CS4410/11: Operating Systems

CPU Scheduling (Recap)

Networking

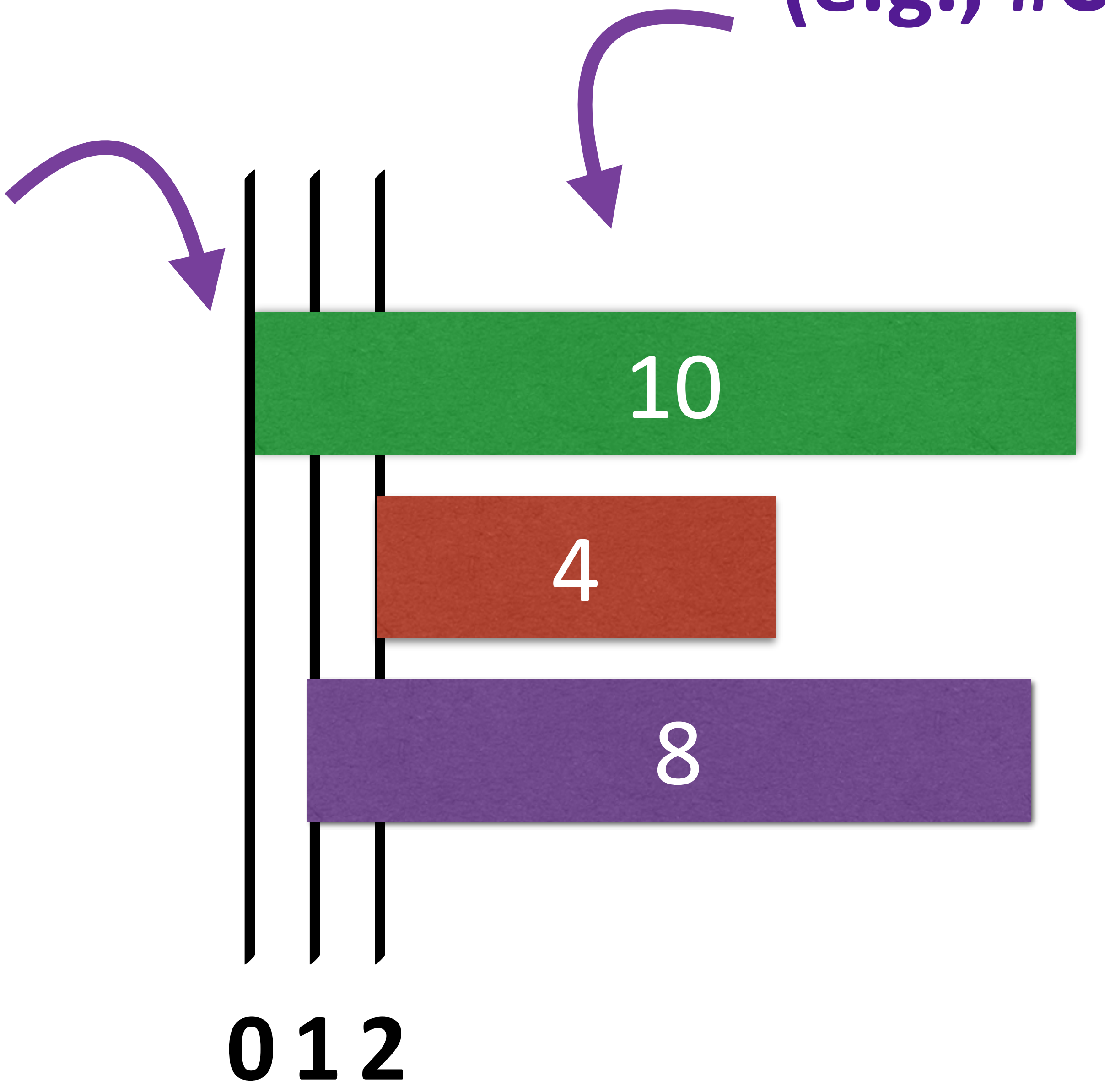
Rachit Agarwal
Anne Bracy

Slides based on material from Sirer, Rennesse, Rexford (Princeton)

CPU Scheduling — Example

Job Length
(e.g., #CPU cycles)

Arrival
Time



FIFO

LIFO

SJF

SRTF

RR

Priority

Networking — What is it about?

