Communication Networks

Spring 2019

Alexander Dietmüller

nsg.ee.ethz.ch

ETH Zürich (D-ITET)

March 11 2019

Materials inspired from Scott Shenker & Jennifer Rexford

Mark the date:

Internet Routing Project

Group Registration

March 18, register in groups of three

Project Start

March 25

Hackathon

April 4, 18:00

Last week on

Communication Networks

Reliable Transport

1 **Correctness condition**

if-and-only if again

2 **Design space**

timeliness *vs* efficiency *vs* …

3 **Examples**

Go-Back-N & Selective Repeat

Reliable Transport

1 **Correctness condition**

if-and-only if again

Design space

timeliness *vs* efficiency *vs* …

Examples

Go-Back-N & Selective Repeat

A reliable transport design is correct if…

attempt #4

A packet is always resent if
the previous packet was lost or corrupted

A packet may be resent at other times

Correct!

Reliable Transport

Correctness condition

if-and-only if again

2 **Design space**

timeliness *vs* efficiency *vs* …

Examples

Go-Back-N & Selective Repeat

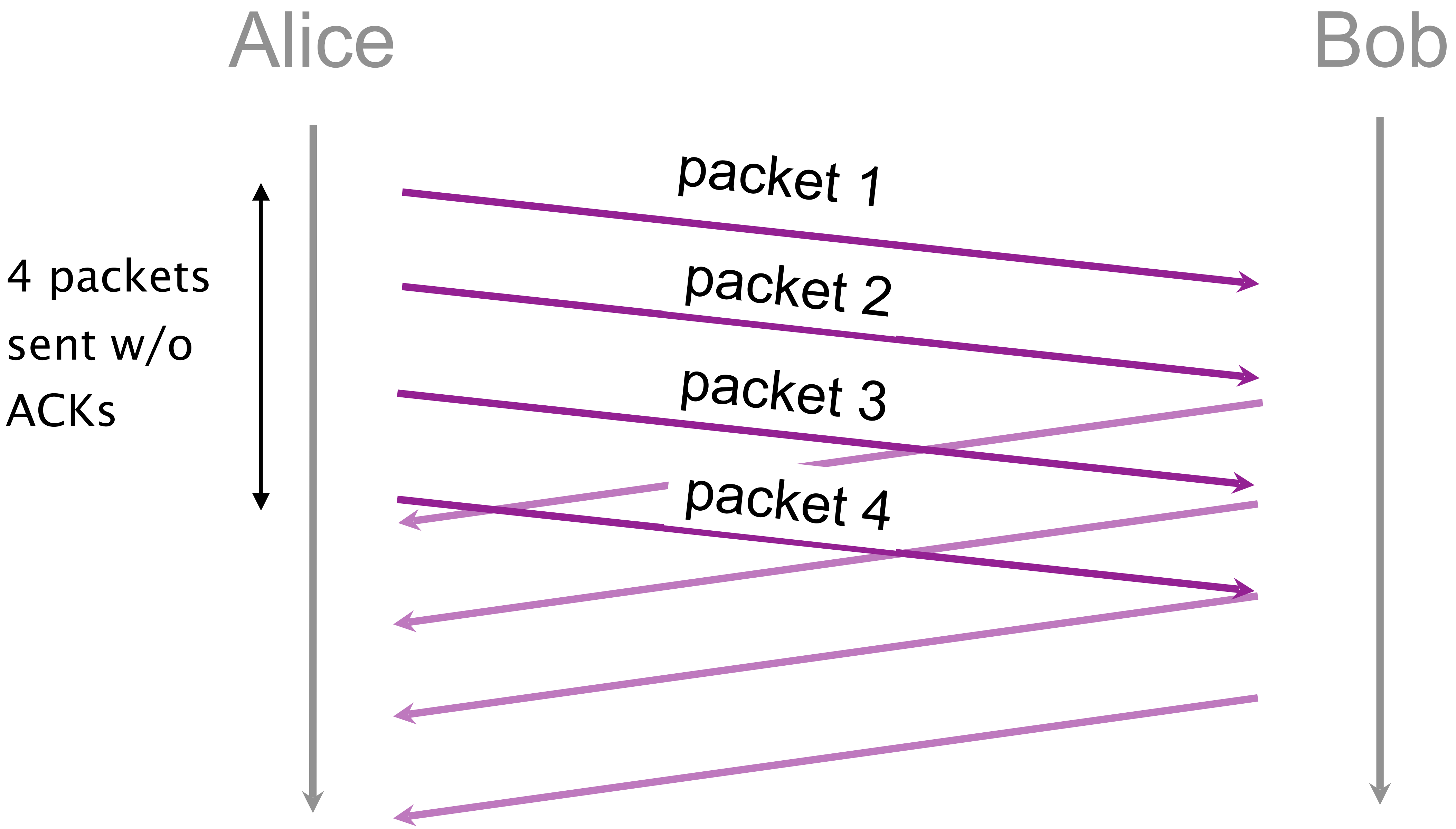
To improve timeliness, reliable transport protocols send **multiple packets at the same time**

approach — add sequence number inside each packet

add buffers to the sender and receiver

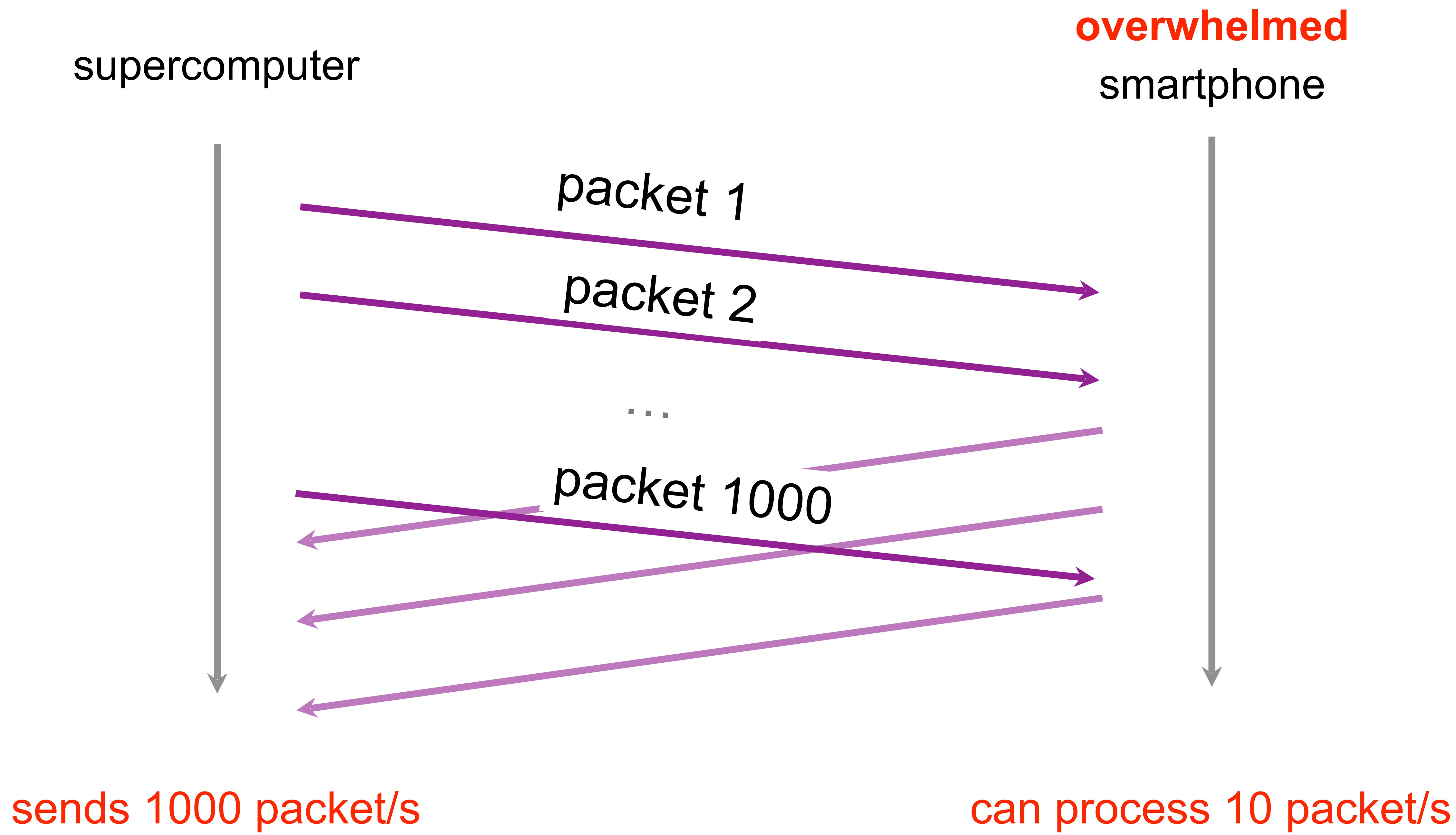
sender — store packets sent & not acknowledged

receiver — store out-of-sequence packets received



Alice
Bob
4 packets
sent w/o
ACKs
packet 1
packet 2
packet 3
packet 4

Sending multiple packets improves timeliness, but it can also **overwhelm the receiver**



Using a sliding window enables flow control

Sender keeps a list of the sequence # it can send

known as the *sending window*

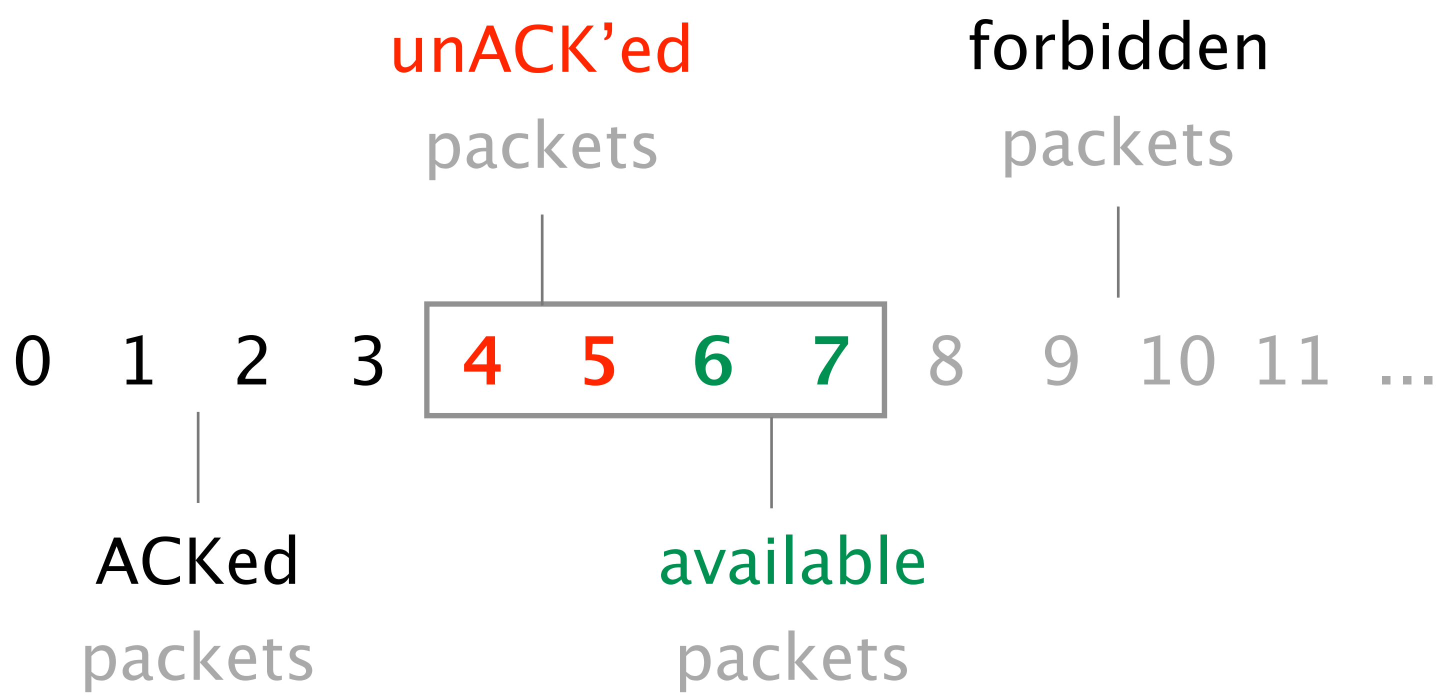
Receiver also keeps a list of the acceptable sequence #

