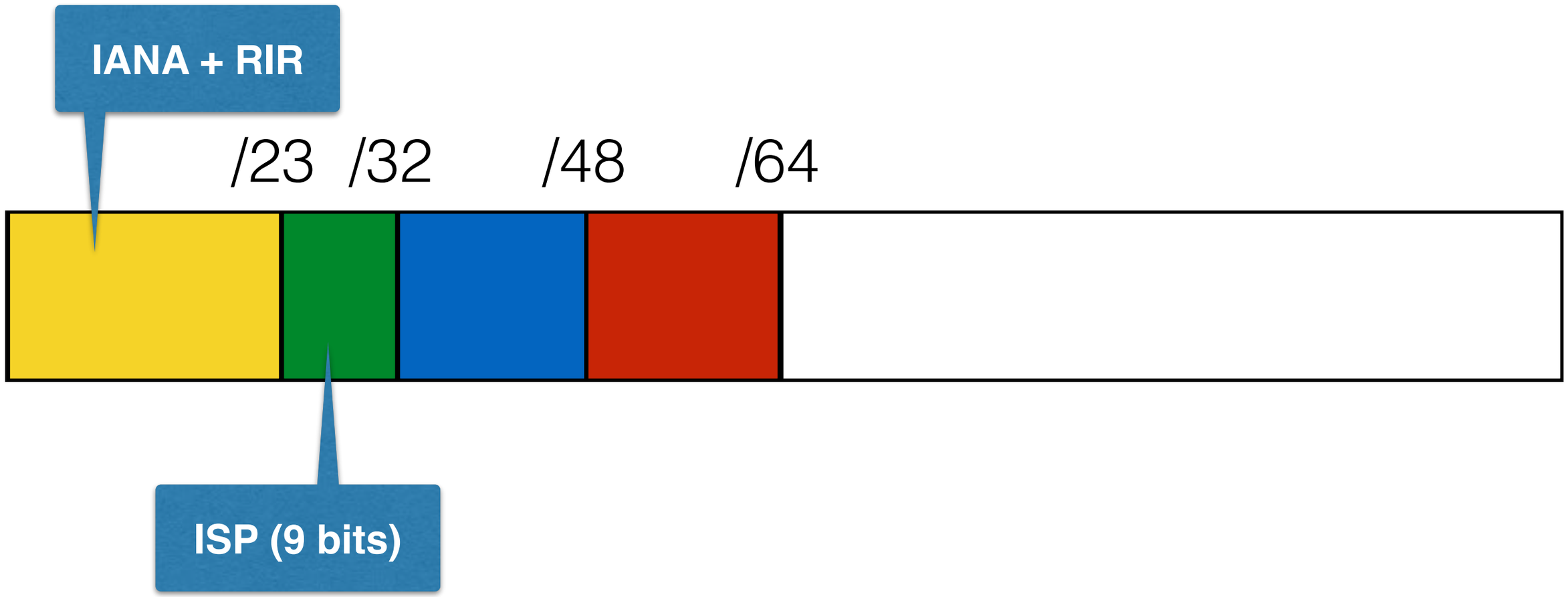
IP version 6 address space

Typical allocation:



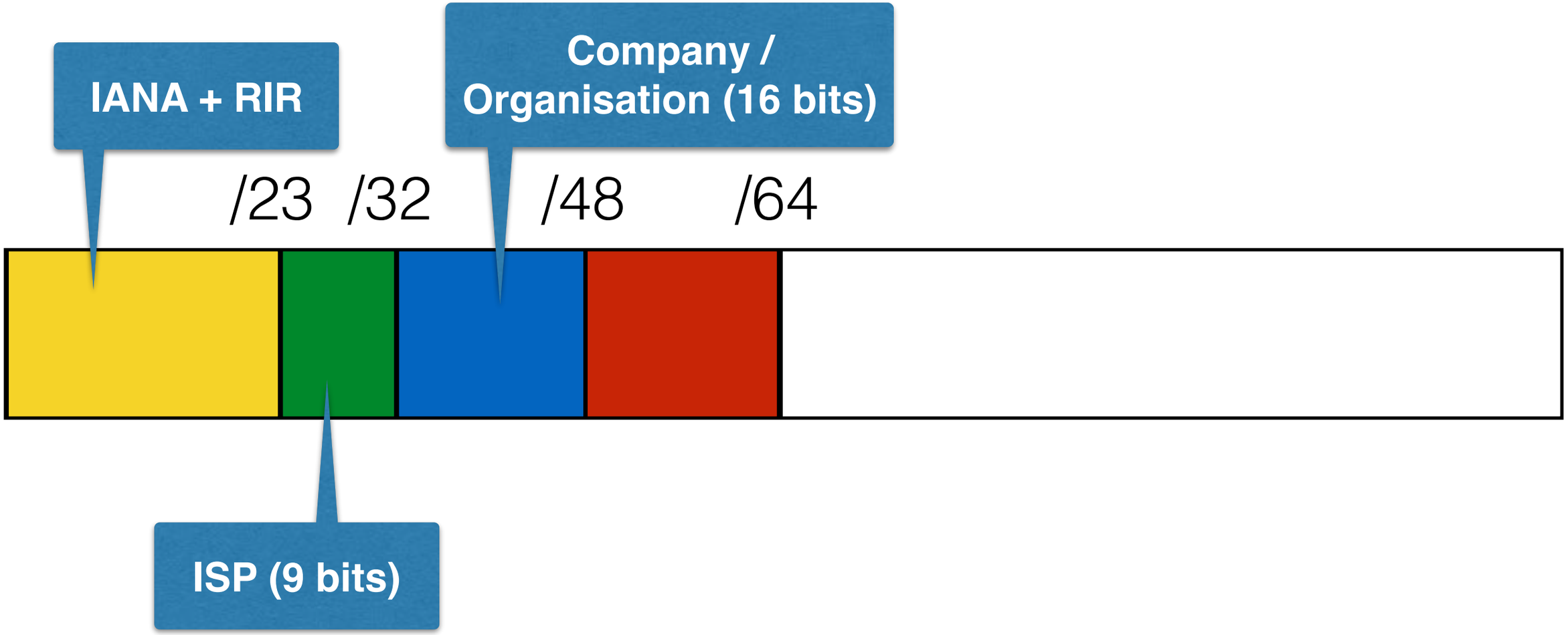
IP version 6 address space

Typical allocation:



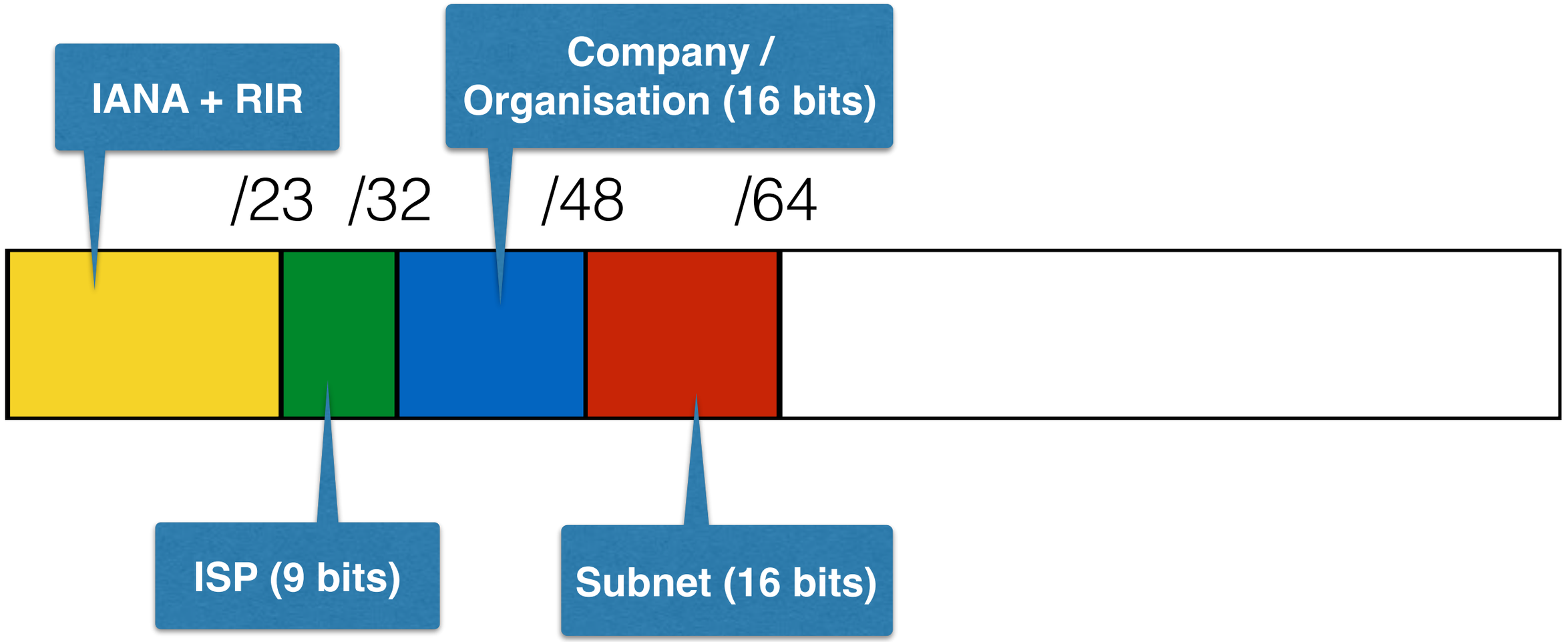
IP version 6 address space

Typical allocation:



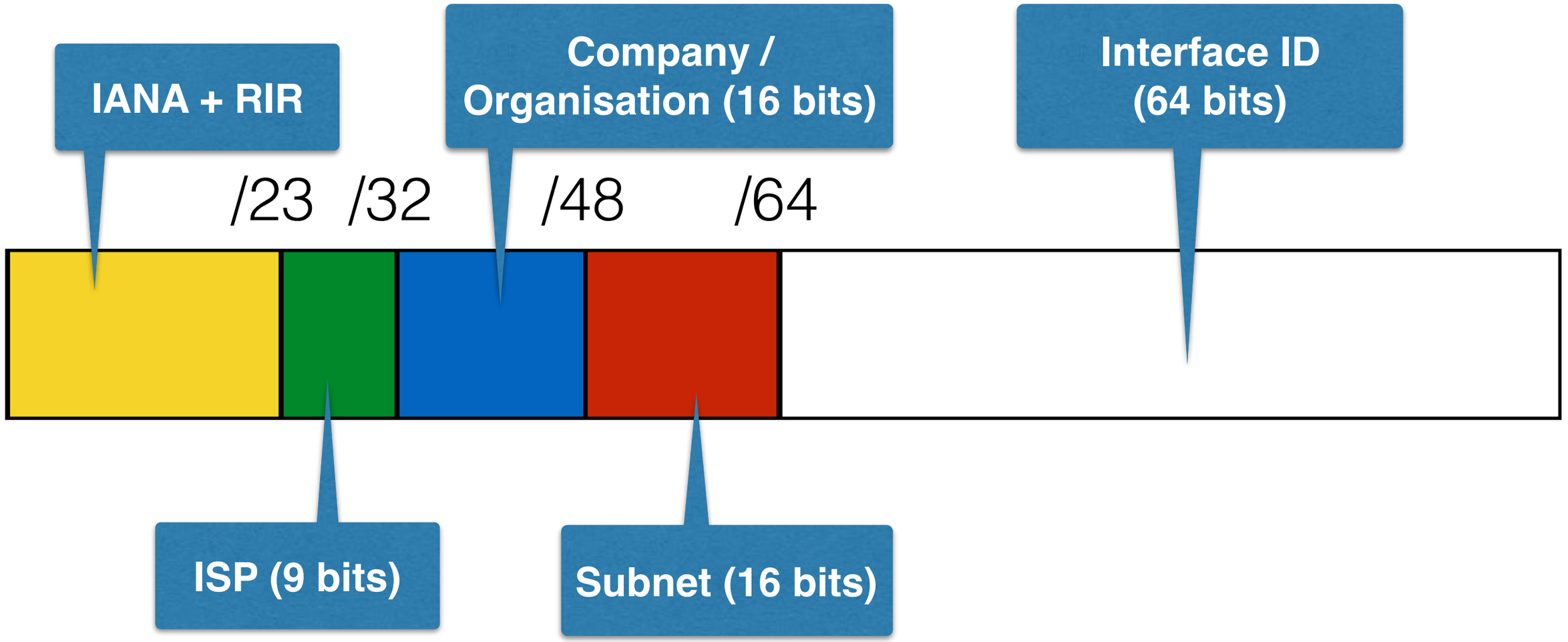
IP version 6 address space

Typical allocation:



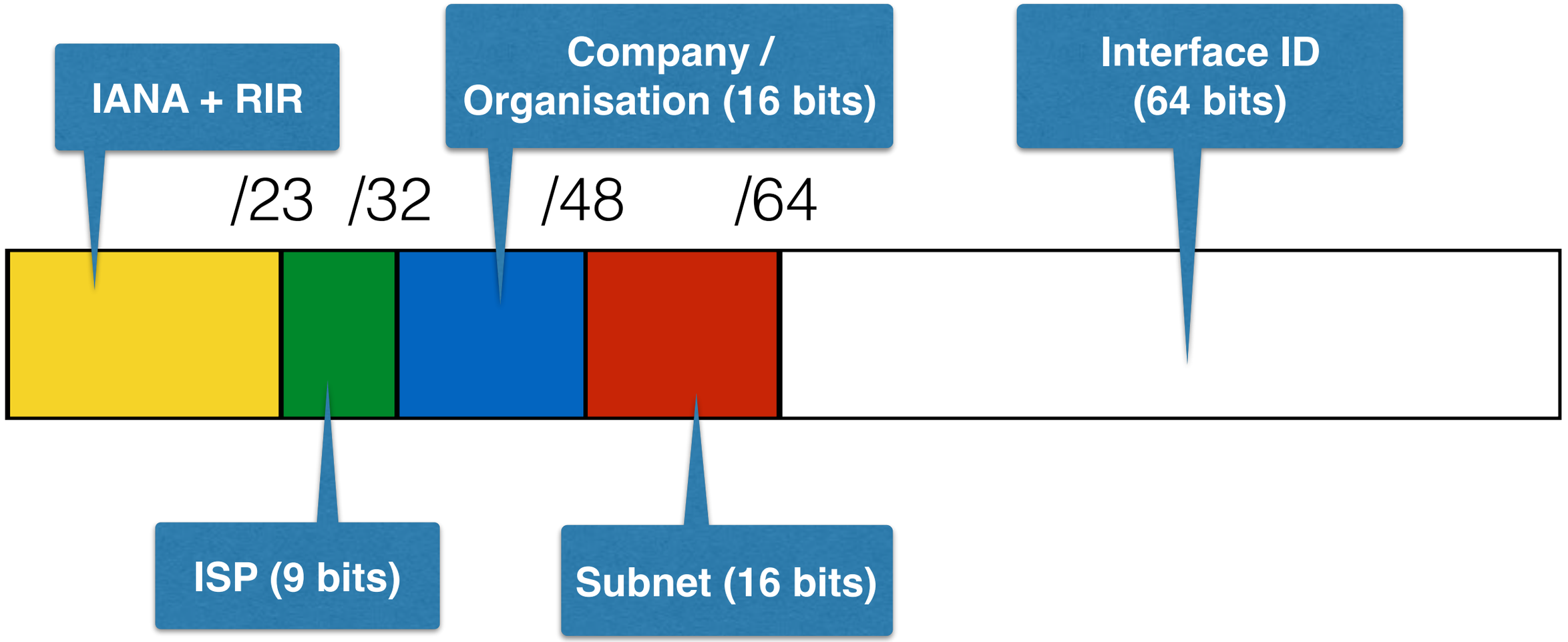
IP version 6 address space

Typical allocation:



IP version 6 address space

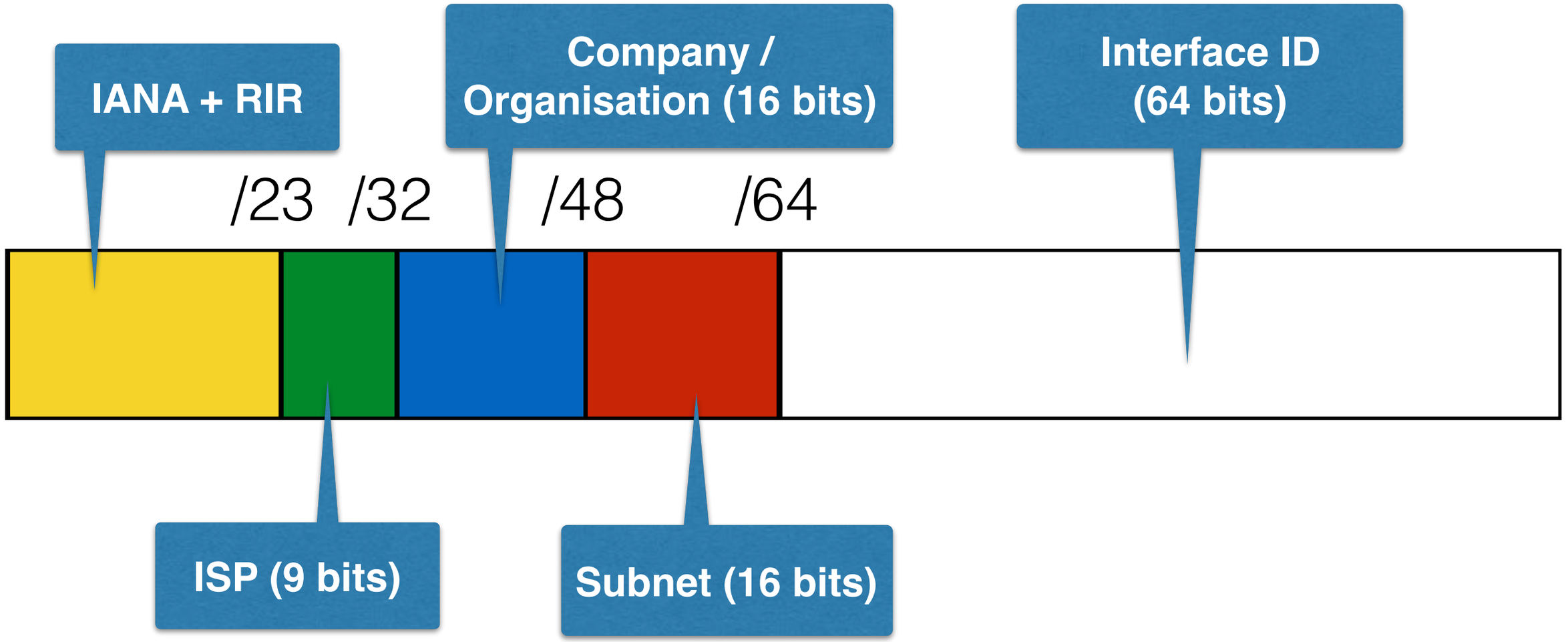
Typical allocation:



2^{32} times more addresses than IPv4 in Interface ID alone!

IP version 6 address space

Typical allocation:

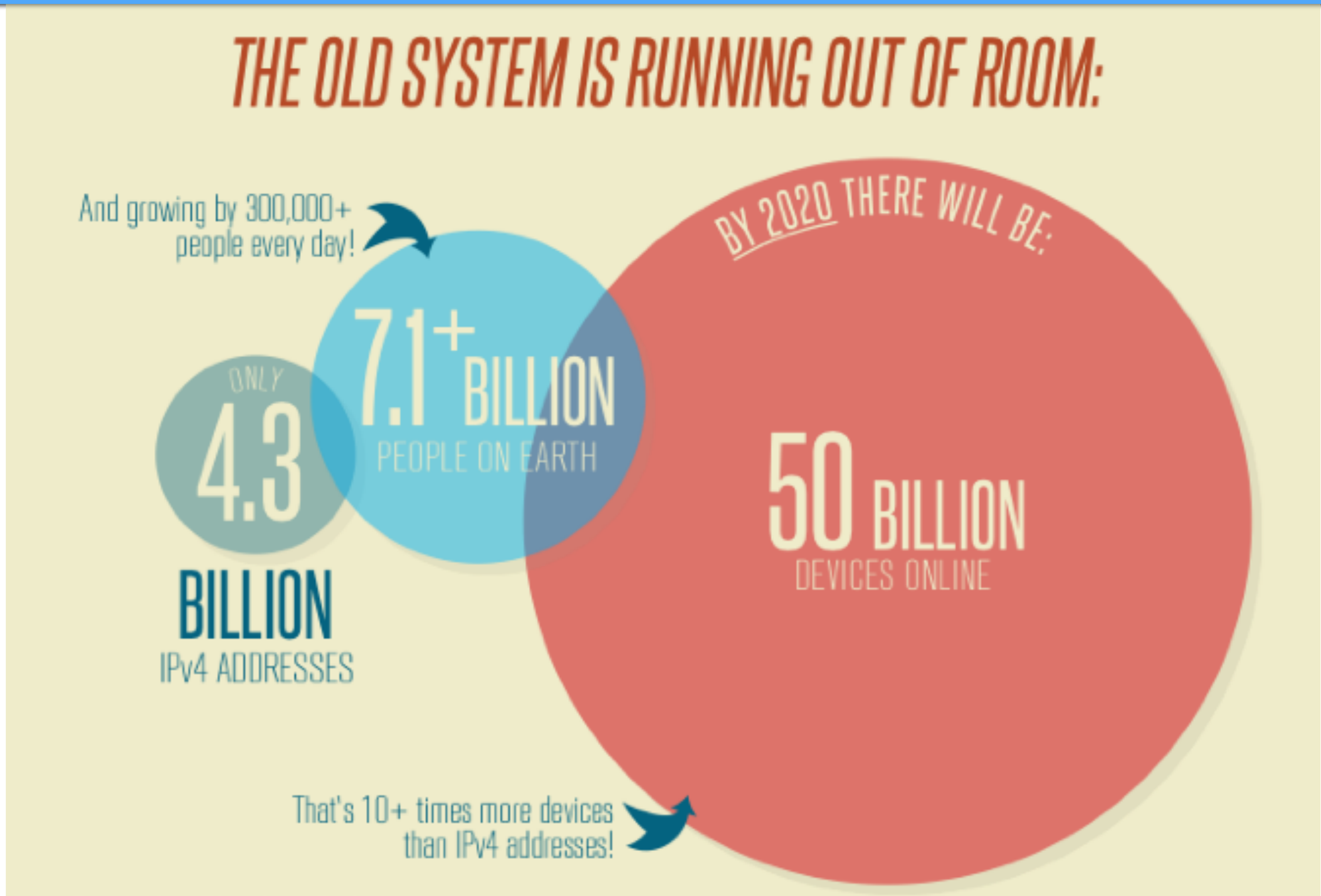


2^{32} times more addresses than IPv4 in Interface ID alone!

Every company can run more than 65,000 different LANs

IPv6 transition

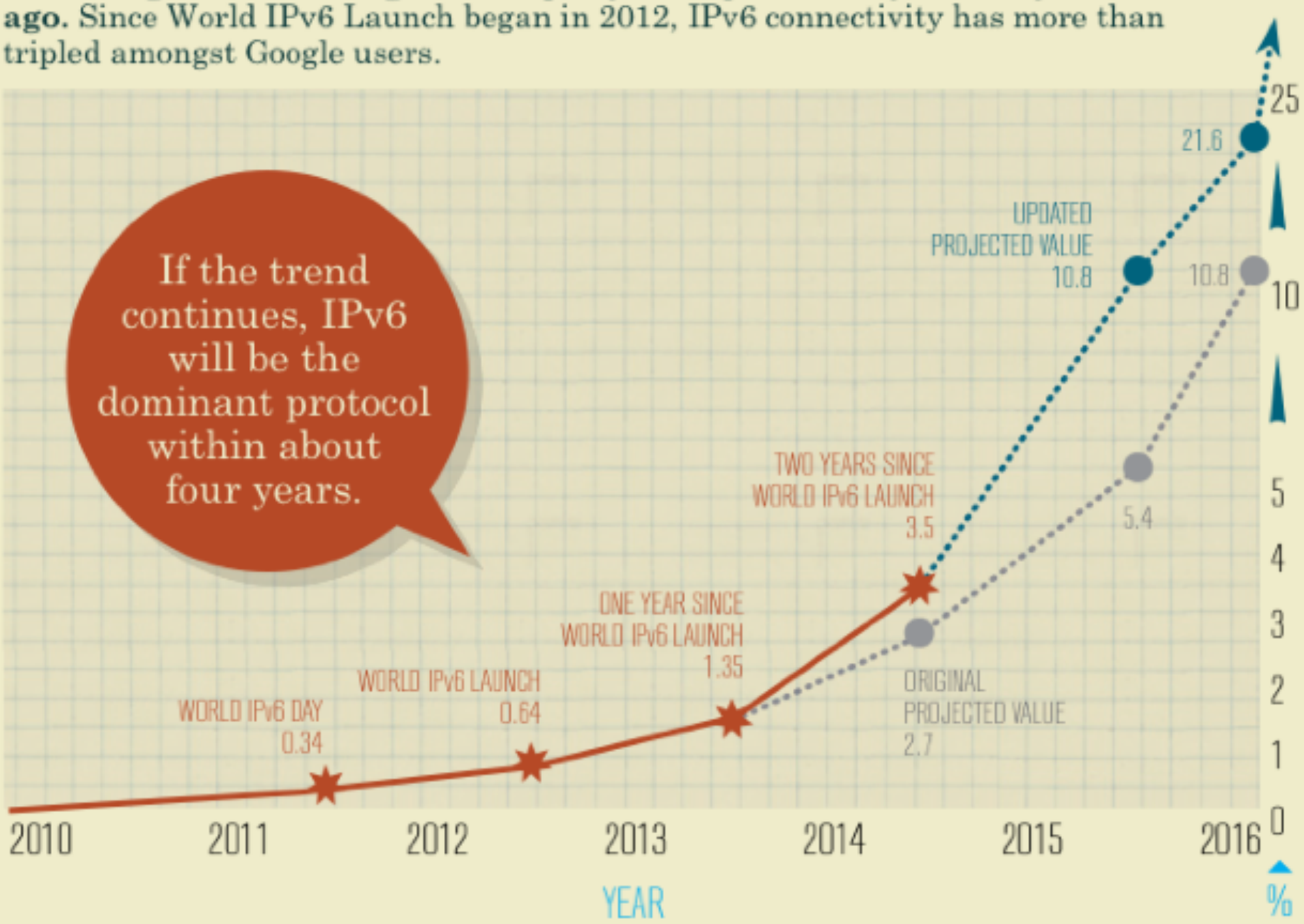
IPv6 transition



IPv6 transition

IPv6 MOMENTUM

IPv6 usage is increasing more rapidly than predicted just two years ago. Since World IPv6 Launch began in 2012, IPv6 connectivity has more than tripled amongst Google users.



Address resolution

Assume we browse to http://www.google.com.au

Address resolution

Assume we browse to http://www.google.com.au

- We have to translate www.google.com.au into an IP address: 216.58.220.99

Address resolution

Assume we browse to http://www.google.com.au

- We have to translate www.google.com.au into an IP address: 216.58.220.99
- We send a request through the Internet to that IP address

Address resolution

Assume we browse to http://www.google.com.au

- We have to translate www.google.com.au into an IP address: 216.58.220.99
- We send a request through the Internet to that IP address
- The router in the LAN of 216.58.220.99 needs to know the MAC address for 216.58.220.99 to deliver the frame

Address resolution

Assume we browse to http://www.google.com.au

- We have to translate www.google.com.au into an IP address: 216.58.220.99
- We send a request through the Internet to that IP address
- The router in the LAN of 216.58.220.99 needs to know the MAC address for 216.58.220.99 to deliver the frame

This is known as address resolution.

Address resolution: Application Layer

DNS (Domain Name System)

- Application layer protocol for address resolution
- Client sends request to DNS server to get IP address registered for a name

Address resolution: Application Layer

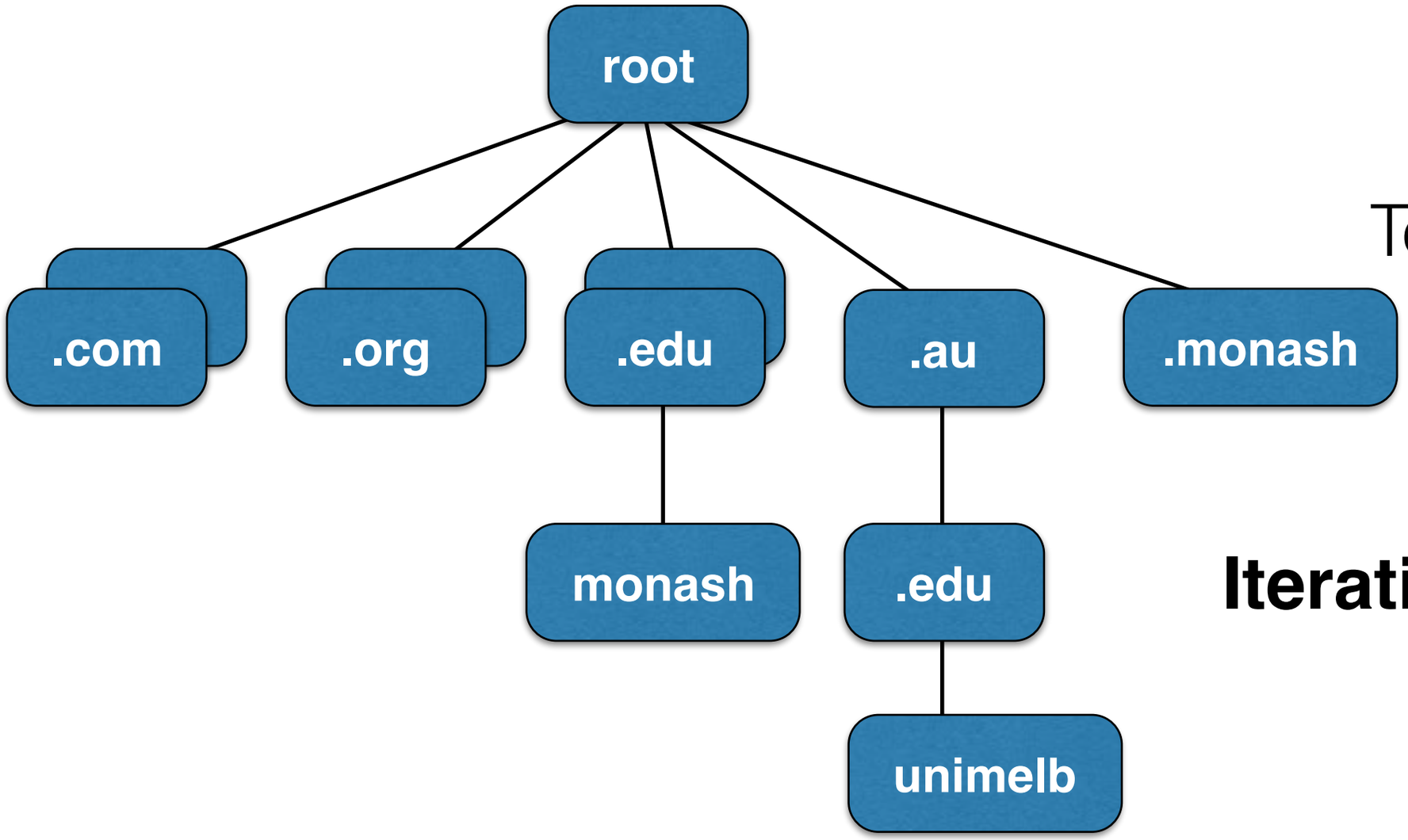
DNS (Domain Name System)

- Application layer protocol for address resolution
- Client sends request to DNS server to get IP address registered for a name

DNS Servers

- Implement a **distributed database** of names
- Are organised in a **hierarchy** reflecting the **structure** of the domain names

DNS

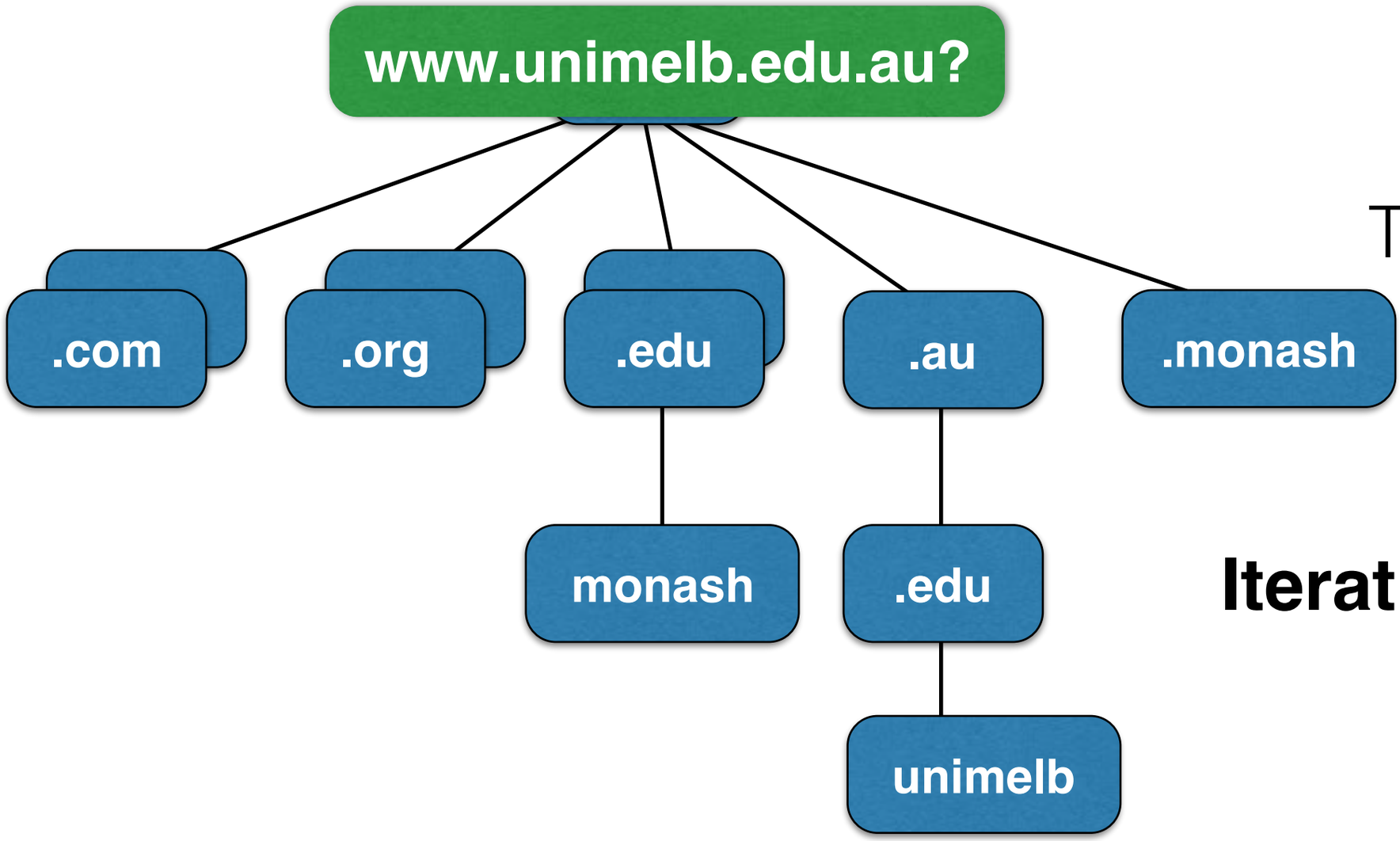


Root Zone

Top-Level Domain (TLD)

Iterative DNS lookup

DNS

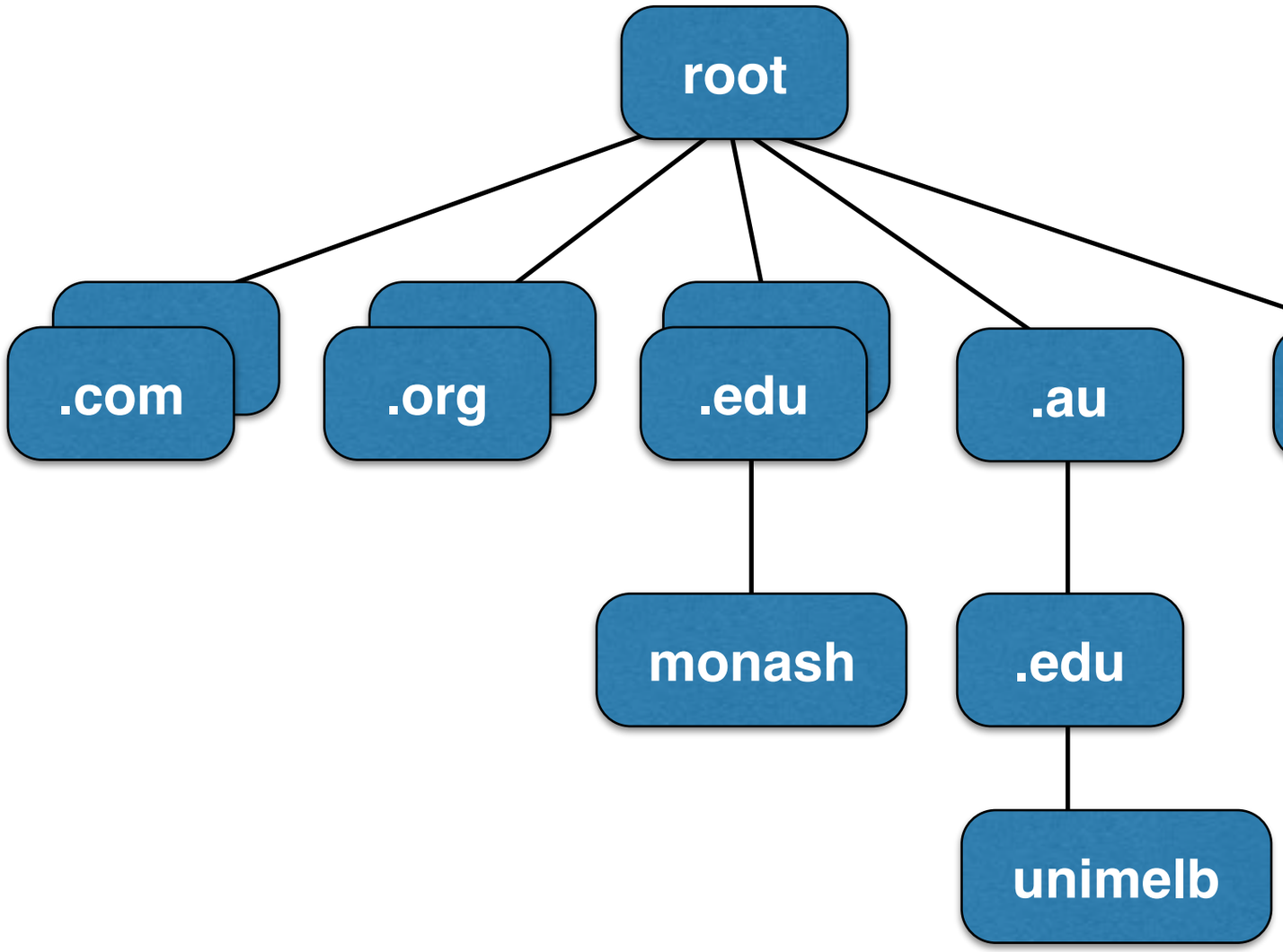


Root Zone

Top-Level Domain (TLD)