So far: focused on what happens on a “machine”!

- **Networking**
  - How do machines communicate?
- **Lets start with a simple analogy**
  - How to move stuff from München to Ithaca?

Networking — Key Concepts

Four "concepts"!

- **Layering**
    - Abstraction is the key to manage complexity
- **Naming**
    - A name for each computer, protocol, ..
- **Protocols**
    - Computers, network devices speaking the same language
- **Resource Allocation**
    - Share resources (bandwidth, wireless spectrum, paths, …)

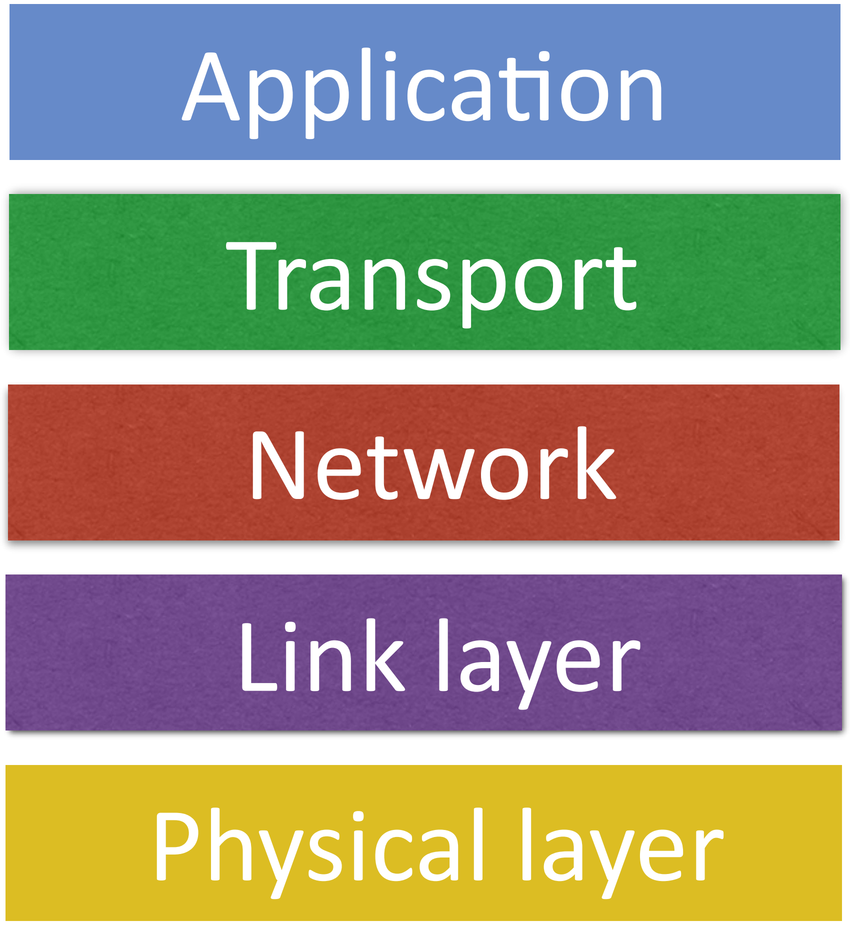
Networking — A Stack of Protocol Layers

Five "layers"!