known as the *receiving window*

Sender and receiver negotiate the window size

sending window <= receiving window

Example with a window composed of 4 packets



Back to the end of last weeks lecture

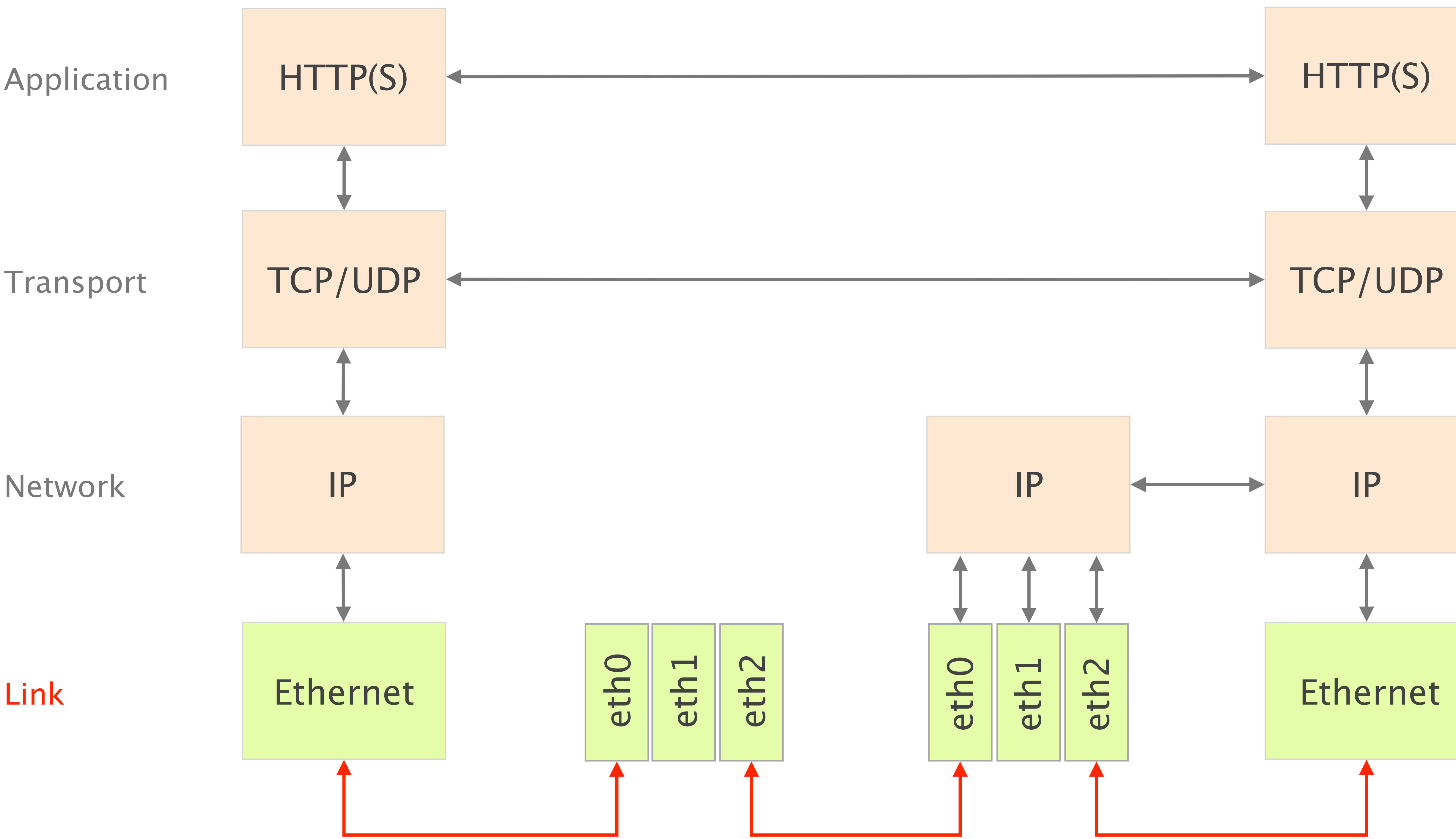
This week on

Communication Networks

This week we start speaking about

How the Internet actually works

We'll do that layer-by-layer, bottom-up, starting with the Link layer



How do local computers communicate?

Communication Networks

Part 2: The Link Layer

#1 What is a link?

#2 How do we identify link adapters?

#3 How do we share a network medium?

#4 What is Ethernet?

#5 How do we interconnect segments at the link layer?

Communication Networks

Part 2: The Link Layer

#1 What is a link?

How do we identify link adapters?

How do we share a network medium?

What is Ethernet?

How do we interconnect segments at the link layer?

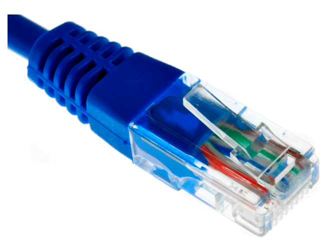
Link

Communication medium

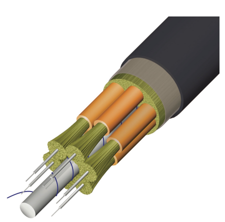
and

Network adapter

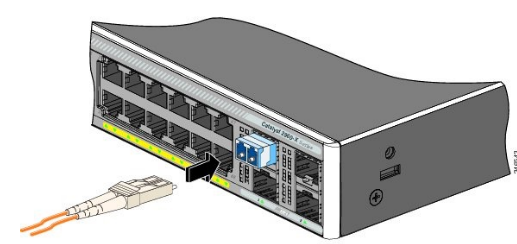
Wifi



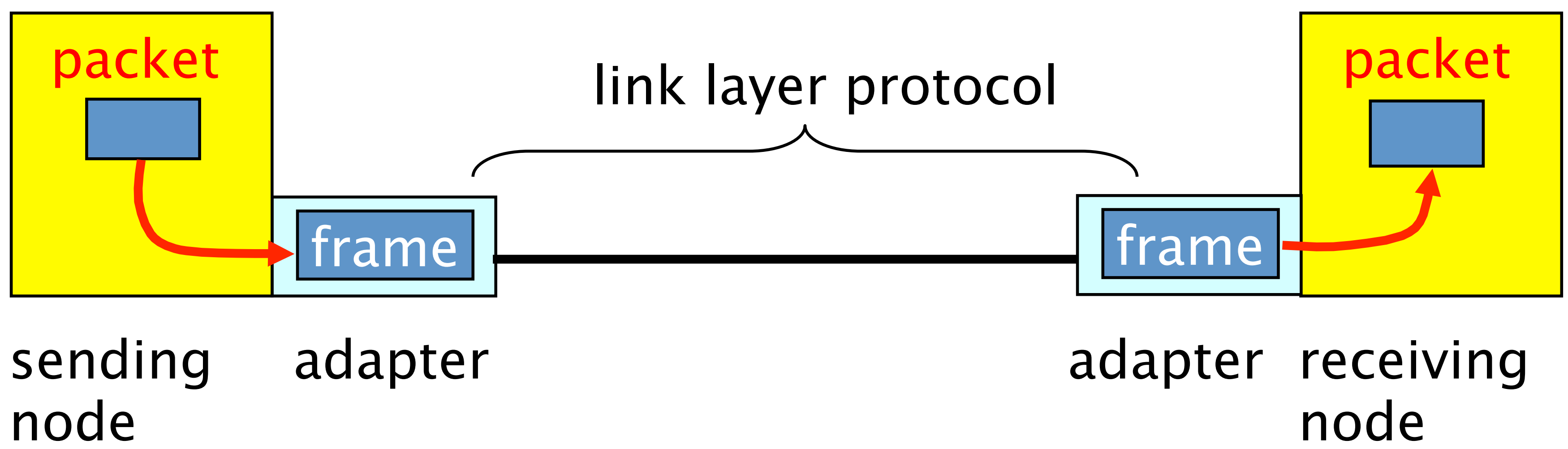
Ethernet



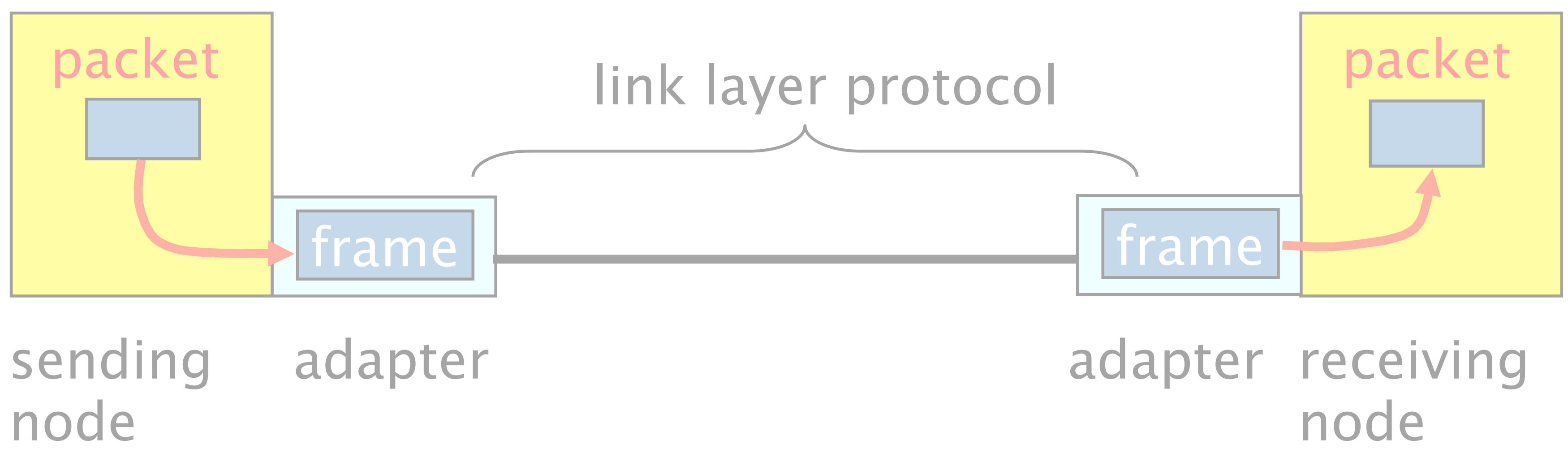
Fiber



Network adapters communicate together through the medium



Network adapters communicate together through the medium



sender

encapsulate packets in a frame

add error checking bits, flow control, …

receiver

look for errors, flow control, ...

extract packet and passes it to the network layer

The Link Layer provides a best-effort delivery service to the Network layer

L3	Network	global best-effort delivery
L2	Link	local best-effort delivery
L1	Physical	physical transfer of bits

The Link Layer provides a best-effort delivery service to the Network layer, composed of 5 sub-services

encoding	represents the 0s and the 1s
framing	encapsulate packet into a frame adding header and trailer
error detection	detects errors with checksum
error correction	optionally correct errors
flow control	pace sending and receiving node

Communication Networks

Part 2: The Link Layer

What is a link?

#2 How do we identify link adapters?

How do we share a network medium?

What is Ethernet?

How do we interconnect segments at the link layer?

Medium Access Control addresses

MAC addresses…

MAC addresses…

identify the sender & receiver adapters

used within a link

are uniquely assigned

hard-coded into the adapter when built

use a flat space of 48 bits

allocated hierarchically

MAC addresses are hierarchically allocated

34:36:3b:d2:8a:86

The first 24 bits blocks are assigned to network adapter vendor by the IEEE

34:36:3b:d2:8a:86

Apple, Inc.
1 Infinite Loop
Cupertino CA 95014
US

see http://standards-oui.ieee.org/oui/oui.txt

The second 24 bits block is assigned by the vendor to each network adapter

34:36:3b:d2:8a:86

assigned by Apple to my adapter

The address with all bits set to 1 identifies the broadcast address

`ff:ff:ff:ff:ff:ff`

enables to send a frame to *all* adapters on the link

By default, adapters only decapsulates frames addressed to the local MAC or the broadcast address

The promiscuous mode enables to decapsulate *everything*, independently of the destination MAC

Why don't we simply use IP addresses?

Links can support any protocol (not just IP)

different addresses on different kind of links

Adapters may move to different locations

cannot assign static IP address, it has to change

Adapters must be identified during bootstrap

need to talk to an adapter to give it an IP address

Adapters must be identified during bootstrap

need to talk to an adapter to give it an IP address

You need to solve two problems when you bootstrap an adapter

Who am I?

MAC-to-IP binding

How do I acquire an IP address?

Who are you?

IP-to-MAC binding

Given an IP address reachable on a link,

How do I find out what MAC to use?

Who am I?

MAC-to-IP binding

How do I acquire an IP address?

Dynamic Host Configuration Protocol

Who are you?

IP-to-MAC binding

Given an IP address reachable on a link,

How do I find out what MAC to use?