Iterative DNS lookup

DNS

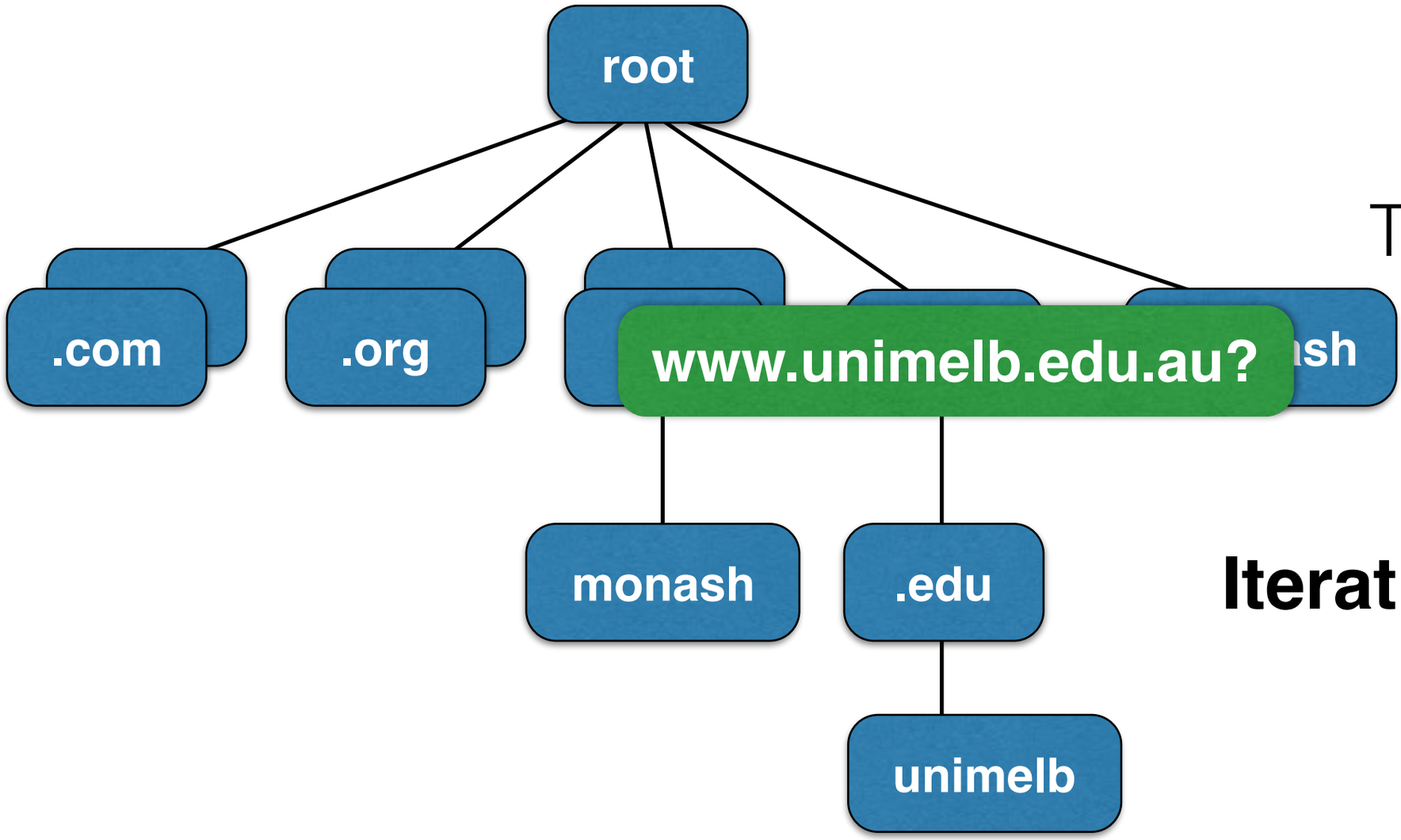


Root Zone

Top-Level Domain (TLD)

Iterative DNS lookup

DNS

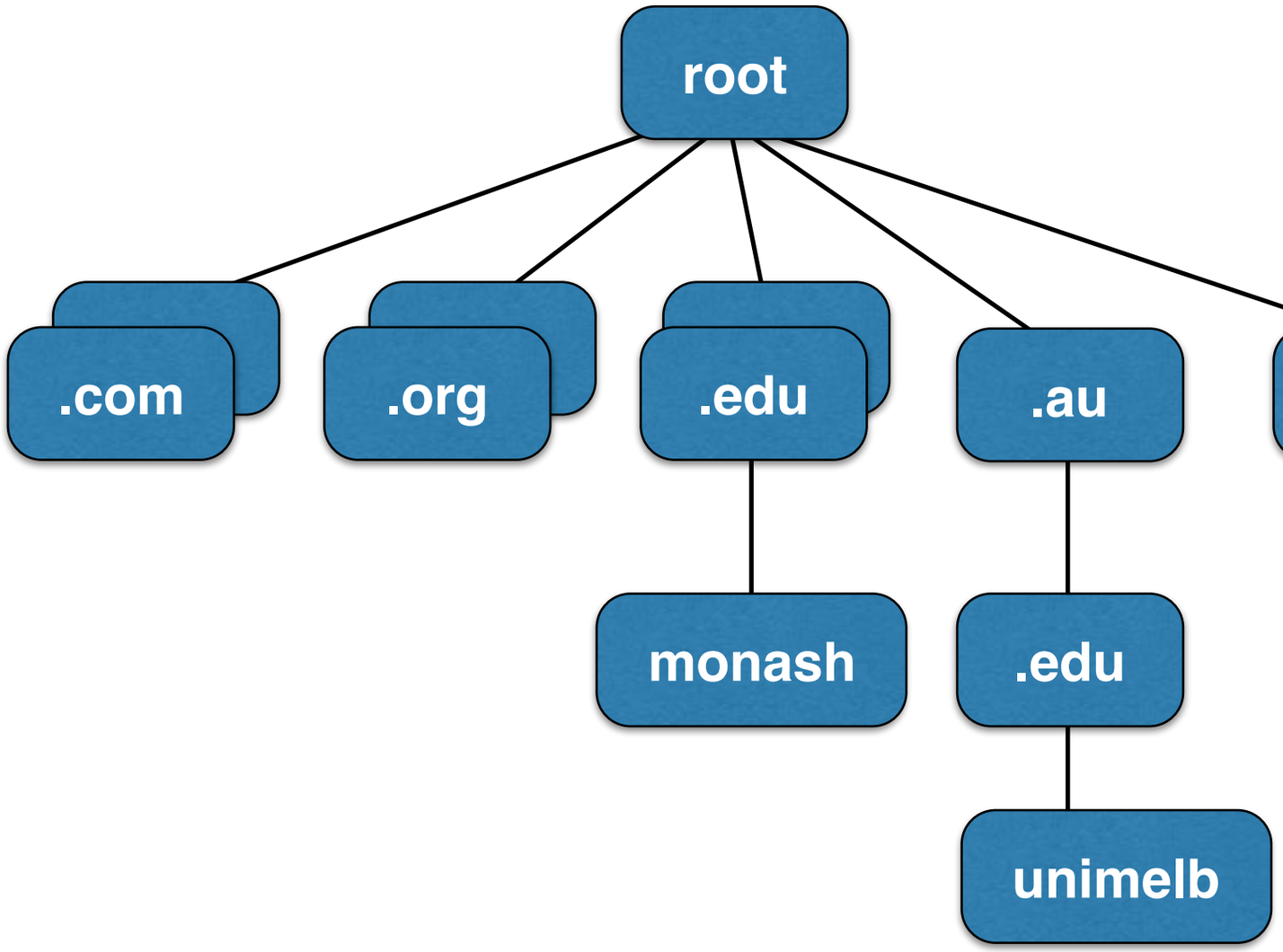


Root Zone

Top-Level Domain (TLD)

Iterative DNS lookup

DNS

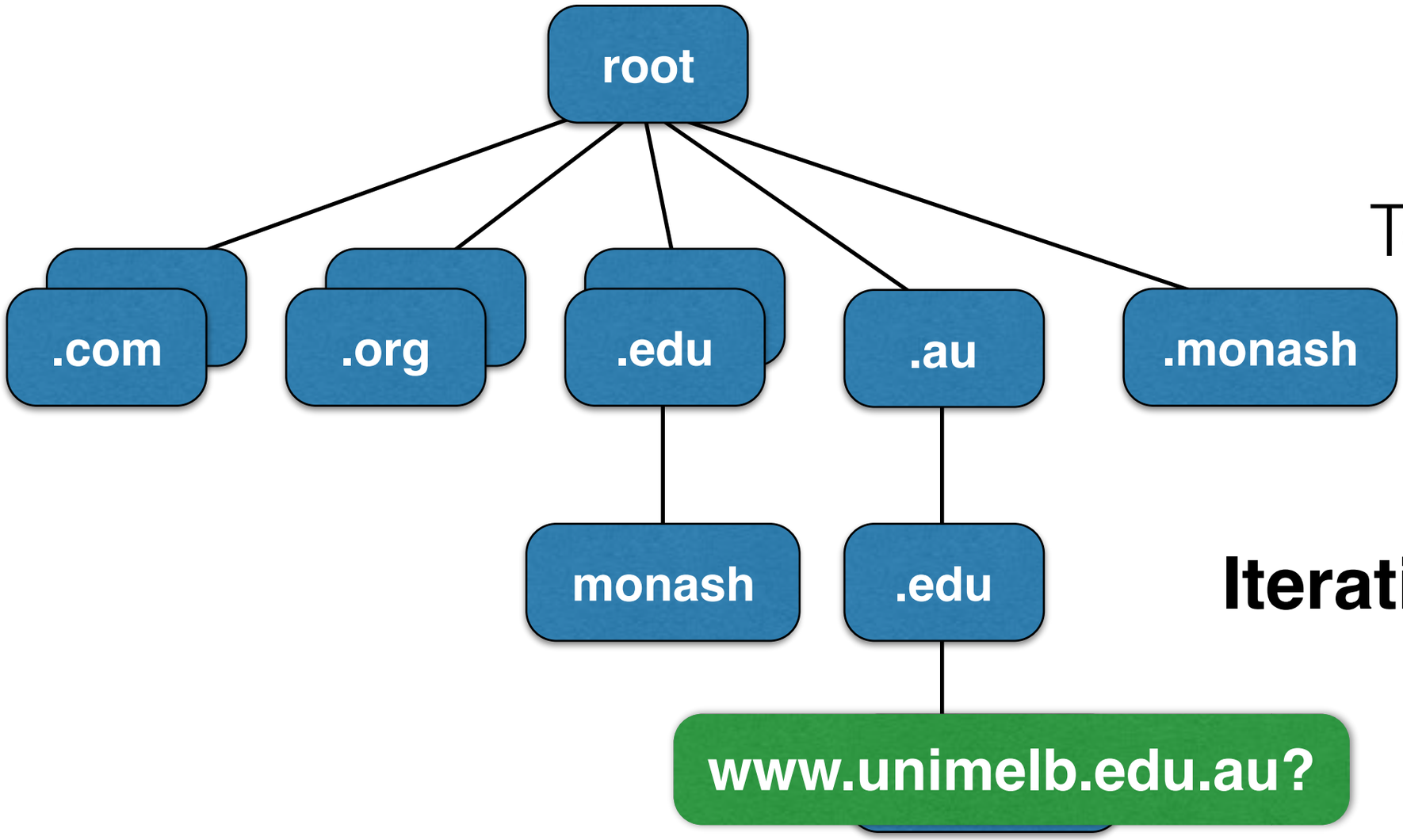


Root Zone

Top-Level Domain (TLD)

Iterative DNS lookup

DNS

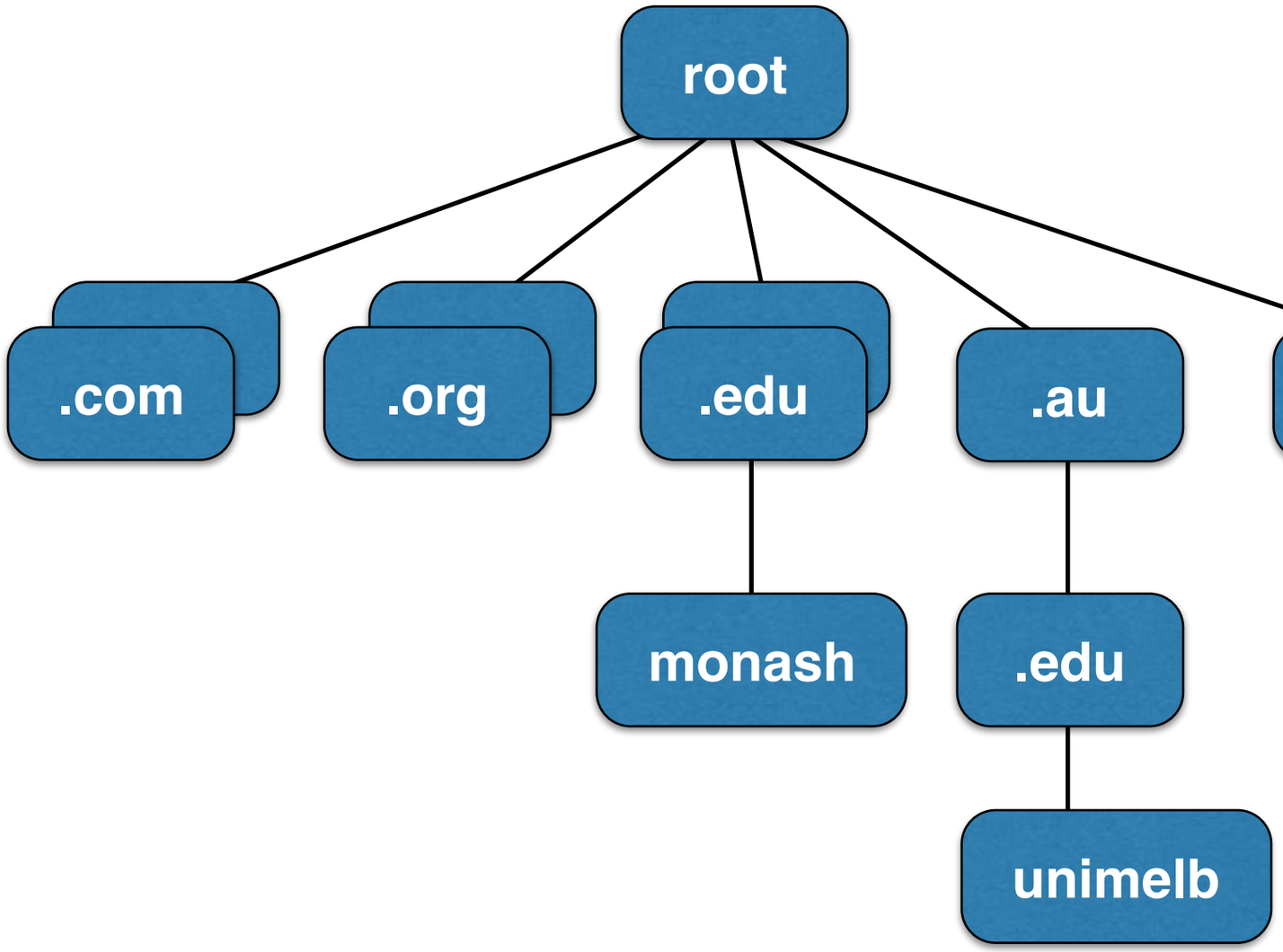


Root Zone

Top-Level Domain (TLD)

Iterative DNS lookup

DNS

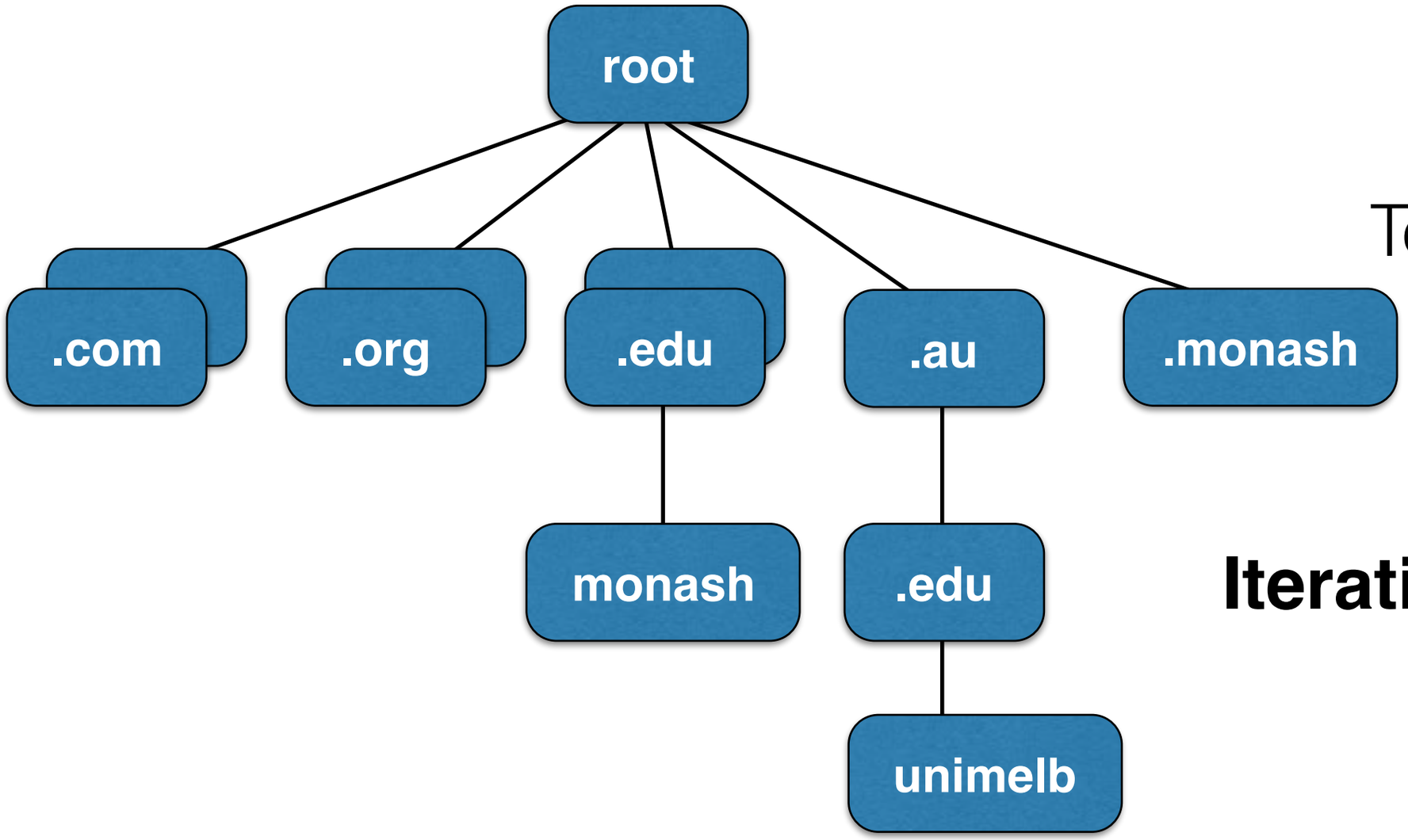


Root Zone

Top-Level Domain (TLD)

Iterative DNS lookup

DNS

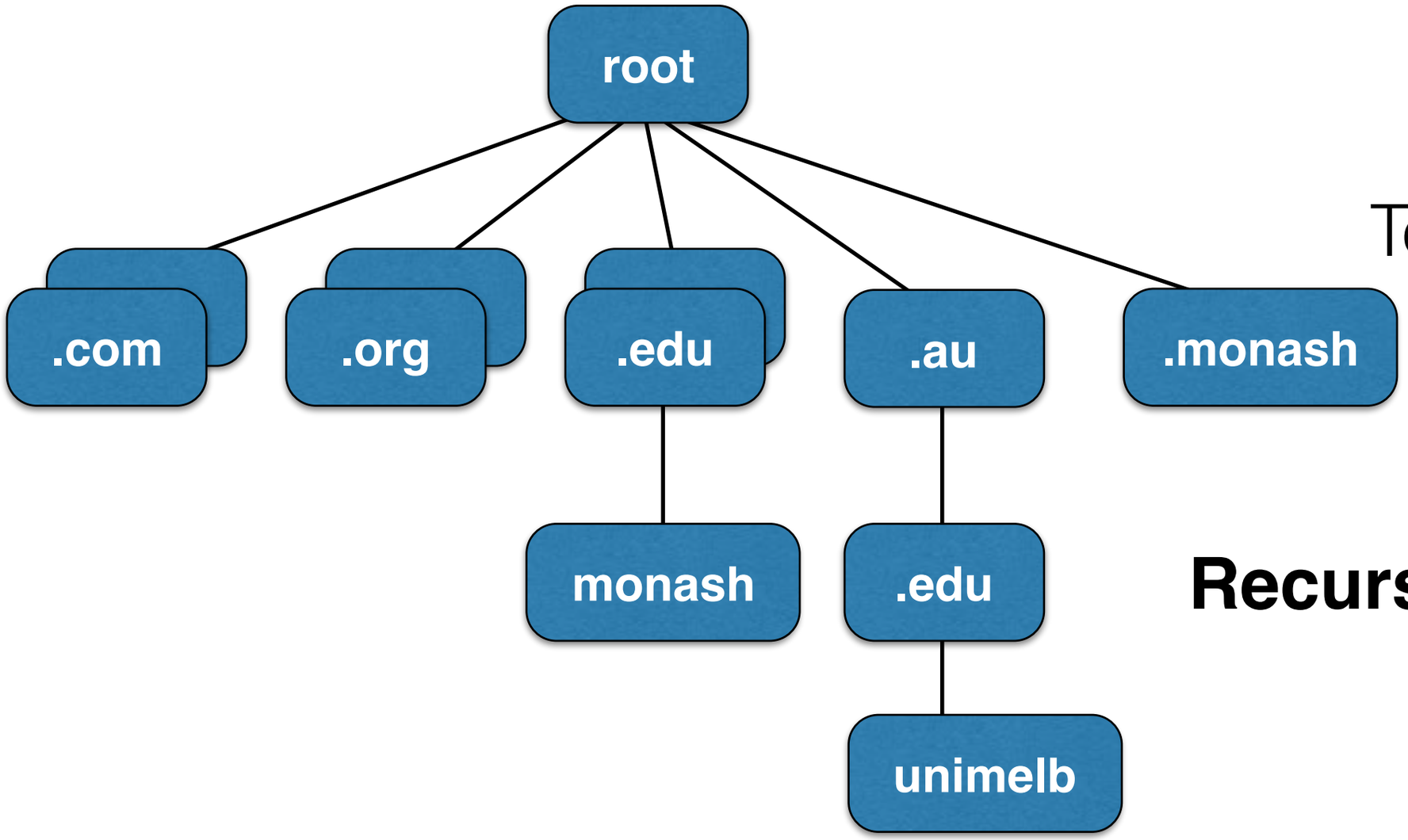


Root Zone

Top-Level Domain (TLD)

Iterative DNS lookup

DNS

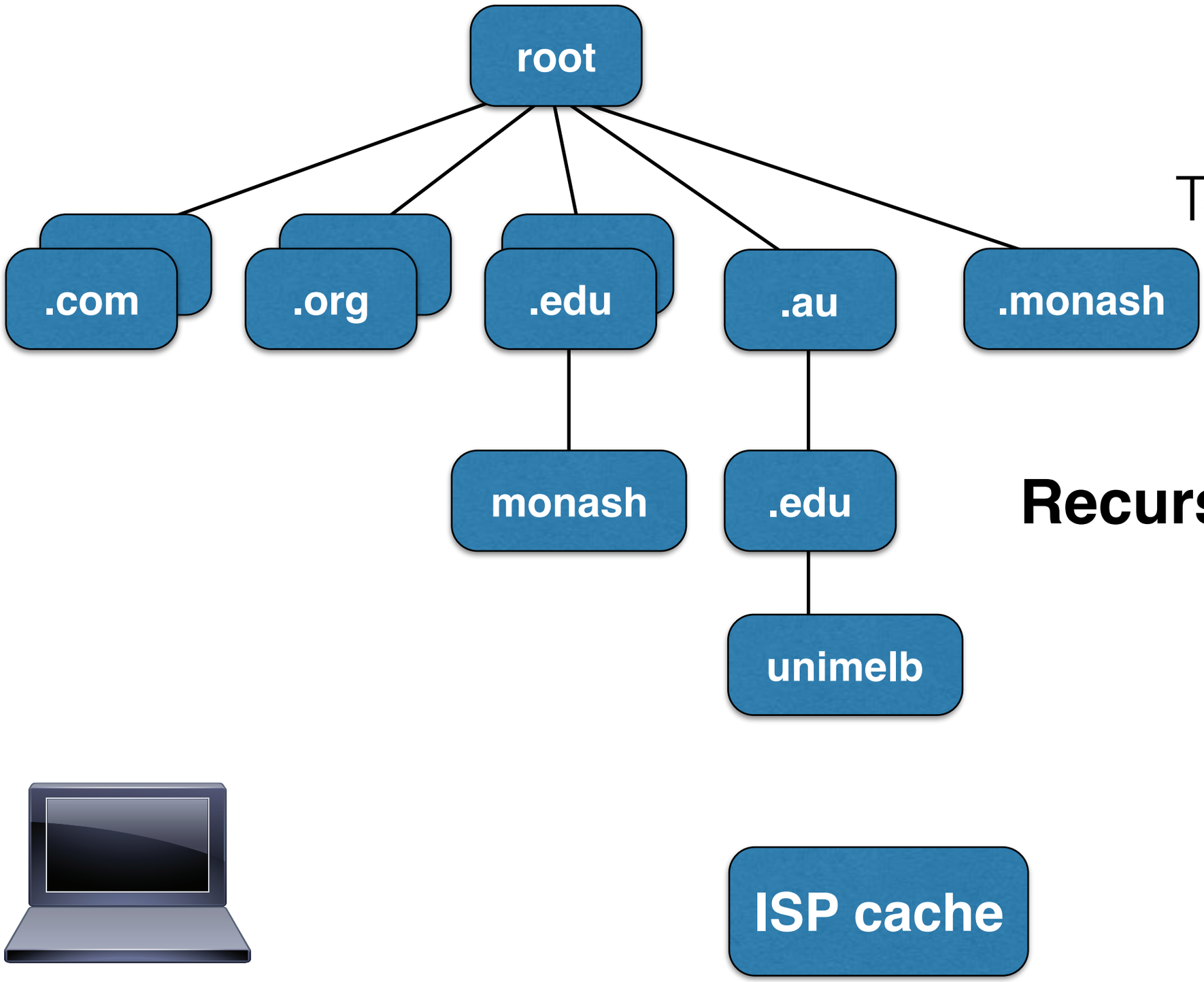


Root Zone

Top-Level Domain (TLD)

Recursive DNS lookup

DNS



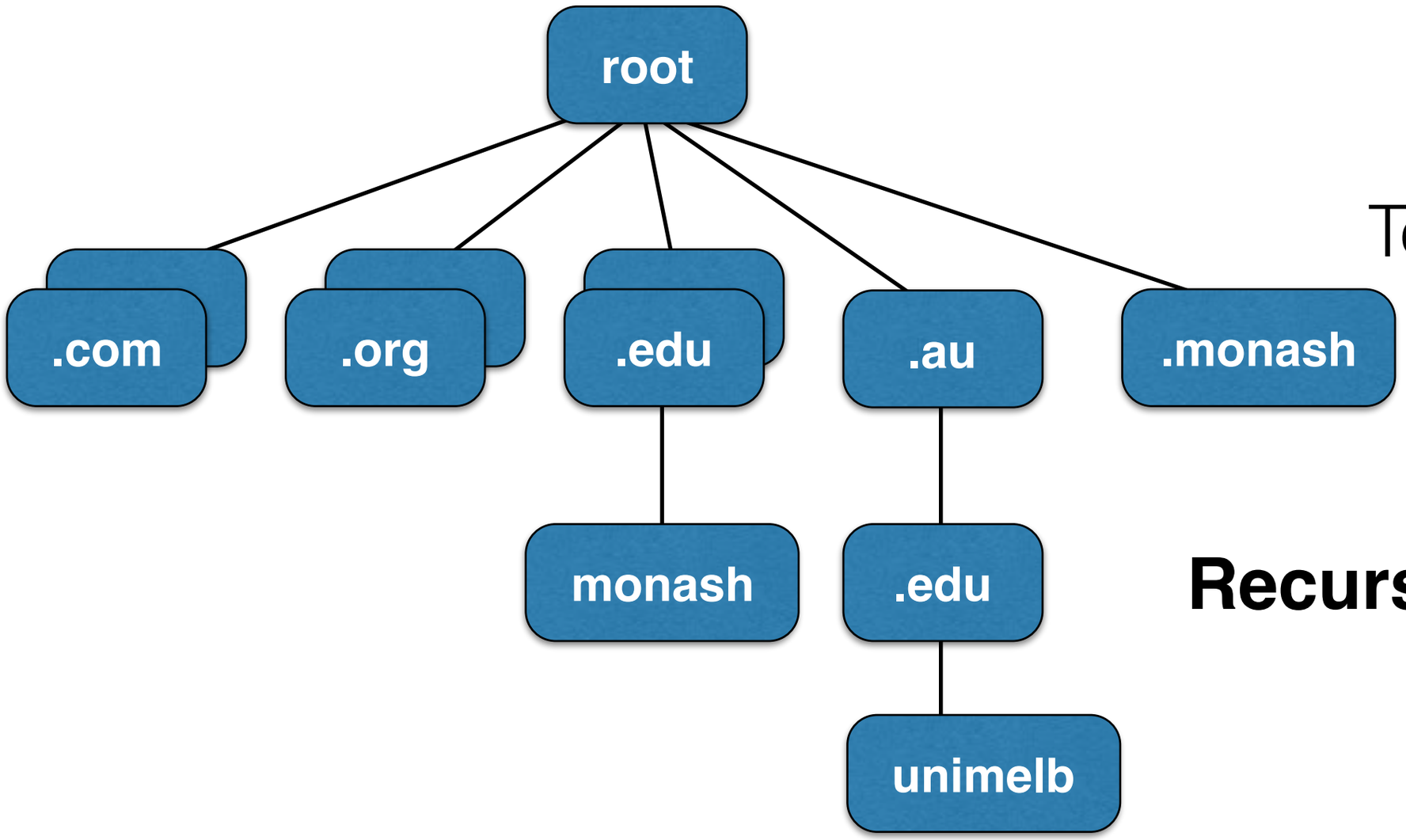
Root Zone

Top-Level Domain (TLD)

Recursive DNS lookup

ISP cache

DNS



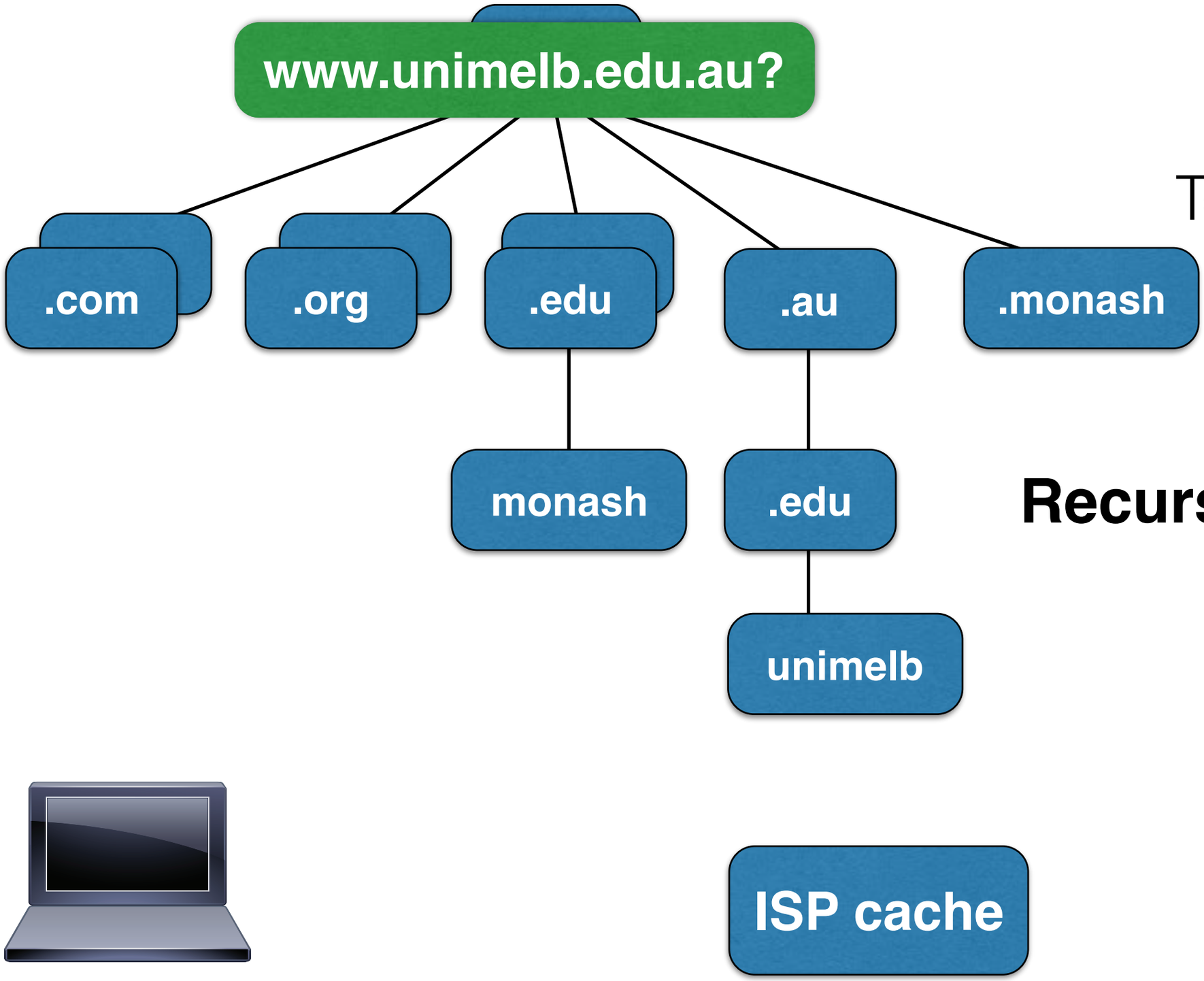
Root Zone

Top-Level Domain (TLD)

Recursive DNS lookup

www.unimelb.edu.au?

DNS

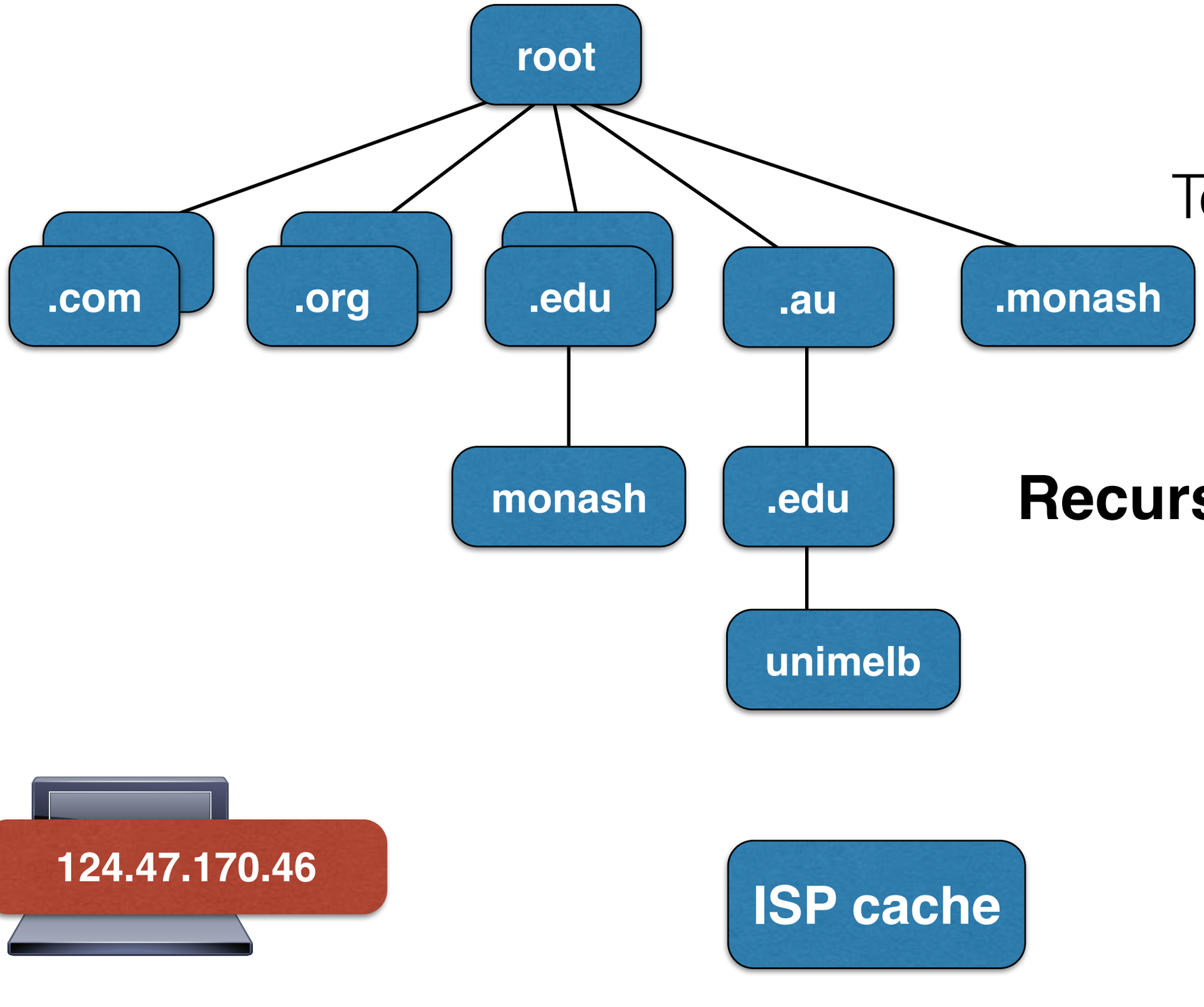


Root Zone

Top-Level Domain (TLD)

Recursive DNS lookup

DNS



Root Zone

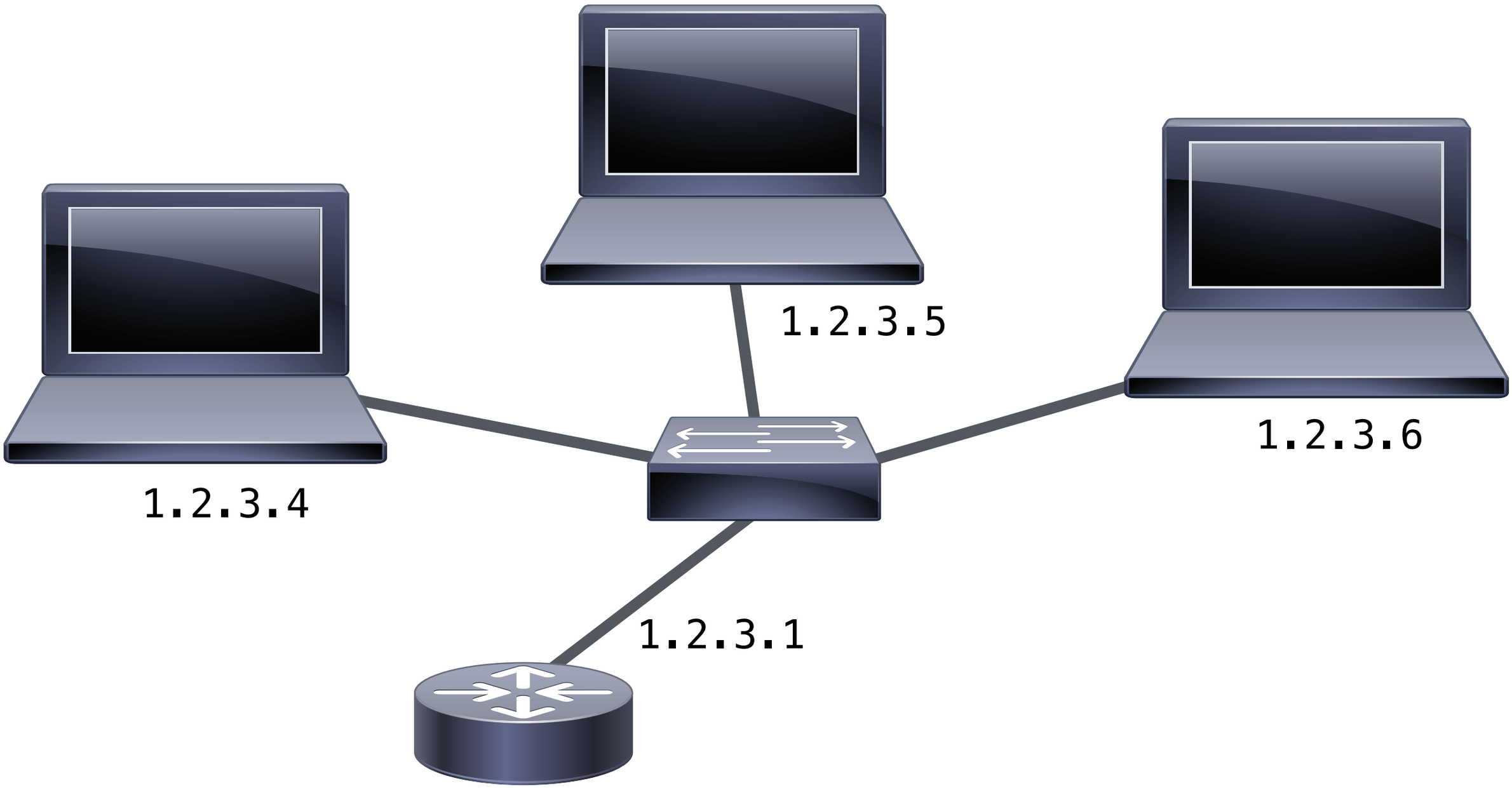
Top-Level Domain (TLD)