- **Modularity**
  - Each layer relies on services from layer below
  - Each layer exports services to layer above
- **Interfaces**
  - Hide implementation details
  - Layers can change without disturbing other layers

Networking — A Stack of Protocol Layers

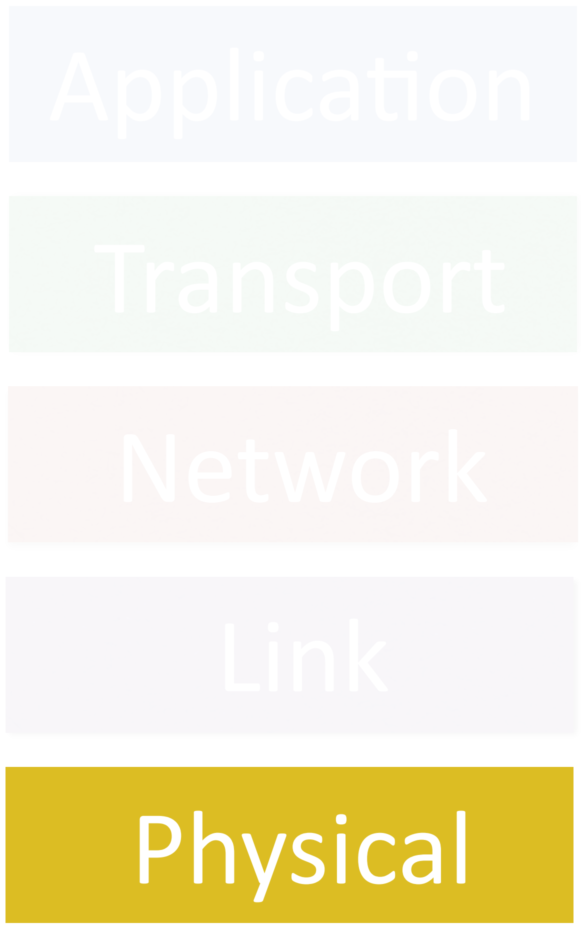
Five “layers”!



Application	You
Transport	Post office
Network	Airplane/rail
Link layer	Postman
Physical layer	Transfer “signals”

Networking — Physical layer

- **Transfer of bits**
  - 0s and 1s
  - Not concerned with protocols



Application

Transport

Network

Link

Physical

Networking — Link layer

Application

Transport

Network

Link

Physical

Link = Medium + Adapters

- Communication Medium

- Network Adapters (e.g., NIC — network interface card)