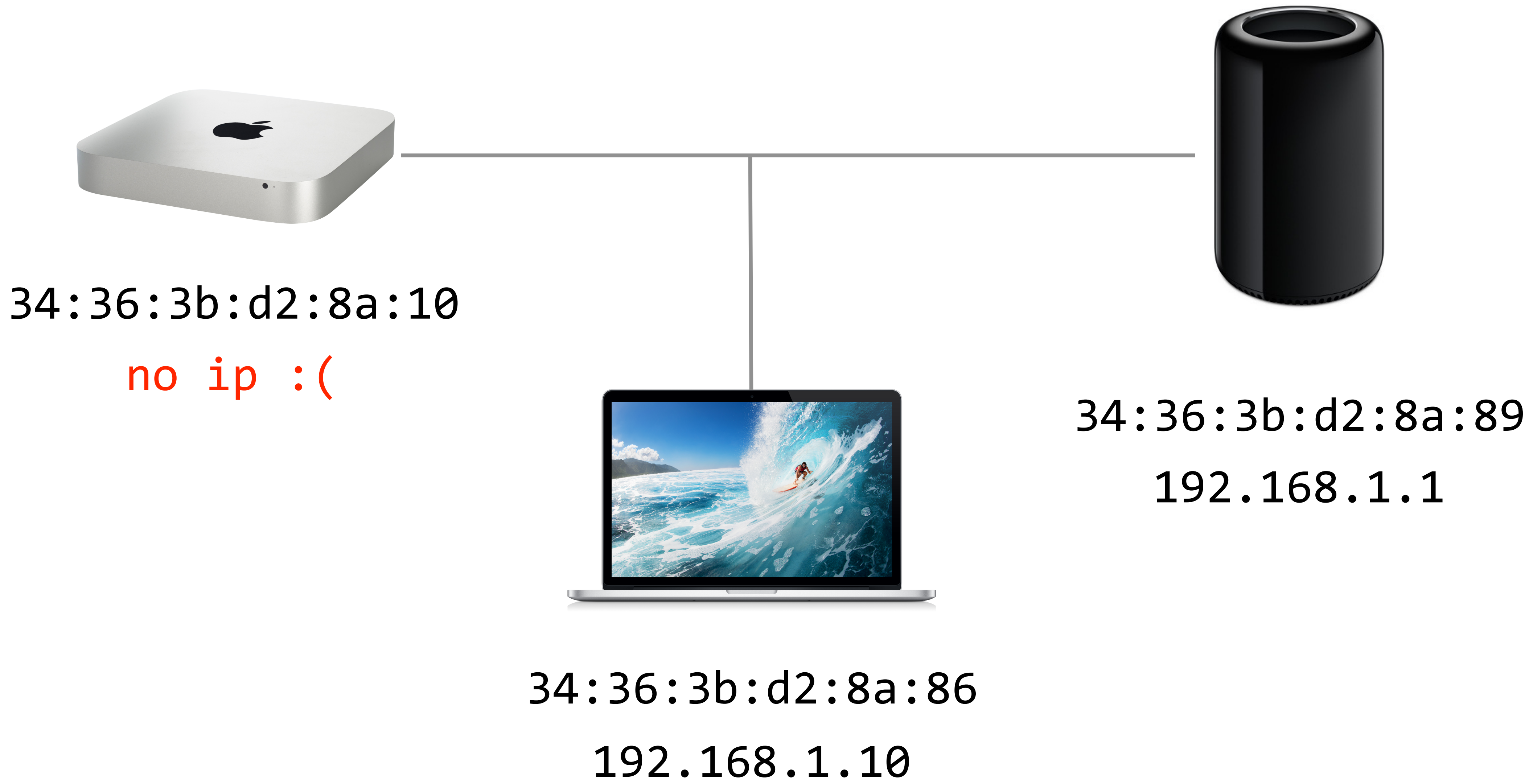
Address Resolution Protocol

Network adapters traditionally acquire an IP address using the Dynamic Host Configuration Protocol (DHCP)

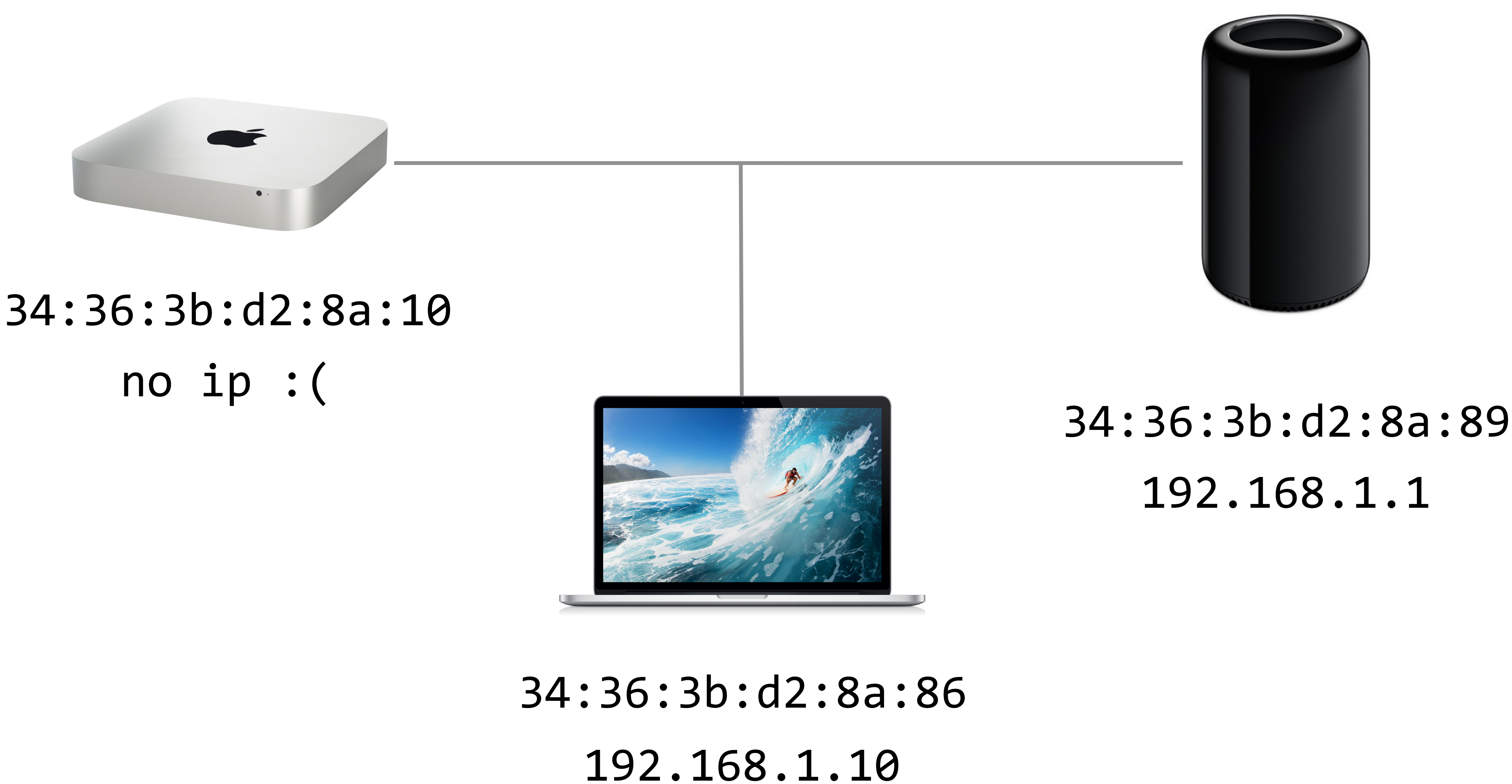
Every connected device needs an IP address…

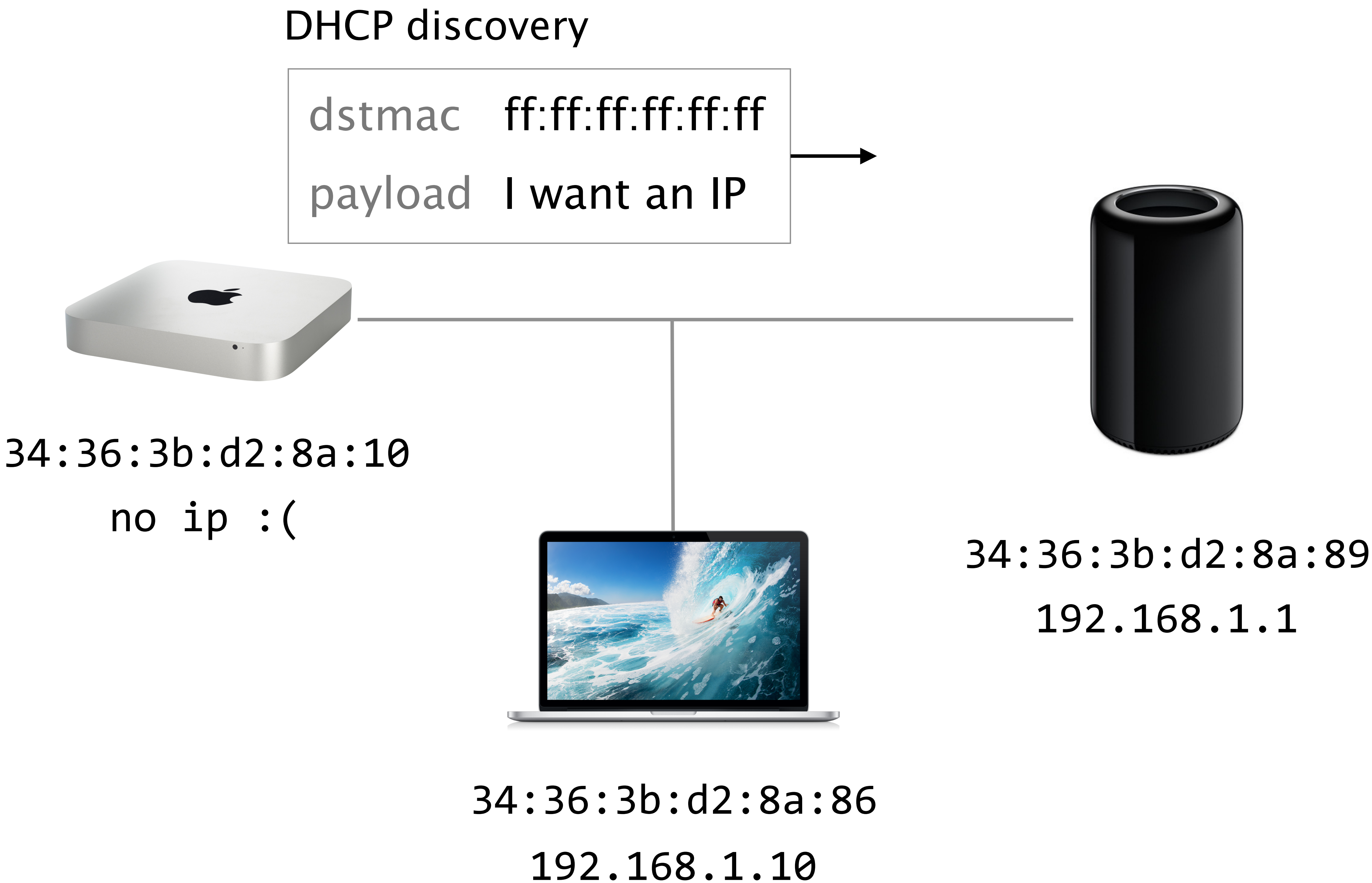
Newark Airport…

source: http://i.imgur.com/m1SQa6W.jpg

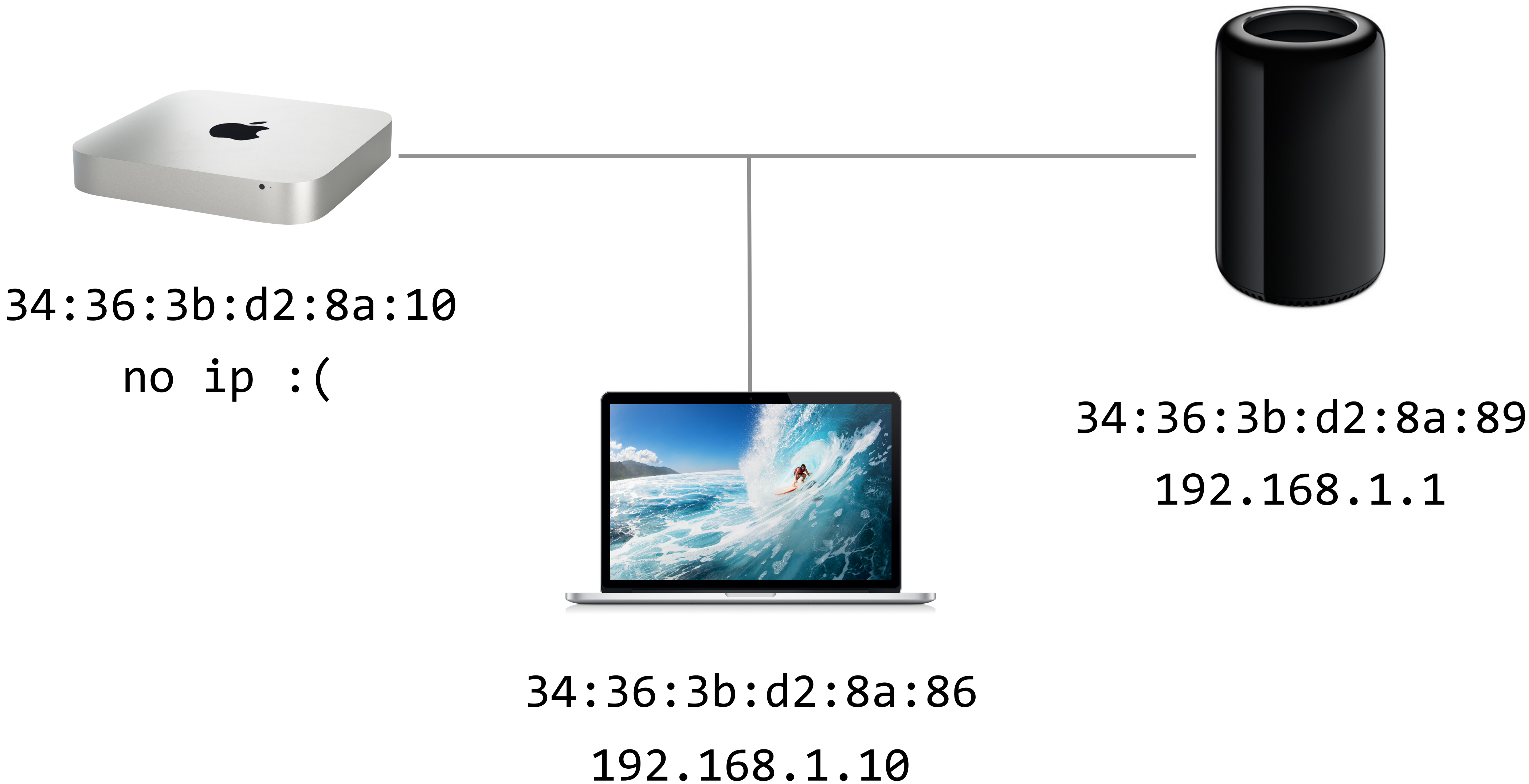

34:36:3b:d2:8a:10
no ip :(
34:36:3b:d2:8a:89
192.168.1.1
34:36:3b:d2:8a:86
192.168.1.10