Recursive DNS lookup

ISP cache

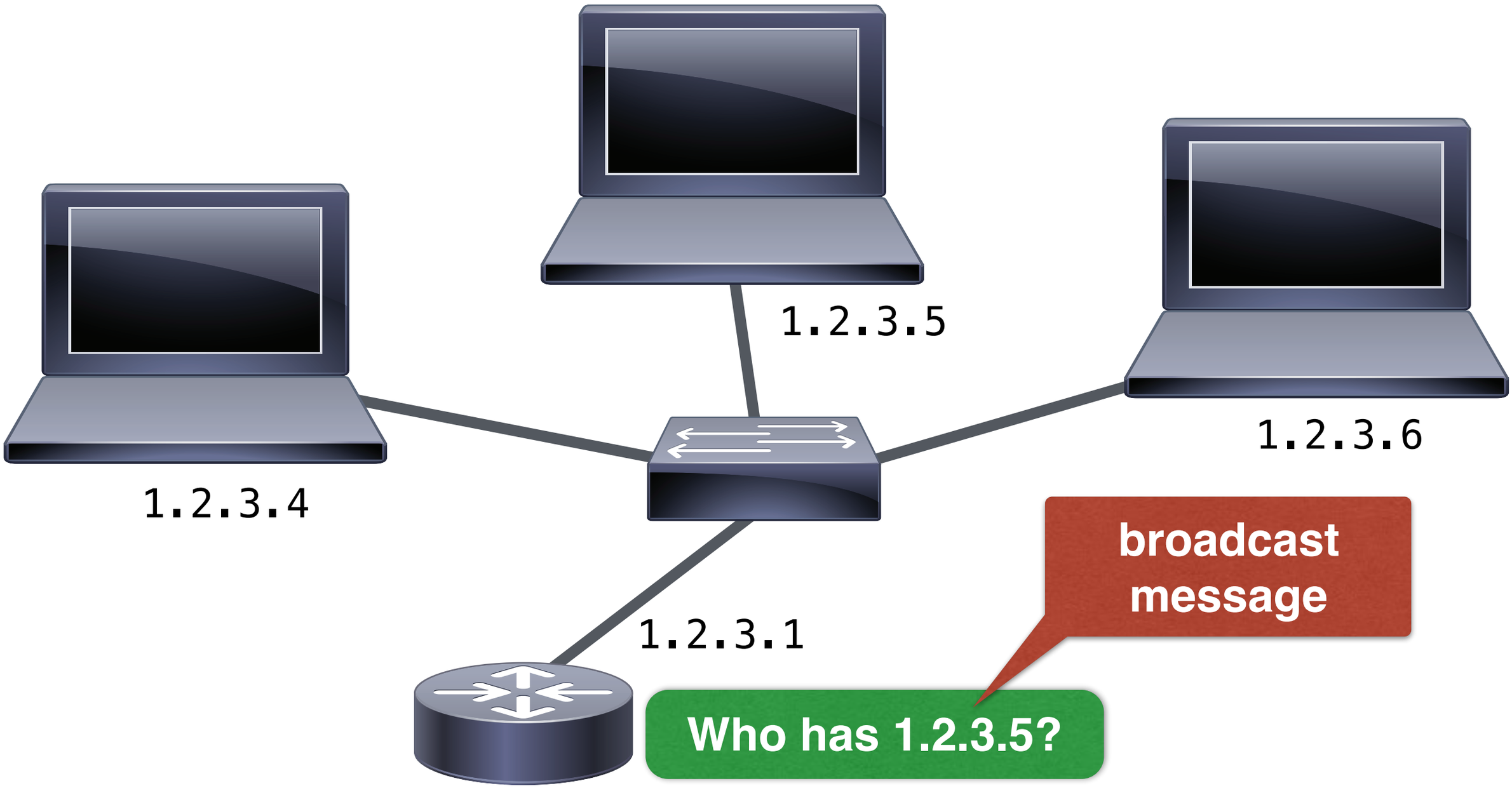
Address resolution: Data Link Layer

How to find the MAC address for an IP address:



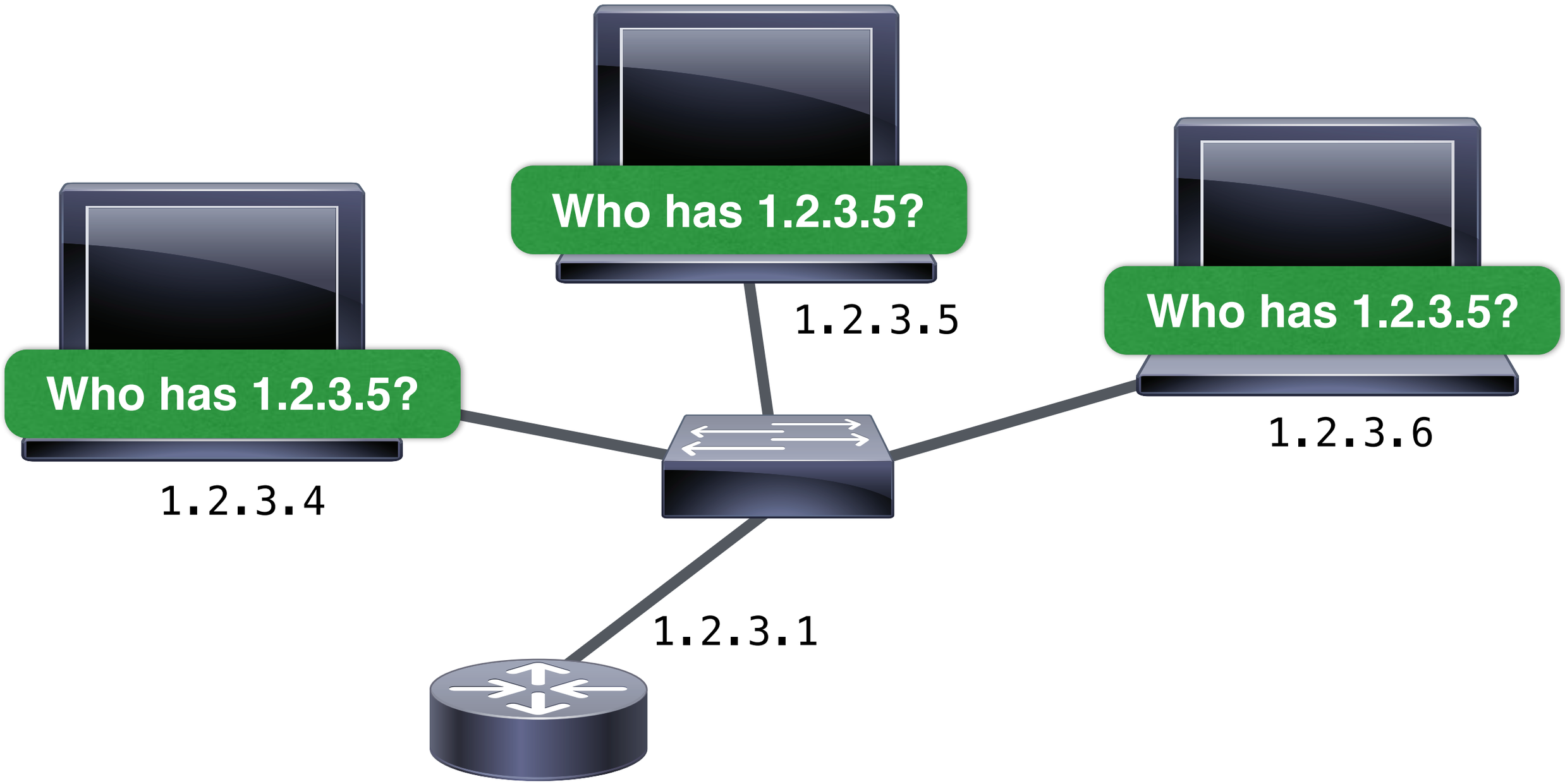
Address resolution: Data Link Layer

How to find the MAC address for an IP address:



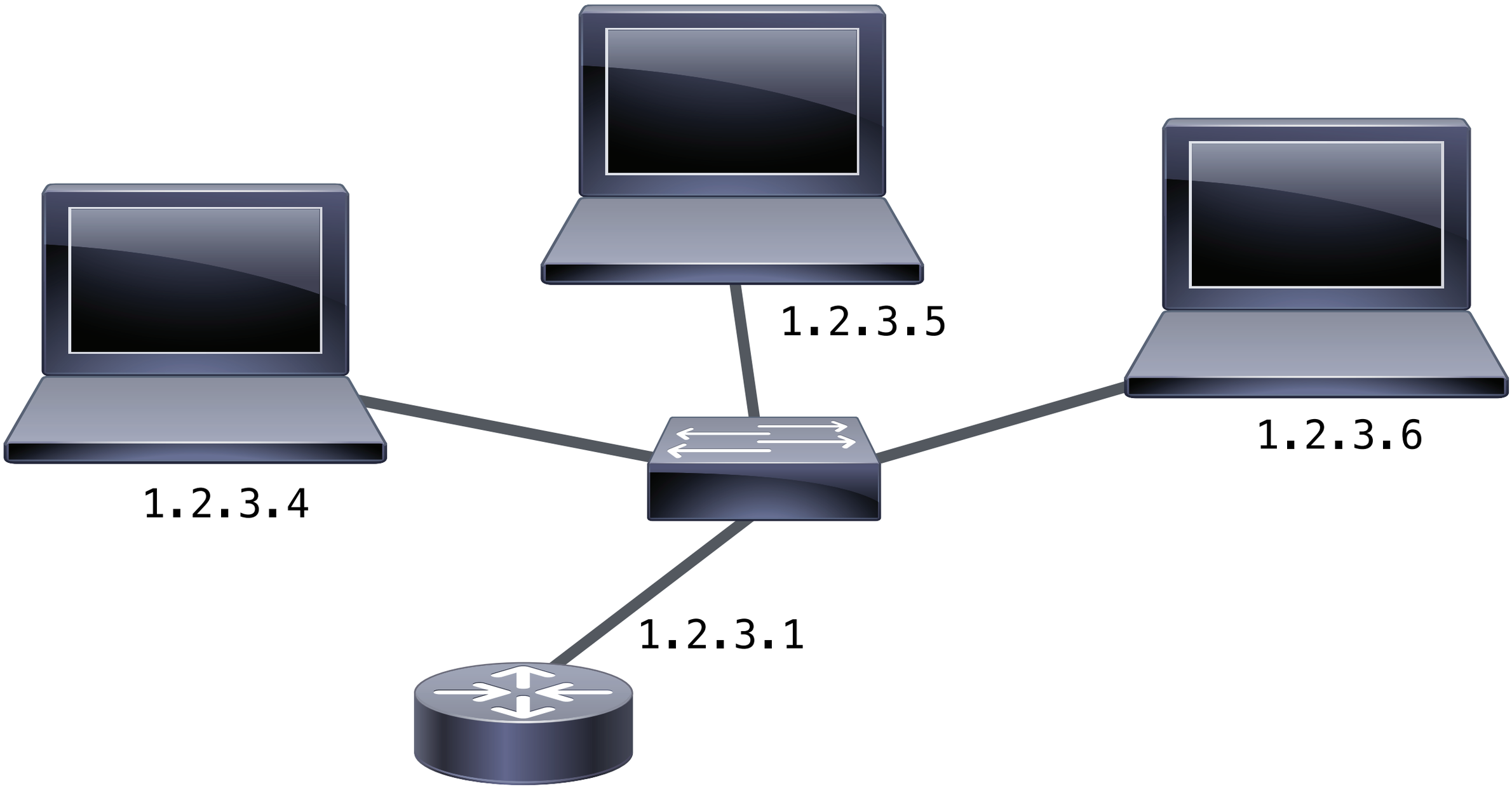
Address resolution: Data Link Layer

How to find the MAC address for an IP address:



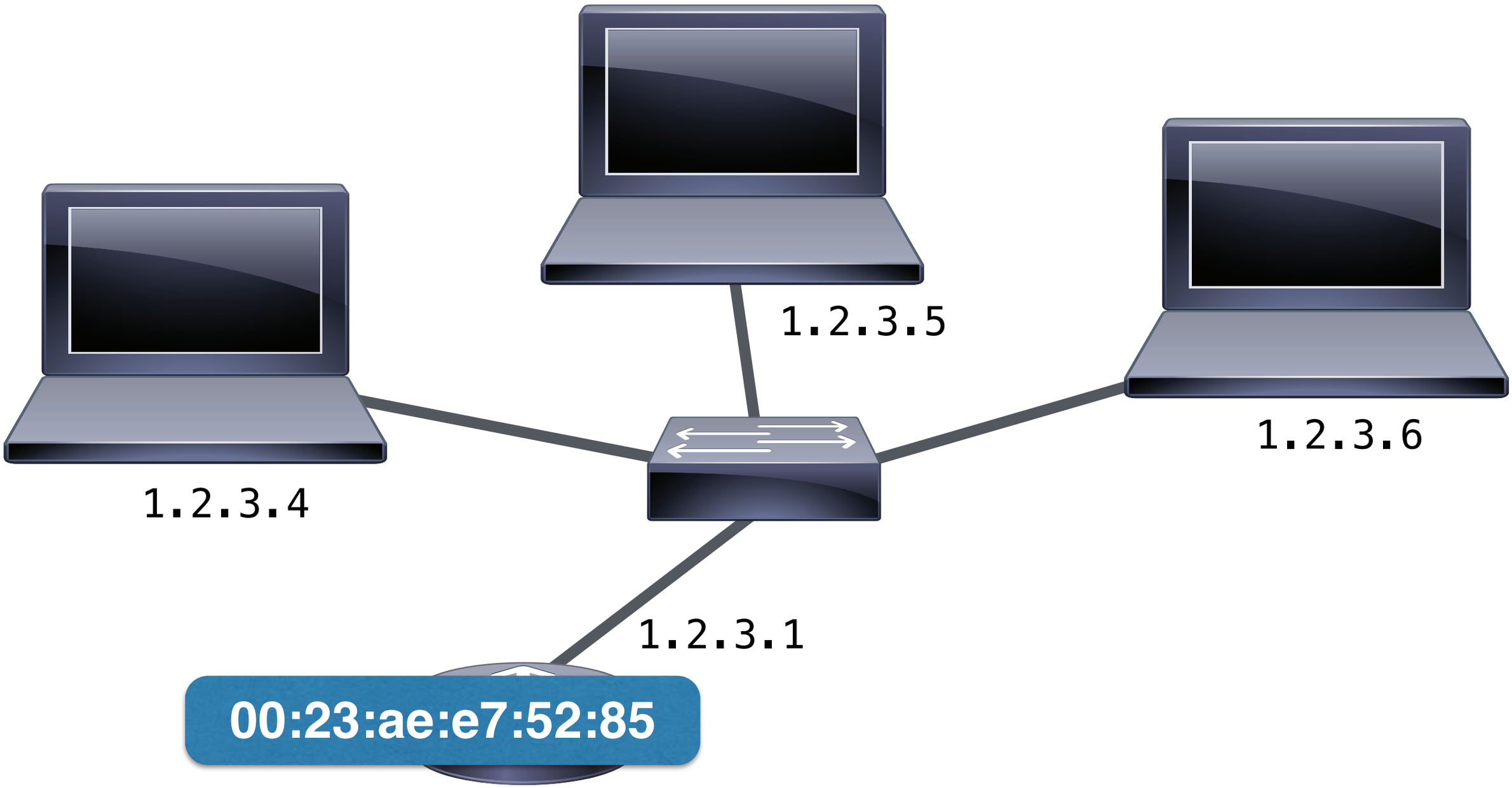
Address resolution: Data Link Layer

How to find the MAC address for an IP address:



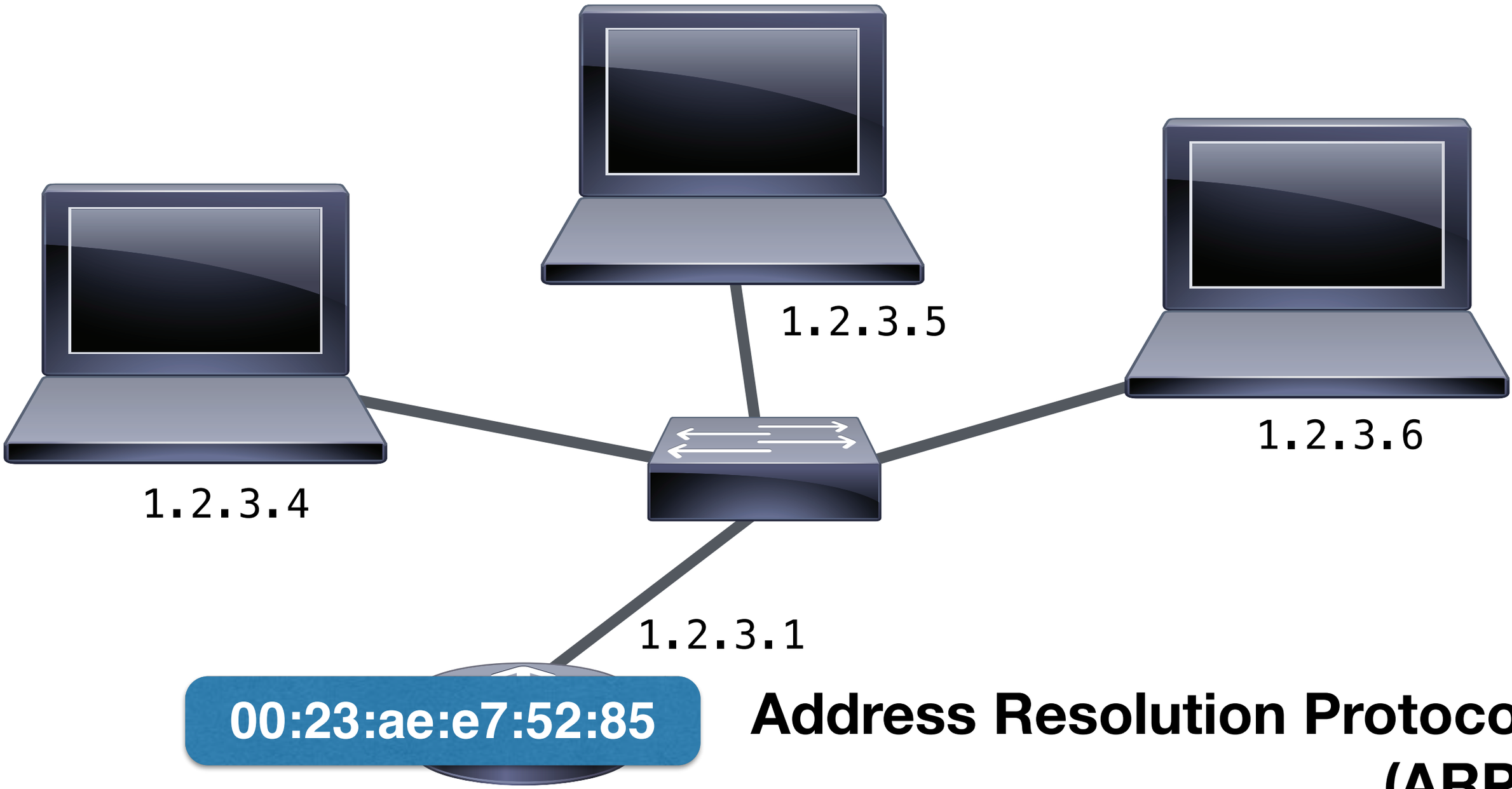
Address resolution: Data Link Layer

How to find the MAC address for an IP address:



Address resolution: Data Link Layer

How to find the MAC address for an IP address:



Address Resolution Protocol (ARP)

The Network Layer:

Routing

Routers

Routers connect networks

- Internet is a network of networks!
- Most important piece of Internet infrastructure

Routers

Routers connect networks

- Internet is a network of networks!
- Most important piece of Internet infrastructure

A router is a layer 3 device

- one IP address per **interface**, i.e. typically per subnet it is connected to
- Clients send packets to routers **if destination is outside their own subnet**
- Routers use IP address to determine **where the packet is sent next**

Routing tables

Routing tables

For each incoming packet, the router

Routing tables

For each incoming packet, the router

- looks at the packet's **destination IP address**

Routing tables

For each incoming packet, the router

- looks at the packet's **destination IP address**
- consults the **routing table:**

  to which other router should I send a packet for this destination, or can I deliver it directly?

Routing tables

For each incoming packet, the router

- looks at the packet's **destination IP address**
- consults the **routing table:**

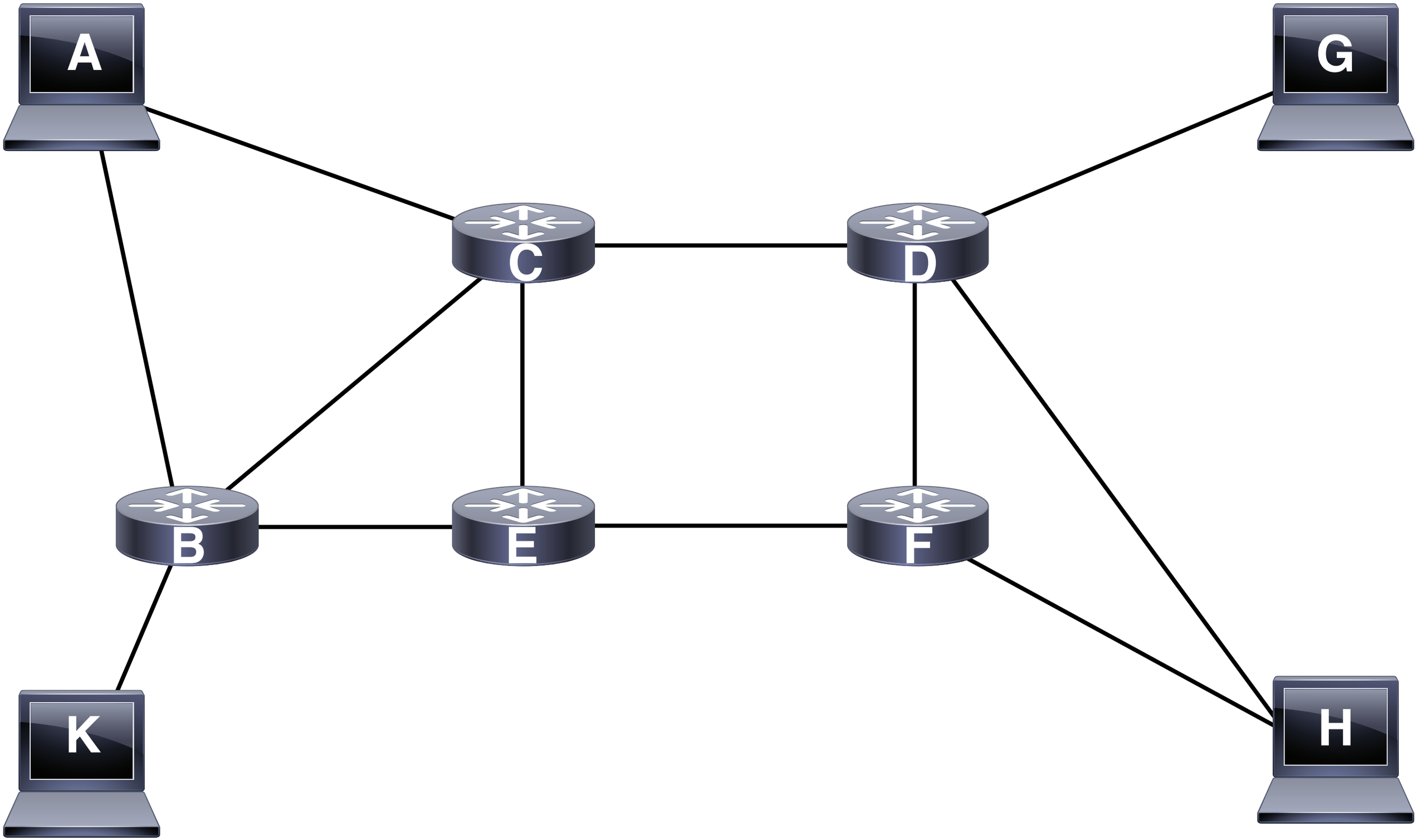
  to which other router should I send a packet for this destination, or can I deliver it directly?
- if destination not in table: send to **default gateway**

Routing tables

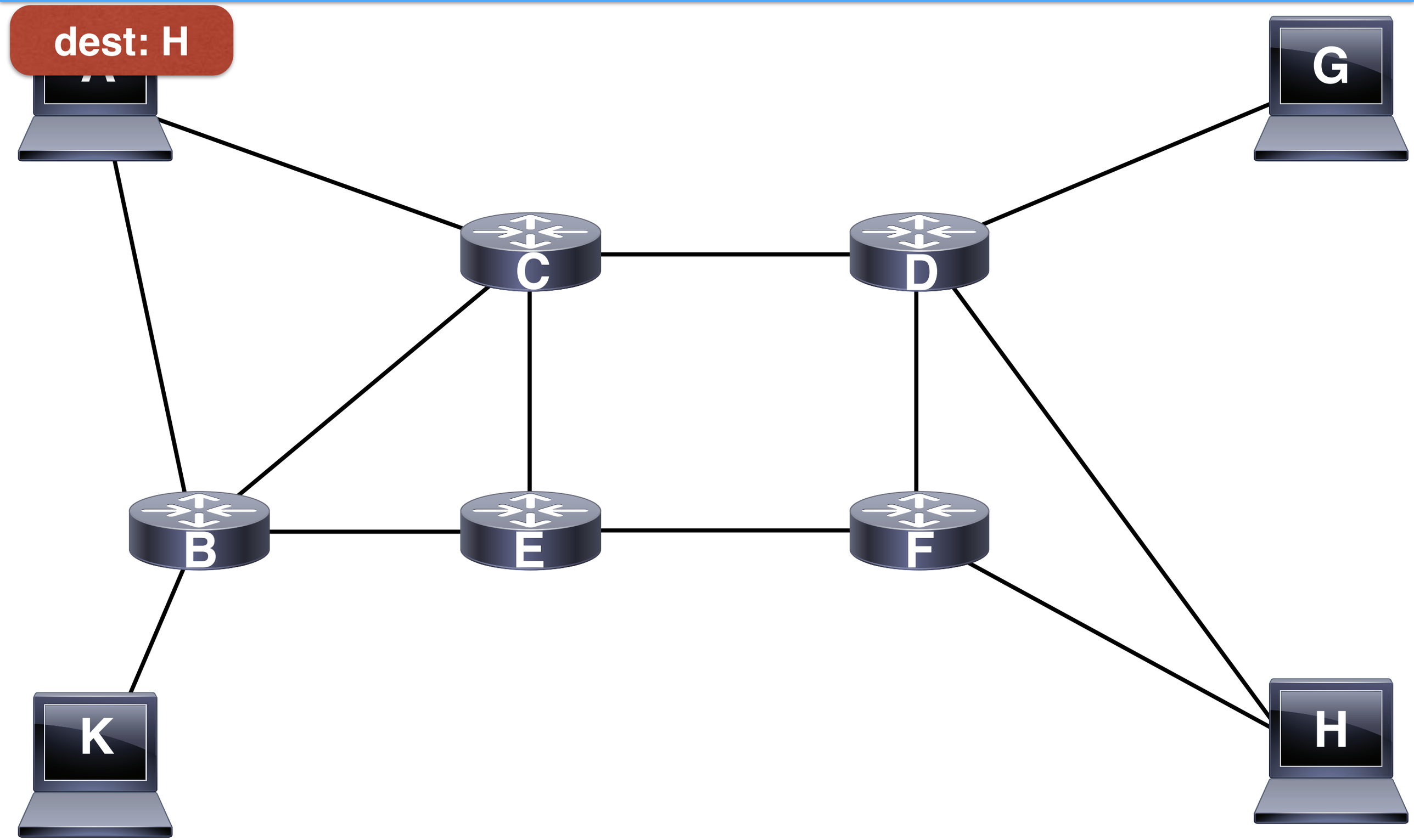
For each incoming packet, the router

- looks at the packet's **destination IP address**
- consults the **routing table:**
  to which other router should I send a packet for this destination, or can I deliver it directly?
- if destination not in table: send to **default gateway**
- if no default gateway configured: **packet can't be routed**