Networking — Link layer

Application

Transport

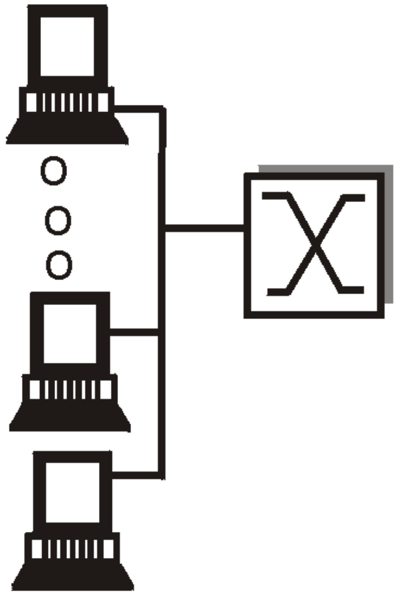
Network

Link

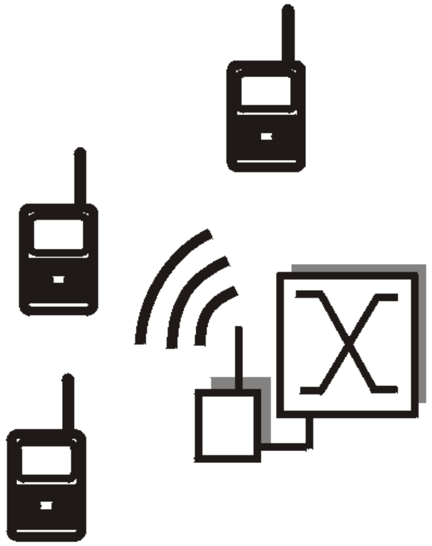
Physical

Broadcast links = Shared Medium

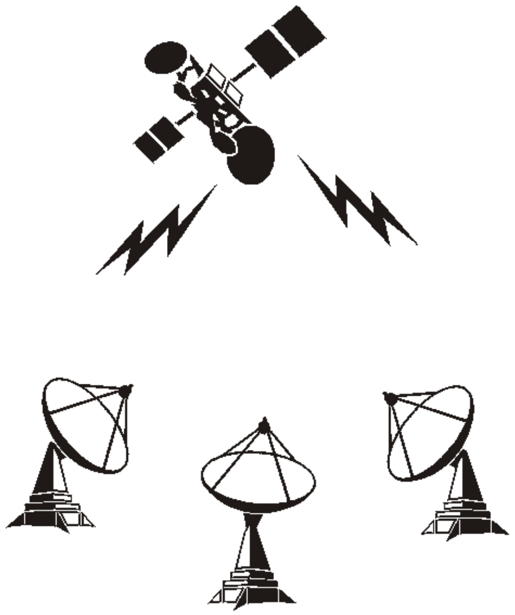
- Everyone listens to everybody



shared wire
(e.g. Ethernet)



shared wireless
(e.g. Wavelan)



satellite

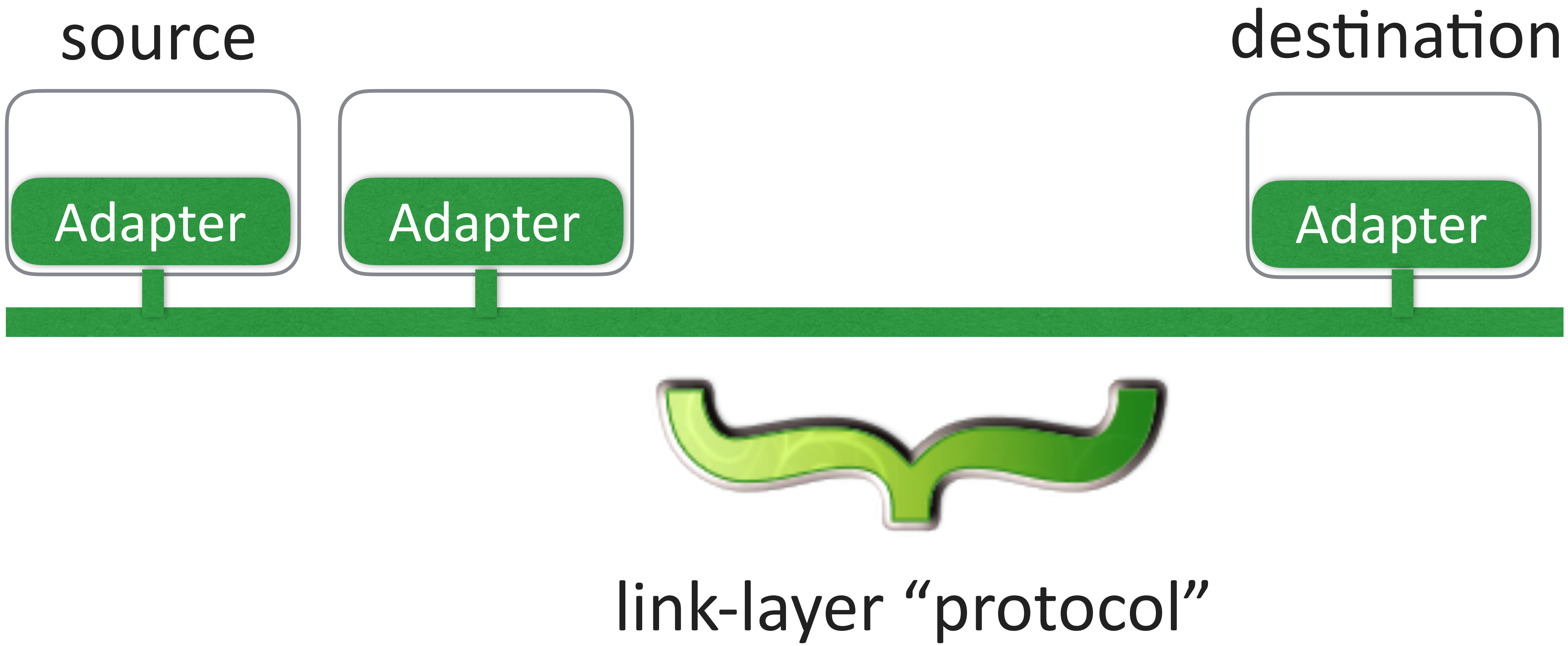
cocktail party

Networking — Link layer

Application
Transport
Network
Link
Physical

Broadcast links = Shared Medium

- Everyone listens to everybody



source

destination

Adapter

Adapter

Adapter

link-layer “protocol”

Networking — Link layer

Five “services”!