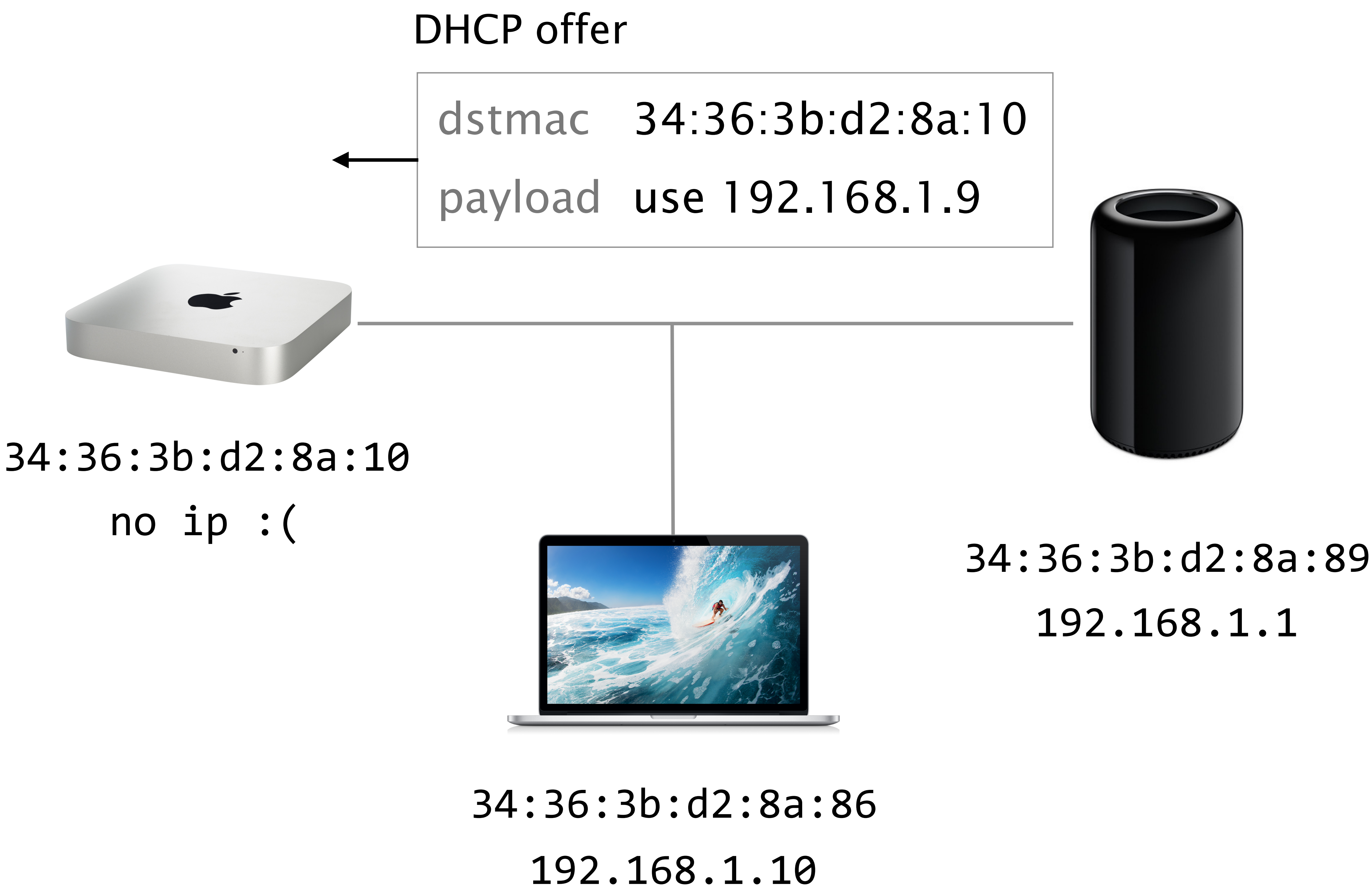
Host sends an “IP request” to everyone on the link using the broadcast address

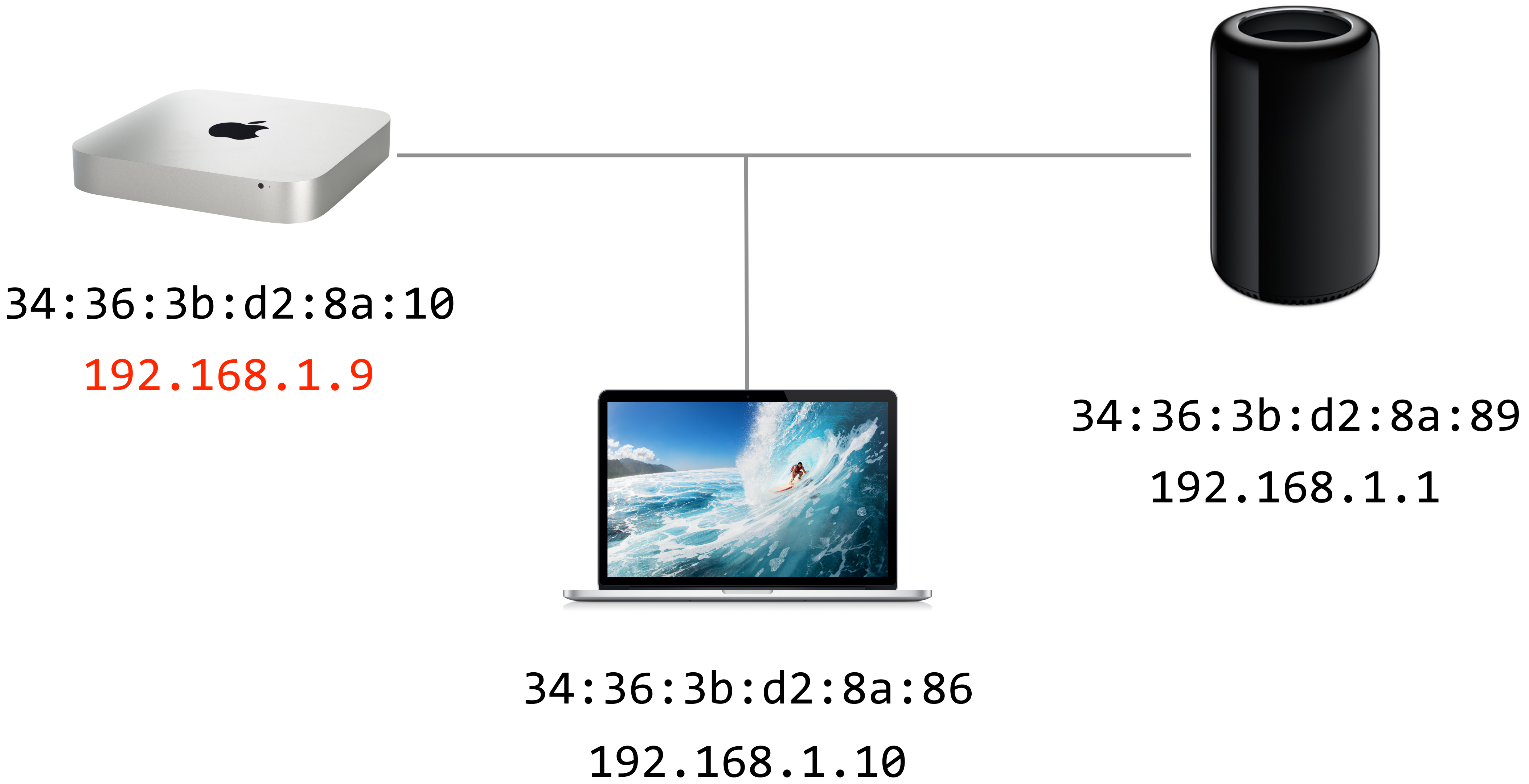



DHCP discovery
dstmac ff:ff:ff:ff:ff:ff
payload I want an IP
34:36:3b:d2:8a:10
no ip :(
34:36:3b:d2:8a:89
192.168.1.1
34:36:3b:d2:8a:86
192.168.1.10

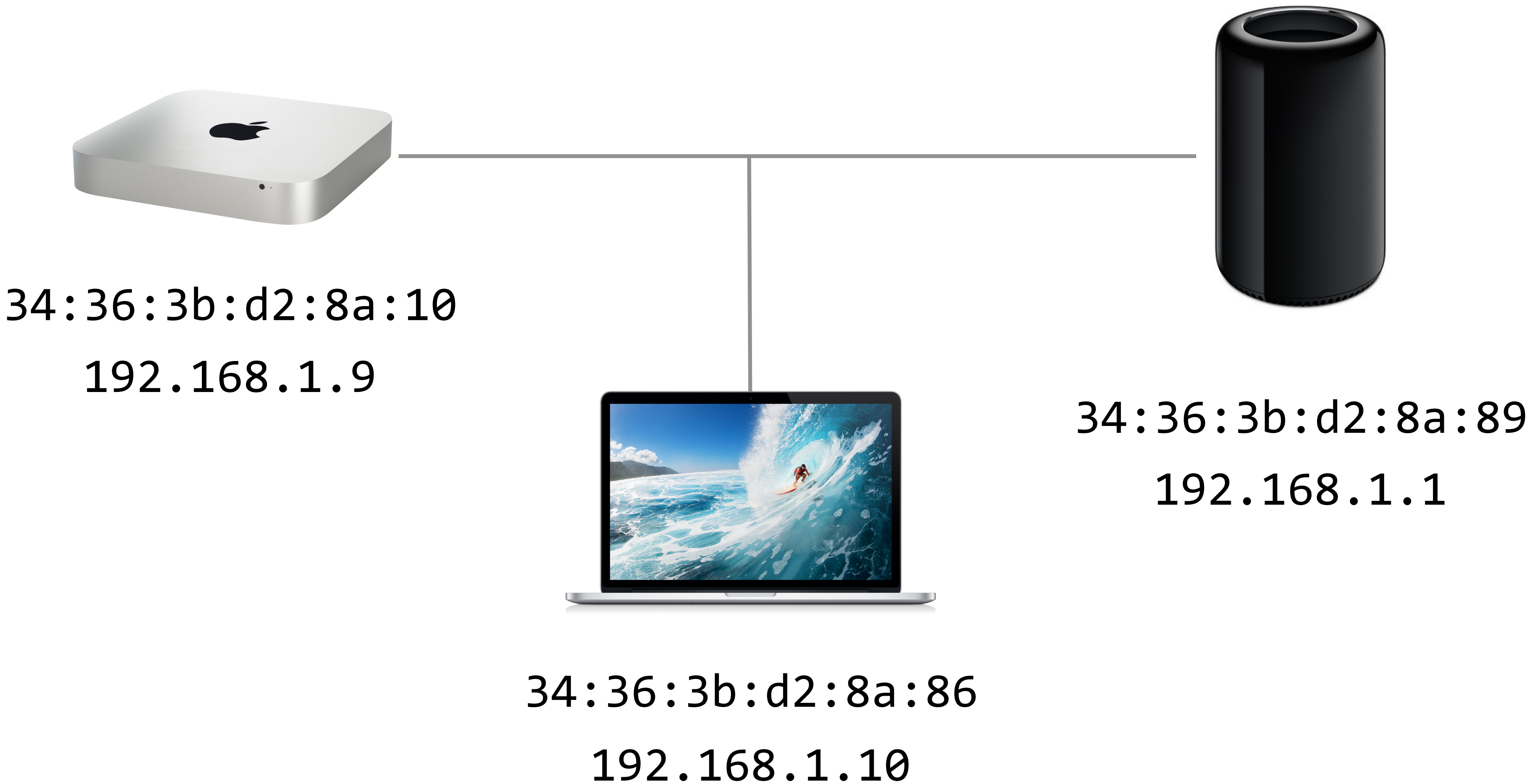
DHCP server (if any)
answers with an IP address