Routing example



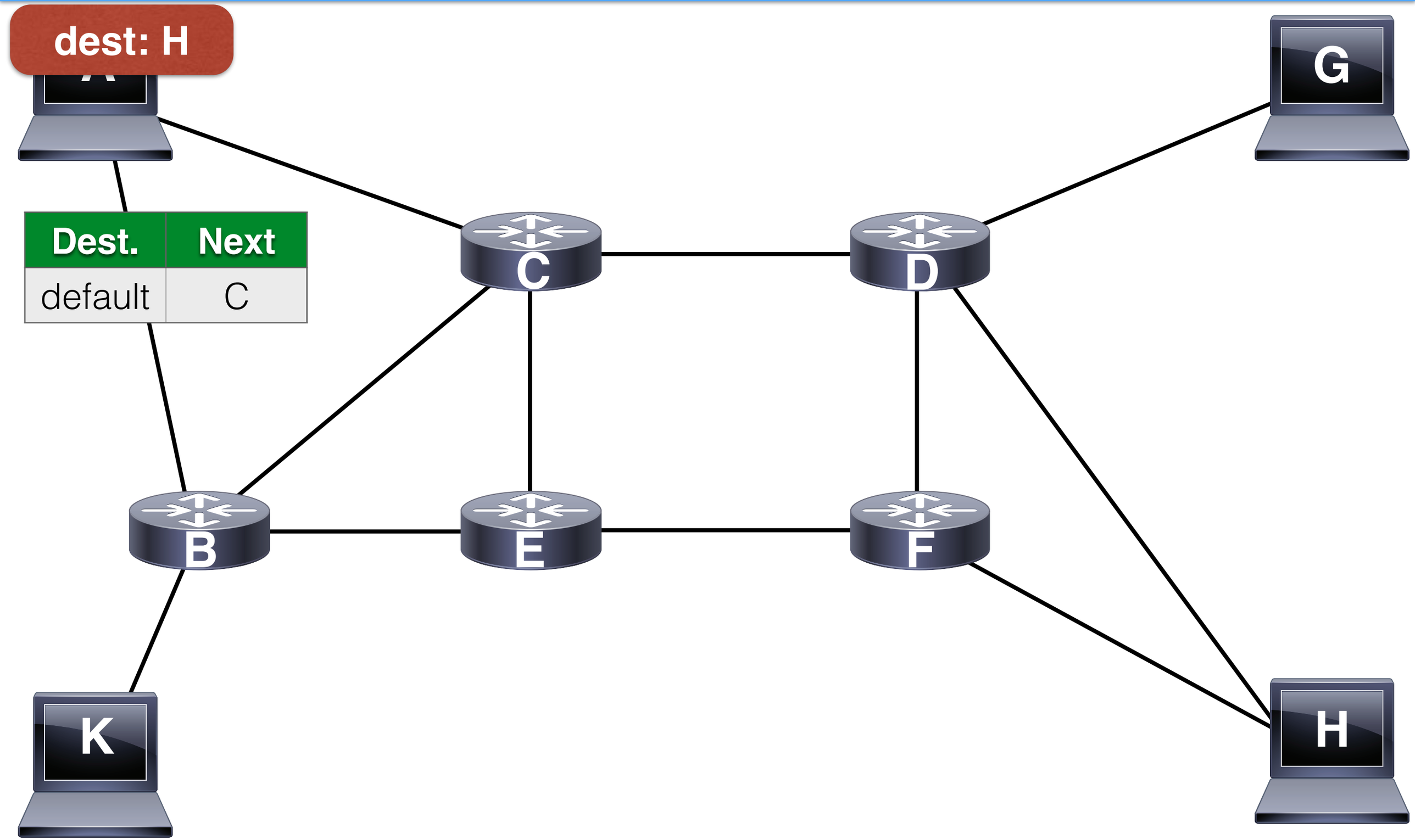
Routing example



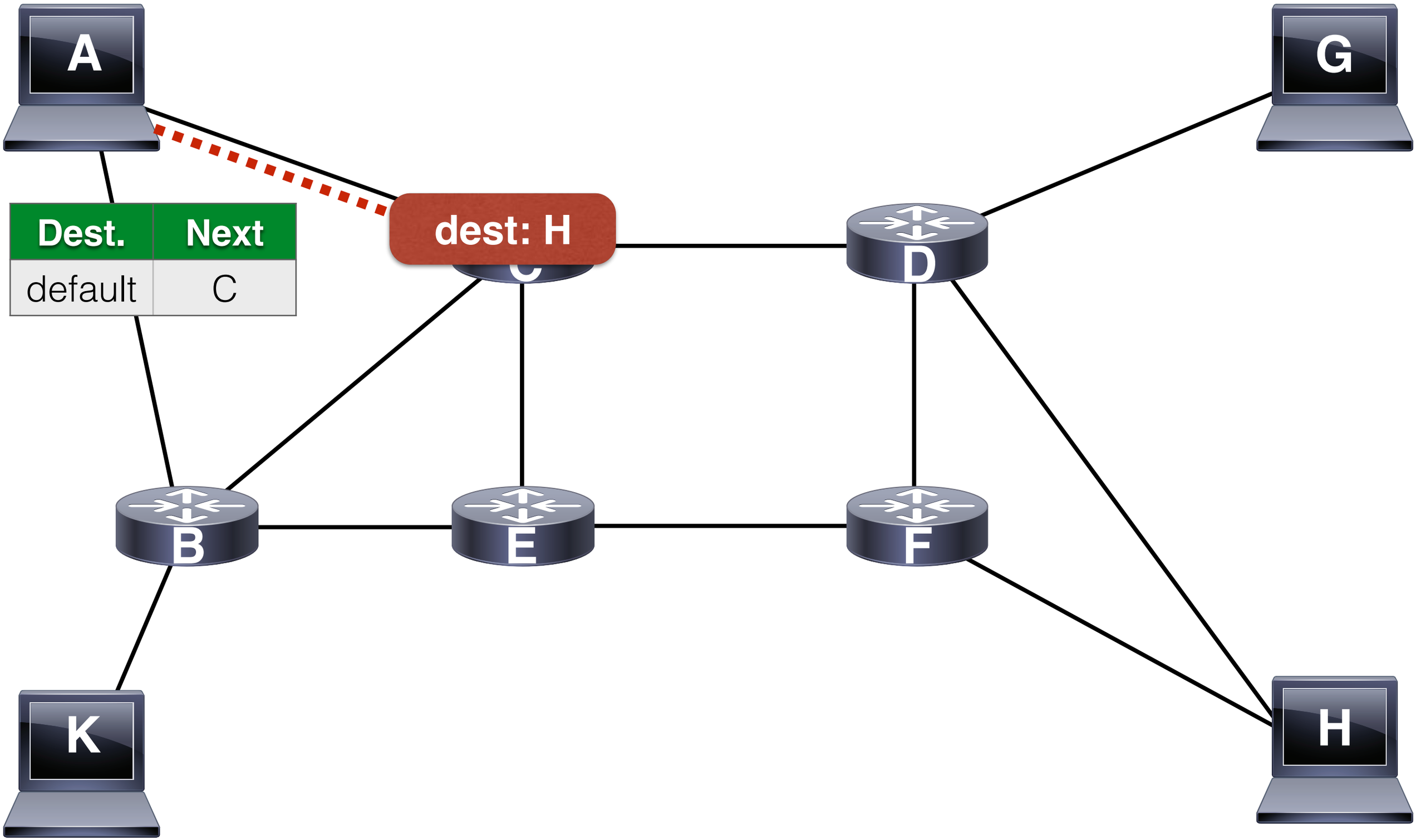
Routing example

dest: H

Dest.	Next
default	C



Routing example



Dest.	Next
default	C

Routing example

Dest.	Next
A	A
G	D
H	E
K	E

Dest.	Next
default	C

dest: H

A

G

C

D

B

E

F

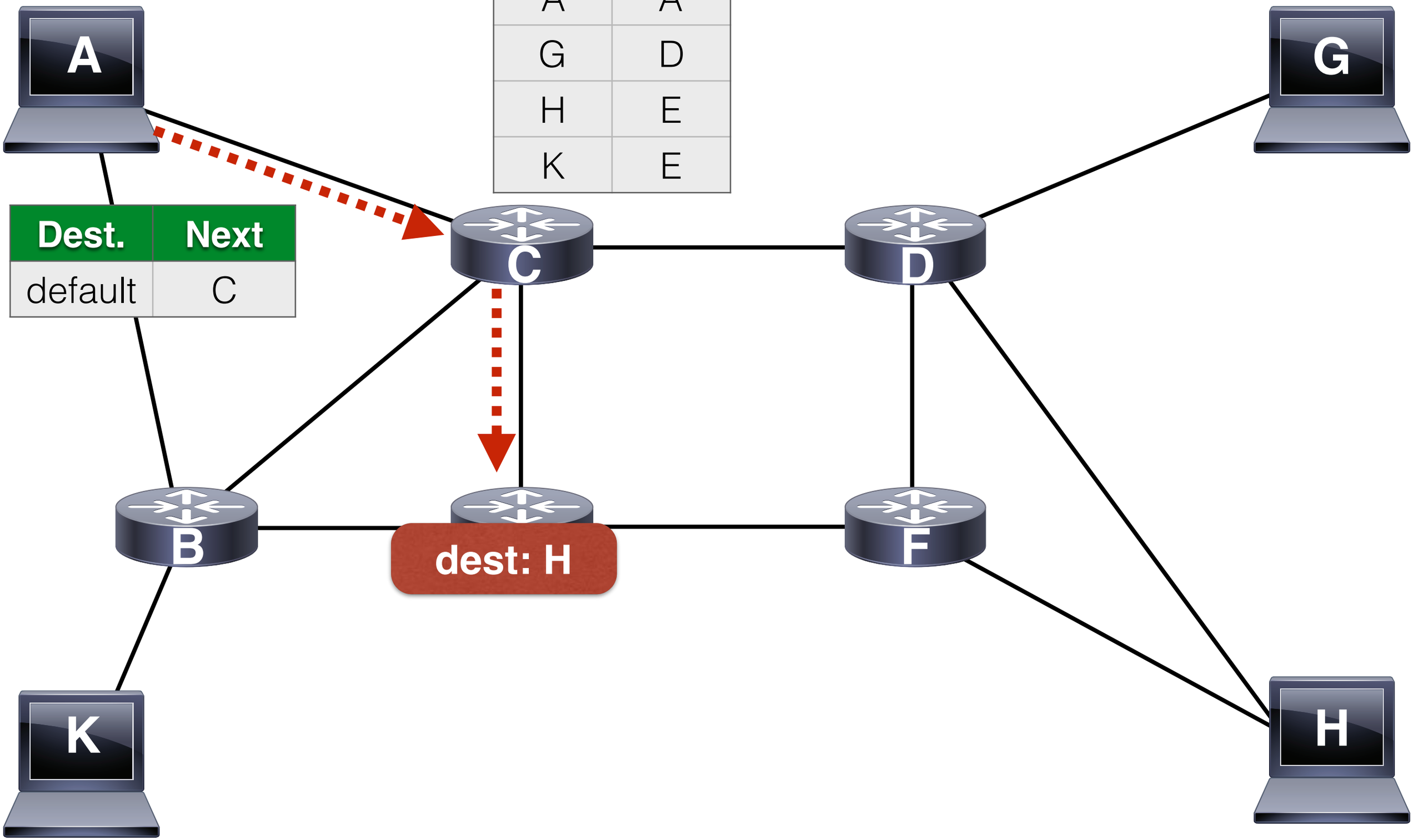
K

H

Routing example

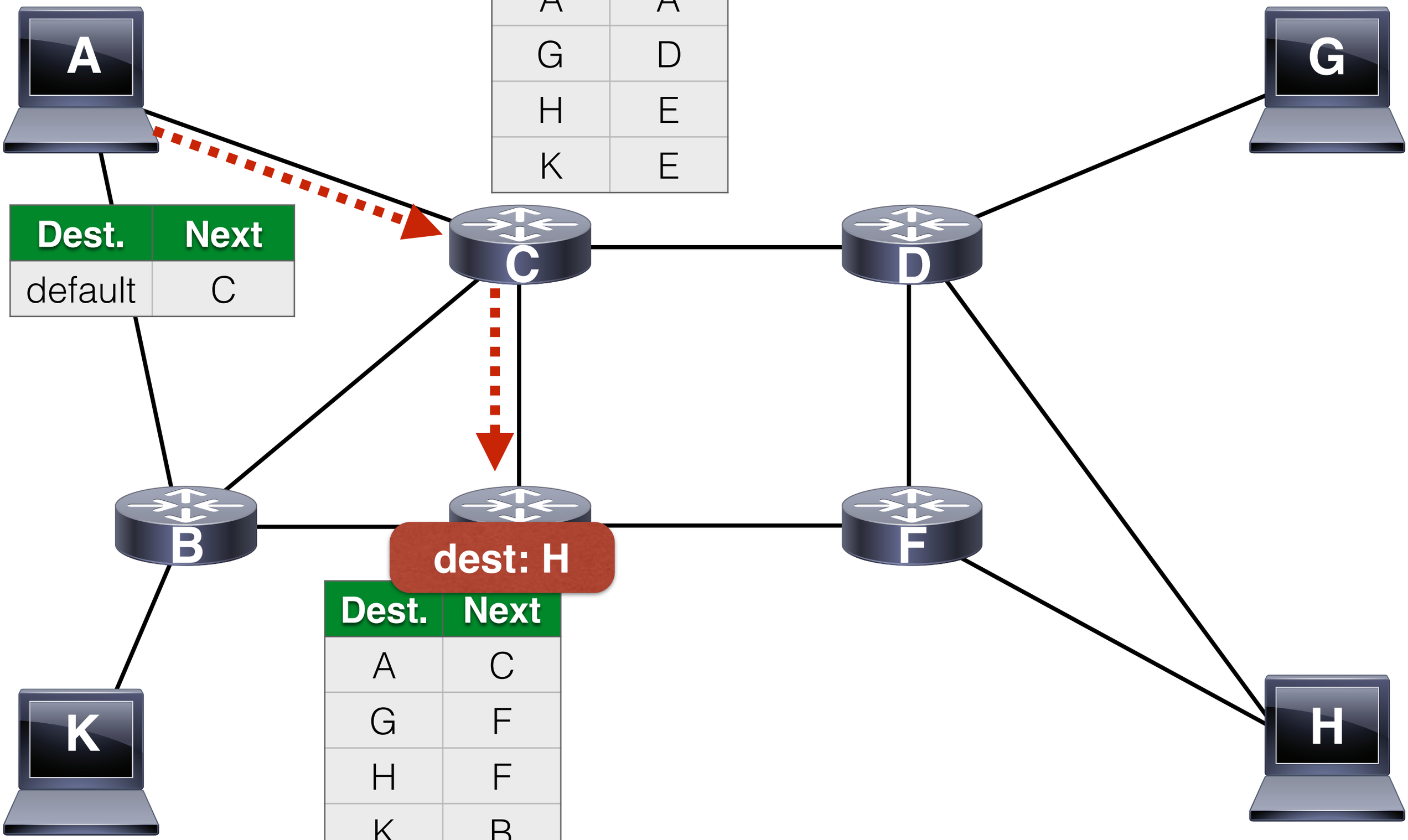
Dest.	Next
A	A
G	D
H	E
K	E

Dest.	Next
default	C



Routing example

Dest.	Next
A	A
G	D
H	E
K	E

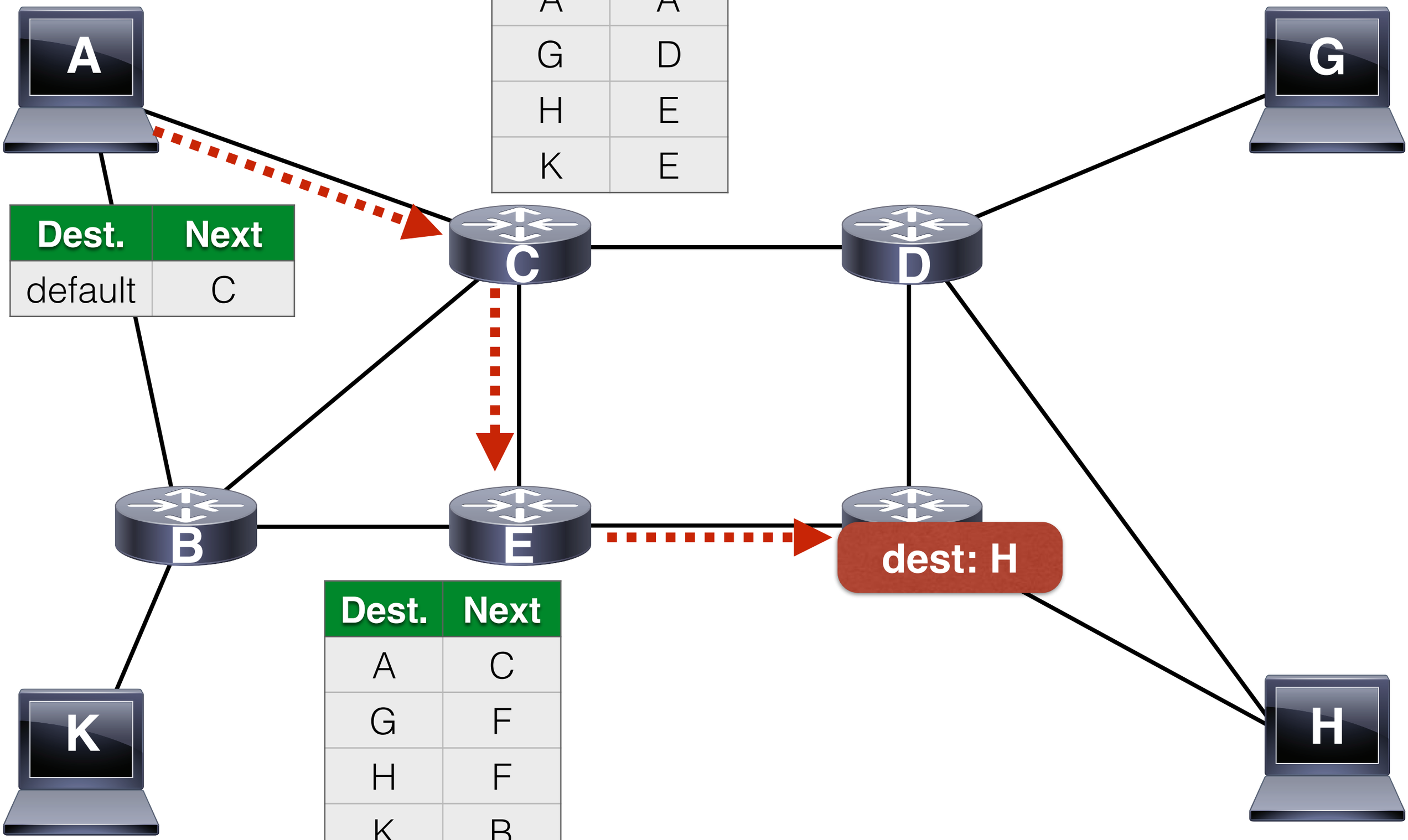


Dest.	Next
default	C

Dest.	Next
A	C
G	F
H	F
K	B

Routing example

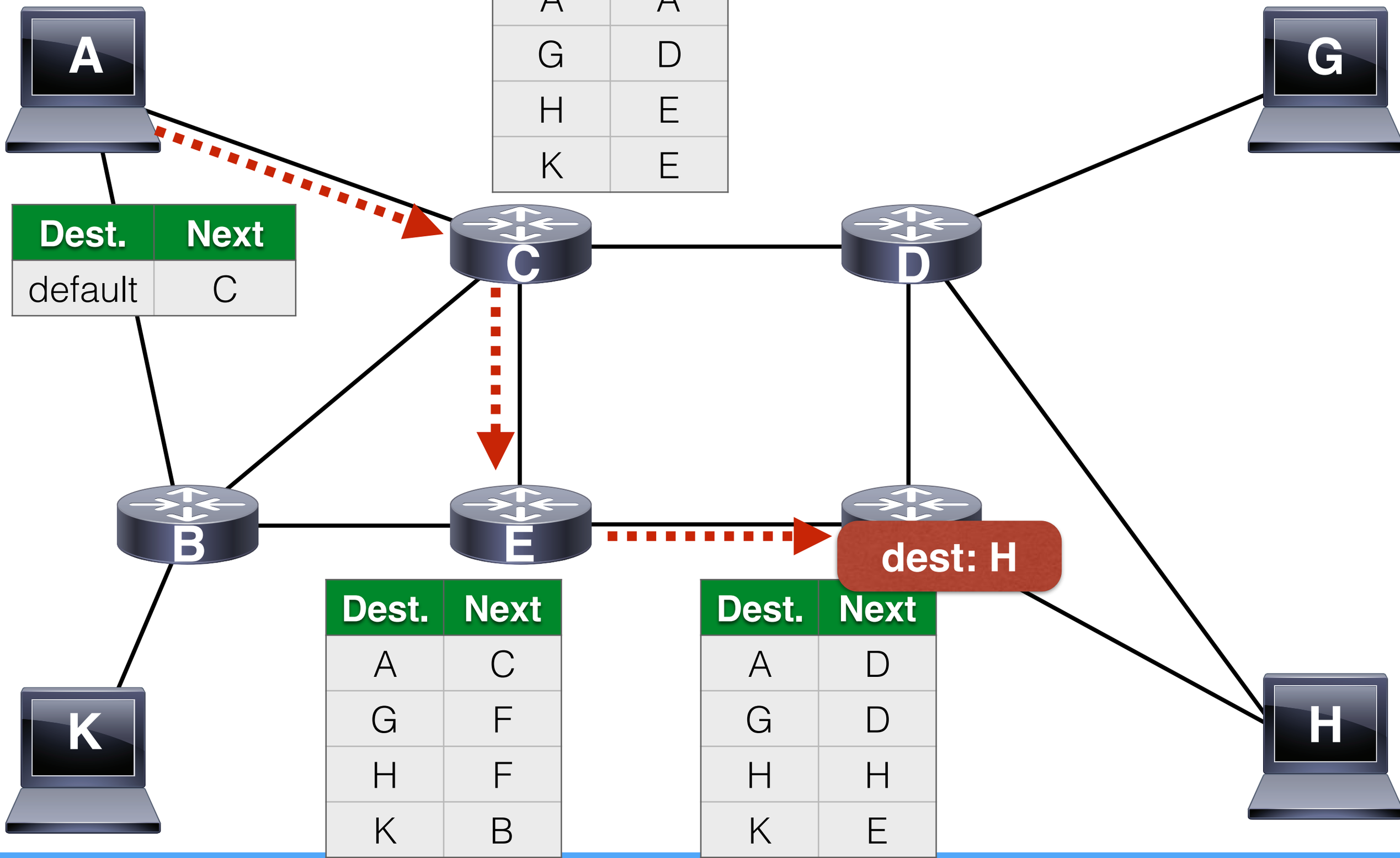
Dest.	Next
A	A
G	D
H	E
K	E



Dest.	Next
default	C

Dest.	Next
A	C
G	F
H	F
K	B

Routing example



Dest.	Next
A	A
G	D
H	E
K	E

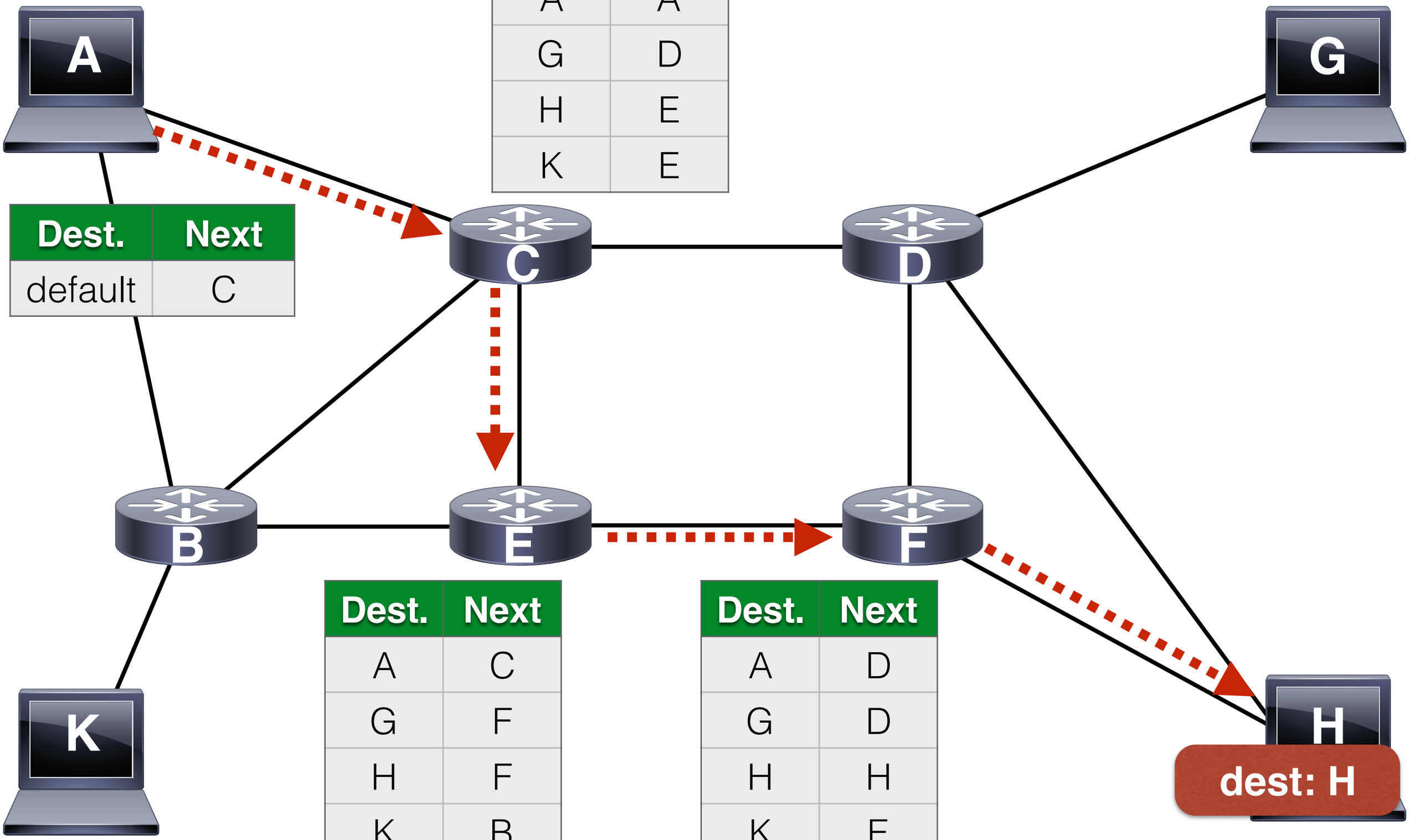
Dest.	Next
default	C

Dest.	Next
A	C
G	F
H	F
K	B

Dest.	Next
A	D
G	D
H	H
K	E

Routing example

Dest.	Next
A	A
G	D
H	E
K	E



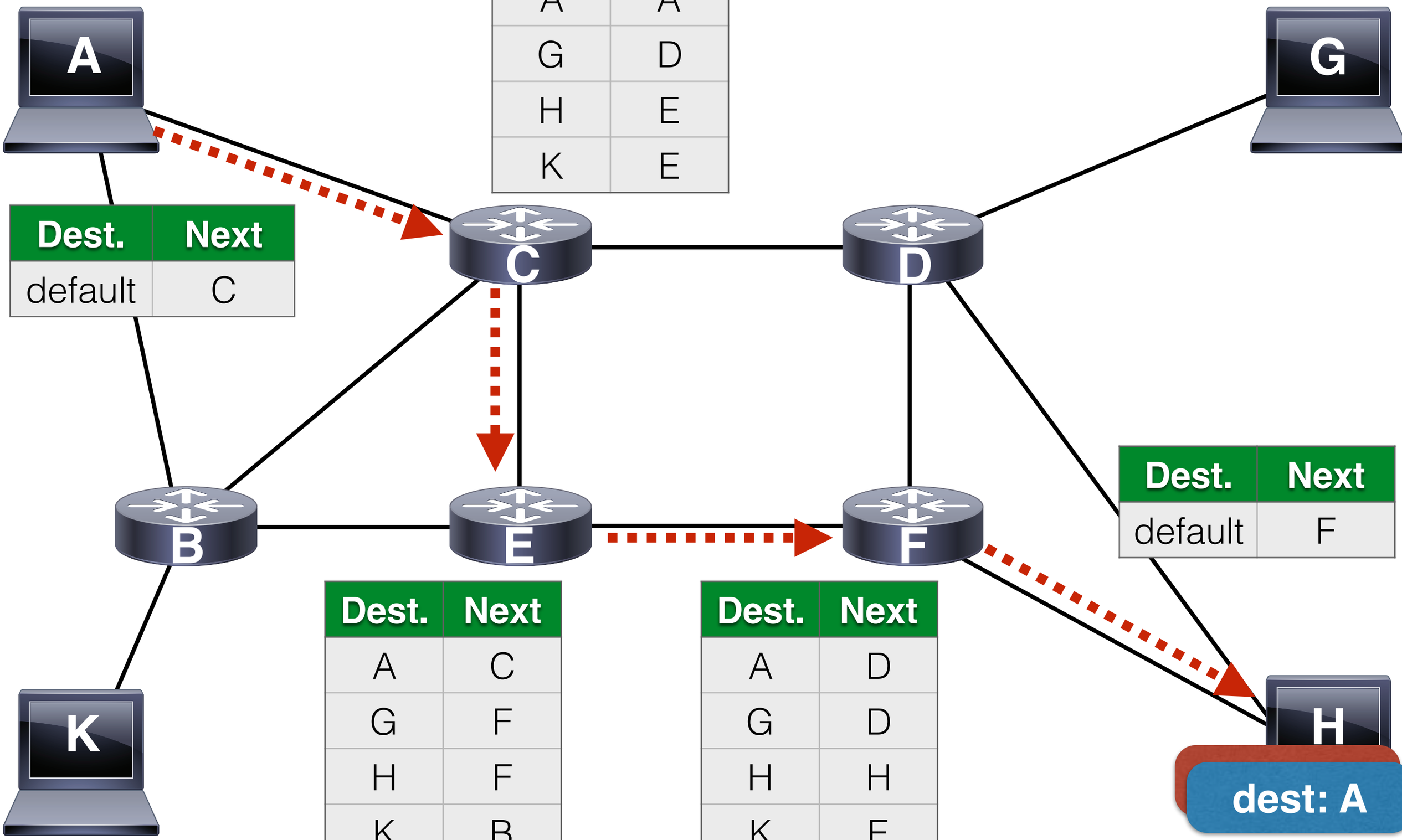
Dest.	Next
default	C

Dest.	Next
A	C
G	F
H	F
K	B

Dest.	Next
A	D
G	D
H	H
K	E

Routing example

Dest.	Next
A	A
G	D
H	E
K	E



Dest.	Next
default	C

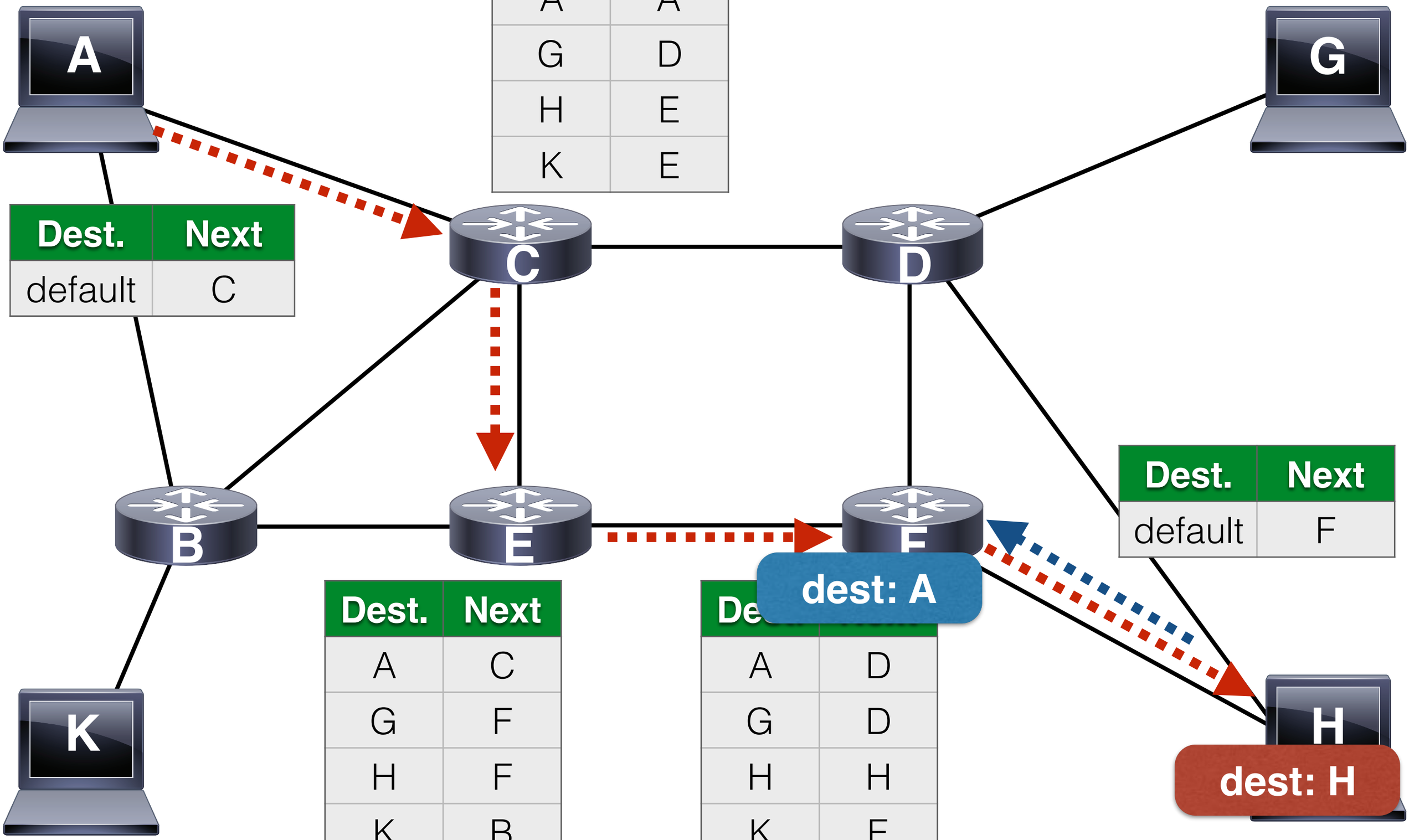
Dest.	Next
A	C
G	F
H	F
K	B

Dest.	Next
A	D
G	D
H	H
K	E

Dest.	Next
default	F

Routing example

Dest.	Next
A	A
G	D
H	E
K	E



Dest.	Next
default	C

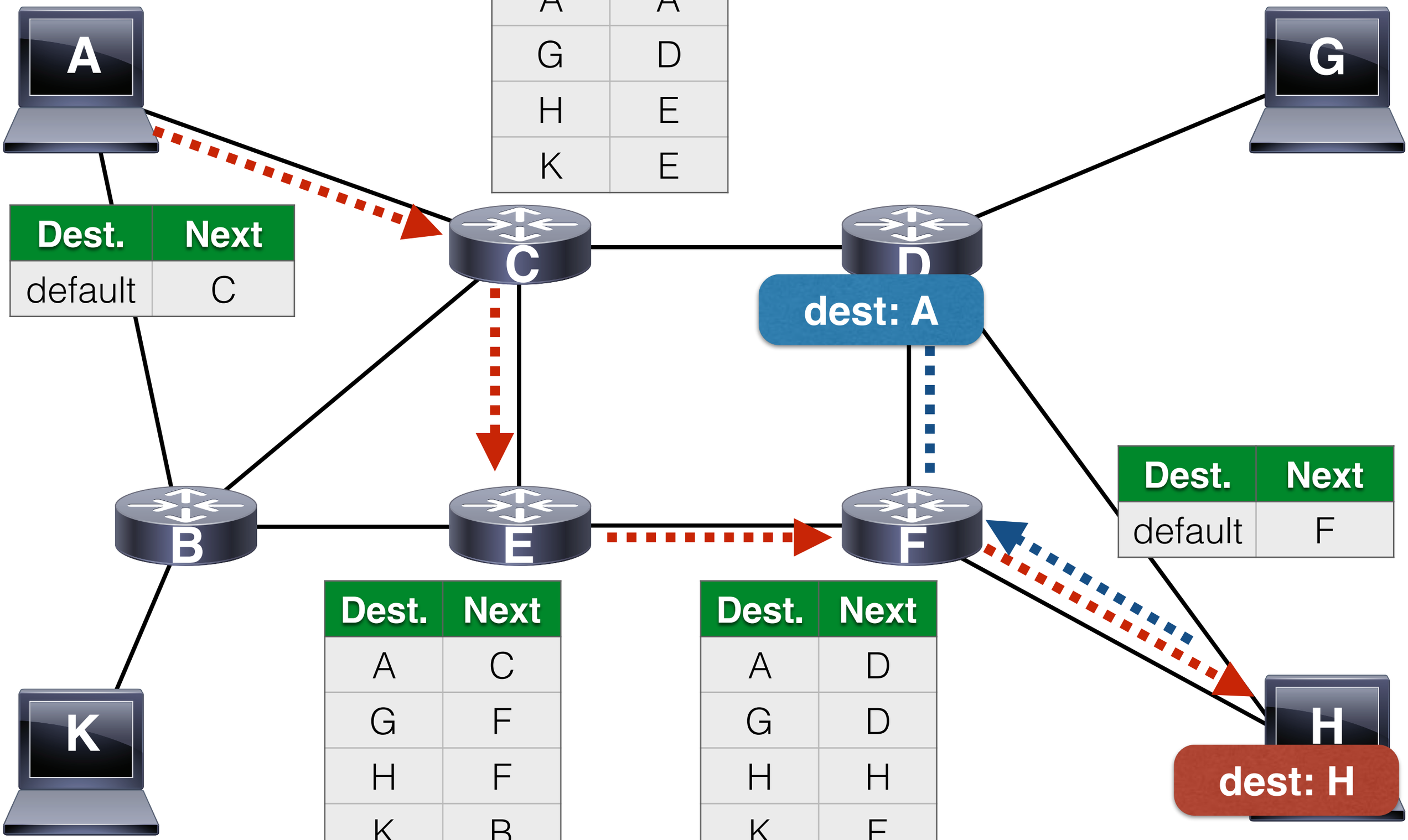
Dest.	Next
A	C
G	F
H	F
K	B

Dest.	Next
A	D
G	D
H	H
K	E

Dest.	Next
default	F

Routing example

Dest.	Next
A	A
G	D
H	E
K	E



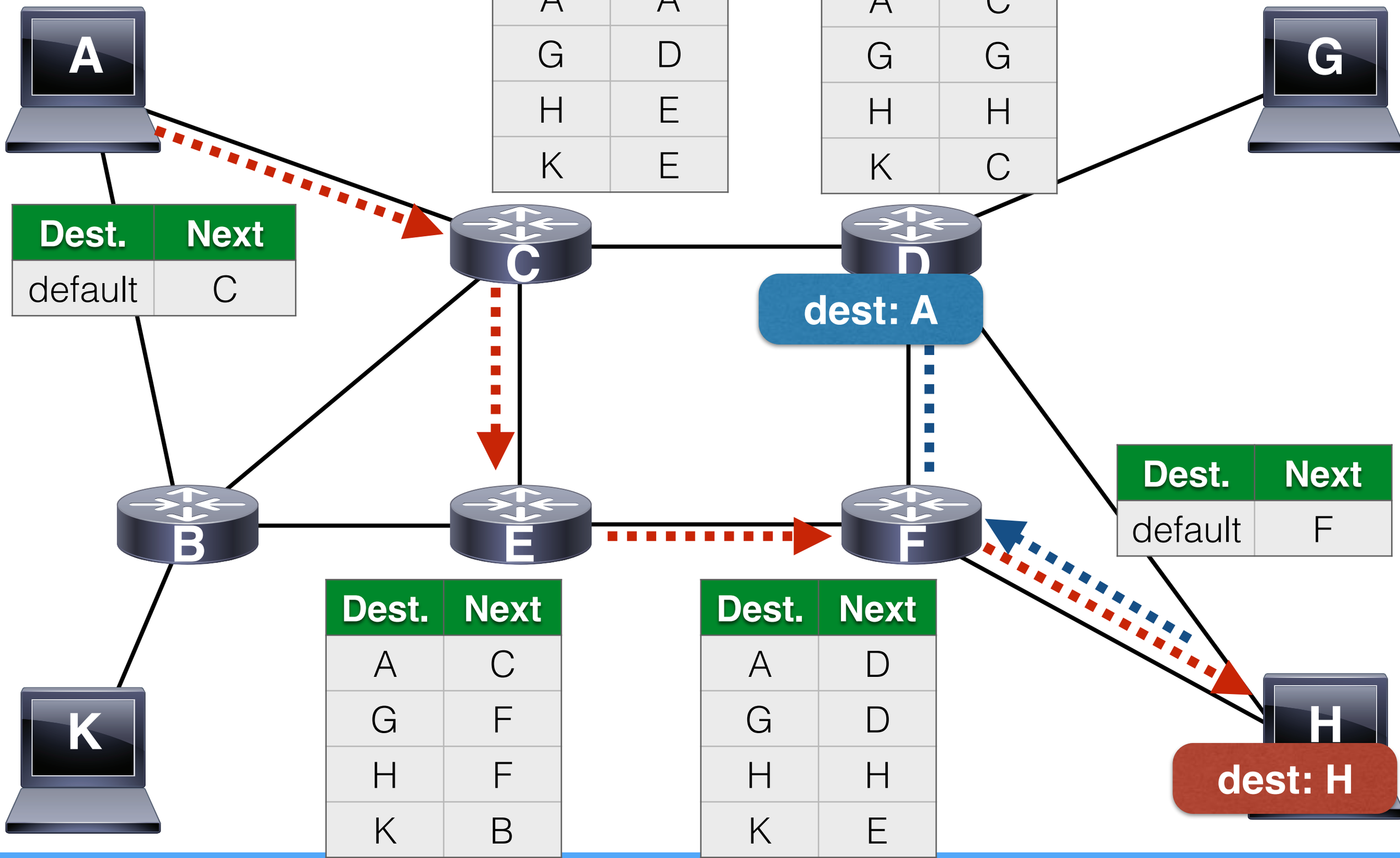
Dest.	Next
default	C

Dest.	Next
default	F

Dest.	Next
A	C
G	F
H	F
K	B

Dest.	Next
A	D
G	D
H	H
K	E

Routing example



Dest.	Next
default	C

Dest.	Next
A	A
G	D
H	E
K	E

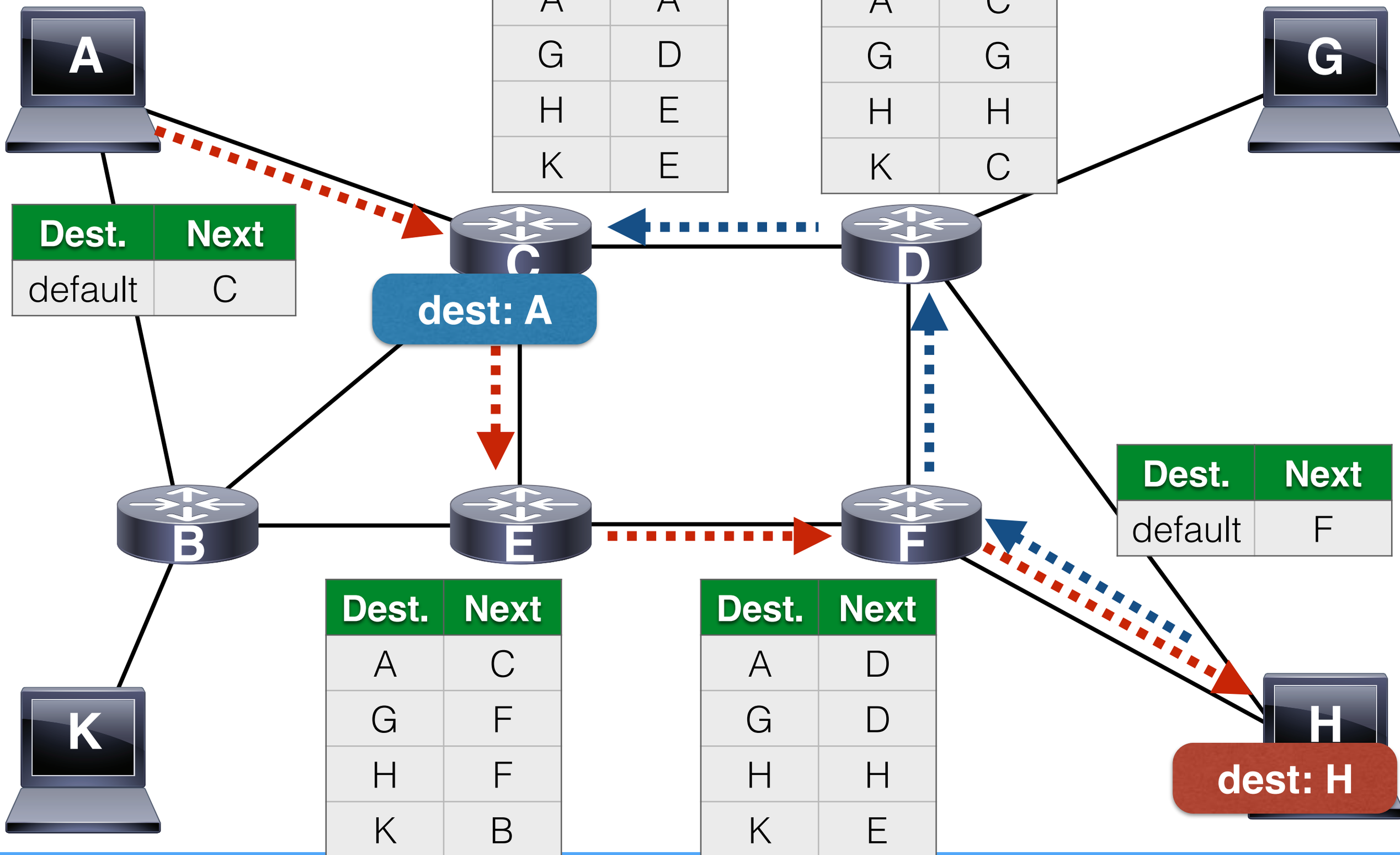
Dest.	Next
A	C
G	G
H	H
K	C

Dest.	Next
A	C
G	F
H	F
K	B

Dest.	Next
A	D
G	D
H	H
K	E

Dest.	Next
default	F

Routing example



Dest.	Next
A	A
G	D
H	E
K	E

Dest.	Next
A	C
G	G
H	H
K	C

Dest.	Next
default	C

Dest.	Next
A	C
G	F
H	F
K	B

Dest.	Next
A	D
G	D
H	H
K	E

Dest.	Next
default	F

Routing example

dest: A

Dest.	Next
A	A
G	D
H	E
K	E

Dest.	Next
A	C
G	G
H	H
K	C

Dest.	Next
default	C

Dest.	Next
default	F

Dest.	Next
A	C
G	F
H	F
K	B

Dest.	Next
A	D
G	D
H	H
K	E

dest: H

Types of routing

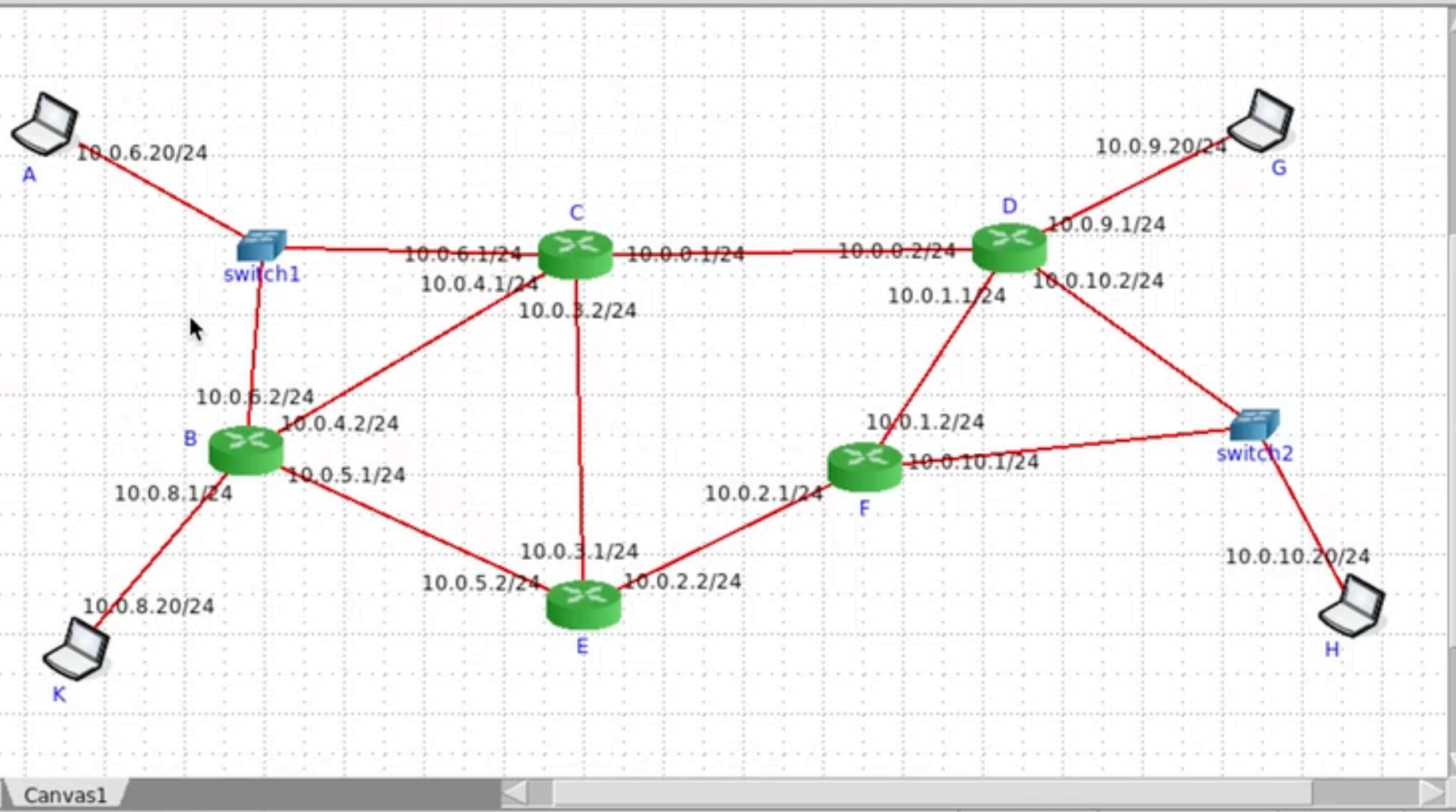
Static routing

- Network manager prepares **fixed routing tables**
- Manually updated when the network changes
- Used in simple networks that don't change a lot

Static routing demo

CORE (60497 on virtual-core) route_static.imn

File Edit Canvas View Tools Widgets Session Help



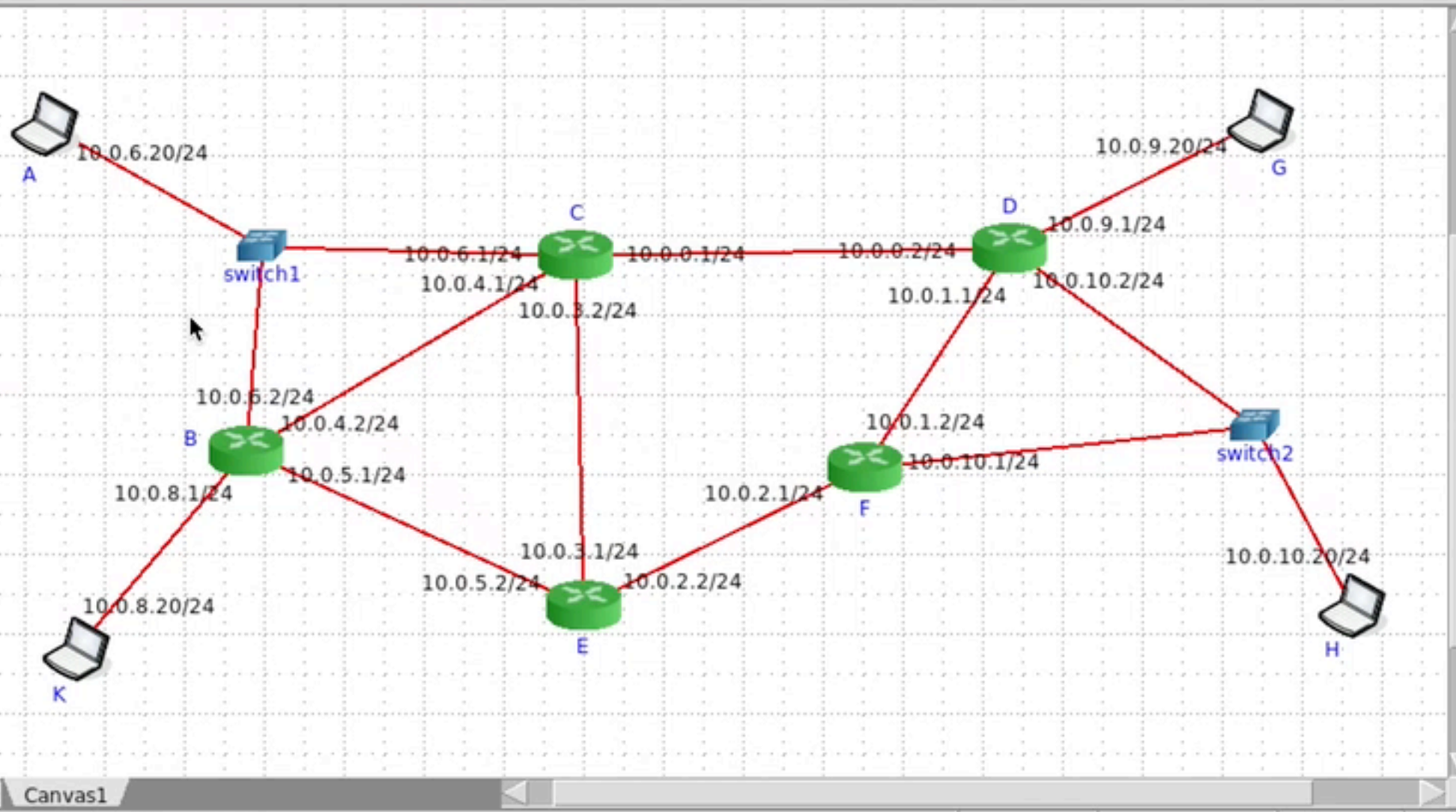
Canvas1

Cleanup completed in 0 seconds.