- **Encoding data**
  - Represented as a collection of 0s and 1s
- **Framing**
  - Put data packet into a frame; add receiver address
- **Error detection and correction**
  - Detect and (optionally) correct errors
- **Flow control**
  - When to send/receive frames
  - Depends on the protocol

Networking — Link layer

Addresses

- **Unique identifiers for sources and destinations**
  - “Hard-coded” in the adapter
  - MAC address (e.g., 00-15-C5-49-04-A9)
  - Hierarchical allocation
    - Blocks: assigned to vendors (e.g., Dell) from IEEE
    - Adapters: assigned by the vendor from its block
- **What if I want to send to everybody?**
  - Special (broadcast) address: FF-FF-FF-FF-FF-FF

Networking — Link layer

Sharing a medium

- **Ever been to a party?**
  - Tried to have an interesting discussion?
- **Collisions**

Link layer — Sending/receiving

Lets try to come up with a protocol to avoid collisions!

- **Attempt 1: Time sharing**
  - Everybody gets a turn to speak
- **Goods**
  - Never have a collision
- **Problem**
  - Underutilization of resources
    - During my turn, I may have nothing to speak
    - When I have something to speak, I wait for my turn

Link layer — Sending/receiving

Lets try another protocol to avoid collisions

- **Attempt 2: Frequency sharing**
  - For wireless and optical mediums
  - Each source assigned a particular frequency; receivers tune
  - E.g., Divide into groups; each group talks among themselves
- **Problem**
  - Overheads …
  - What if I want to talk to only a few people in the group?
  - What if I want to talk to people in different groups?
  - E.g., one person wants to announce something …

Link layer — Sending/receiving

Attempt 3: Carrier sense, Collision detection, Random access

- **Carrier Sense**
  - Listen before speaking
  - …. and don’t interrupt
- **Collision detection**
  - Detect simultaneous speaking
  - …. and shut up!
- **Random access**
  - Wait for a random period of time
  - …. before trying to talk again

Link layer — Sending/receiving

Comparing the three approaches

- **Time division**
  - No collisions
  - Underutilization of resources!
  - What if token is lost?
- **Frequency division**
  - Overheads
- **Random access**
  - Efficient at low load, inefficient at high load (collisions)

Ethernet — Sending/receiving at Link layer

Ethernet uses CSMA/CD

- **Carrier Sense: continuously listen to the channel**
  - If idle: start transmitting
  - If busy: wait until idle
- **Collision Detection: listen while transmitting**
  - No collision: transmission complete
  - Collision: abort transmission; send jam signal
- **Random access: exponential back off**
  - After collision, transmit after "waiting time"
  - After k collisions, choose "waiting time" from {0, …, 2^k-1)
  - (Exponentially increasing waiting times)

Networking — Link layer (Ethernet)

Interesting Properties

- **Distributed**
    - **No** Central arbitrer
    - Why is that good?
- **Inexpensive**
    - No state in the network
    - Cheap physical links