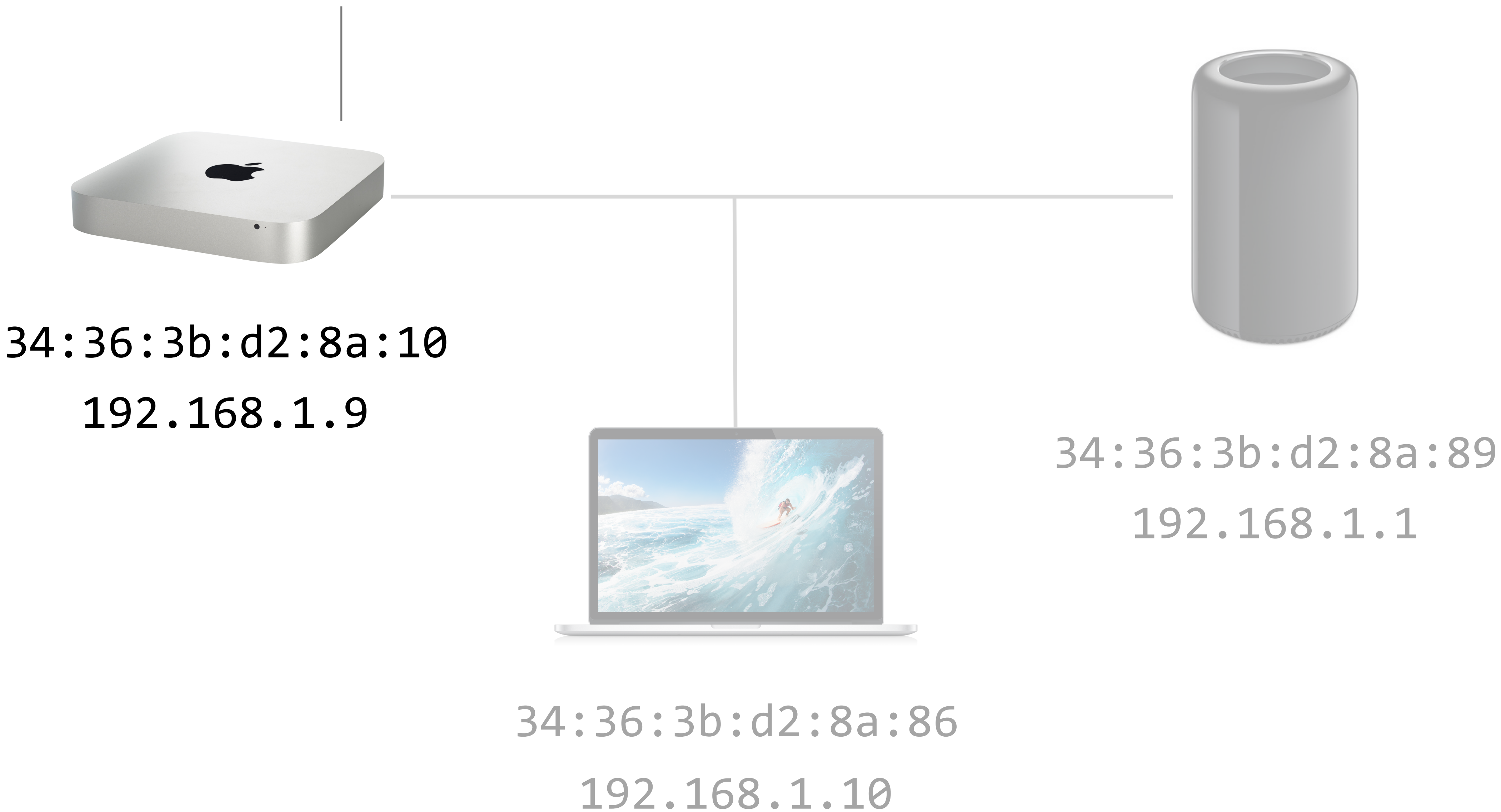

DHCP offer
dstmac 34:36:3b:d2:8a:10
payload use 192.168.1.9
34:36:3b:d2:8a:10
no ip :(
34:36:3b:d2:8a:89
192.168.1.1
34:36:3b:d2:8a:86
192.168.1.10

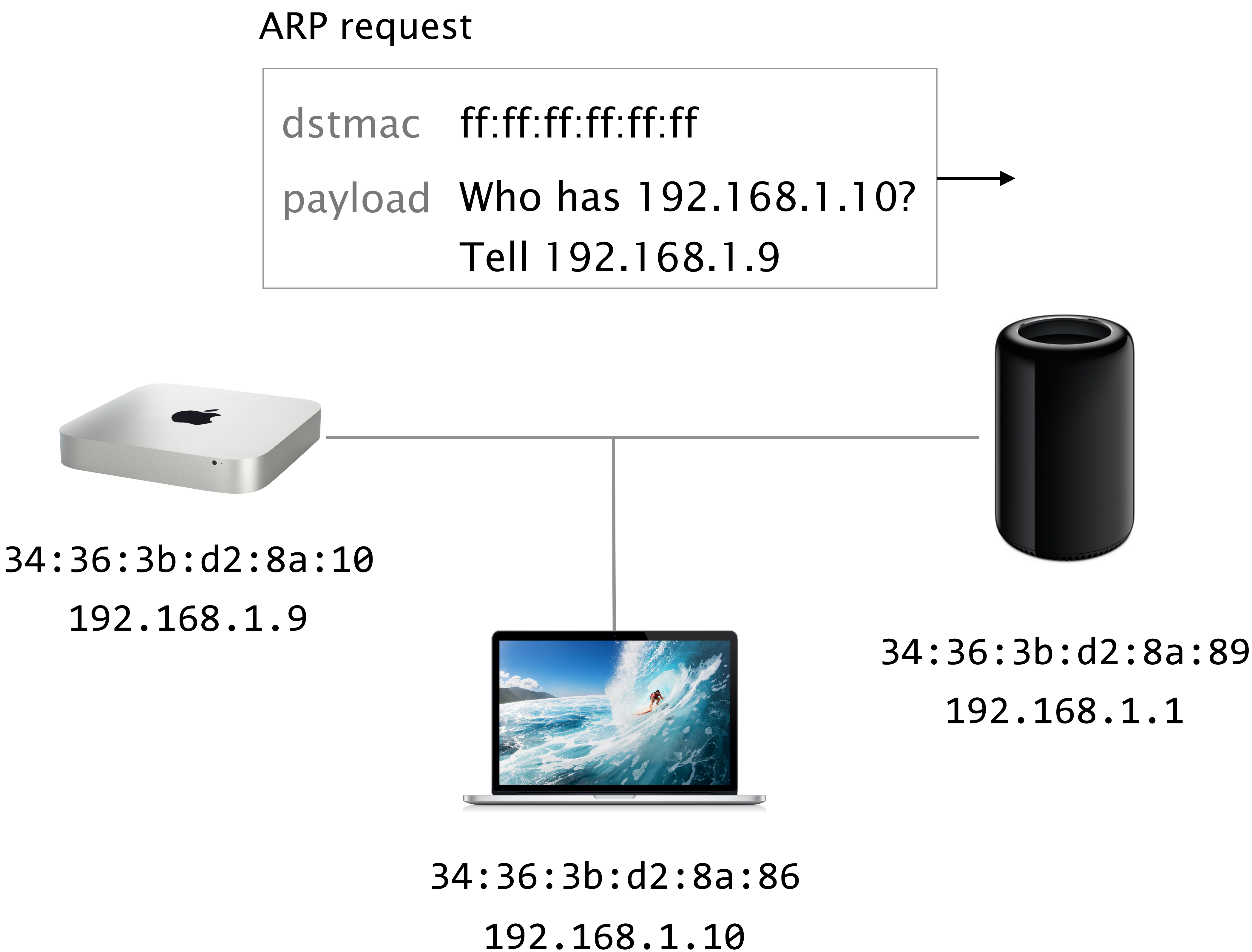

34:36:3b:d2:8a:10
192.168.1.9
34:36:3b:d2:8a:89
192.168.1.1
34:36:3b:d2:8a:86
192.168.1.10

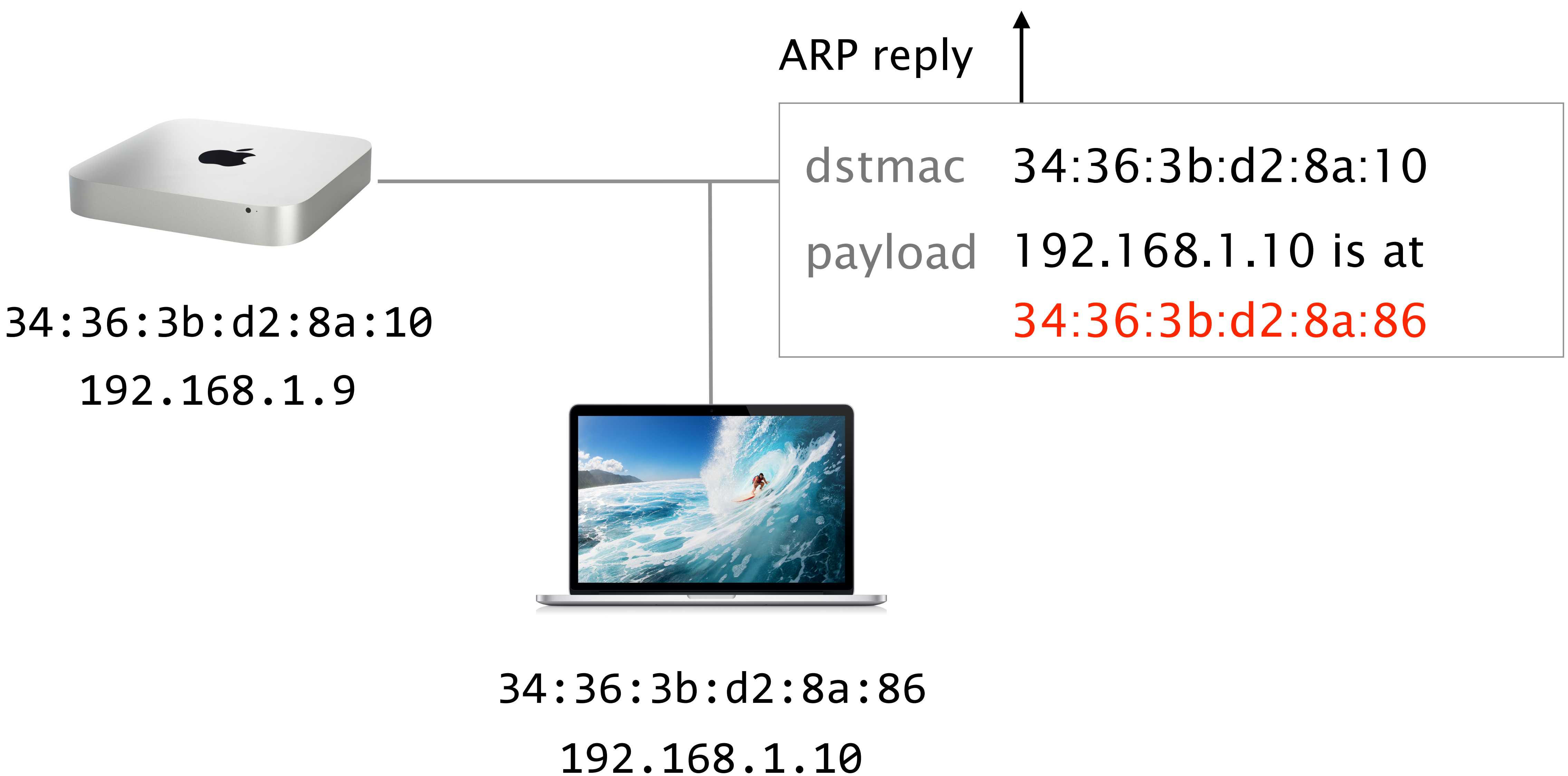
The Address Resolution Protocol (ARP) enables a host to discover the MAC associated to an IP

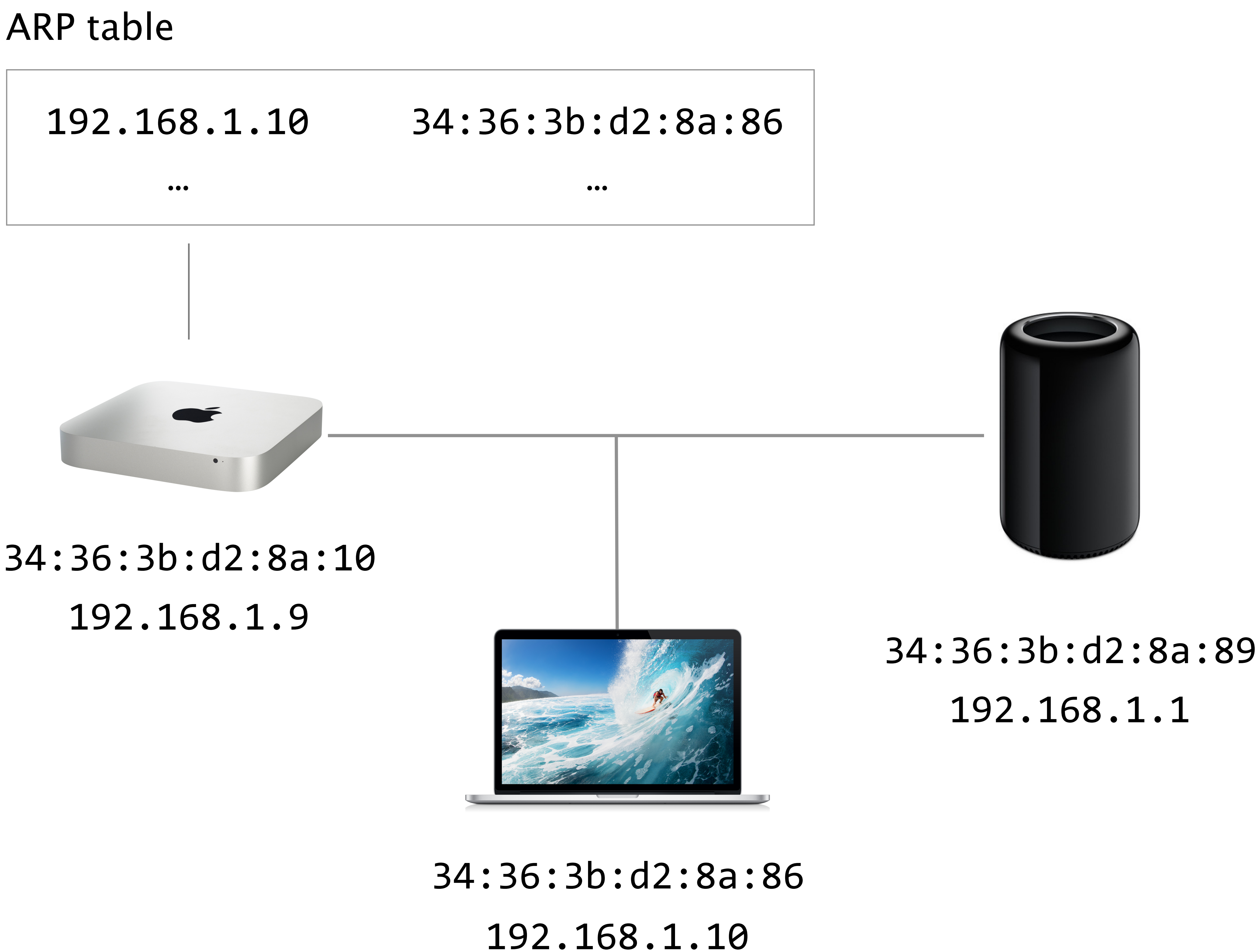

34:36:3b:d2:8a:10
192.168.1.9
34:36:3b:d2:8a:89
192.168.1.1
34:36:3b:d2:8a:86
192.168.1.10

I want to send an IP packet
to 192.168.1.10?
What destination MAC do I use?!