zoom 100%

CORE (60497 on virtual-core) route_static.imn

File Edit Canvas View Tools Widgets Session Help



Canvas1

Cleanup completed in 0 seconds.

zoom 100%

Types of routing

Static routing

- Network manager prepares **fixed routing tables**
- Manually updated when the network changes
- Used in simple networks that don't change a lot

Types of routing

Static routing

- Network manager prepares **fixed routing tables**
- Manually updated when the network changes
- Used in simple networks that don't change a lot

Dynamic routing

- Routers **exchange information** to build routing tables **dynamically**
- Initial tables can be set up by network managers

Dynamic routing algorithms

Distance vector

- Exchange information about **distance to destination**, choose **shortest route**
- EIGRP (Enhanced Interior Gateway Routing Protocol)
- RIP (Routing Information Protocol)
- BGP (Border Gateway Protocol)

Dynamic routing algorithms

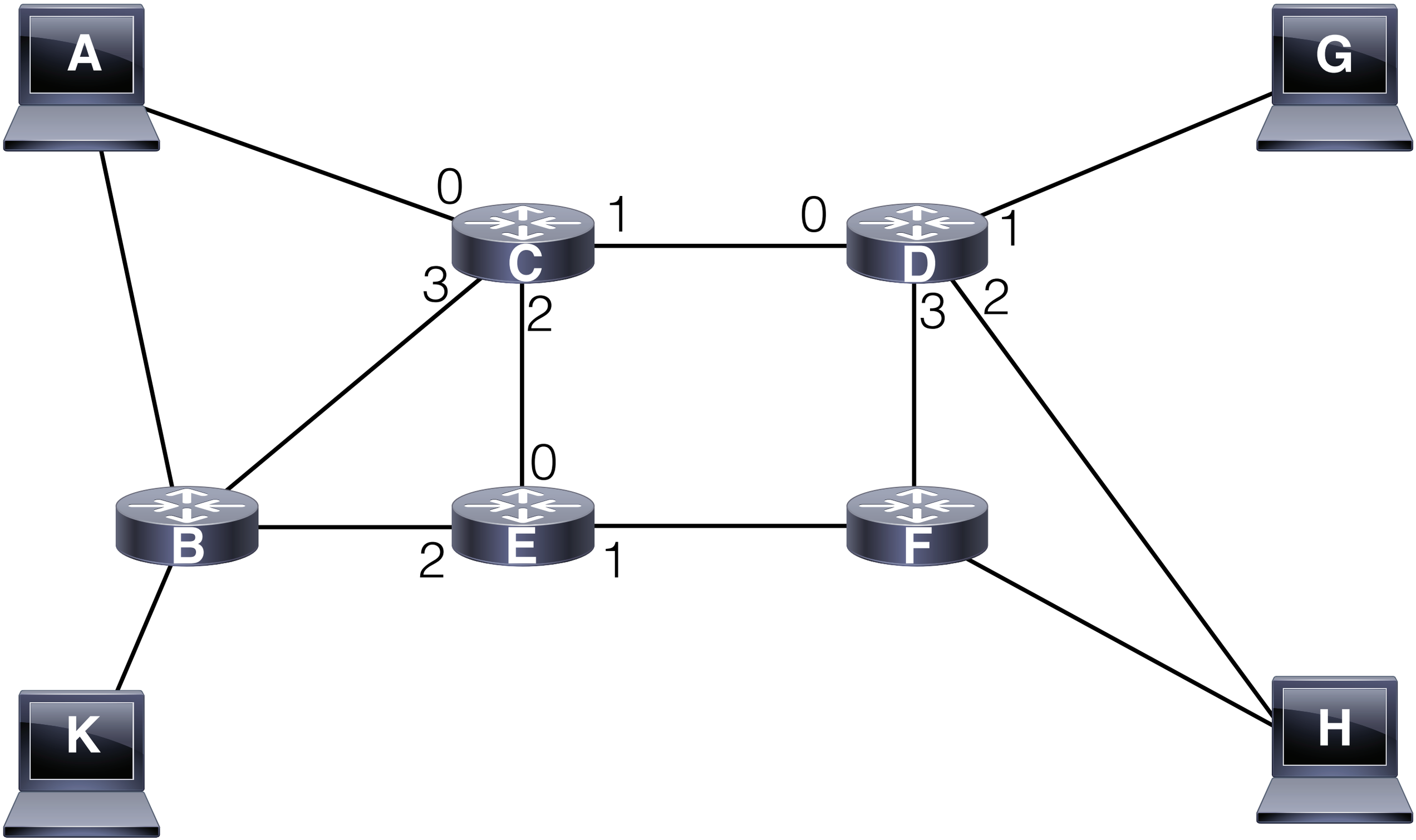
Distance vector

- Exchange information about **distance to destination**, choose **shortest route**
- EIGRP (Enhanced Interior Gateway Routing Protocol)
- RIP (Routing Information Protocol)
- BGP (Border Gateway Protocol)

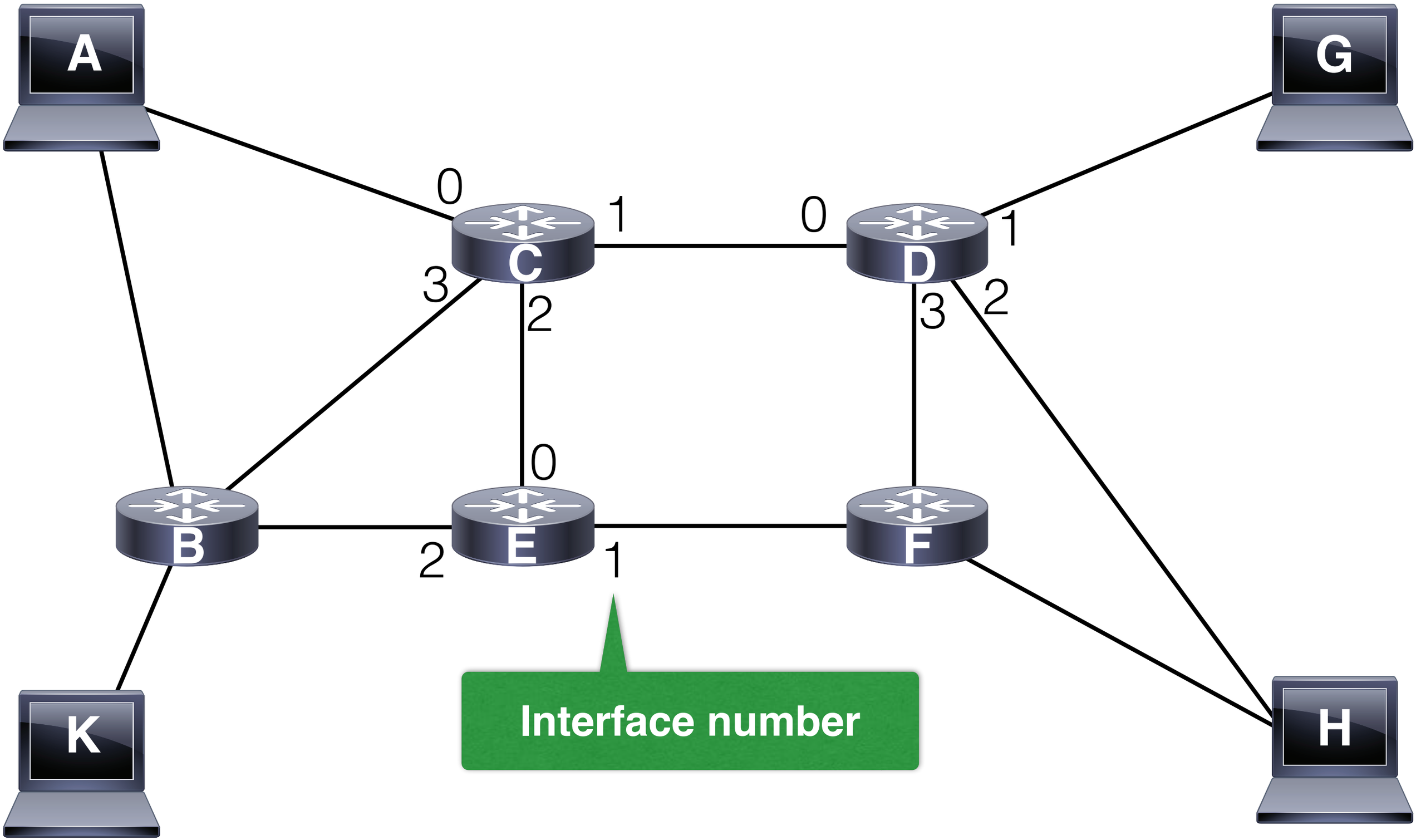
Link state

- Exchange information about **quality of links**, choose **fastest route**
- OSPF (Open Shortest Path First)

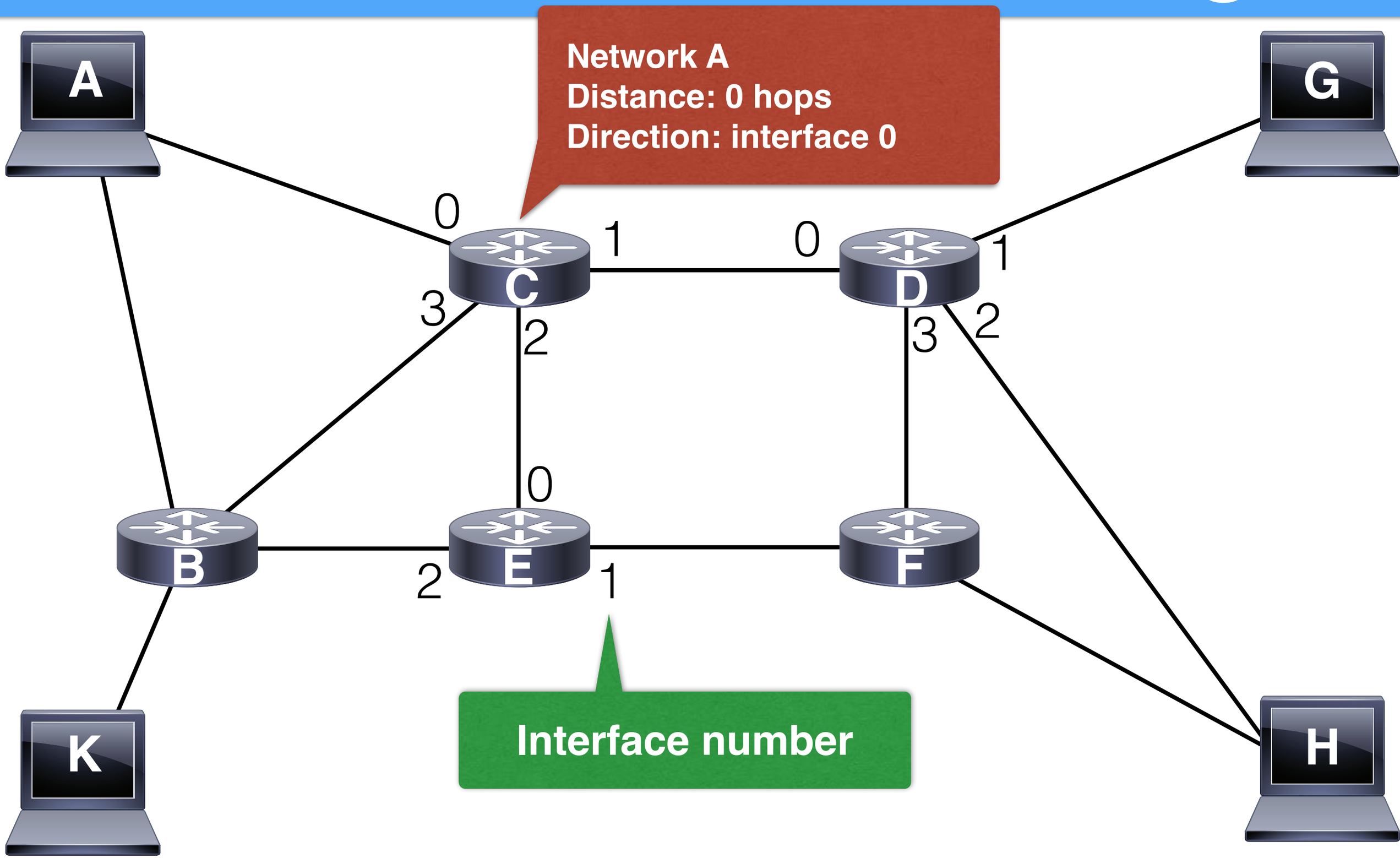
Distance vector routing



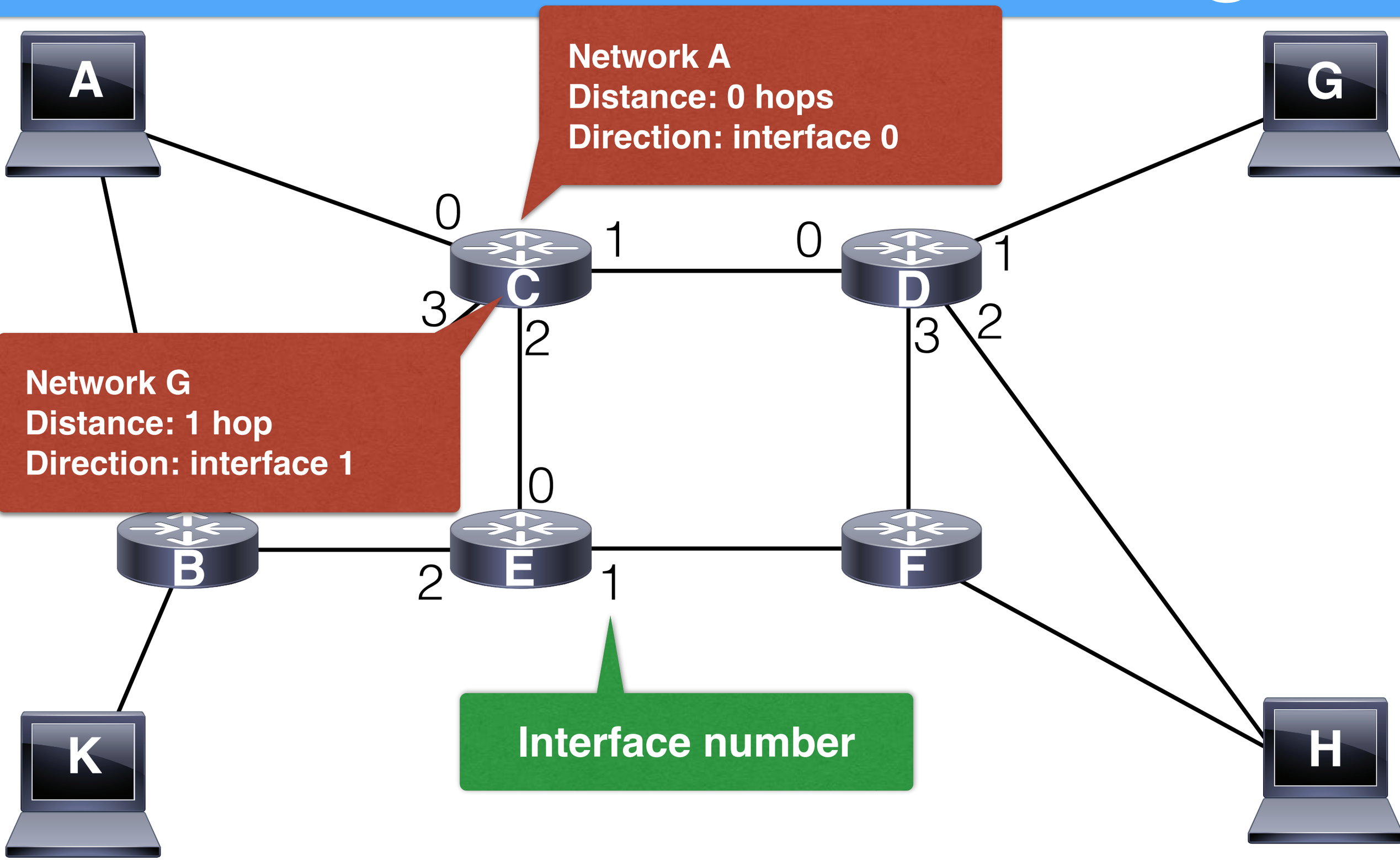
Distance vector routing



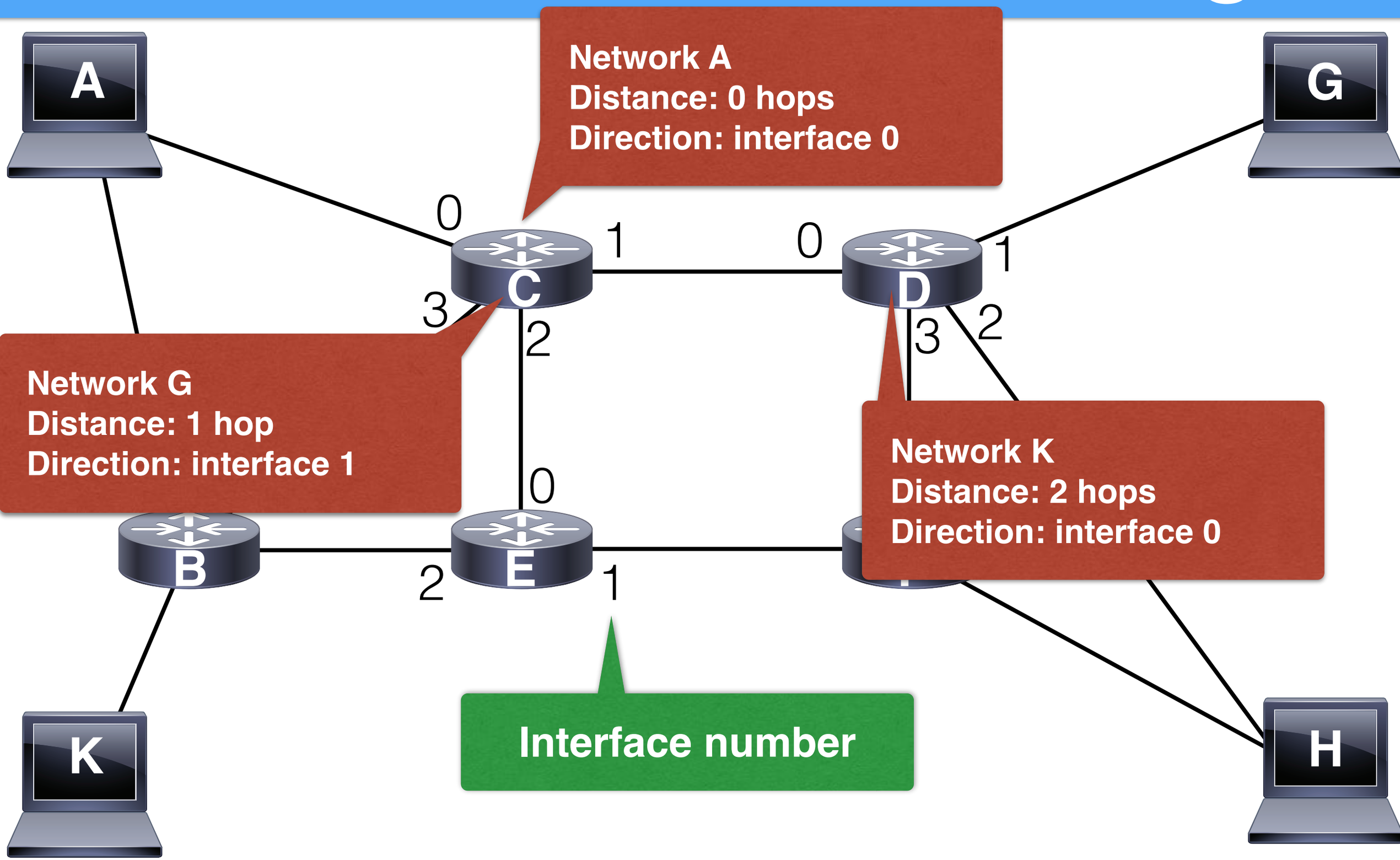
Distance vector routing



Distance vector routing



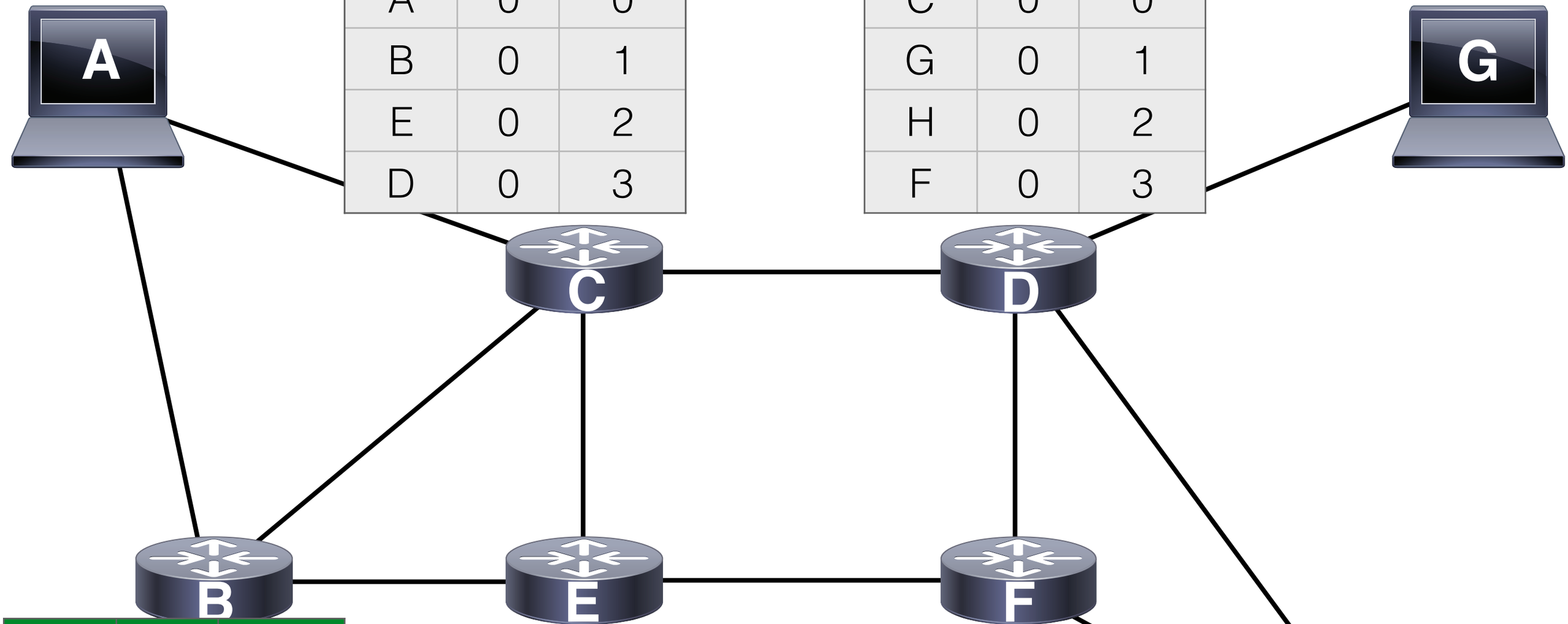
Distance vector routing



Routing Information Protocol (RIP)

Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3

Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3



Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

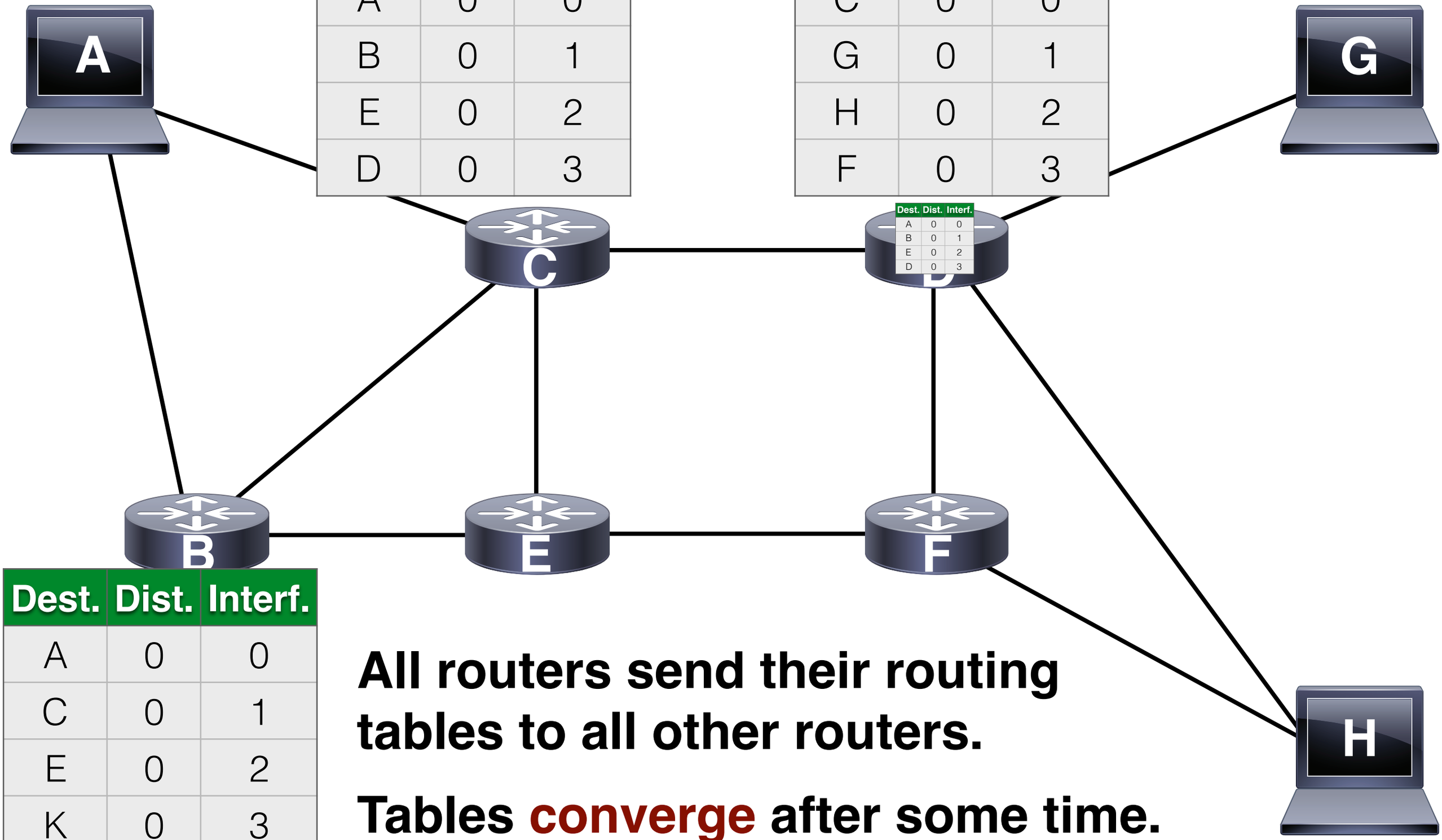
All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)

Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3

Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3



Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

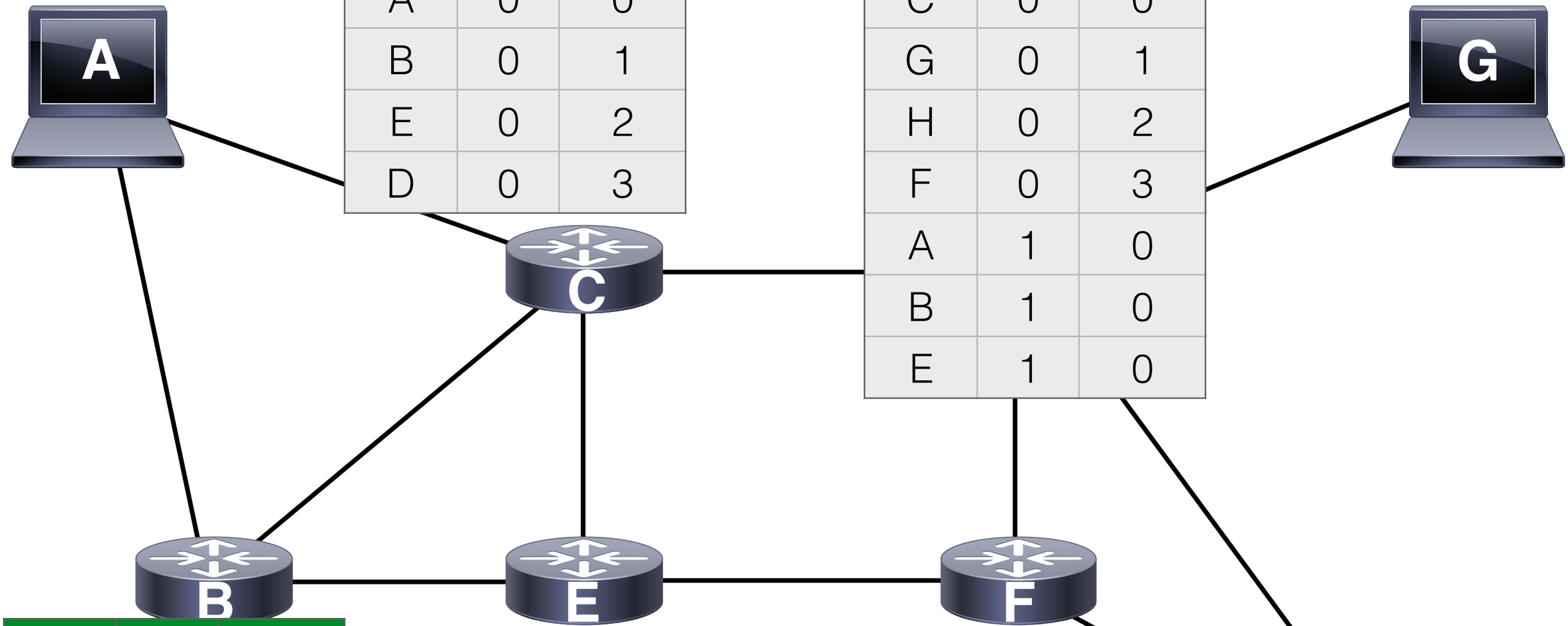
All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)

Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3

Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3
A	1	0
B	1	0
E	1	0



Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

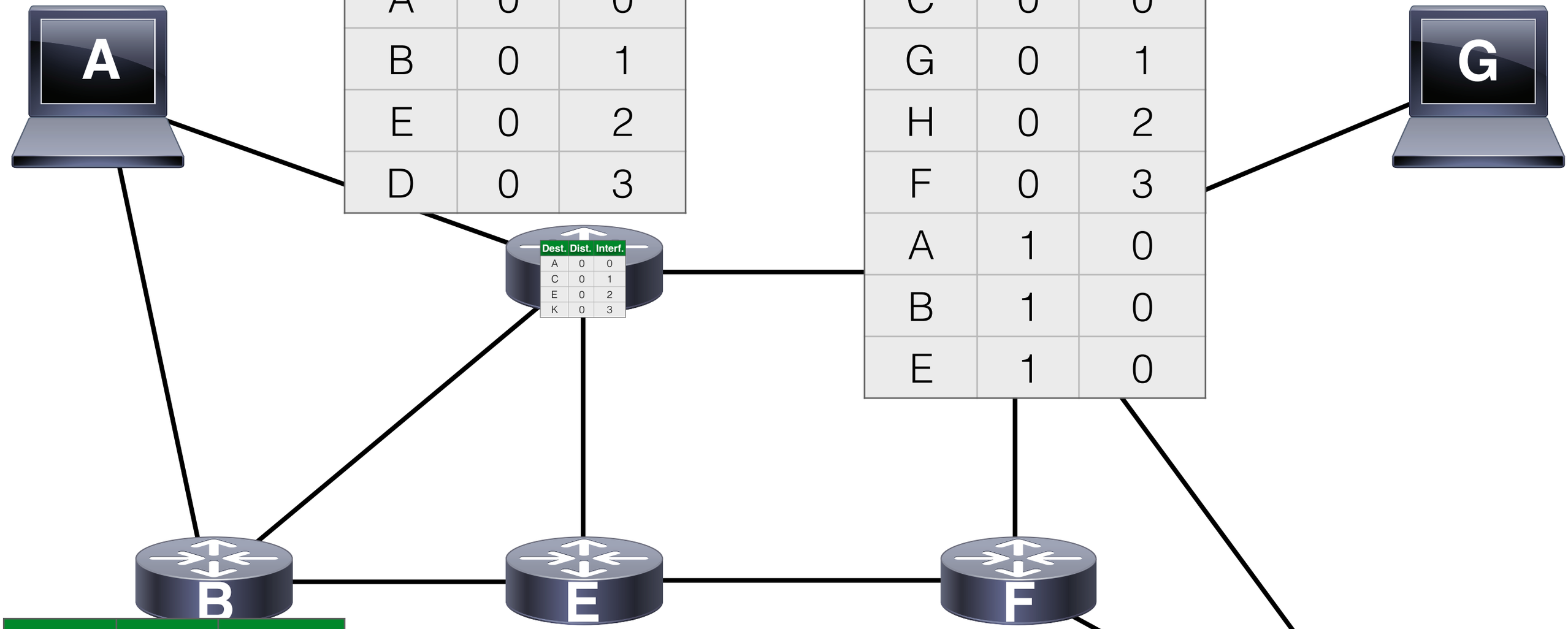
All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)

Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3

Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3
A	1	0
B	1	0
E	1	0

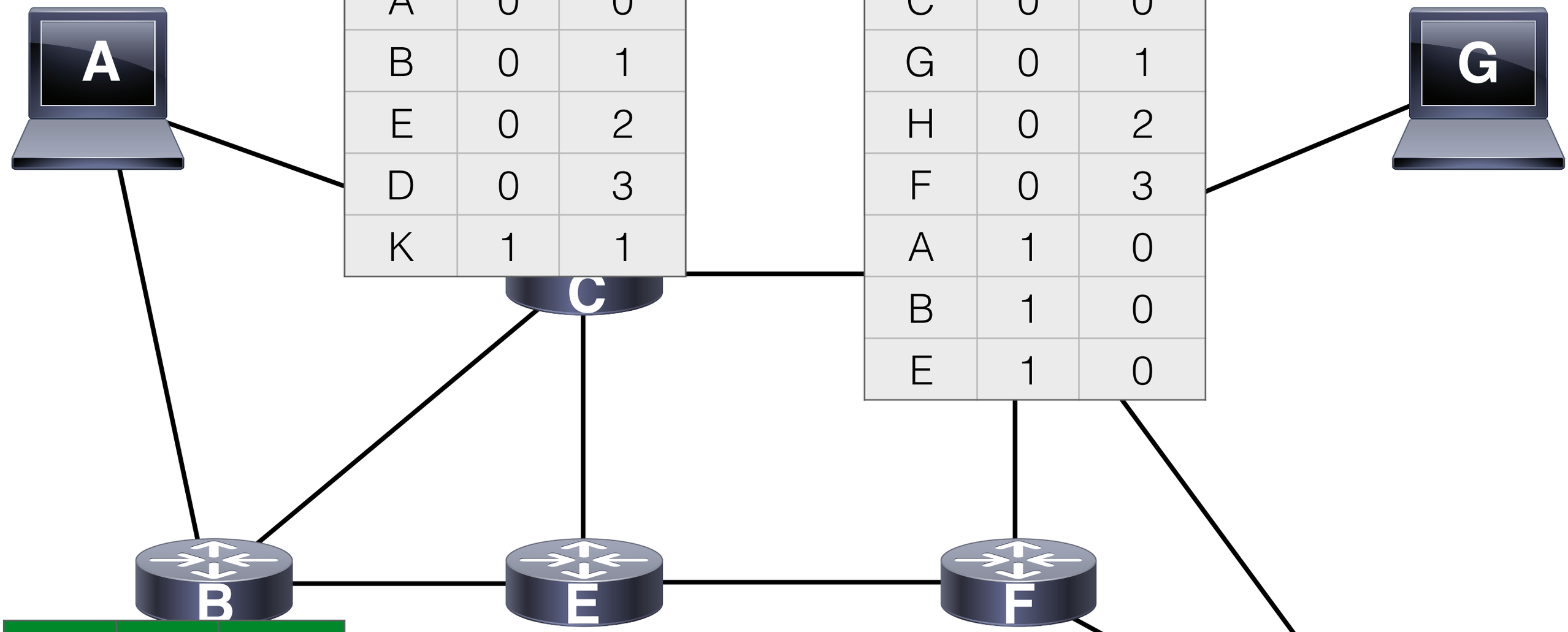


Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)



Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3
K	1	1

Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3
A	1	0
B	1	0
E	1	0

Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

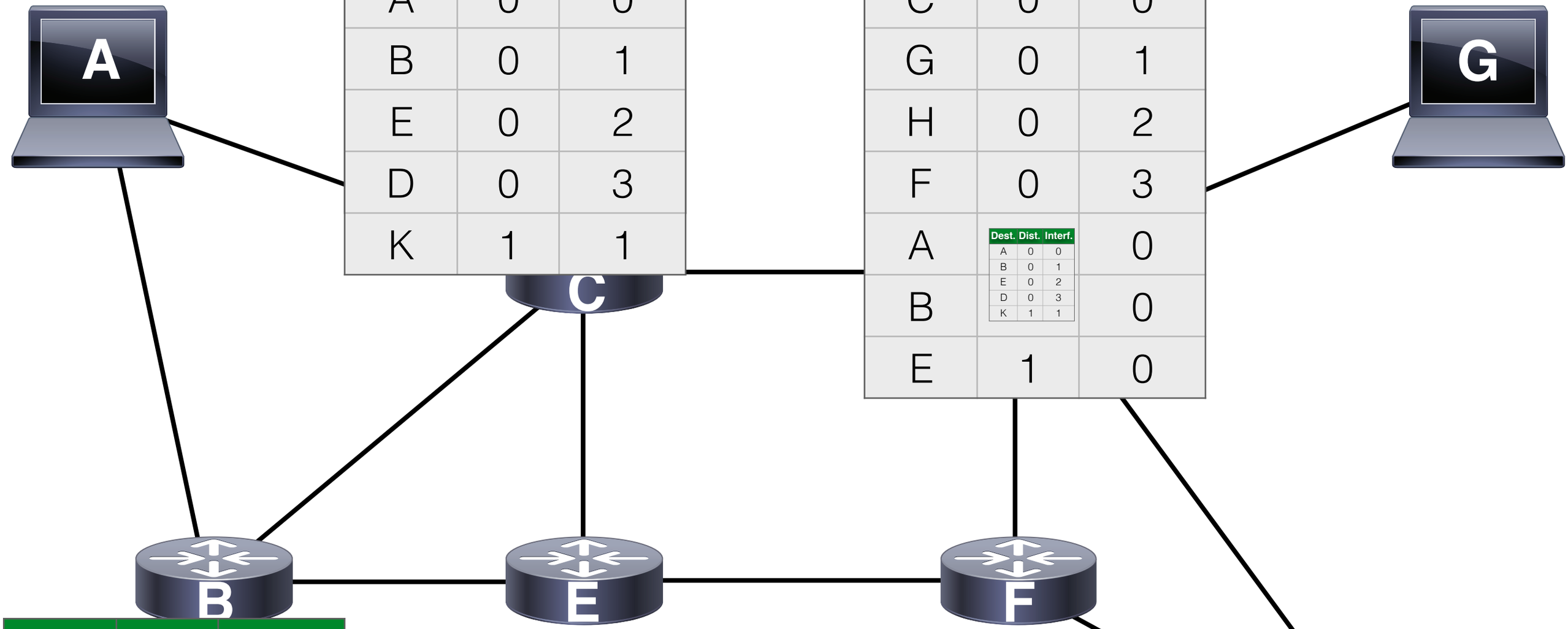
All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)

Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3
K	1	1

Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3
A	Dest. Dist. Interf. A 0 0 B 0 1 E 0 2 D 0 3 K 1 1	0
B		0
E	1	0