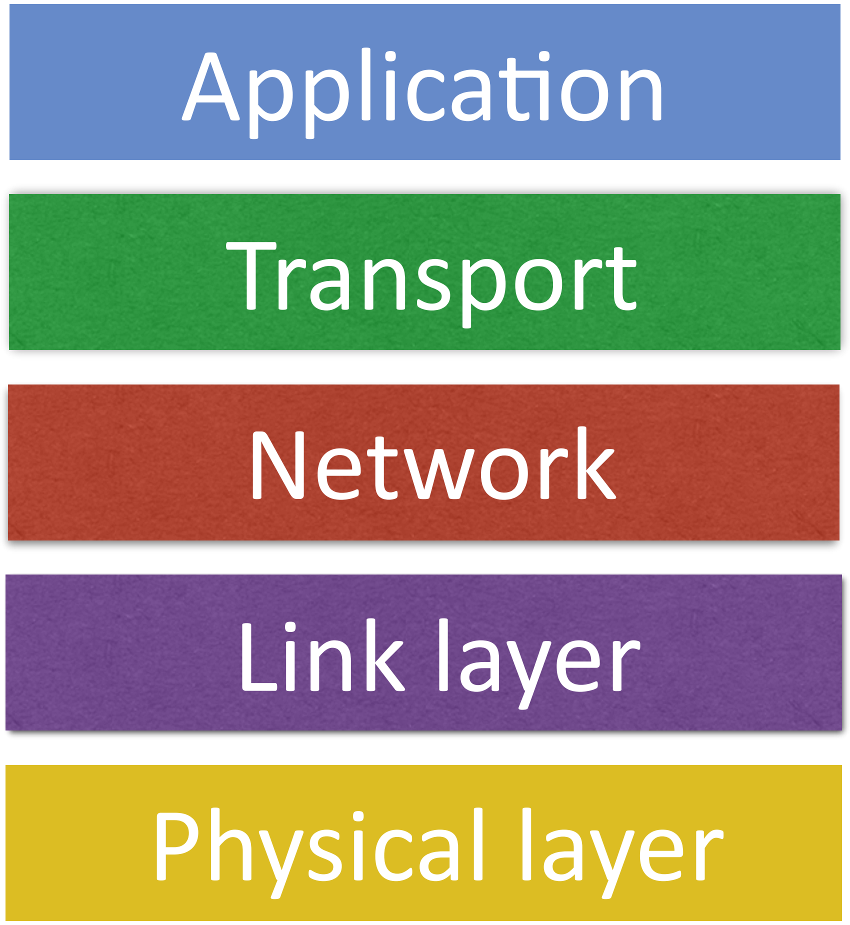
Networking — Link layer (Ethernet)

Connection-less, unreliable service

- **Connection less**
  - E.g., I am going to talk to you without getting permission first
  - Networking terminology: No "handshaking"
- **Unreliable**
  - Destination adapter does not acknowledge
    - Did you listen to what I said?
  - Adversarial behavior could bring the connections down
    - I am going to ignore the protocol
  - Untrusted data access
    - I want to listen to what others are talking

Networking — A Stack of Protocol Layers

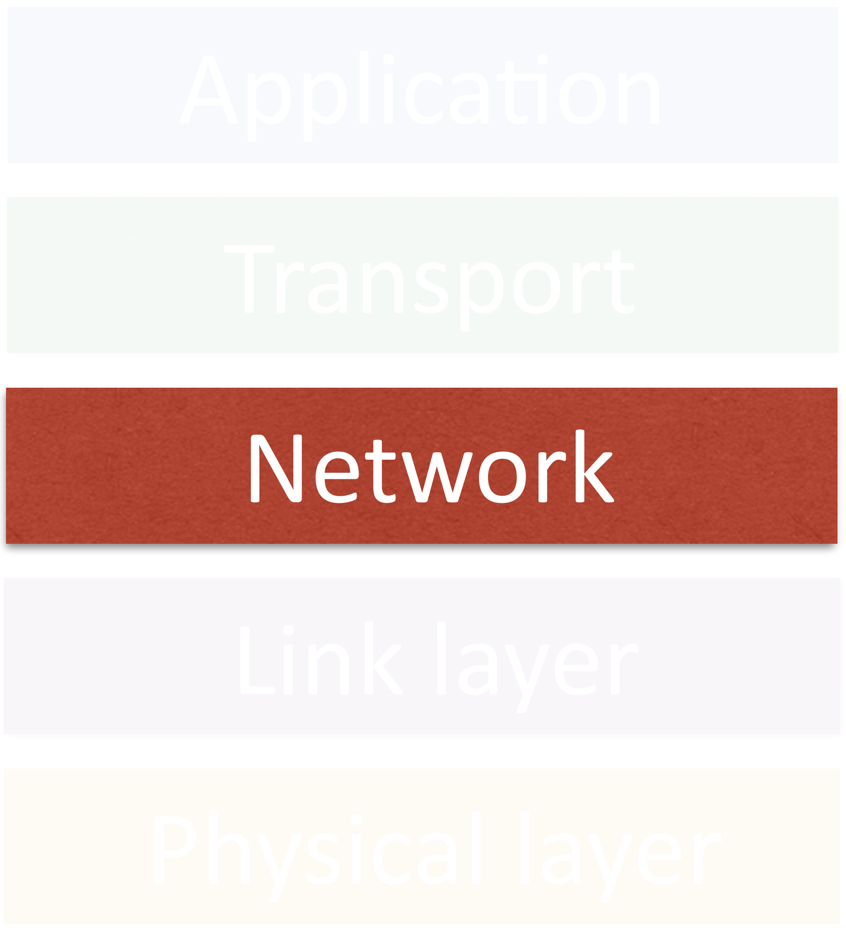
Five “layers”!