ARP request
dstmac ff:ff:ff:ff:ff:ff
payload Who has 192.168.1.10?
Tell 192.168.1.9
34:36:3b:d2:8a:10
192.168.1.9
34:36:3b:d2:8a:89
192.168.1.1
34:36:3b:d2:8a:86
192.168.1.10


ARP reply
dstmac 34:36:3b:d2:8a:10
payload 192.168.1.10 is at
34:36:3b:d2:8a:86
34:36:3b:d2:8a:10
192.168.1.9
34:36:3b:d2:8a:86
192.168.1.10


ARP table
192.168.1.10
34:36:3b:d2:8a:86
...
...
34:36:3b:d2:8a:10
192.168.1.9
34:36:3b:d2:8a:89
192.168.1.1
34:36:3b:d2:8a:86
192.168.1.10

Communication Networks

Part 2: The Link Layer

What is a link?

How do we identify link adapters?

#3 How do we share a network medium?

What is Ethernet?

How do we interconnect segments at the link layer?

Some medium are multi-access:

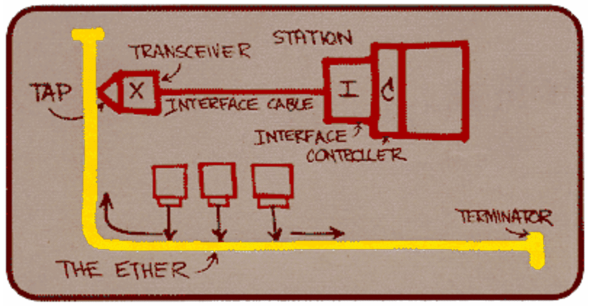
>1 host can communicate at the same time

Some medium are multi-access:

>1 host can communicate at the same time

Wireless
networks

Satellite
networks



Ethernet
networks

Cellular
networks

Some medium are multi-access:
>1 host can communicate at the same time

Problem

collisions lead
to garbled data

Solution

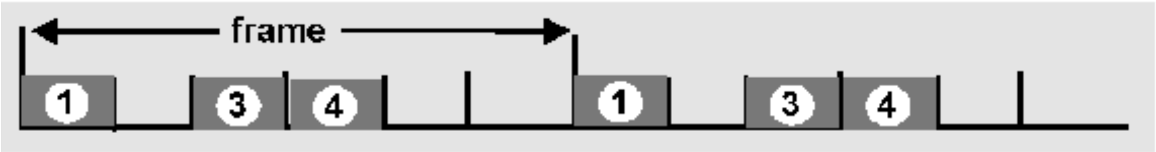
distributed algorithm
for sharing the channel

When can each node transmit?

Essentially, there are three techniques to deal with Multiple Access Control (MAC)

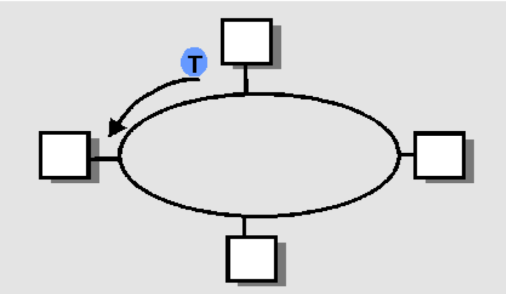
Divide the channel into pieces

either in time or in frequency



Take turns

pass a token for the right to transmit



Random access

allow collisions, detect them and then recover

Communication Networks

Part 2: The Link Layer

What is a link?

How do we identify link adapters?

How do we share a network medium?

#4 What is Ethernet?

How do we interconnect segments at the link layer?

Ethernet…

was invented as a broadcast technology

each packet was received by all attached hosts

is now *the* dominant wired LAN technology

by far the most widely used

has managed to keep up with the speed race

from 10 Mbps to 400 Gbps (next goal: 1 Tbps!)

Ethernet offers an unreliable, connectionless service

unreliable — Receiving adapter does not acknowledge anything

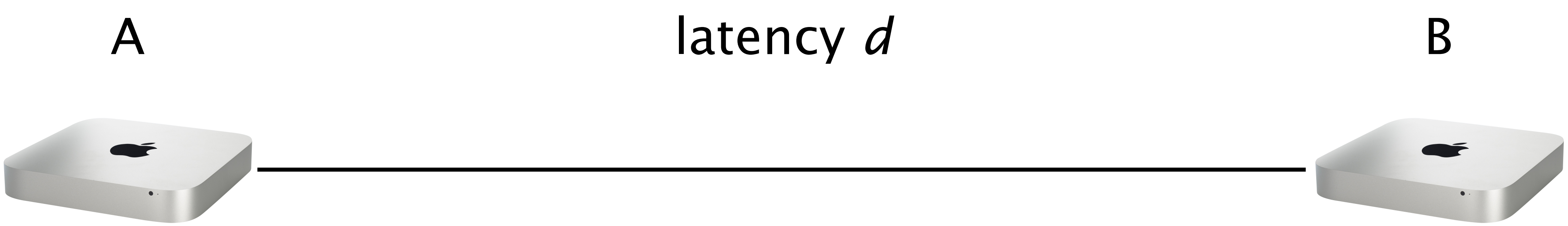
Packets passed to the network layer can have gaps

which can be filled by the transport protocol (TCP)

connectionless — No handshaking between the send and receive adapter

“Traditional” Ethernet relies on CSMA/CD

CSMA/CD imposes limits on the network length



Suppose A sends a packet at time *t*

B sees an idle line just before t+d and sends a packet

Effect

B would detect a collision and sends a jamming signal

A can detect the collision only after t+2d

For this reason, Ethernet imposes a minimum packet size (512 bits)

This imposes restriction on the length of the network

$$\text{Network length [m]} = \frac{\text{min\_frame\_size} * \text{speed of light}}{2 * \text{bandwidth}}$$

$$= 768 \text{ meters} \quad \text{for 100 Mbps}$$

What about for 1 Gbps, 10 Gbps, 100 Gbps?

Modern Ethernet links interconnects *exactly* two hosts, in full-duplex, rendering collisions impossible!

CSMA/CD is only needed for half-duplex communications

10 Gbps Ethernet does not even allow half-duplex anymore

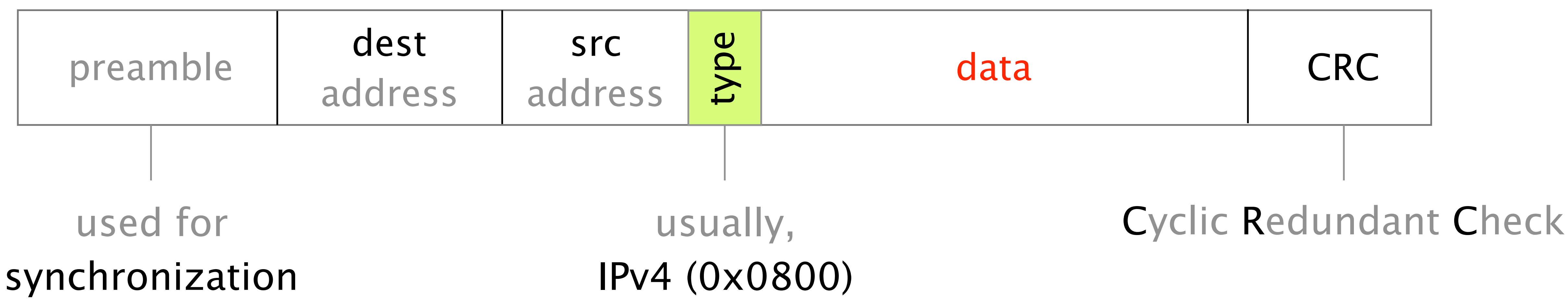
This means the 64 bytes restriction is not strictly needed

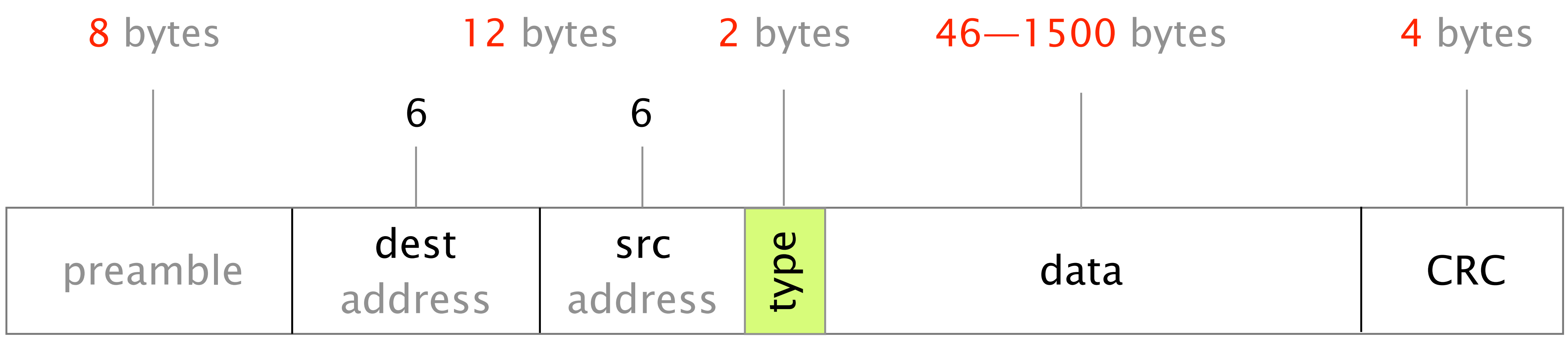
but IEEE chose to keep it

Multiple Access Protocols are still important for Wireless

important concepts to know in practice

The Ethernet header is simple, composed of 6 fields only





Ethernet efficiency (payload/tot. frame size): ~97.5%

Maximum throughput for 100 Mbps: ~97.50 Mbps

Communication Networks

Part 2: The Link Layer

What is a link?

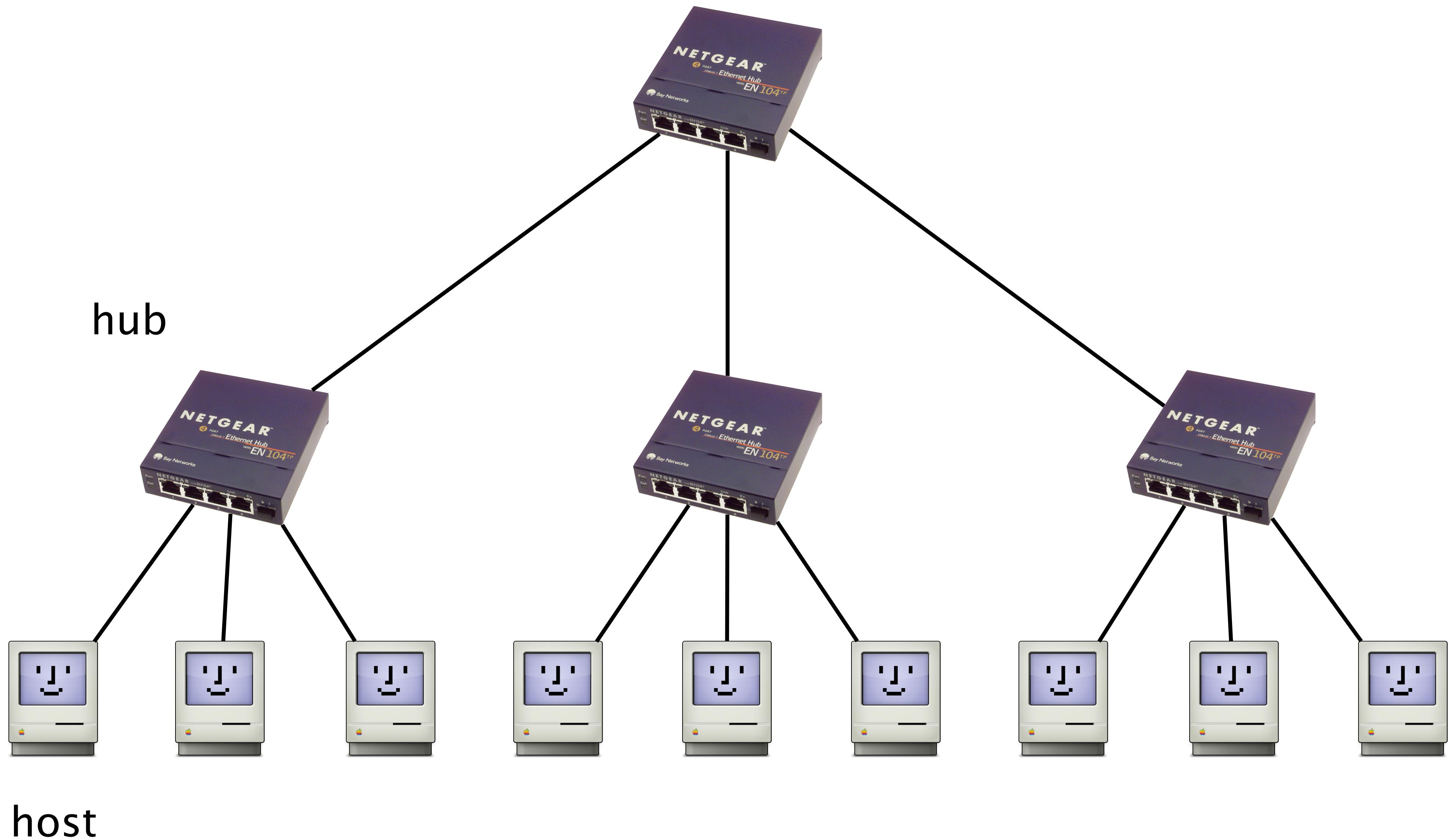
How do we identify link adapters?