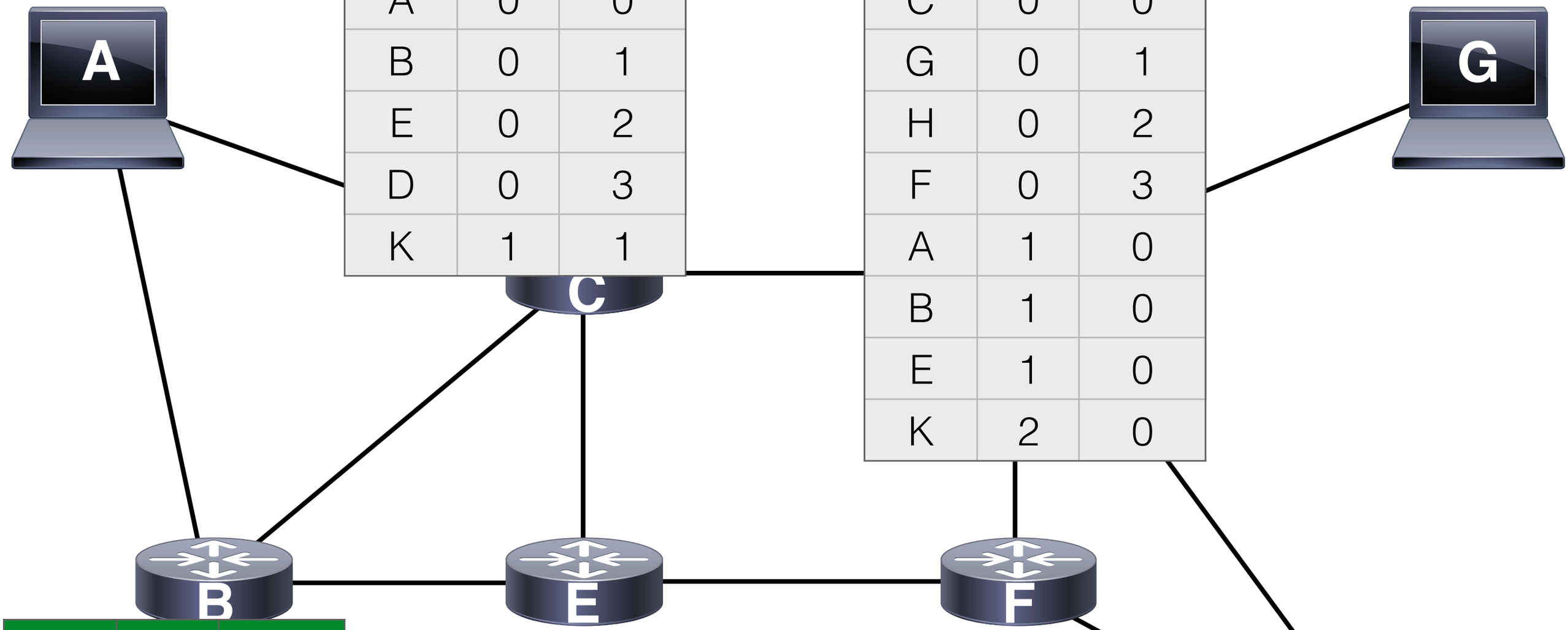


Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)



Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3
K	1	1

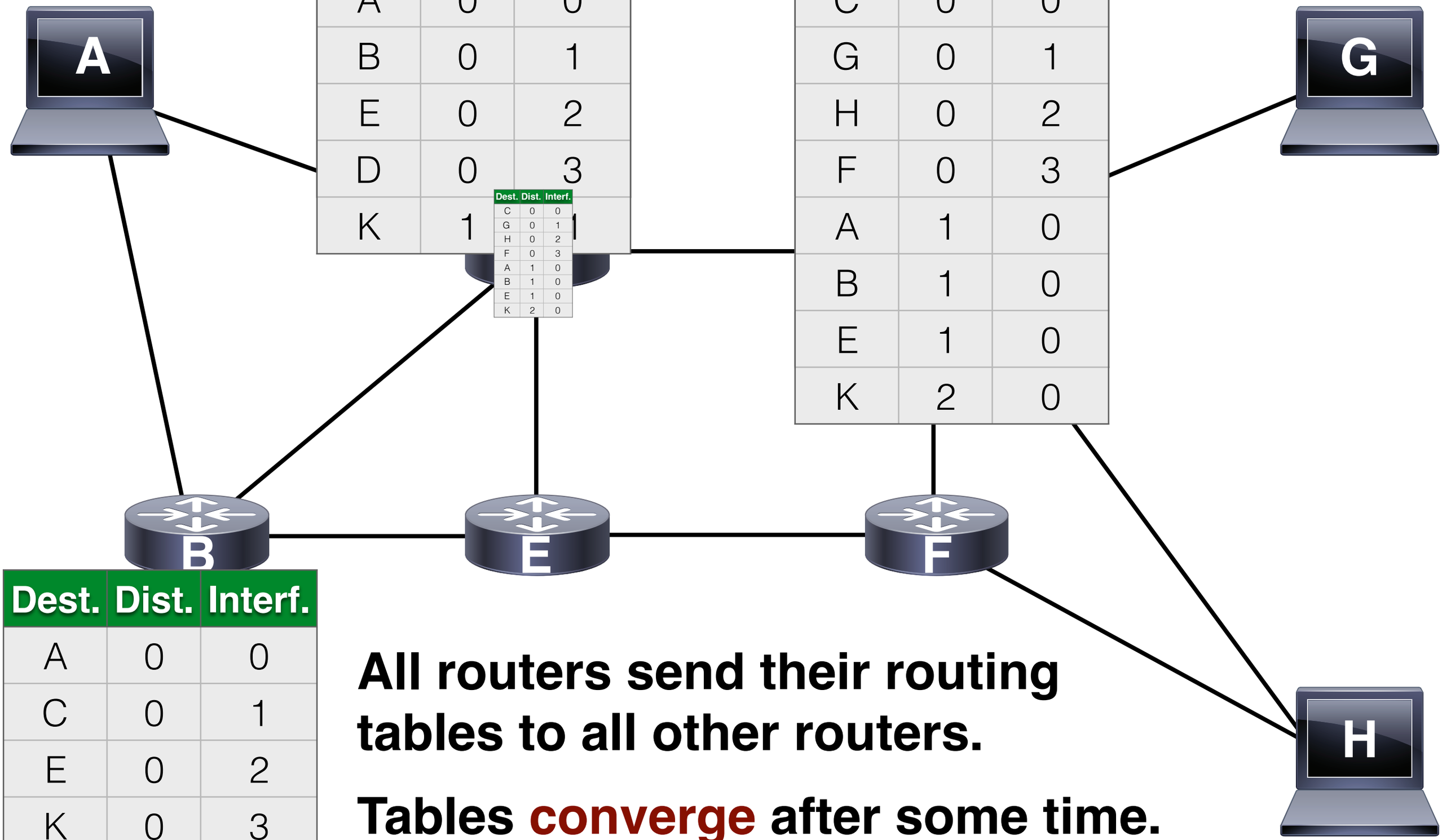
Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3
A	1	0
B	1	0
E	1	0
K	2	0

Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)



Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3
K	1	1

Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3
A	1	0
B	1	0
E	1	0
K	2	0

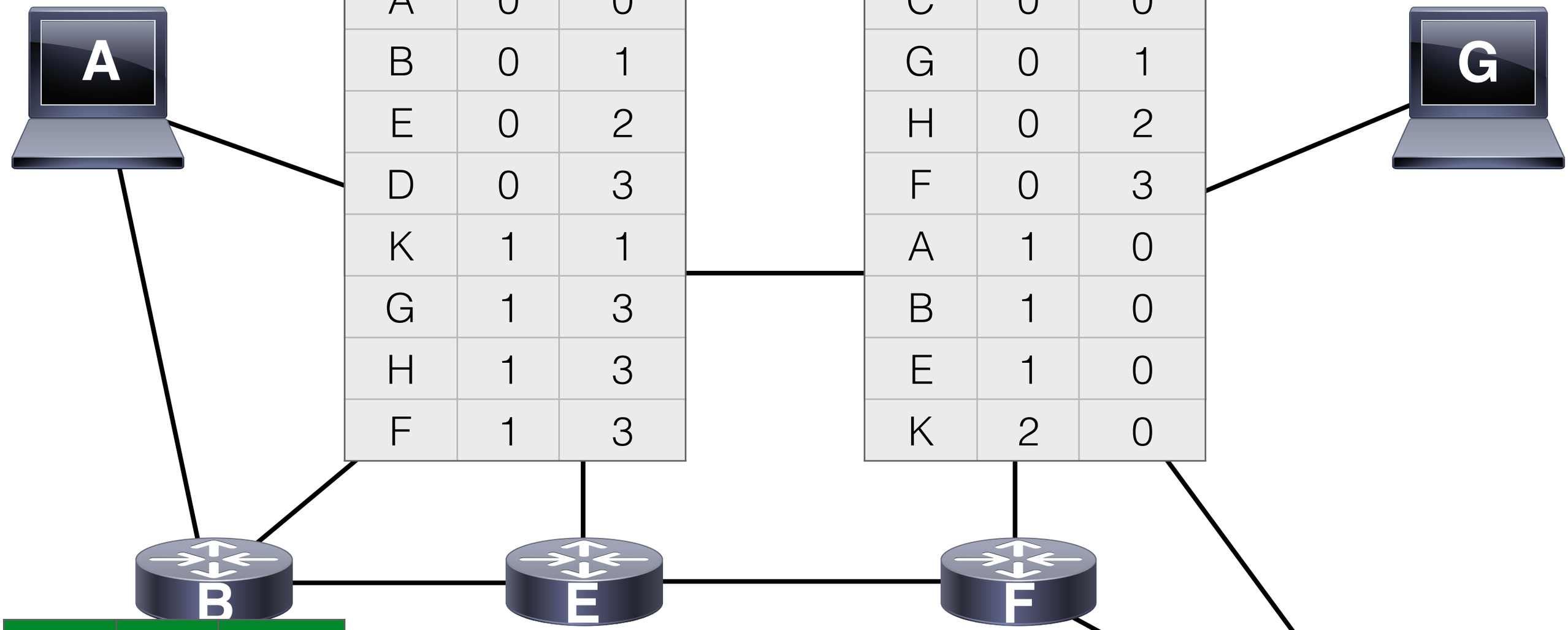
Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3
A	1	0
B	1	0
E	1	0
K	2	0

Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)



Dest.	Dist.	Interf.
A	0	0
B	0	1
E	0	2
D	0	3
K	1	1
G	1	3
H	1	3
F	1	3

Dest.	Dist.	Interf.
C	0	0
G	0	1
H	0	2
F	0	3
A	1	0
B	1	0
E	1	0
K	2	0

Dest.	Dist.	Interf.
A	0	0
C	0	1
E	0	2
K	0	3

All routers send their routing tables to all other routers.

Tables converge after some time.

Routing Information Protocol (RIP)

Distance = hop count

- Max. hop count 15
- Avoids loops

Only used in small networks

- At most 15 hops
- Updates transmit whole routing table
- Can be slow to converge

Link-state routing

Link-state routing

Routers exchange information about connectivity

- not just routing table (best routes)
- but **all** of the network it knows about
- use a **metric** (usually link speed) to describe the **quality** of each connection
- each router can **independently** compute best route to every subnet

Link-state routing

Routers exchange information about connectivity

- not just routing table (best routes)
- but **all** of the network it knows about
- use a **metric** (usually link speed) to describe the **quality** of each connection
- each router can **independently** compute best route to every subnet

Most common protocol: OSPF
(video available on Moodle in case you're interested)

Transport Layer

Transmission Control Protocol (TCP)

Connection-oriented

- A **virtual circuit** is established between two devices
- To the application it always looks like a **point-to-point full duplex** connection
- Messages split into **segments** for transmission

Transmission Control Protocol (TCP)

Connection-oriented

- A **virtual circuit** is established between two devices
- To the application it always looks like a **point-to-point full duplex** connection
- Messages split into **segments** for transmission

Reliable

- Errors are **detected** and **corrected**
- Segments are re-assembled in the **correct order**

Transmission Control Protocol (TCP)

Connection-oriented

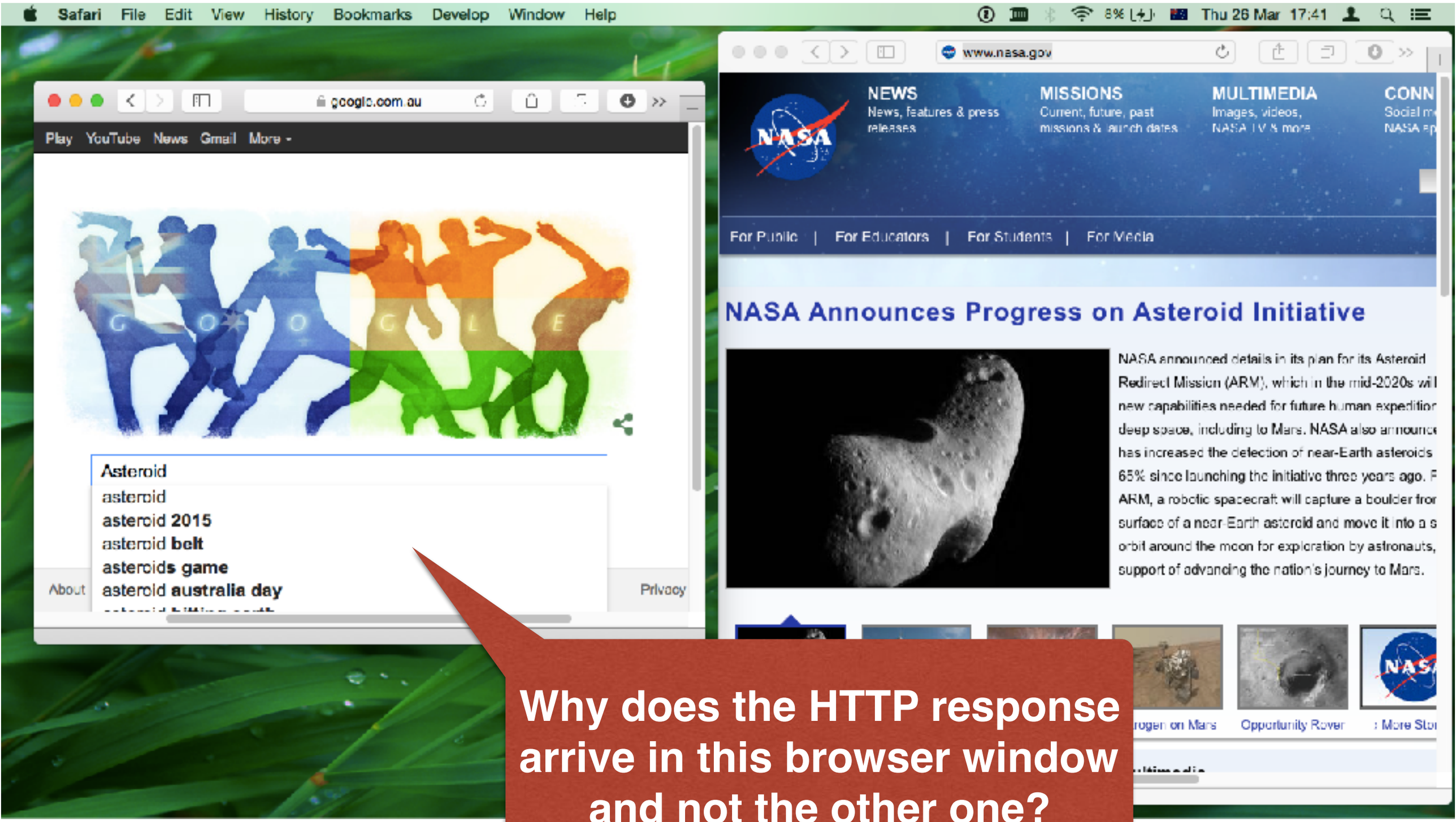
- A **virtual circuit** is established between two devices
- To the application it always looks like a **point-to-point full duplex** connection
- Messages split into **segments** for transmission

Reliable

- Errors are **detected** and **corrected**
- Segments are re-assembled in the **correct order**

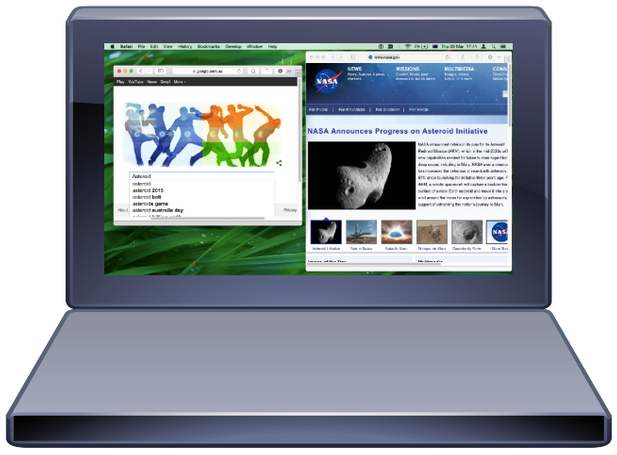
Used by e.g. HTTP, SMTP, IMAP, SSH

Addressing applications



Why does the HTTP response arrive in this browser window and not the other one?

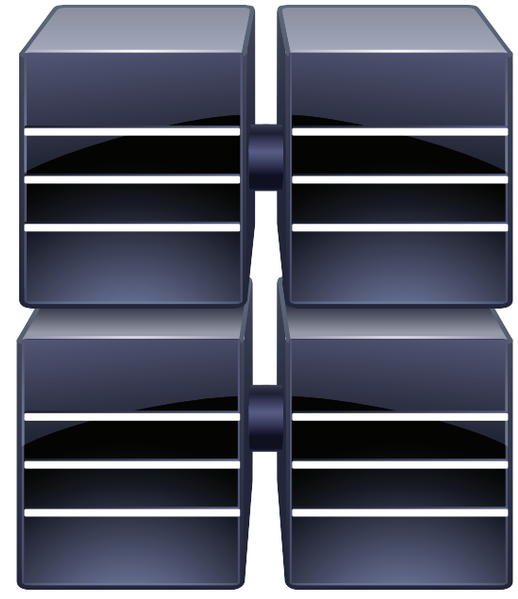
Addressing applications



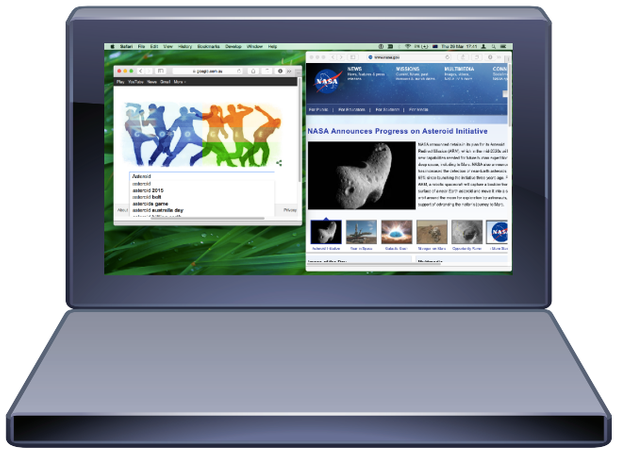
130.194.77.37

http://www.google.com.au

216.58.220.99

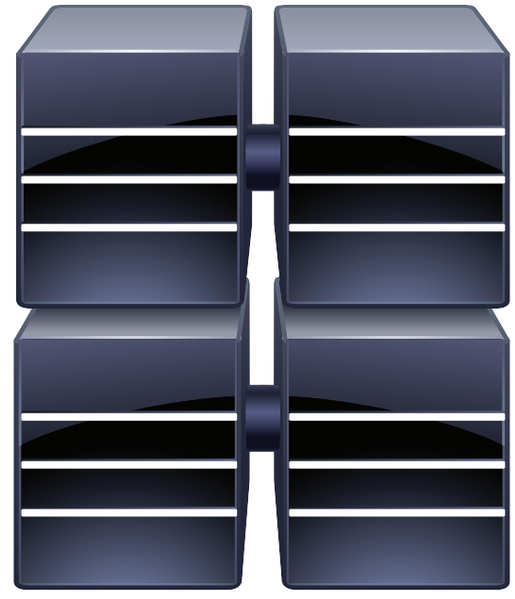


Addressing applications



http://www.google.com.au

216.58.220.99:80



130.194.77.37:57017

Addressing applications

http://www.google.com.au

216.58.220.99:80

Random client port, one per connection

130.194.77.37:57017

Addressing applications

http://www.google.com.au

216.58.220.99:80

Random client port, one per connection

130.194.77.37:57017

Fixed server port (80 = HTTP)

Addressing applications

http://www.google.com.au

216.58.220.99:80

Random client port, one per connection

Fixed server port (80 = HTTP)

130.194.77.37:57017

130.194.77.37:57018

111.119.8.38:80

http://www.nasa.gov

Addressing applications

Random client port, one per connection

http://www.google.com.au

216.58.220.99:80

Fixed server port (80 = HTTP)

130.194.77.37:57017

130.194.77.37:57018

130.194.77.37:57019

130.194.11.146:25

smtp.monash.edu

111.119.8.38:80

http://www.nasa.gov

Addressing applications

http://www.google.com.au

216.58.220.99:80

Random client port, one per connection

Fixed server port (80 = HTTP)

130.194.77.37:57017

130.194.77.37:57018

130.194.77.37:57019

130.194.11.146:25

smtp.monash.edu

Fixed server port (25 = SMTP)

111.119.8.38:80

http://www.nasa.gov

One address per layer

One address per layer

Application Layer

- URL (e.g. http://www.csse.monash.edu)

One address per layer

Application Layer

- URL (e.g. http://www.csse.monash.edu)

Transport Layer (TCP)

- Port number (e.g. 80 for HTTP)
- identifies the application that handles a message

One address per layer

Application Layer

- URL (e.g. http://www.csse.monash.edu)

Transport Layer (TCP)

- Port number (e.g. 80 for HTTP)
- identifies the application that handles a message

Network Layer (IP)

- IP address (e.g. 130.194.66.43)
- used for identifying devices across networks

One address per layer

Application Layer

- URL (e.g. http://www.csse.monash.edu)

Transport Layer (TCP)

- Port number (e.g. 80 for HTTP)
- identifies the application that handles a message

Network Layer (IP)

- IP address (e.g. 130.194.66.43)
- used for identifying devices across networks

Data Link Layer (Ethernet)

- MAC address (e.g. 00:23:ae:e7:52:85)
- used for sending frames in a LAN

TCP ARQ

Error control

- Data Link Layer **discards** frames that have errors
- Frames may not arrive at all
- But TCP should be a **reliable channel!**

TCP ARQ

Error control

- Data Link Layer **discards** frames that have errors
- Frames may not arrive at all
- But TCP should be a **reliable channel!**

Solution: Automatic **R**epeat Re**Q**uest

- Exchange **acknowledgements (ACK)**, letting sender know that packets were received correctly
- Sender re-transmits if no ACK within certain time

TCP ARQ

When we send a TCP packet, it includes two numbers.

Sequence number:

- how many bytes we've already transmitted (before this one)

Acknowledgement number:

- how many bytes we've received from the other side

Sender can therefore check how many bytes have been received correctly!

Establishing a connection

Three-way handshake:

Establishing a connection

Three-way handshake:

- Client sends a **SYN** packet with **random sequence number *A***

Establishing a connection

Three-way handshake:

- Client sends a **SYN** packet with **random sequence number *A***
- Server replies with **SYN, ACK, acknowledgement number *A+1,*** and **random sequence number *B***

Establishing a connection

Three-way handshake:

- Client sends a **SYN** packet with **random sequence number *A***
- Server replies with **SYN, ACK, acknowledgement number *A+1,*** and **random sequence number *B***
- Client sends **ACK** with **sequence number *A+1*** and **acknowledgement number *B+1***

Closing a connection

Four-way handshake:

Closing a connection

Four-way handshake:

- Computer A (client or server!) sends a **FIN** packet

Closing a connection

Four-way handshake:

- Computer A (client or server!) sends a **FIN** packet
- Computer B acknowledges with an **ACK**

Closing a connection

Four-way handshake:

- Computer A (client or server!) sends a **FIN** packet
- Computer B acknowledges with an **ACK**
- Computer B sends a **FIN** packet

Closing a connection

Four-way handshake:

- Computer A (client or server!) sends a **FIN** packet
- Computer B acknowledges with an **ACK**
- Computer B sends a **FIN** packet
- Computer A acknowledges with an **ACK**

Closing a connection

Four-way handshake:

- Computer A (client or server!) sends a **FIN** packet
- Computer B acknowledges with an **ACK**
- Computer B sends a **FIN** packet
- Computer A acknowledges with an **ACK**
- **Can be simplified to three-way** (combining a FIN/ACK)

Closing a connection

Four-way handshake:

- Computer A (client or server!) sends a **FIN** packet
- Computer B acknowledges with an **ACK**
- Computer B sends a **FIN** packet
- Computer A acknowledges with an **ACK**
- **Can be simplified to three-way** (combining a FIN/ACK)

Necessary because TCP is full duplex!

TCP session

Client

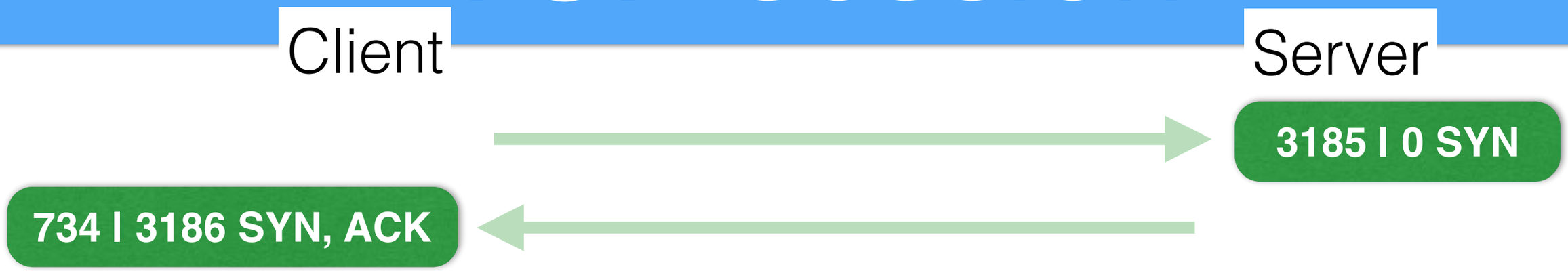
Server

TCP session

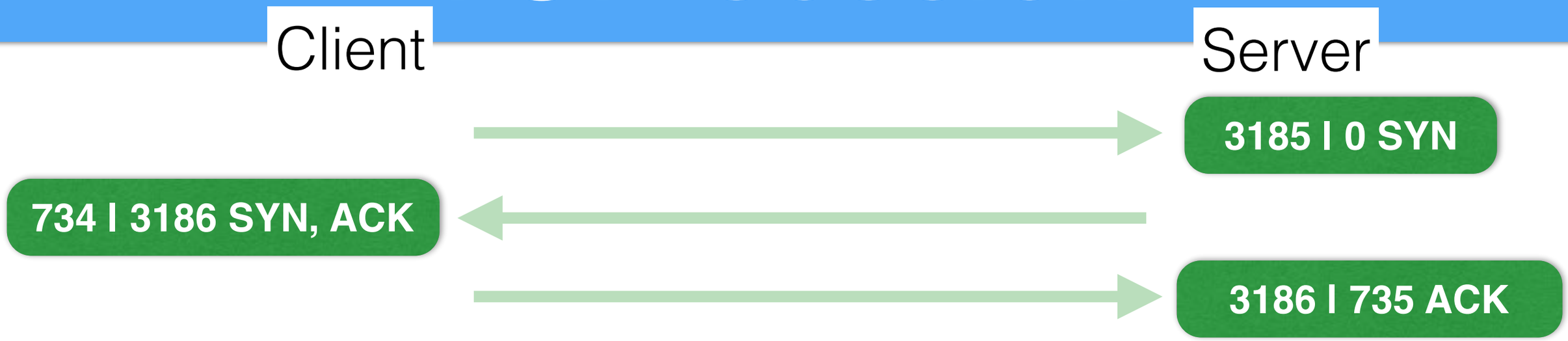
Client

Server

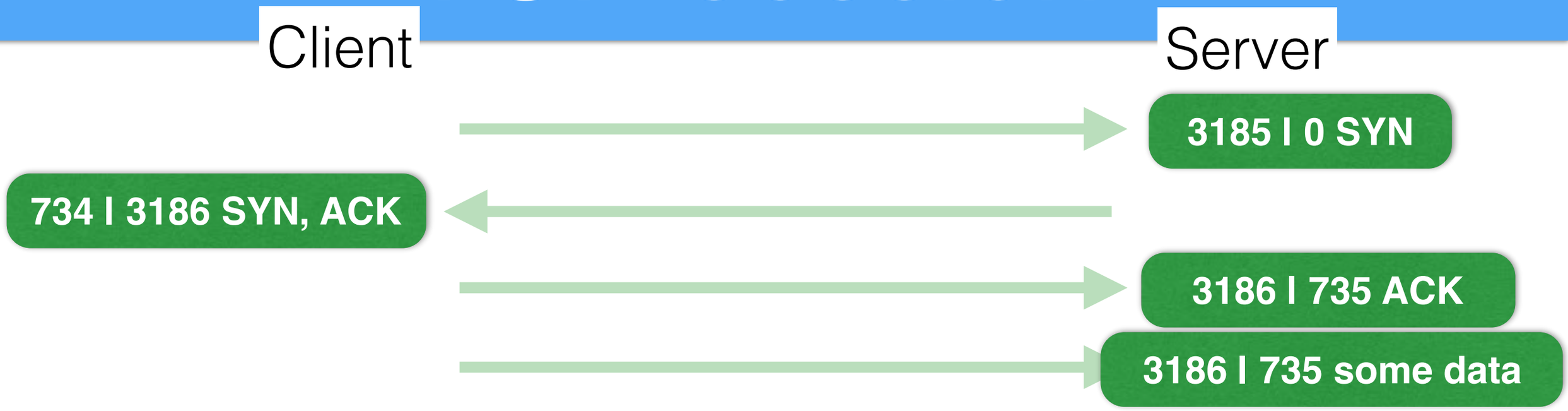
TCP session



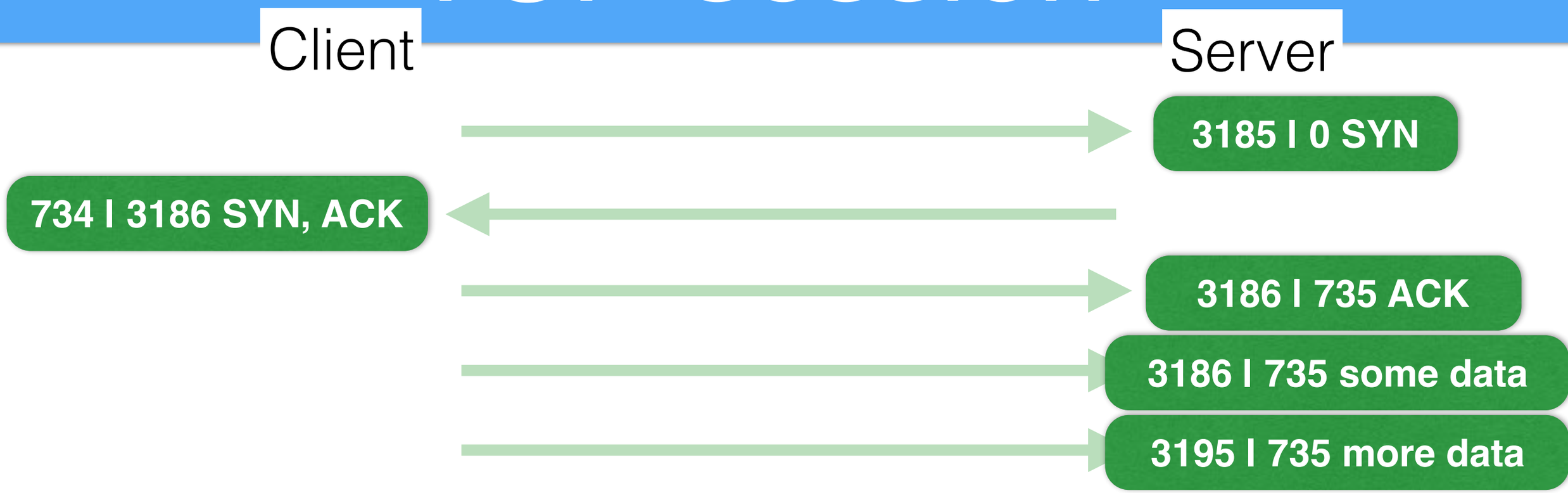
TCP session



TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data

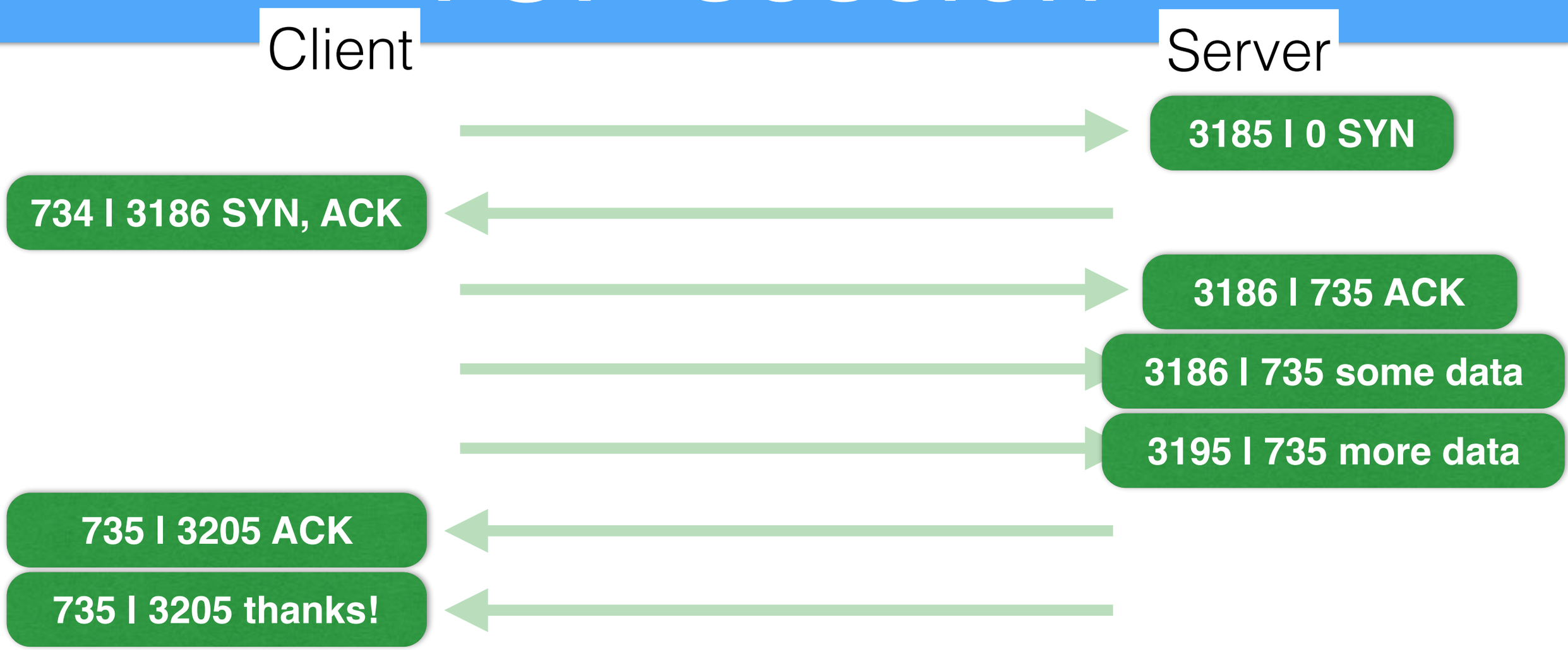
TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data

TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK

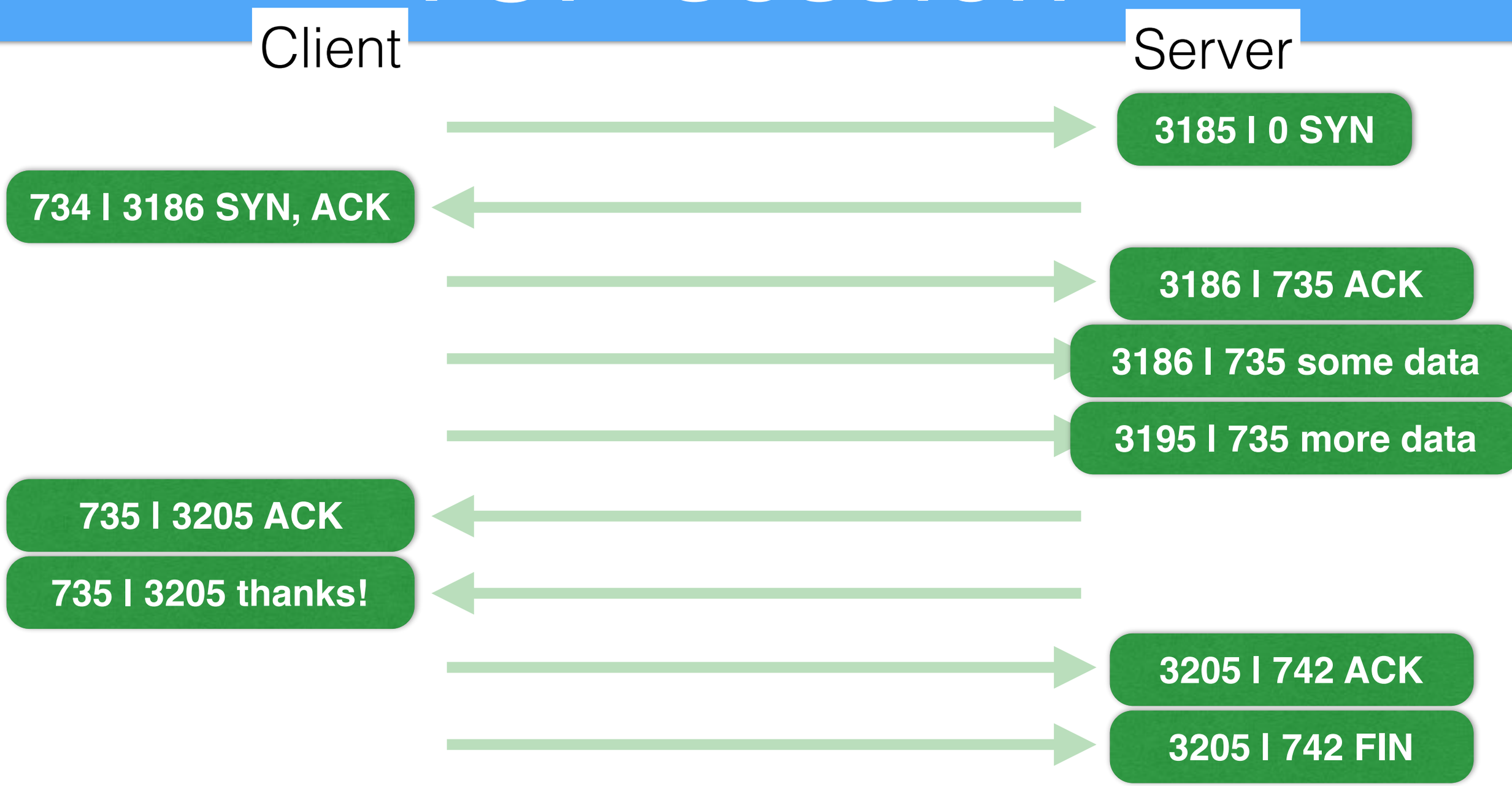
TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK
735 I 3205 thanks!

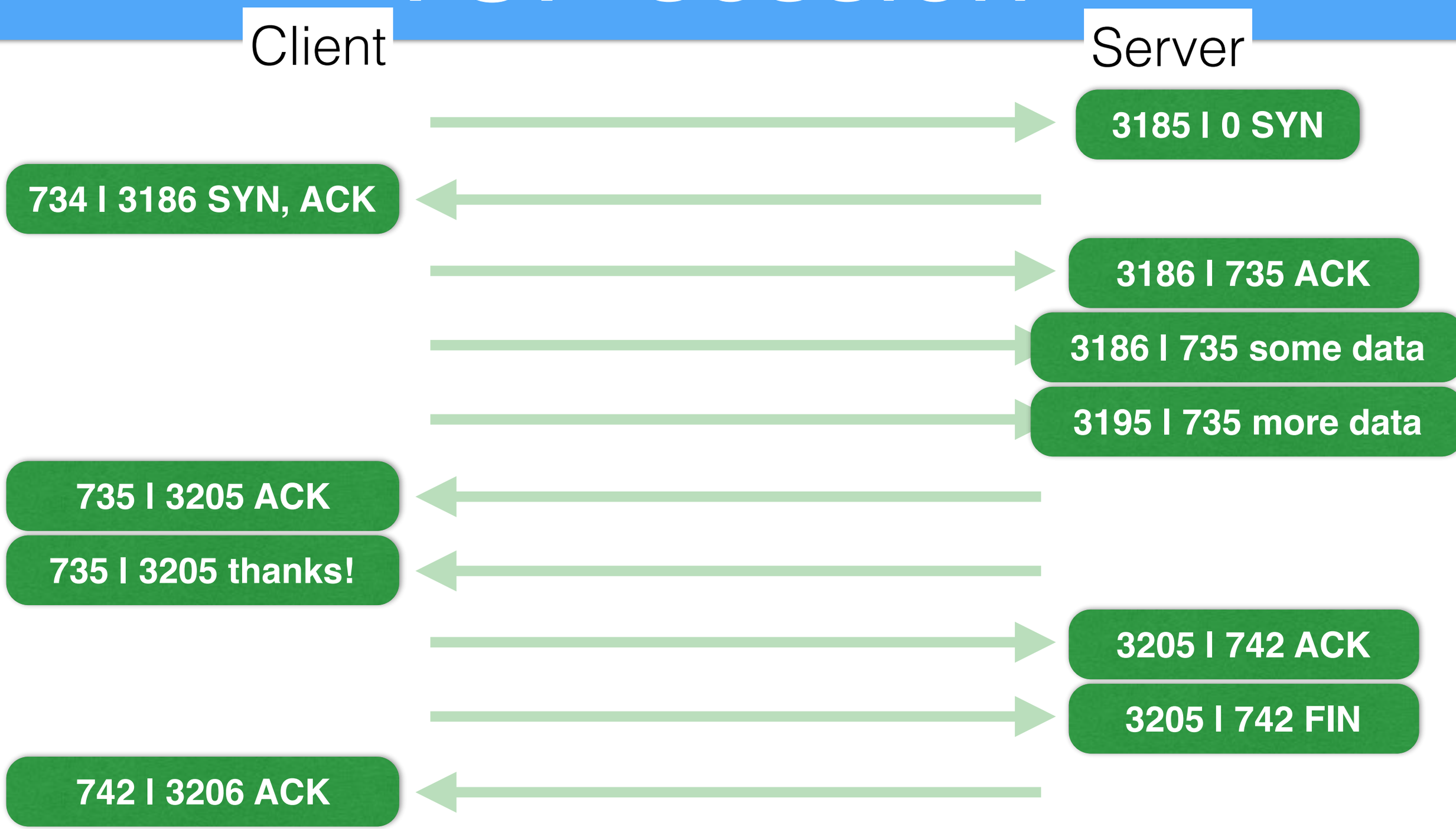
TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK
735 I 3205 thanks!
3205 I 742 ACK

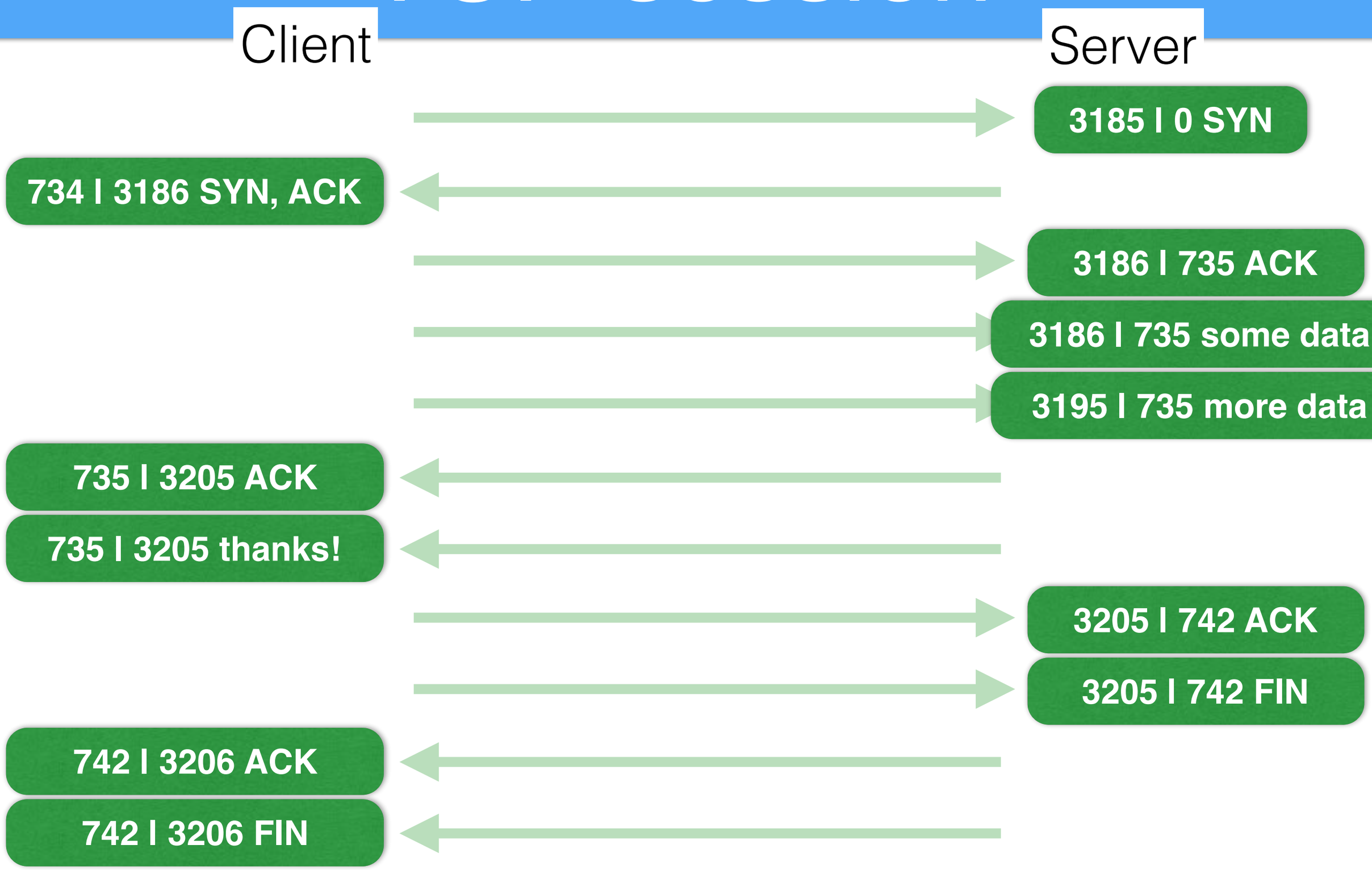
TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK
735 I 3205 thanks!
3205 I 742 ACK
3205 I 742 FIN

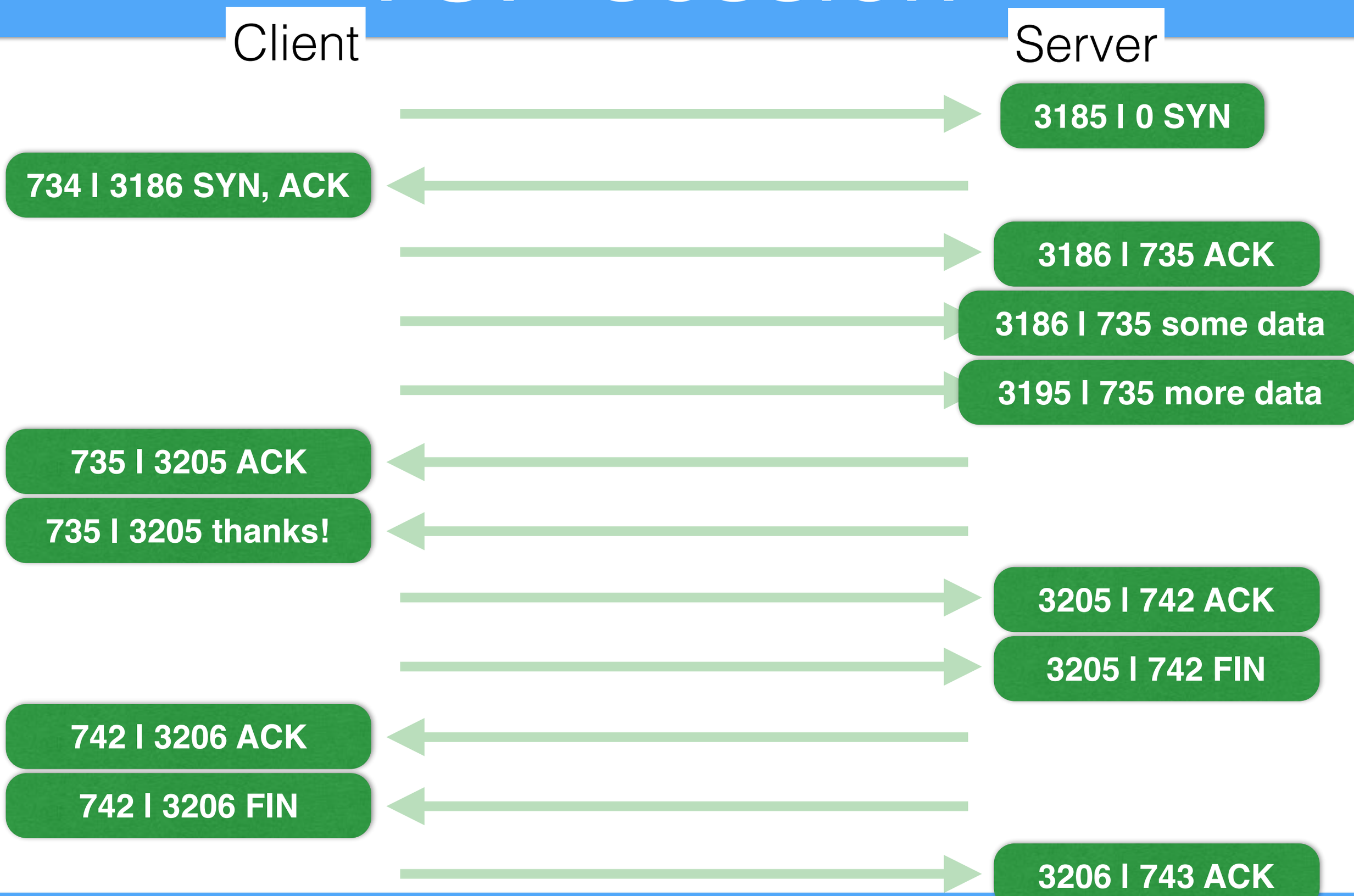
TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK
735 I 3205 thanks!
3205 I 742 ACK
3205 I 742 FIN
742 I 3206 ACK

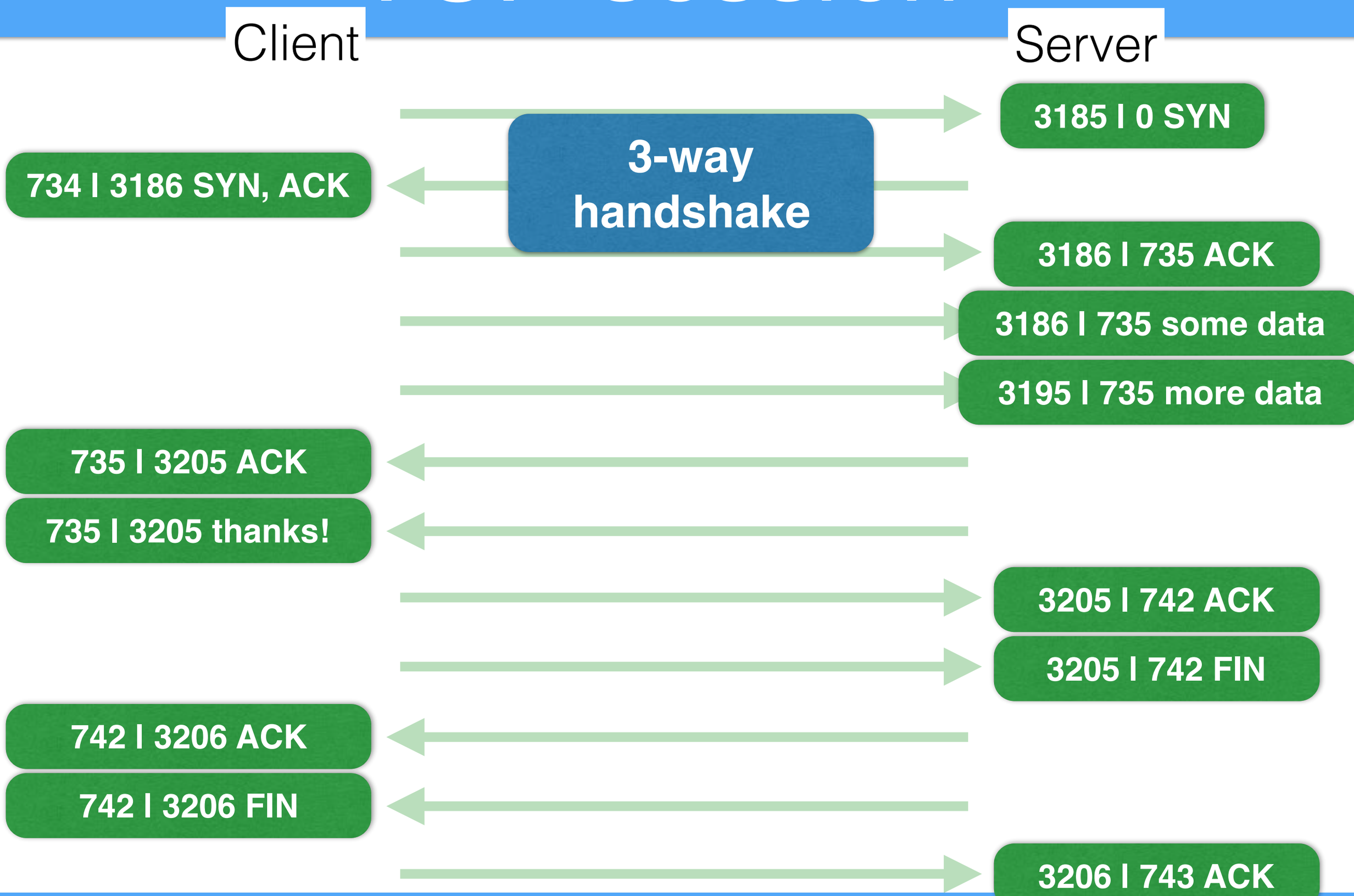
TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK
735 I 3205 thanks!
3205 I 742 ACK
3205 I 742 FIN
742 I 3206 ACK
742 I 3206 FIN

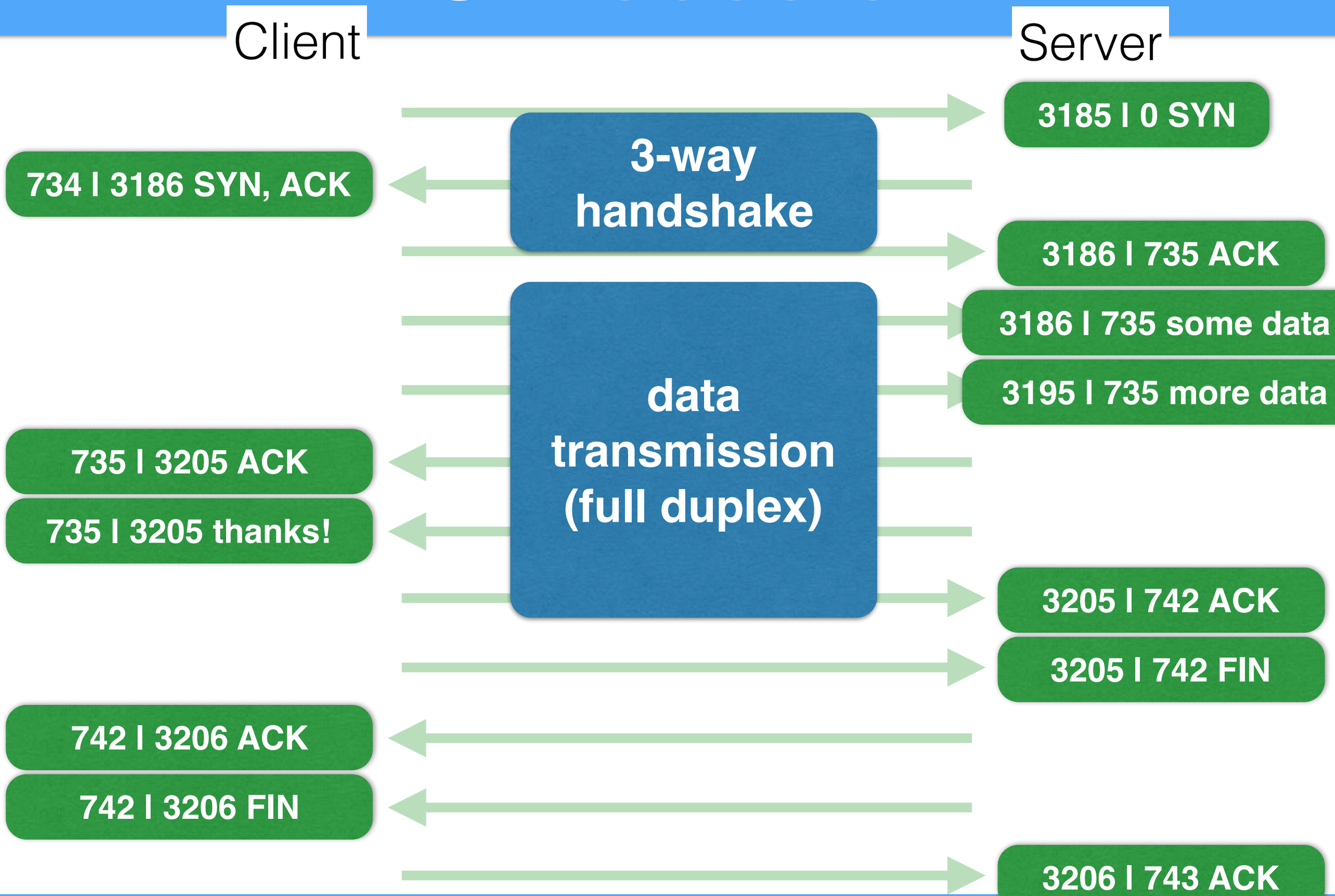
TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK
735 I 3205 thanks!
3205 I 742 ACK
3205 I 742 FIN
742 I 3206 ACK
742 I 3206 FIN
3206 I 743 ACK

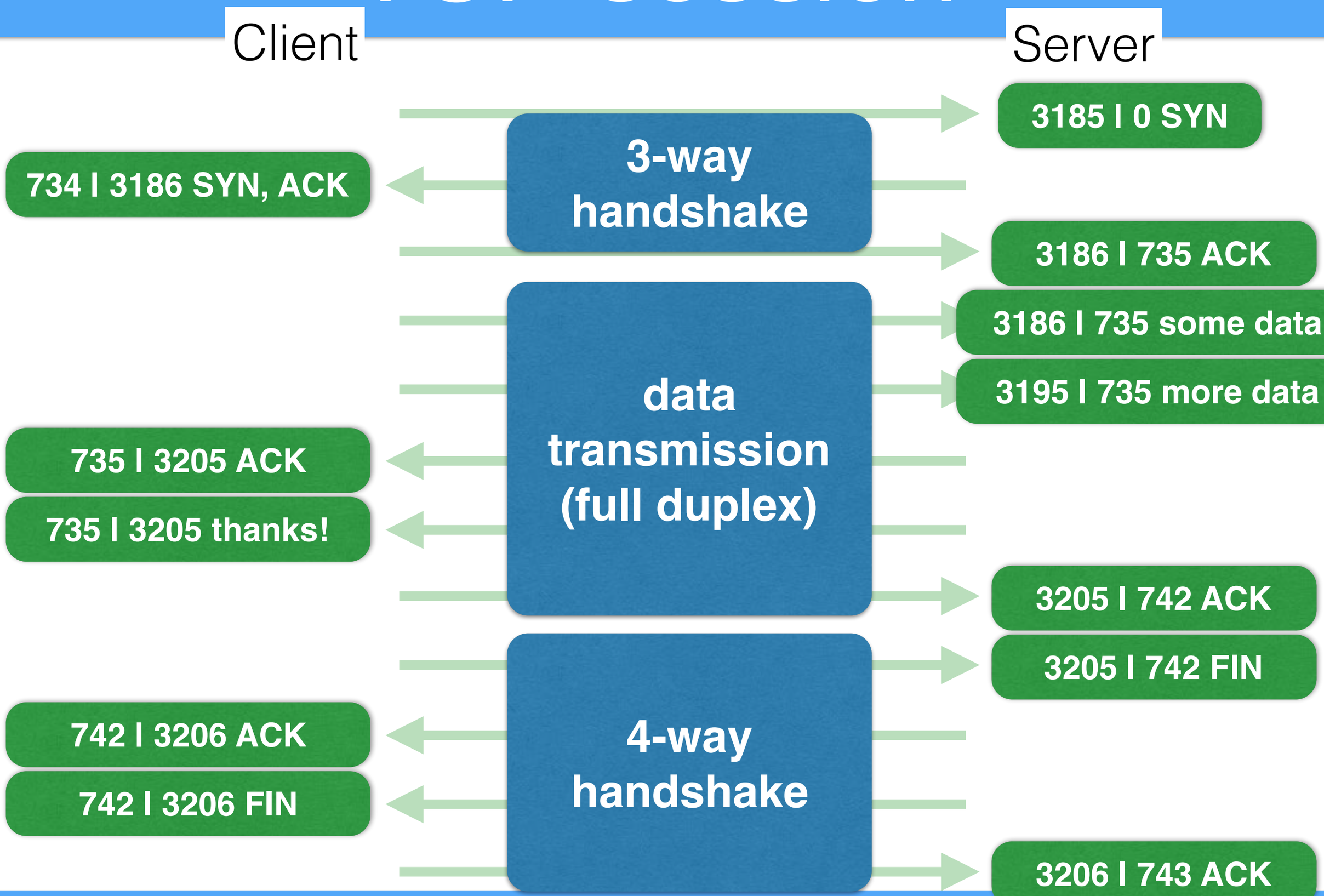
TCP session


Client
Server
3185 I 0 SYN
3-way handshake
734 I 3186 SYN, ACK
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK
735 I 3205 thanks!
3205 I 742 ACK
3205 I 742 FIN
742 I 3206 ACK
742 I 3206 FIN
3206 I 743 ACK

TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3-way handshake
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
data transmission (full duplex)
735 I 3205 ACK
735 I 3205 thanks!
3205 I 742 ACK
3205 I 742 FIN
742 I 3206 ACK
742 I 3206 FIN
3206 I 743 ACK

TCP session


Client
Server
3185 I 0 SYN
734 I 3186 SYN, ACK
3-way handshake
3186 I 735 ACK
3186 I 735 some data
3195 I 735 more data
735 I 3205 ACK
735 I 3205 thanks!
data transmission (full duplex)
3205 I 742 ACK
3205 I 742 FIN
742 I 3206 ACK
742 I 3206 FIN
4-way handshake
3206 I 743 ACK

TCP parameters

TCP implements segmentation

- large application layer messages are split into segments

TCP parameters

TCP implements segmentation

- large application layer messages are split into segments

How fast to send segments?

- Sending too many at once may overload receiver or intermediate path with lower bandwidth

TCP parameters

TCP implements segmentation

- large application layer messages are split into segments

How fast to send segments?

- Sending too many at once may overload receiver or intermediate path with lower bandwidth

How to decide the segment size?

- Sending too large segments requires IP to **fragment**
- Large segments also increase errors

TCP segment size

Two approaches:

- use “reasonable” MTU, accept that IP sometimes needs to fragment

TCP segment size

Two approaches:

- use “reasonable” MTU, accept that IP sometimes needs to fragment
- use **Path MTU Discovery** (PMTUD):

TCP segment size

Two approaches:

- use “reasonable” MTU, accept that IP sometimes needs to fragment
- use **Path MTU Discovery** (PMTUD):
  - send IP packets to destination, asking routers to **never** fragment

TCP segment size

Two approaches:

- use “reasonable” MTU, accept that IP sometimes needs to fragment
- use **Path MTU Discovery** (PMTUD):
  - send IP packets to destination, asking routers to **never** fragment
  - if router would have to fragment, it sends back an error message

TCP segment size

Two approaches:

- use “reasonable” MTU, accept that IP sometimes needs to fragment
- use **Path MTU Discovery** (PMTUD):
  - send IP packets to destination, asking routers to **never** fragment
  - if router would have to fragment, it sends back an error message
  - increase packet length until error occurs, then use last known error-free MTU

TCP congestion control

How fast to send?

- Receiver transmits its maximum buffer size
- Sender sends segments without waiting for ACK up to buffer size

TCP congestion control

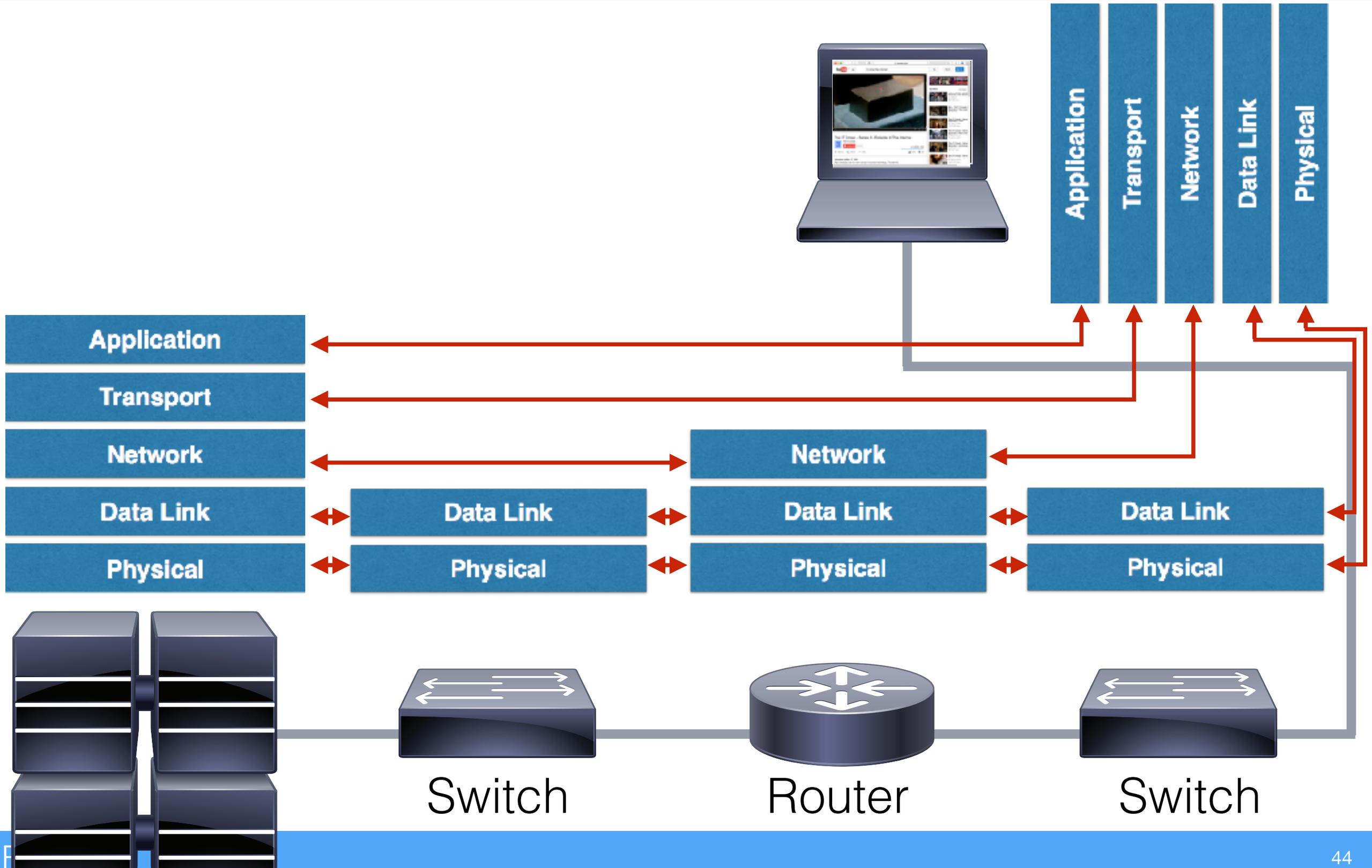
How fast to send?

- Receiver transmits its maximum buffer size
- Sender sends segments without waiting for ACK up to buffer size

What if network cannot cope?

- Start slow: wait for ACK after each segment
- Increase with every ACK: send two, four etc segments after each ACK
- Fall back to slower speed when no ACK arrives

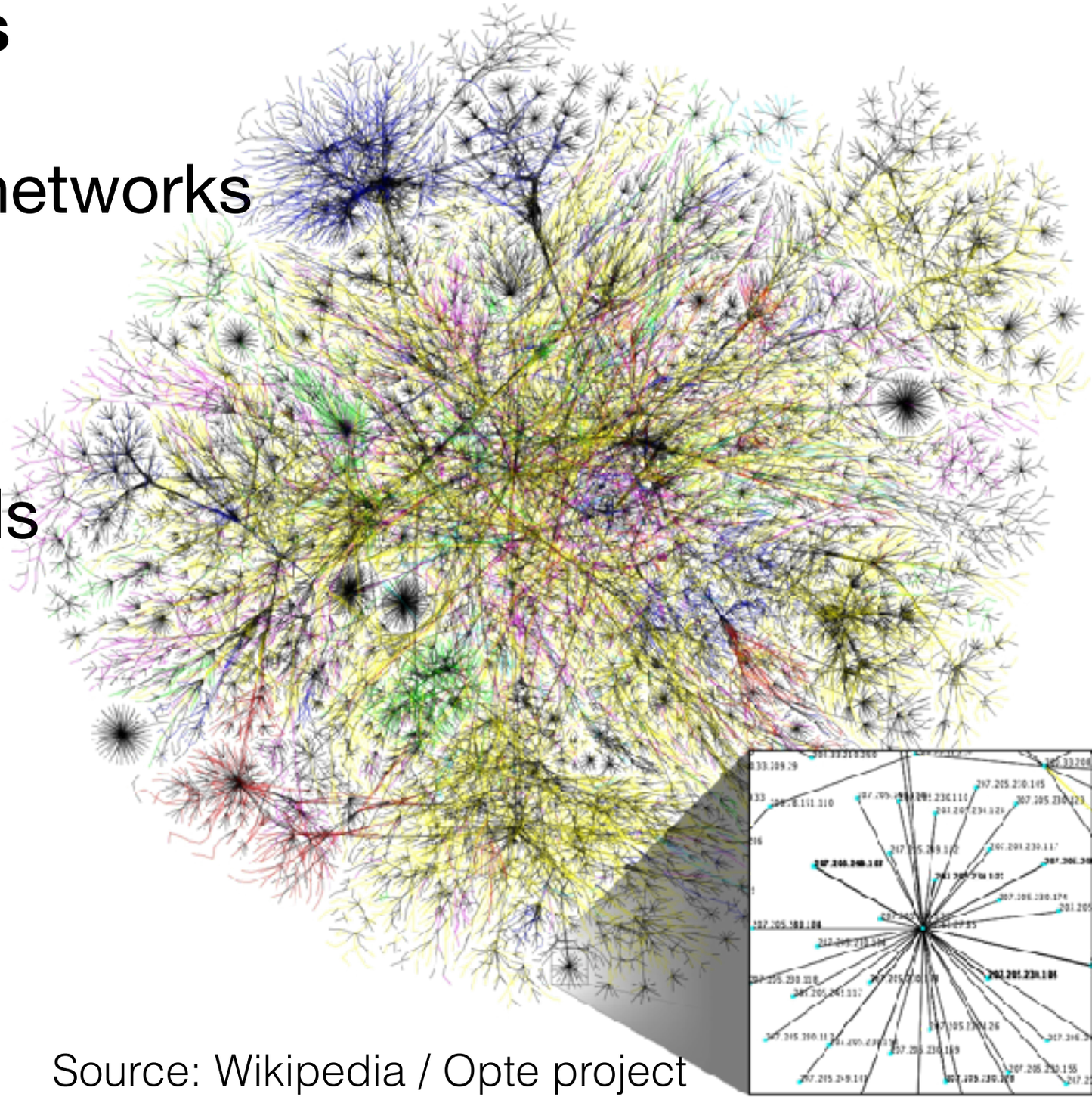
Layers of Abstraction



The Internet

The Internet

- A **network of networks**
- Connecting millions of networks and billions of devices
- Based on a common, standard set of protocols



Source: Wikipedia / Opte project

Autonomous Systems

Networks operated by a single organisation

- e.g. Monash University's or your ISP's network

Autonomous Systems

Networks operated by a single organisation

- e.g. Monash University's or your ISP's network

Interior routing

- for routing packets **within** an AS
- uses RIP, OSPF, EIGRP

Autonomous Systems

Networks operated by a single organisation

- e.g. Monash University's or your ISP's network

Interior routing

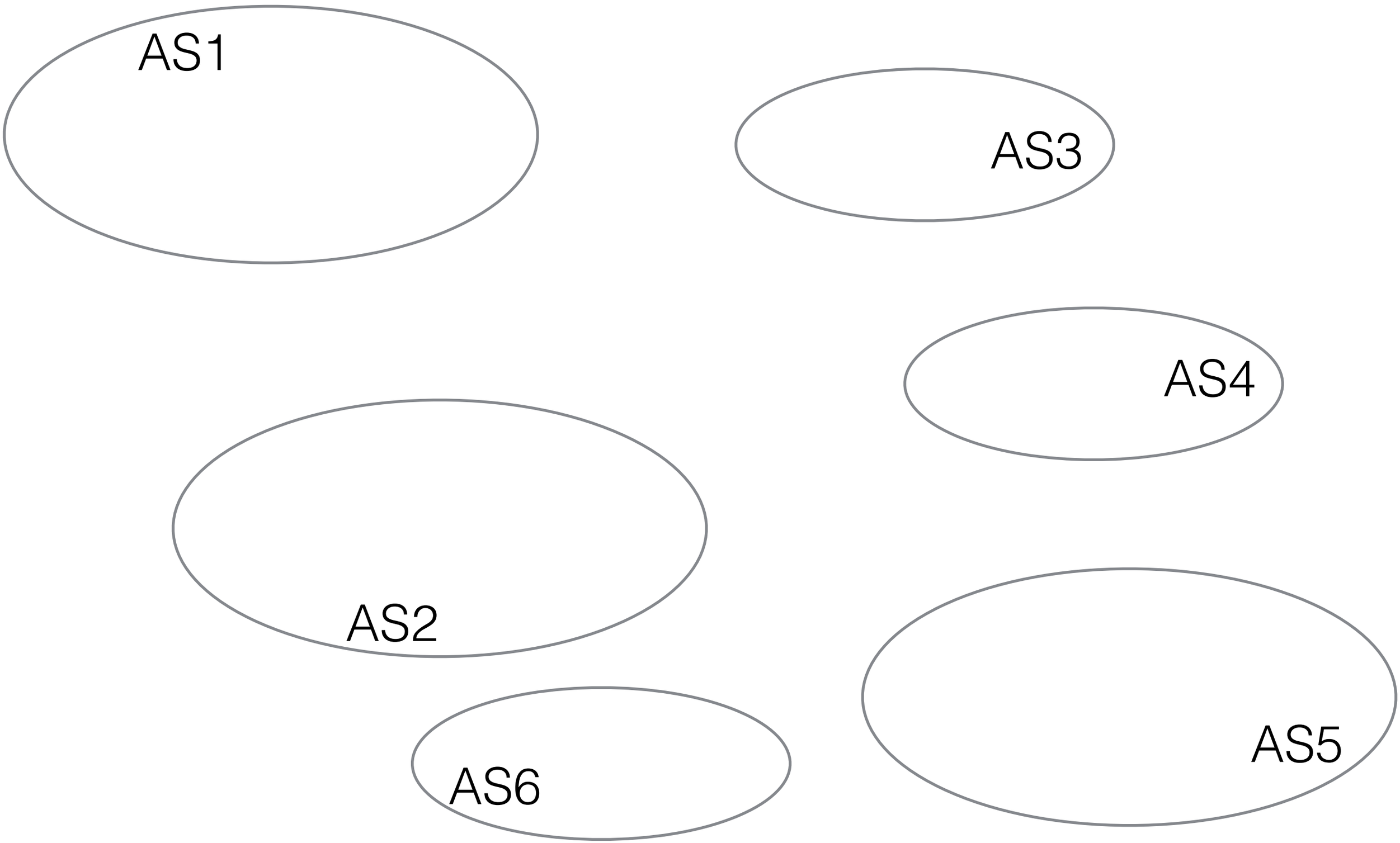
- for routing packets **within** an AS
- uses RIP, OSPF, EIGRP

Exterior routing

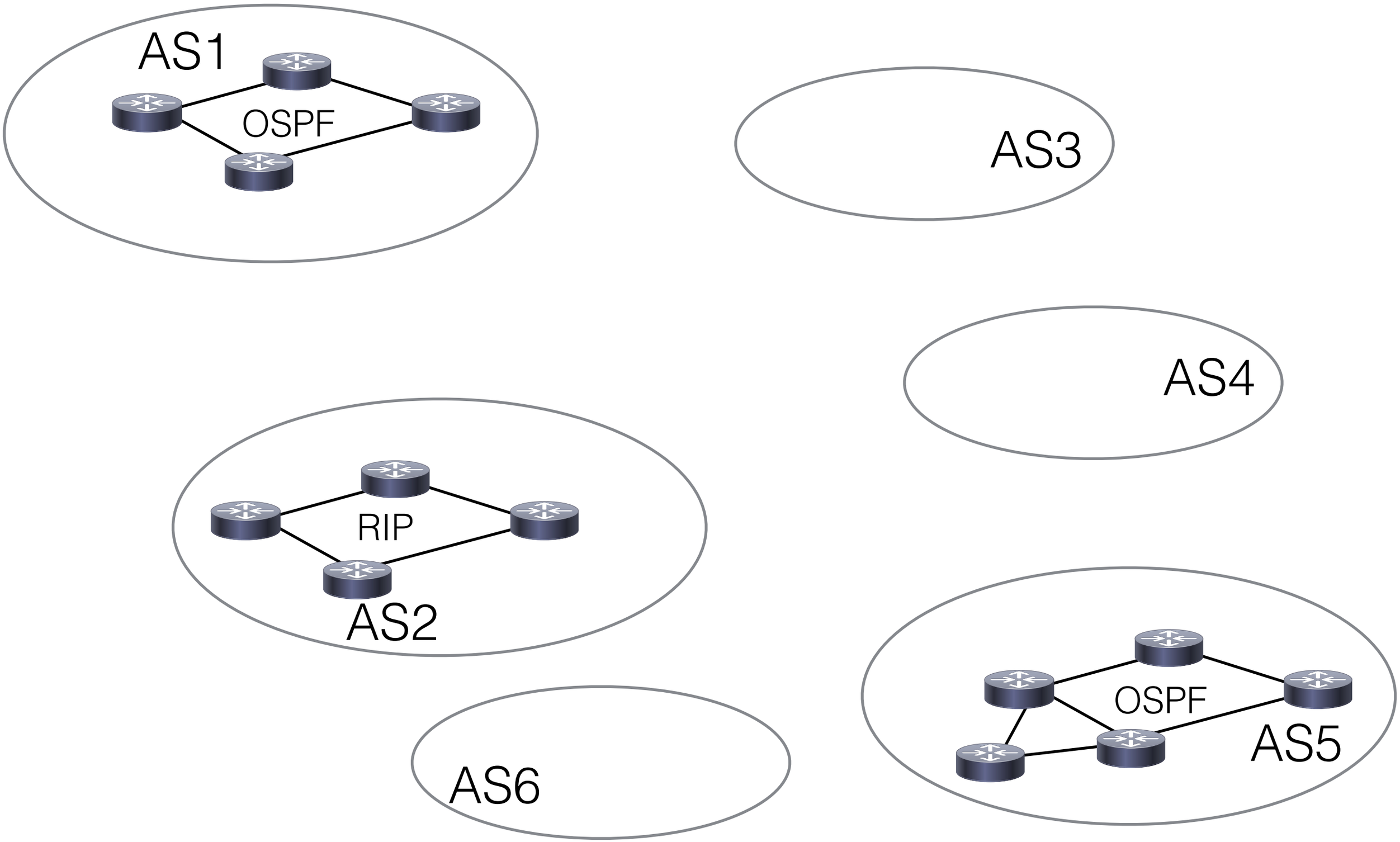
- for routing packets **between** different AS
- Internet uses BGP (Border Gateway Protocol)