How do we share a network medium?

What is Ethernet?

#5 How do we interconnect segments at the link layer?

Historically, people connected Ethernet segments together at the physical level using Ethernet hubs



Hubs work by repeating bits from one port to all the other ones

Hubs are now 

advantages

simple, cheap

disadvantages

inefficient, each bit is sent everywhere

limits the aggregates throughput

limited to one LAN technology

can't interconnect different rates/formats

limited number of nodes and distances

cannot go beyond 2500m on Ethernet

Local Area Networks are now almost exclusively composed of Ethernet switches

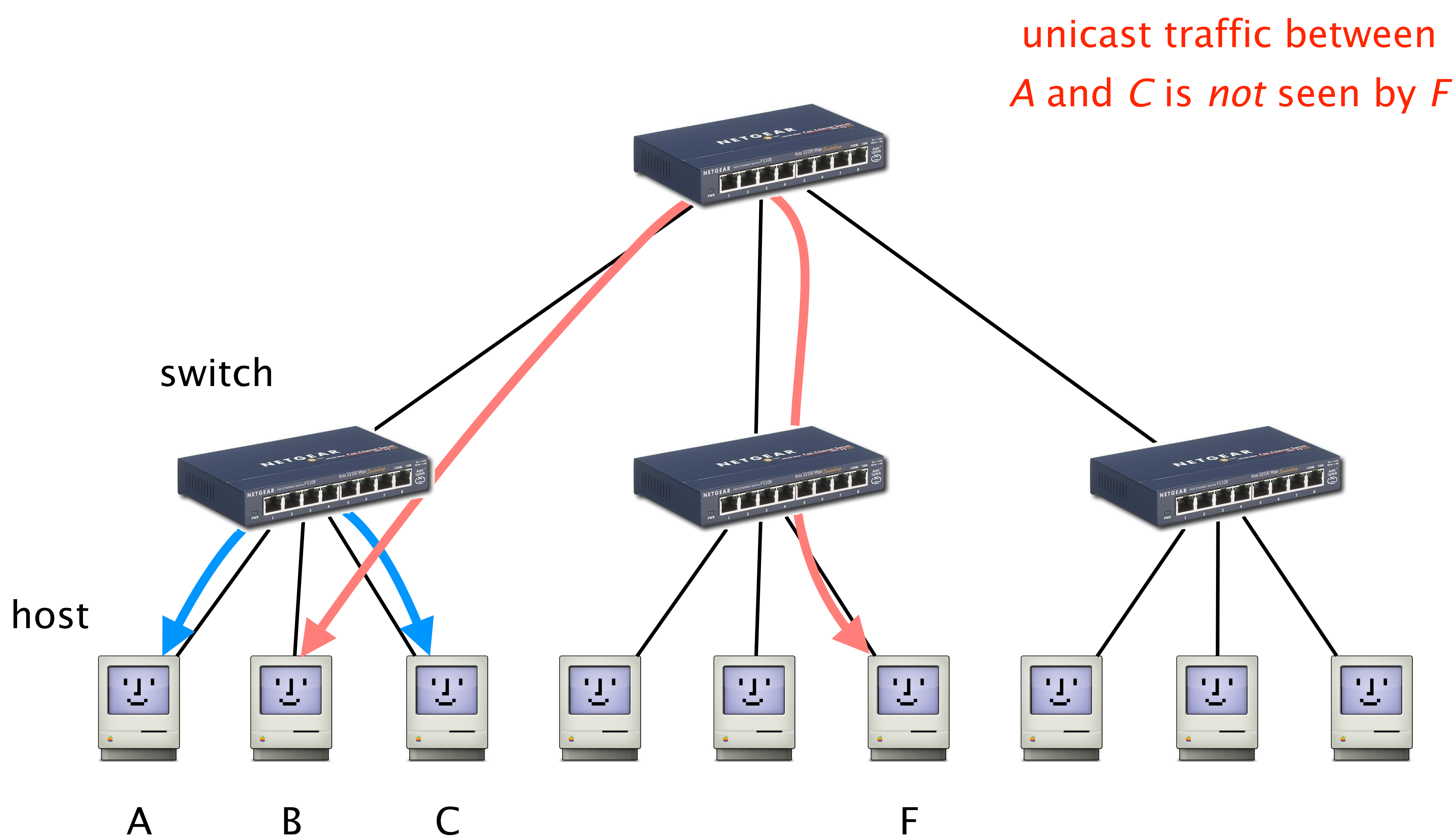
Switches connect two or more LANs together at the Link layer, acting as L2 gateways

Switches are "store-and-forward" devices, they

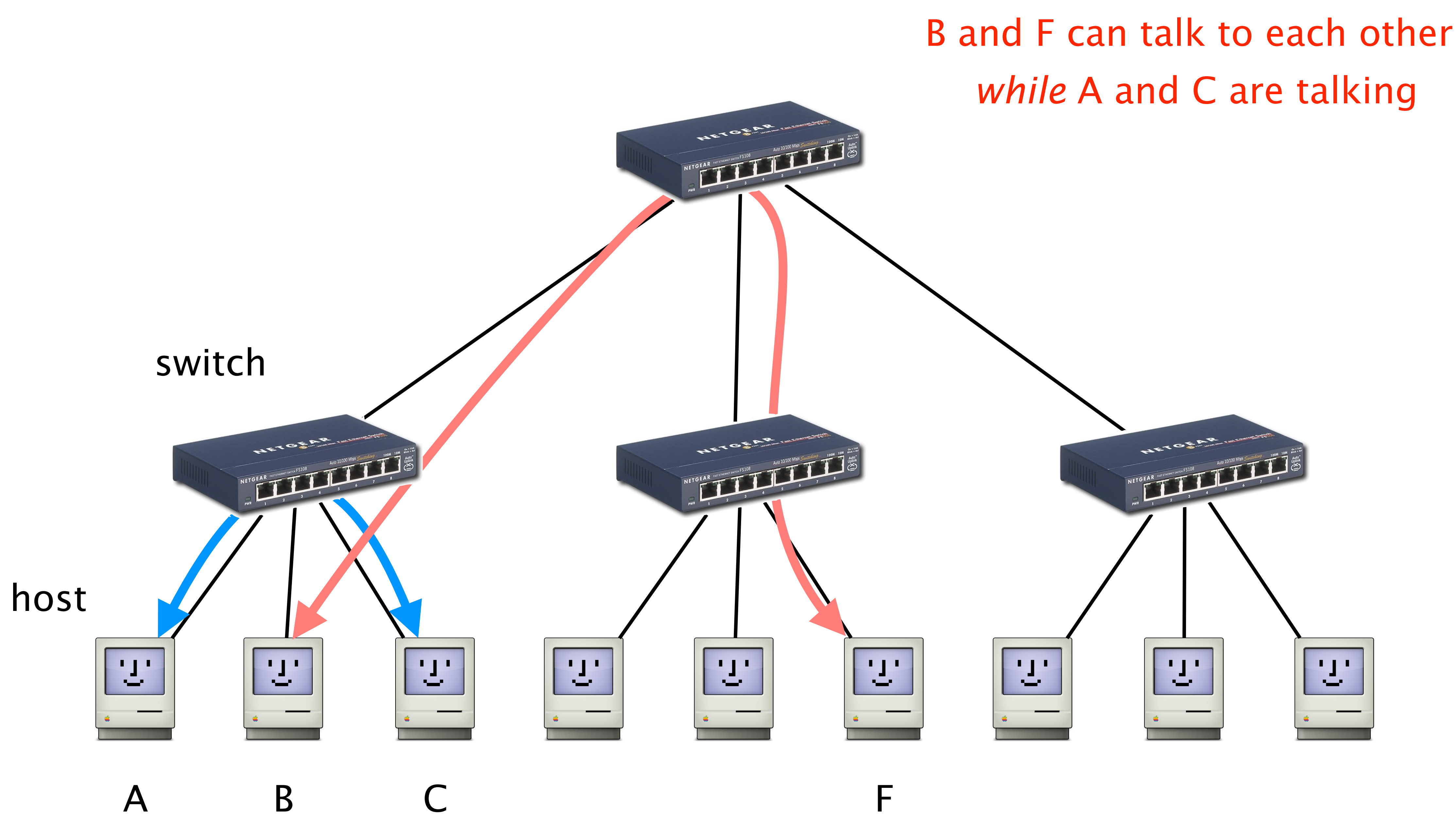
- extract the destination MAC from the frame
- look up the MAC in a table (using exact match)
- forward the frame on the appropriate interface

Switches are similar to IP routers,
except that they operate one layer below

Unlike with hubs, switches enable each LAN segment to carry its own traffic



Unlike with hubs,
switches supports concurrent communication



The advantages of switches are numerous

advantages

only forward frames where needed

avoids unnecessary load on segments

join segment using different technologies

improved privacy

hosts can only snoop traffic traversing their segment

wider geographic span

separates segments allow longer distance

Switches are plug-and-play devices,
they build their forwarding table on their own

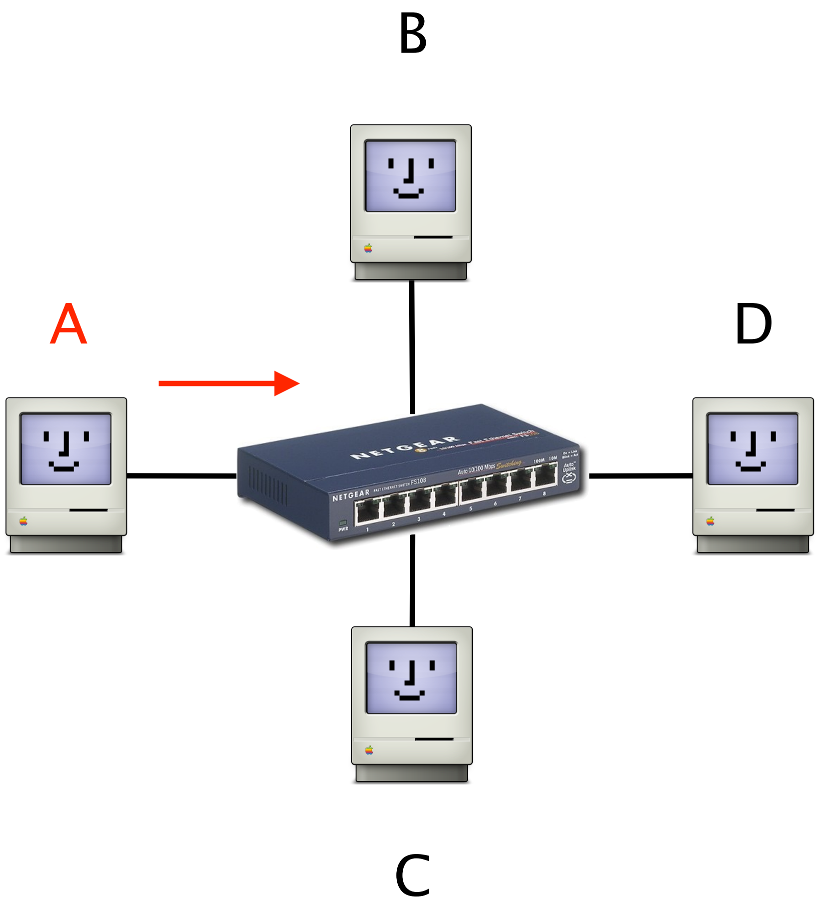
Switches are “store-and-forward” devices, they

- extract the destination MAC from the frame
- look up the MAC in a table (using exact match)
- forward the frame on the appropriate interface

Switches are plug-and-play devices, they build their forwarding table on their own

When a frame arrives:

- inspect the source MAC address
- associate the address with the port
- store the mapping in the switch table
- launch a timer to eventually forget the mapping



switch learns how to reach A

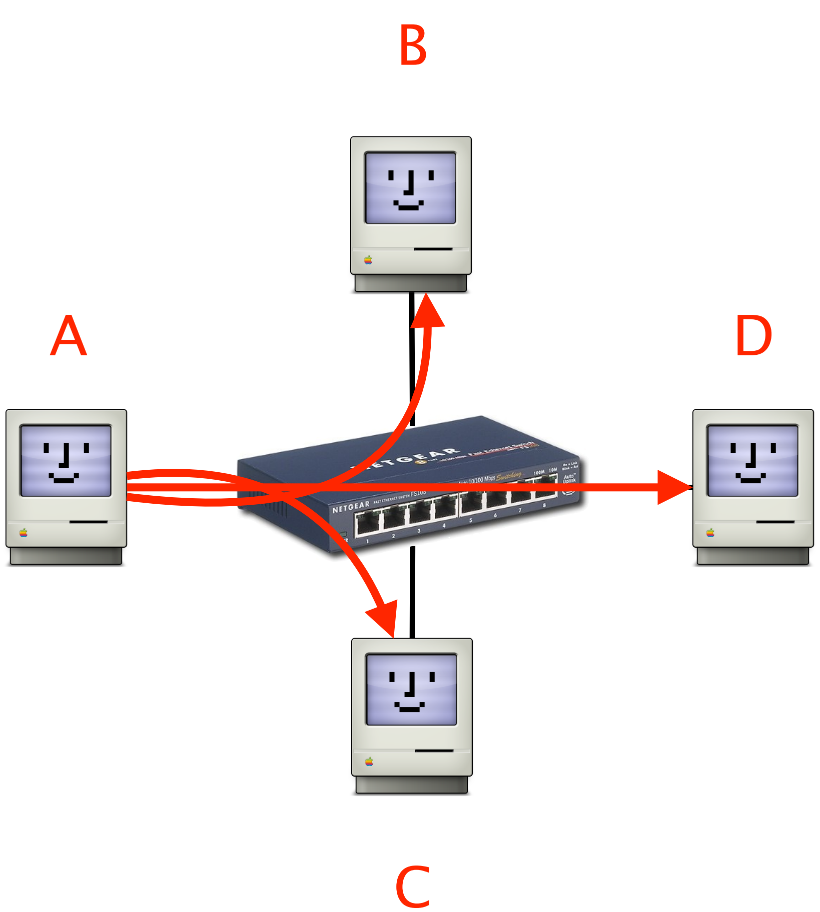
In cases of misses, switches simply floods the frames