Internet Architecture

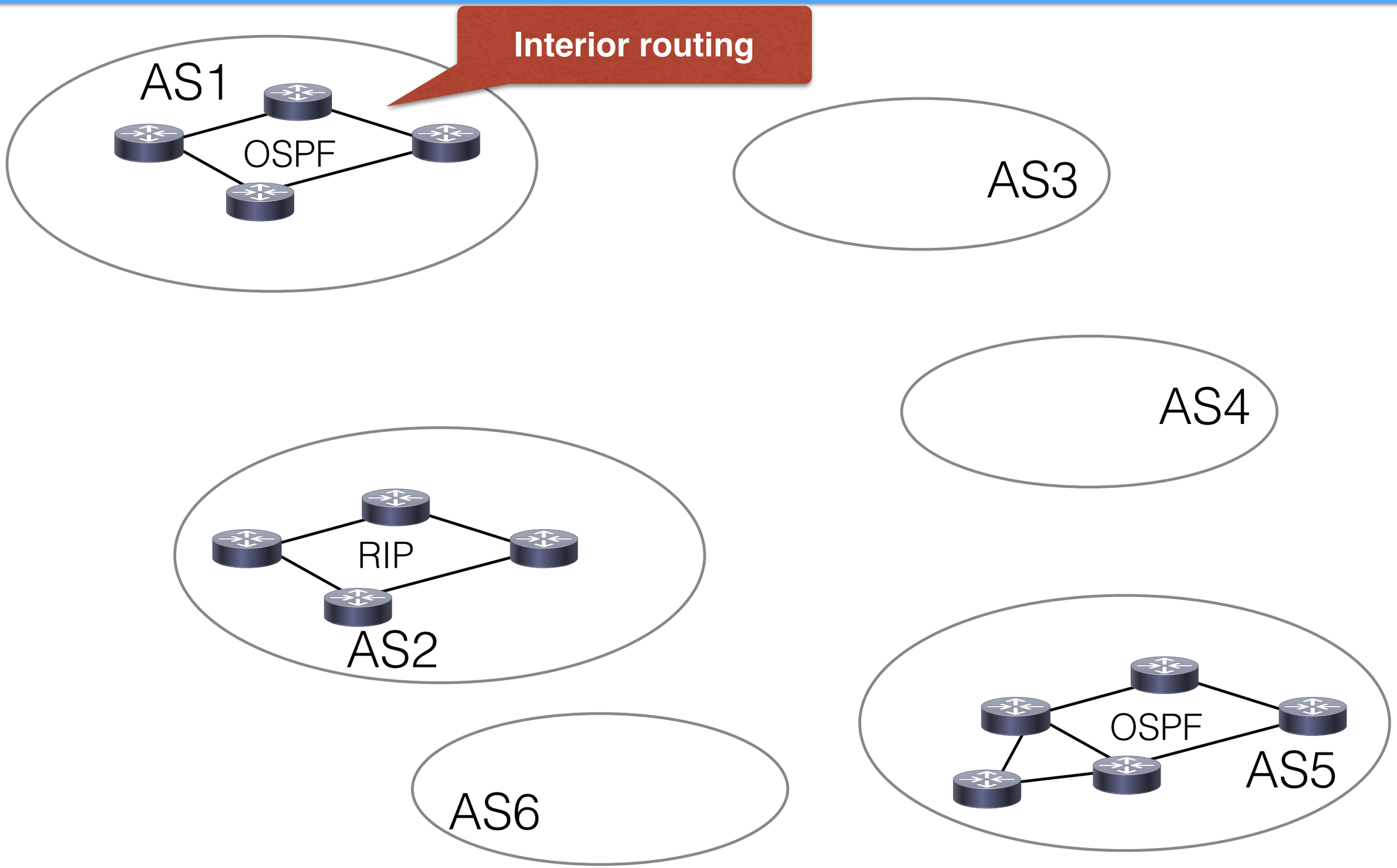
Internet Architecture



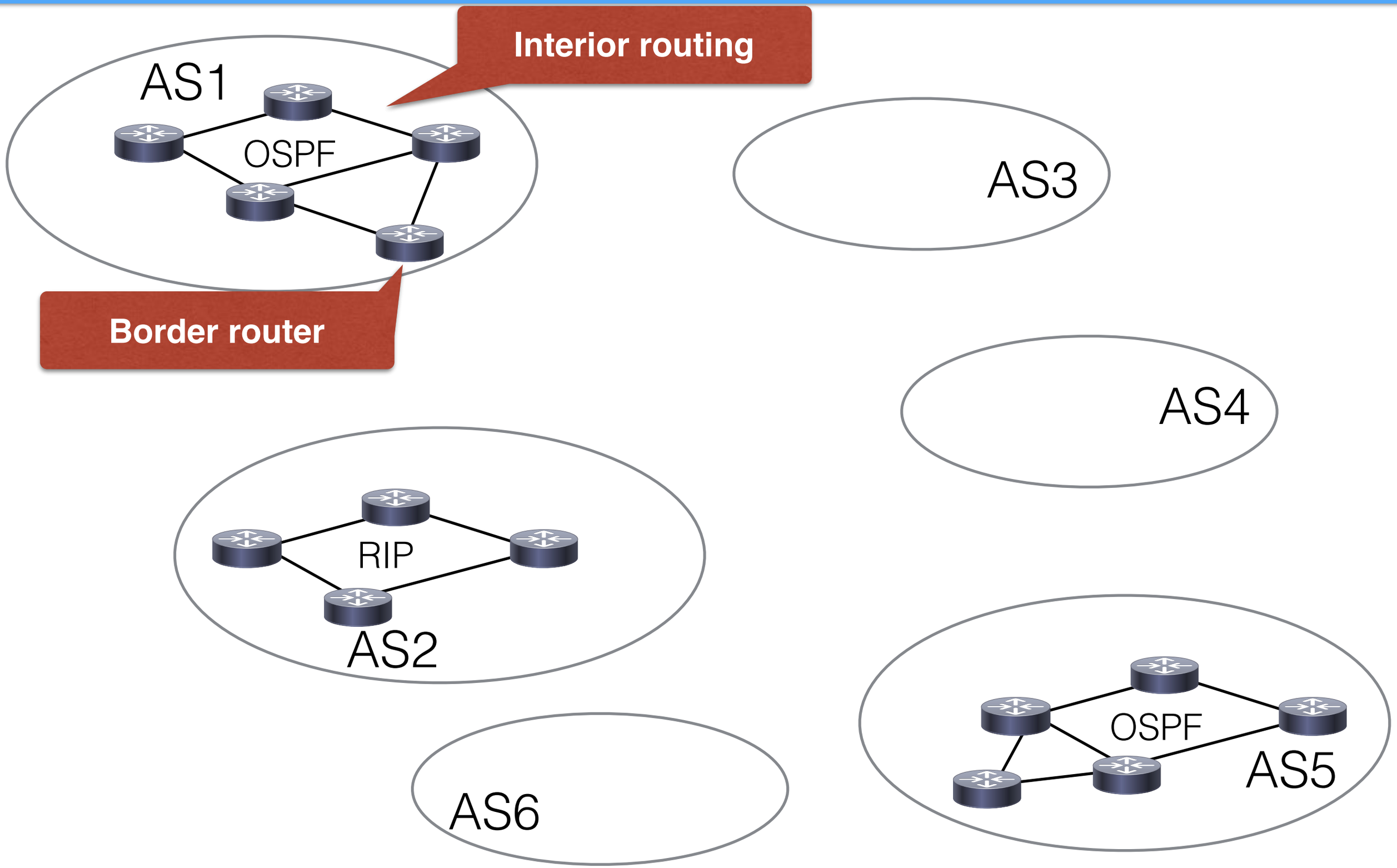
Internet Architecture



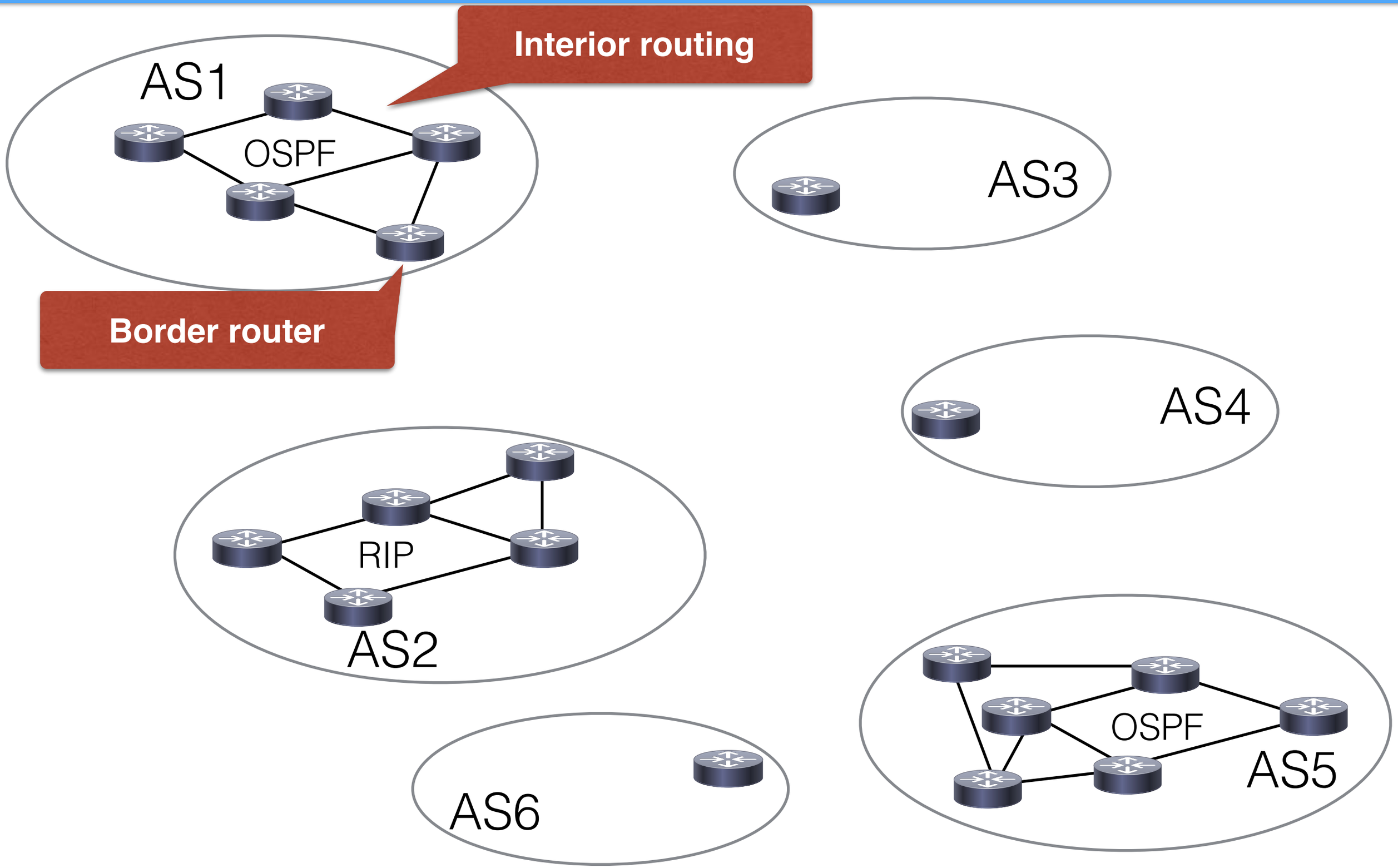
Internet Architecture



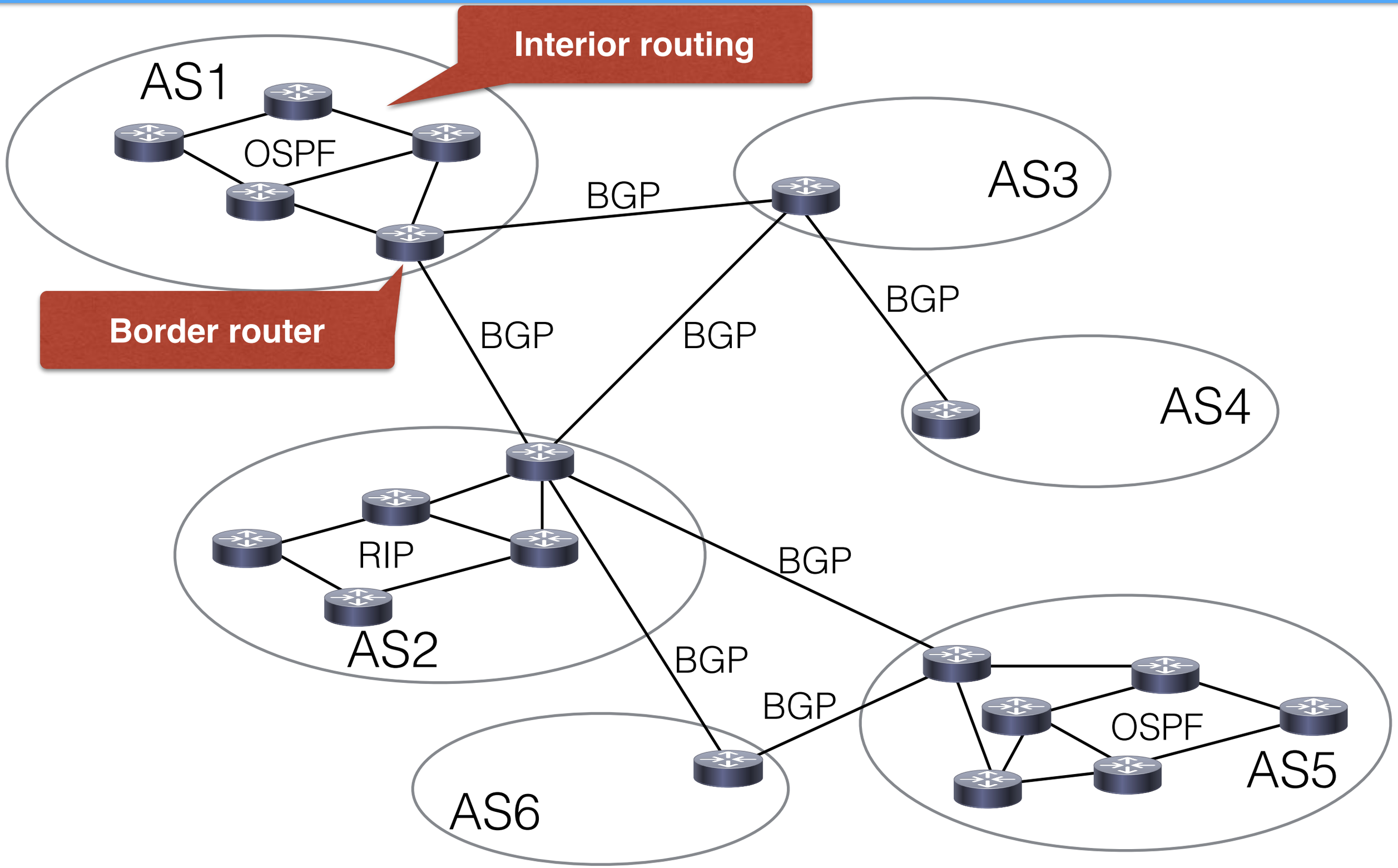
Internet Architecture



Internet Architecture



Internet Architecture



Internet Structure

Internet Structure

Internet Service Providers

- Commercial companies
- You connect via ADSL, 4G, cable, NBN, …
- How are they connected with each other?

Internet Structure

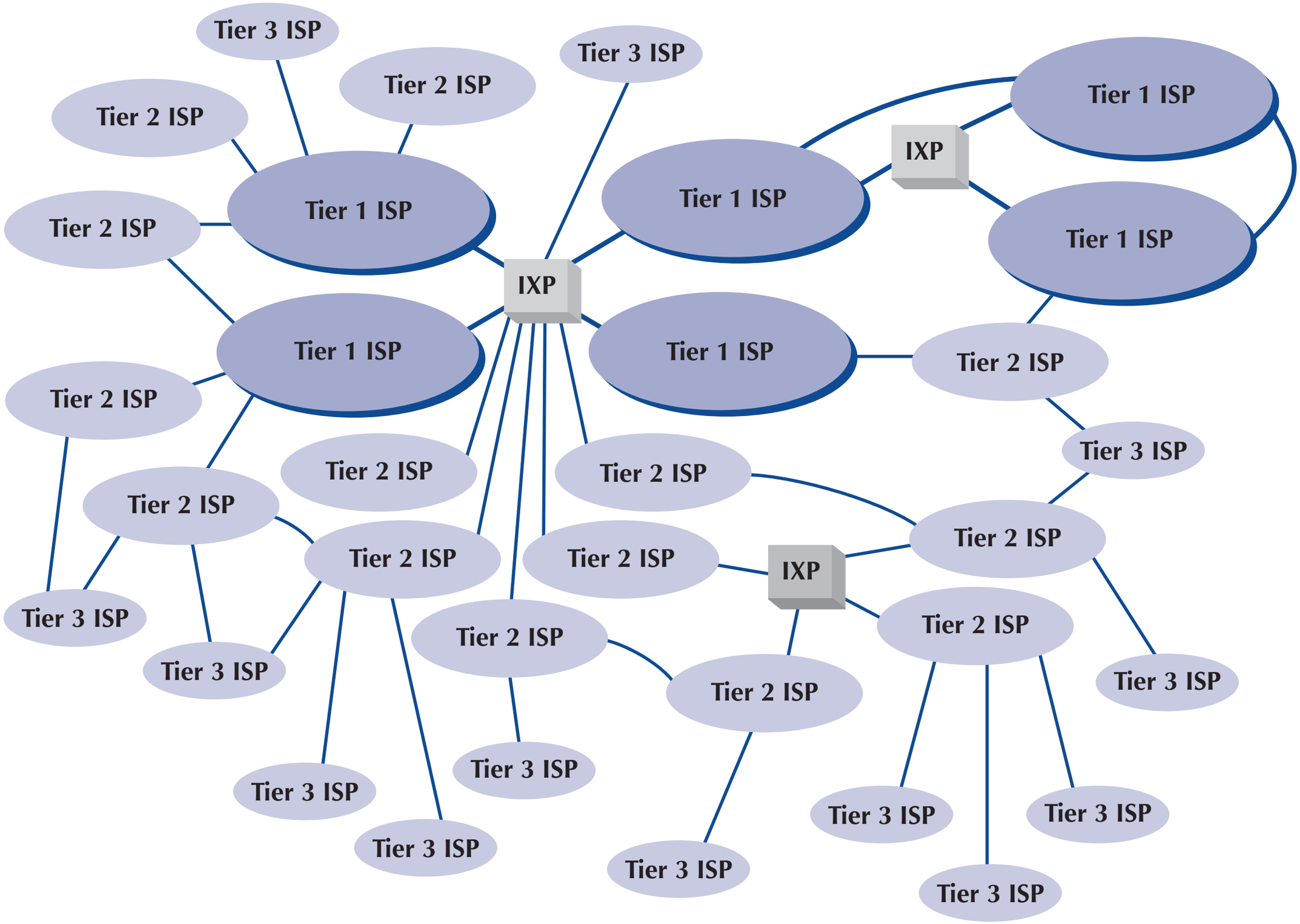
Internet Service Providers

- Commercial companies
- You connect via ADSL, 4G, cable, NBN, …
- How are they connected with each other?

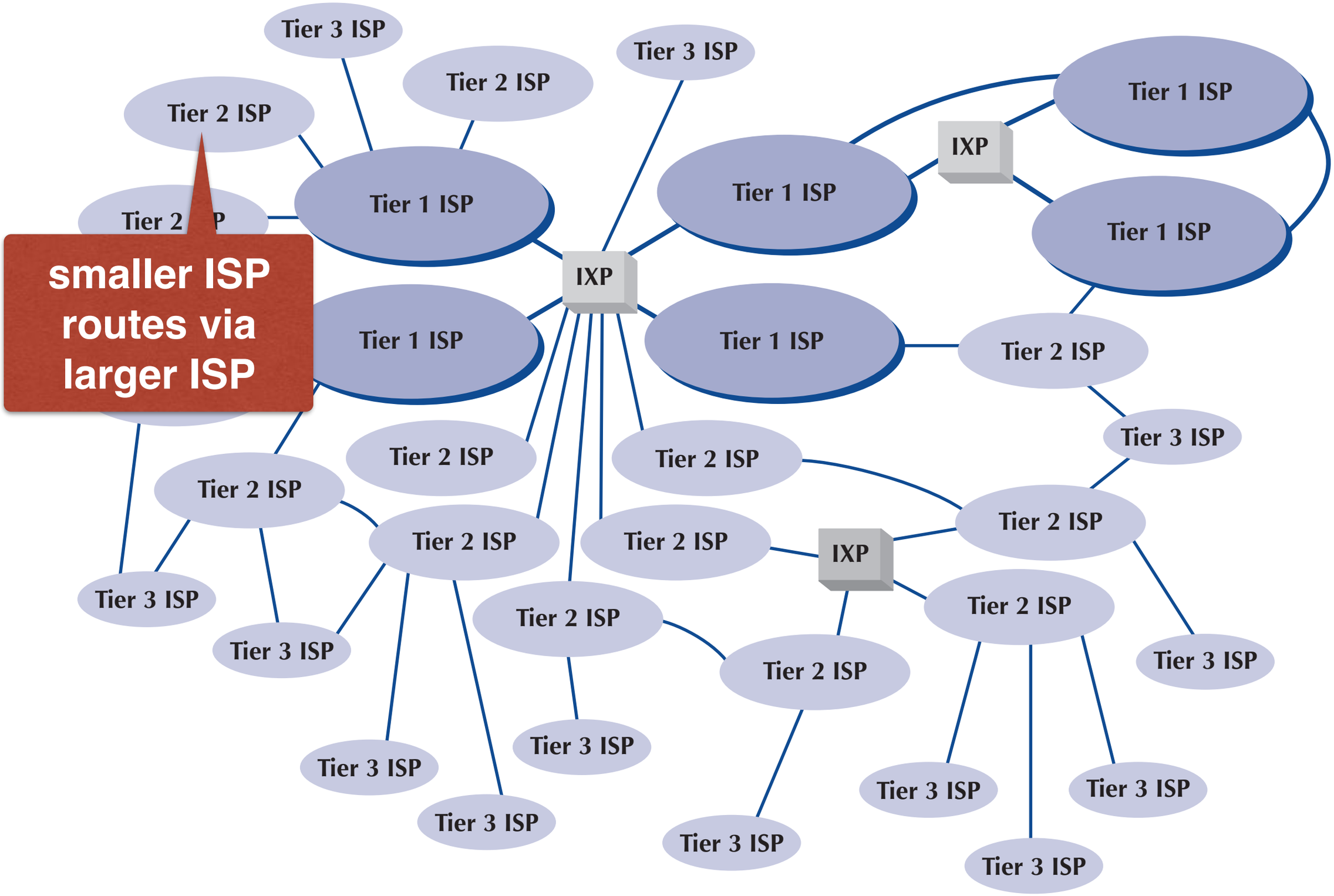
Hierarchy of ASs

- Each ISP operates an AS
- Routing information shared between ASs using BGP
- ISPs connect at **IXPs**

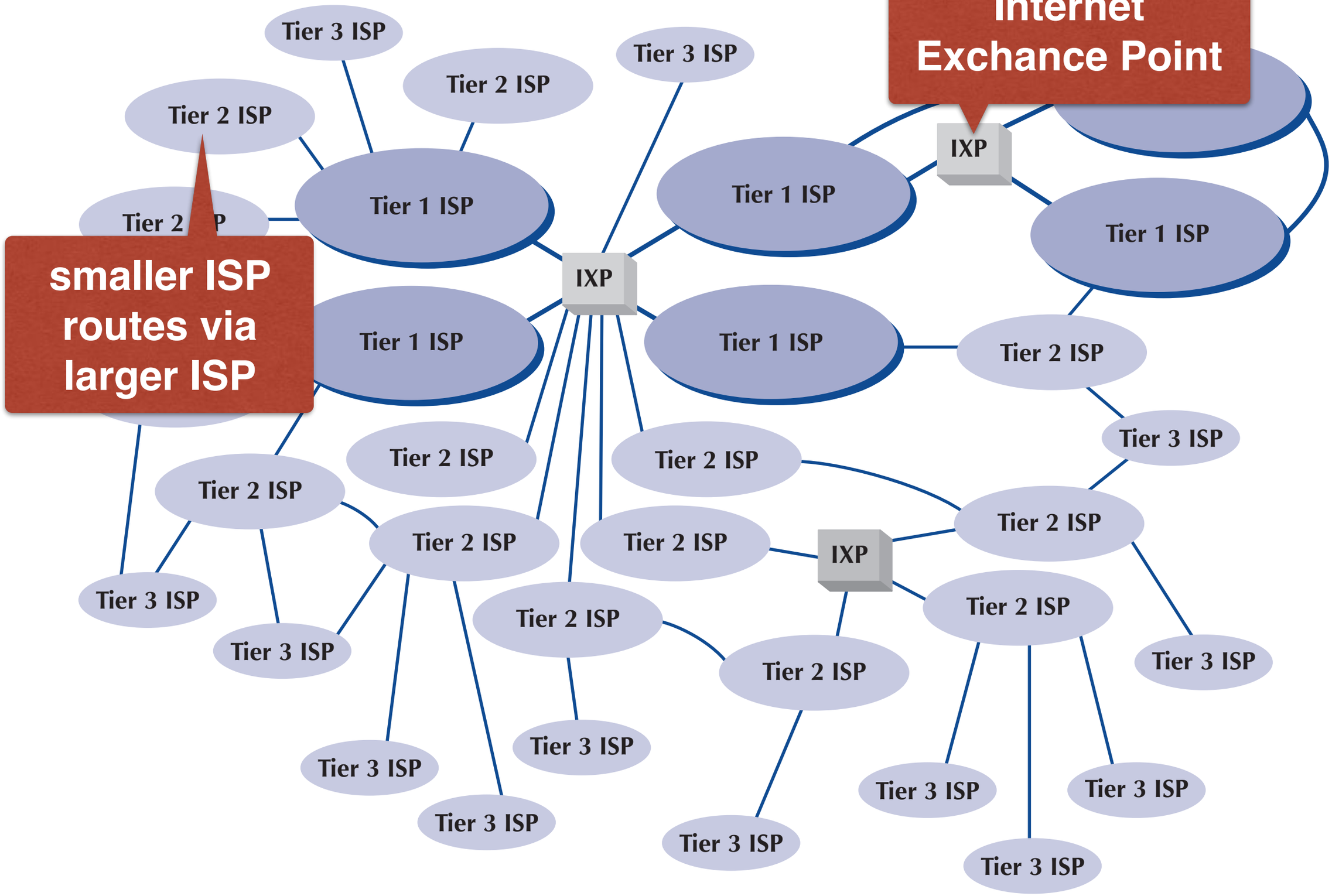
Internet Structure



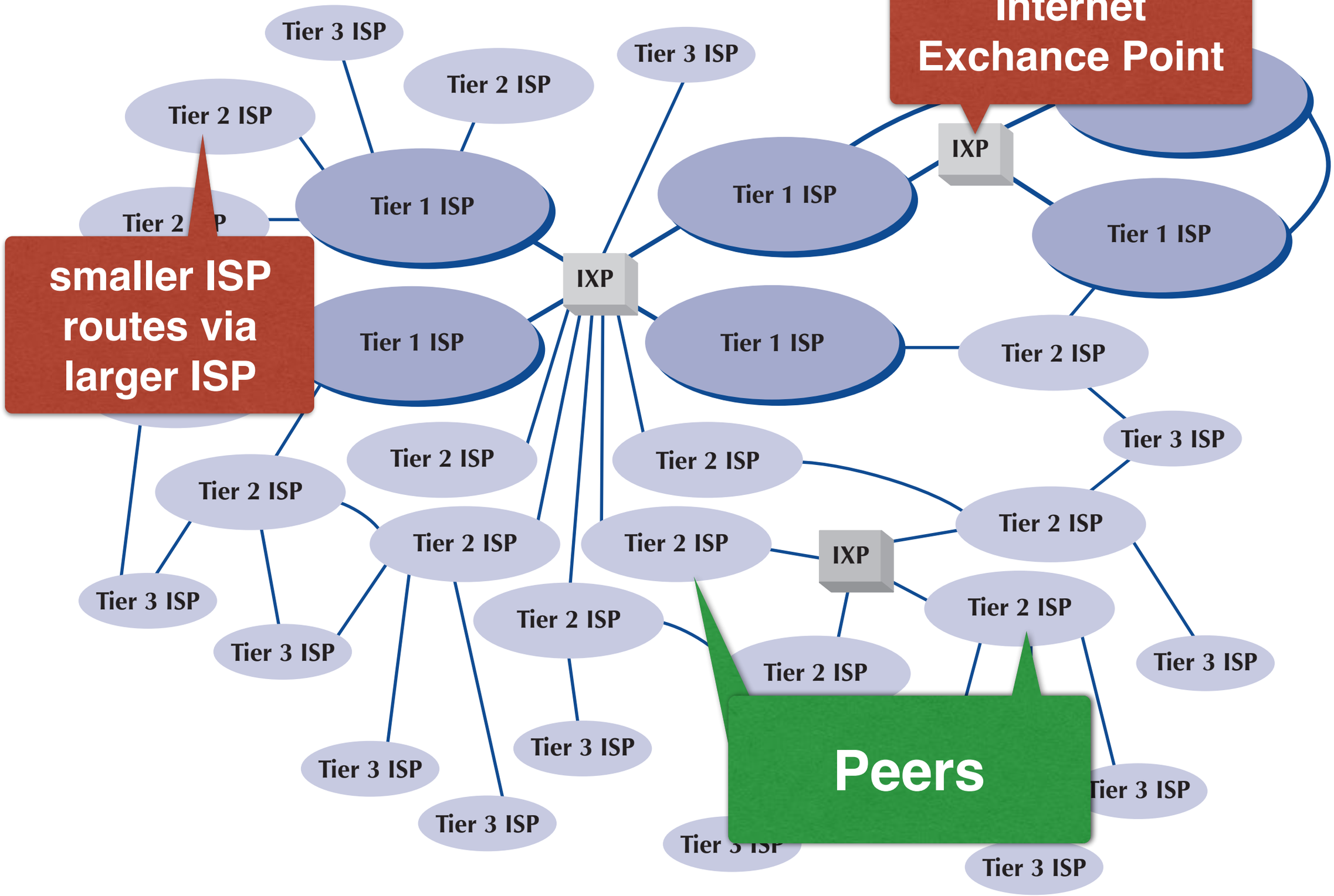
Internet Structure



Internet Structure



Internet Structure



Peering

Peering

Tier-1 ISPs

- Large ISPs with large WANs
- Charge smaller ISPs for routing their traffic

Peering

Tier-1 ISPs

- Large ISPs with large WANs
- Charge smaller ISPs for routing their traffic

Peering agreement between two ISPs

- accept each other's traffic without charge
- usually because both are similar size (similar amount of traffic)
- connect at an IXP

Peering

Tier-1 ISPs

- Large ISPs with large WANs
- Charge smaller ISPs for routing their traffic

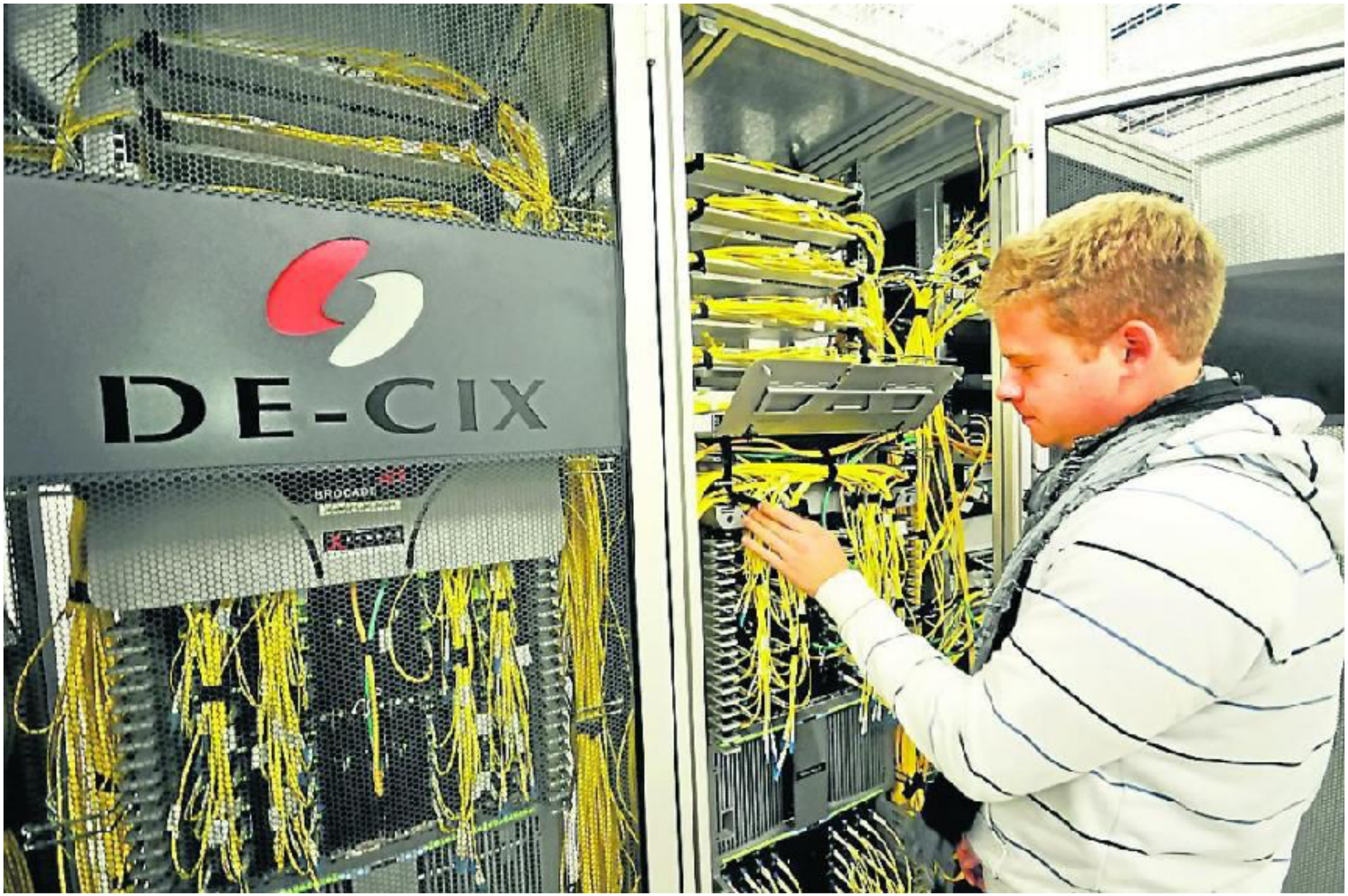
Peering agreement between two ISPs

- accept each other's traffic without charge
- usually because both are similar size (similar amount of traffic)
- connect at an IXP

IXP

- provides the hardware for several ISPs to connect
- often owned by a consortium of ISPs

Inside an IXP



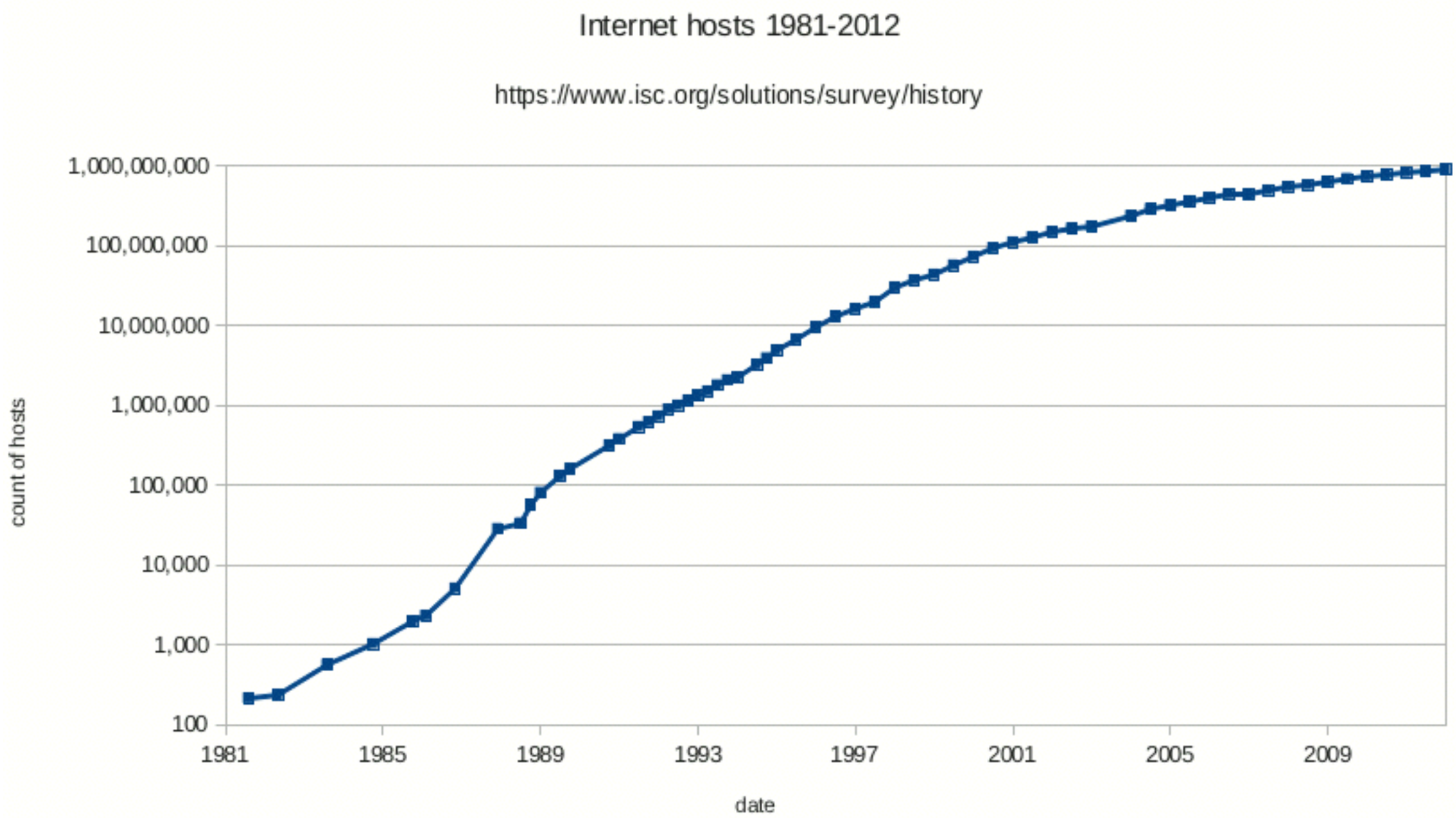
Delivering Content over the Internet

The problem

The Internet has grown

The problem

The Internet has grown



The problem

The Internet has grown

- not only the number of hosts, but also their **distance**
- many applications rely on **low latency**
  (video, streaming music, modern web sites)
- some services have millions of users
- the main protocols (HTTP, TCP/IP and BGP) were not designed for this growth

The problem

The Internet has grown

- not only the number of hosts, but also their **distance**
- many applications rely on **low latency**
  (video, streaming music, modern web sites)
- some services have millions of users
- the main protocols (HTTP, TCP/IP and BGP) were not designed for this growth

So why does it still work?

- Load balancing
- Content Delivery Networks (CDNs)

Load Balancing

Load Balancing

Many services impossible for single server

- Google processes over 40,000 search queries per second
- 6000 tweets are sent per second
- Netflix streams around 10.2 Tbps on average

Load Balancing

Many services impossible for single server

- Google processes over 40,000 search queries per second
- 6000 tweets are sent per second
- Netflix streams around 10.2 Tbps on average

Spread load over multiple servers

- DNS-based: host name maps to multiple IPs
- Special hardware: load balancer accepts requests, routes them to different servers

DNS-based load balancing

DNS-based load balancing

Inside Monash network:

- PING www.google.com (216.58.220.132)
  64 bytes from 216.58.220.132: time=13.752 ms

DNS-based load balancing

Inside Monash network:

- PING www.google.com (216.58.220.132)
  64 bytes from 216.58.220.132: time=13.752 ms

From Optus network:

- PING www.google.com (74.125.237.209)
  64 bytes from 74.125.237.209: time=52.074 ms

DNS-based load balancing

Inside Monash network:

- PING www.google.com (216.58.220.132)
  64 bytes from 216.58.220.132: time=13.752 ms

From Optus network:

- PING www.google.com (74.125.237.209)
  64 bytes from 74.125.237.209: time=52.074 ms

From Germany:

- PING www.google.com (173.194.112.176)
  64 bytes from 173.194.112.176: time=1.43 ms

DNS-based load balancing

Inside Monash network:

- PING www.google.com (216.58.220.132)
  64 bytes from 216.58.220.132: time=13.752 ms

From Optus network:

- PING www.google.com (74.125.237.209)
  64 bytes from 74.125.237.209: time=52.074 ms

From Germany:

- PING www.google.com (173.194.112.176)
  64 bytes from 173.194.112.176: time=1.43 ms

From France:

- PING www.google.com (74.125.21.105)
  64 bytes from 74.125.21.105: time=104 ms

Content Caching

Content Caching

Store web data closer to users

- replicate web pages etc. in *caches*
- can be implemented *transparently:*
  - user makes request
  - router on path to server queries *cache engine*
  - if content available, serve from local cache

Content Caching

Store web data closer to users

- replicate web pages etc. in *caches*
- can be implemented *transparently:*
  - user makes request
  - router on path to server queries *cache engine*
  - if content available, serve from local cache

This is explicitly supported by HTTP

- GET requests can be cached
- HTTP headers can contain `Expires:` field
- Cache only serves GET requests that are not expired

Content Delivery Networks (CDNs)

Content Delivery Networks (CDNs)

Load balancing only solves half the problem

- once the requests arrive in your network, you can distribute them to all your servers
- but the requests and responses need to be routed through the Internet

Content Delivery Networks (CDNs)

Load balancing only solves half the problem

- once the requests arrive in your network, you can distribute them to all your servers
- but the requests and responses need to be routed through the Internet

CDNs

- operate servers in multiple locations
- operate their own high-bandwidth network
- locate points of presence close to end users

CDNs and Peering

CDNs and Peering

Get close to your customers

- improves user experience (fast page load times)
- network inefficiencies are not blamed on you

CDNs and Peering

Get close to your customers

- improves user experience (fast page load times)
- network inefficiencies are not blamed on you

CDNs are present at IXPs

- peer with anybody for free
- small ISPs avoid paying e.g. for YouTube content downloaded from the upstream ISP
- Example: Netflix peers with Australian ISPs, which can offer “unmetered” access

Summary (I)

Network layer

- subnet mask identifies other devices in same subnet
- routers use routing tables to determine next hop
- static vs dynamic routing
- IPv4 vs IPv6

Transport layer

- splits up long messages into segments
- uses ARQ for error control
- uses ports to identify applications

Summary (II)

The Internet

- consists of Autonomous Systems
- connected by routers that use BGP for sharing routes
- ISPs run autonomous systems, connect at IXP
- Load balancing and Content Delivery Networks enabled the massive growth