When a frame arrives with an unknown destination

- forward the frame out of all interfaces

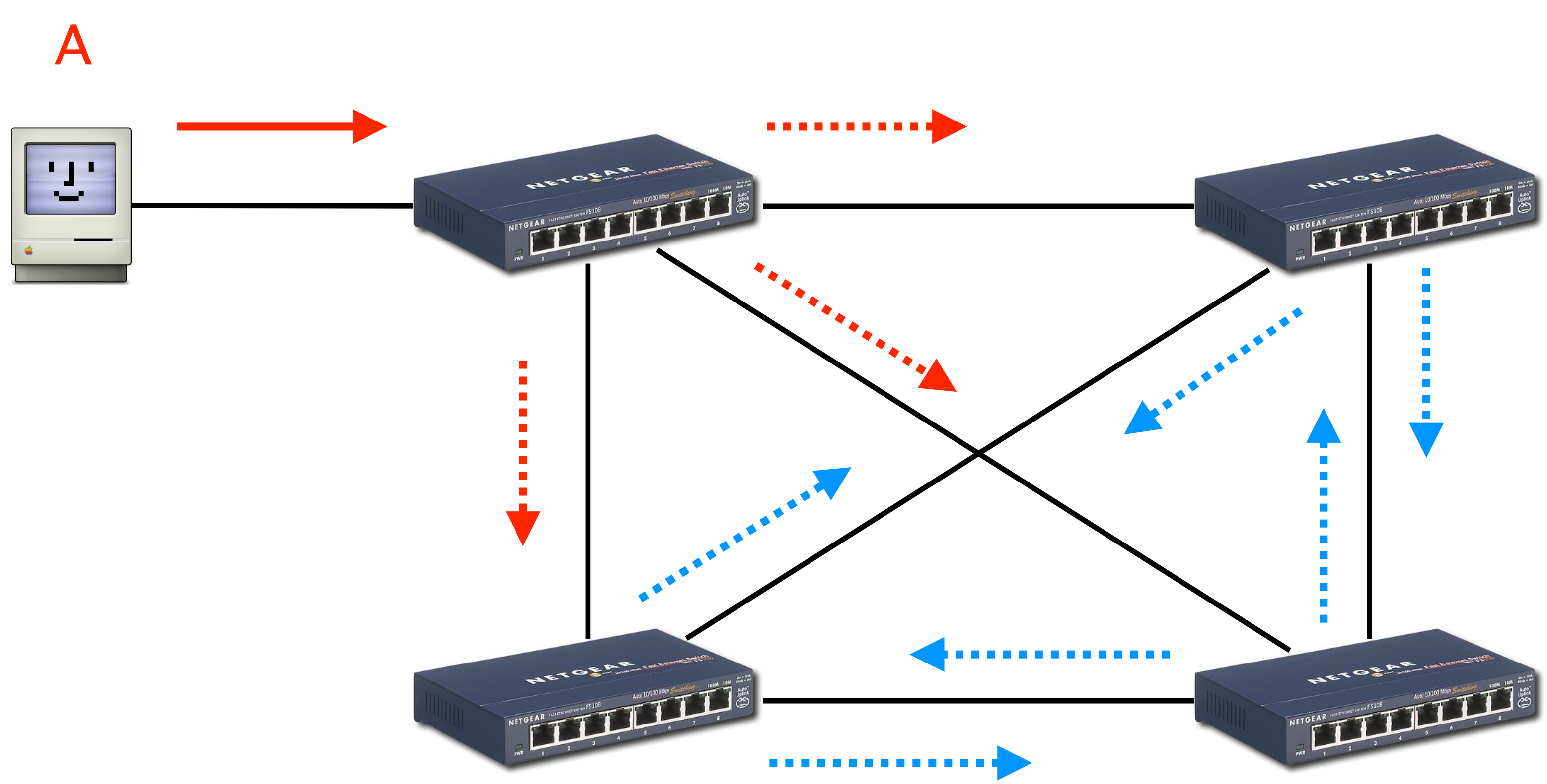
  except for the one where the frame arrived

Hopefully, this is an unlikely event



when in doubt, **shout!**

While flooding enables automatic discovery of hosts,
it also creates problems when the networks has loops



Each frame leads to the creation of *at least two new frames!*

exponential increase, with no TTL to remove looping frames…

While loops create major problems,
networks need redundancy for tolerating failures!

solution

Reduce the network
to one logical spanning tree

Upon failure,
automatically rebuild a spanning tree

In practice, switches run
a *distributed* Spanning-Tree Protocol (STP)

Algorhyme

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.

A tree that must be sure to span
So packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.

Least-cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
Then bridges find a spanning tree.

— *Radia Perlman*

A tree that must be sure to span
So packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.

Least-cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
Then bridges find a spanning tree.

Constructing a Spanning Tree in a nutshell

Switches…

elect a root switch

the one with the smallest identifier

determine if each interface is

on the shortest-path from the root

and disable it if not

For this switches exchange Bridge Protocol Data Unit (BDPU) messages

Each switch *X* iteratively sends

BPDU (Y, d, X) to each neighboring switch

the switch ID it considers as root

the # hops to reach it

initially

Each switch proposes itself as root

sends (X,0,X) on all its interfaces

Upon receiving (Y, d, X), checks if Y is a better root

if so, considers Y as the new root, flood updated message

Switches compute their distance to the root, for each port

simply add 1 to the distance received, if shorter, flood

Switches disable interfaces not on shortest-path tie-breaking

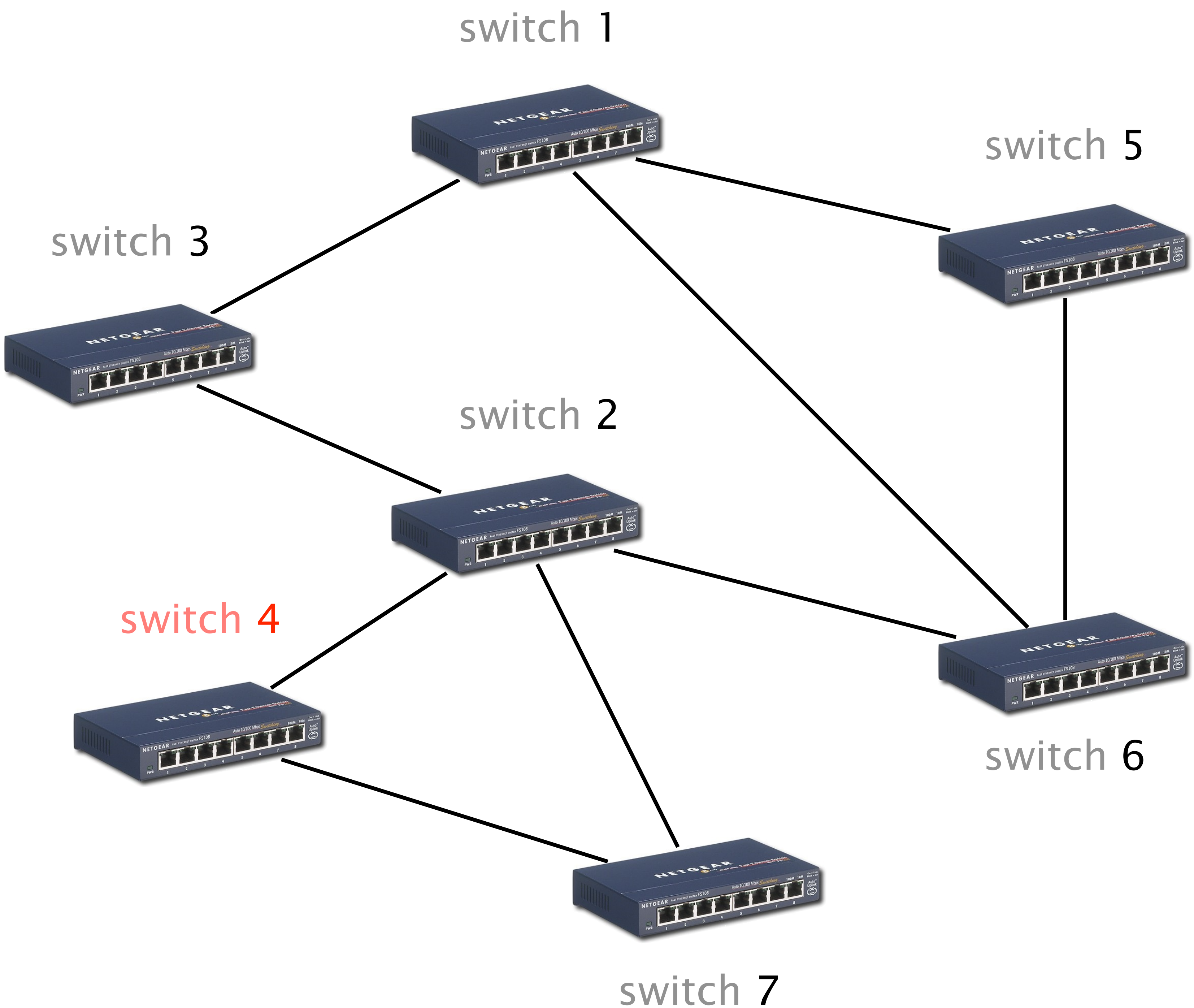
Upon receiving ≠ BPDUs from ≠ switches with = cost

Pick the BPDU with the lower switch sender ID

Upon receiving ≠ BPDUs from a neighboring switch

Pick the BPDU with the lowest port ID (e.g. port 2 < port 3)

Apply the algorithm starting with switch 4



To be robust,
STP must react to failures

Any switch, link or port can fail

including the root switch

Root switch continuously sends messages

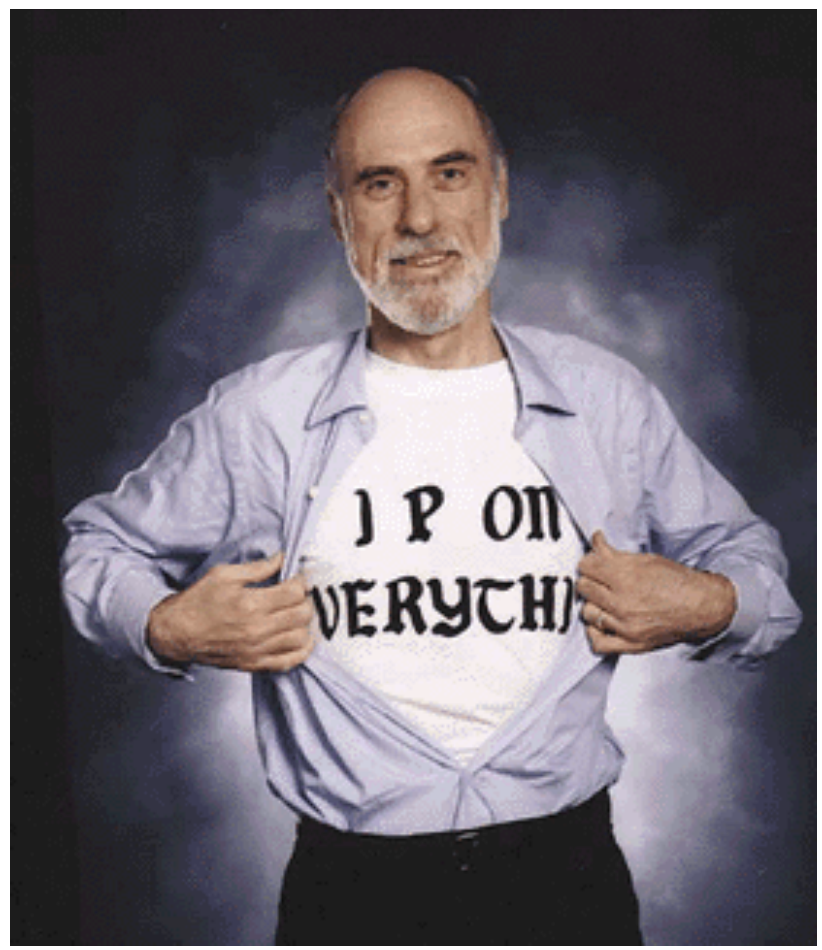
announcing itself as the root (1,0,1), others forward it

Failures is detected through timeout (soft state)

if no word from root in *X*, times out and claims to be the root

Next week on Communication Networks

Ethernet (end) + Internet Protocol (IP)



source: Boardwatch Magazine

Mark the date:
Internet Routing Project

Group Registration

March 18, register in groups of three

Project Start

March 25

Hackathon

April 4, 18:00

Communication Networks

Spring 2019

Alexander Dietmüller

nsg.ee.ethz.ch

ETH Zürich (D-ITET)

March 11 2019