Application	**Deliver**
Transport	Deliver **(un)reliably**
Network	Deliver **globally**
Link layer	Deliver **locally**
Physical layer	Deliver **signals**

Networking — A Stack of Protocol Layers

Five “layers”!



Application	Deliver
Transport	Deliver (un)reliably
Network	Deliver **globally**
Link layer	Deliver locally
Physical layer	Deliver signals

Networking — Network layer

Three concepts

- **Naming**
  - A way to identify the source/destination
  - E.g., house address
- **Routing**
  - Finding “how to” move towards the destination
  - E.g., which airplane should the stuff go on
- **Forwarding**
  - Actually “moving” towards the destination
  - E.g., Using airplane/truck/rail

Networking — Network layer

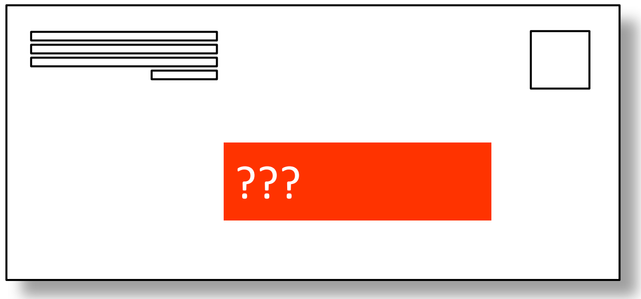
Naming

- **Give every computer a unique name**
  - Challenges?
  - Scalability — why?
  - Assignment — why?

Networking — Network layer

Naming

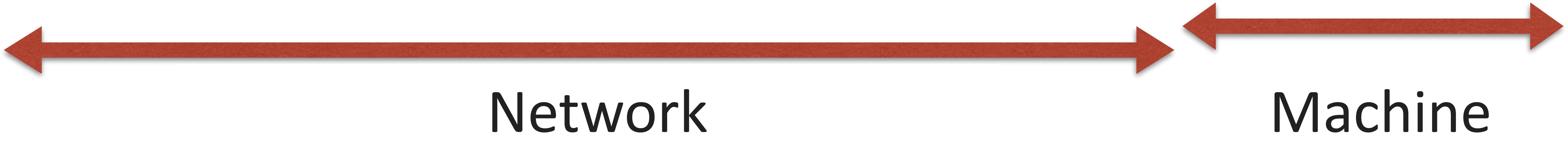
- **Hierarchical addressing**
    - E.g., addresses for houses
    - **Country:** USA
    - **City, State:** Ithaca, NY
    - **Number, Street**: 306 State St.
    - **Name:** Rachit Agarwal



Networking — Network layer

Hierarchical addressing

Country	City, State	Street, Number	Occupant
(8 bits)	(8 bits)	(8 bits)	(8 bits)
10000000	0-1010100	10001011	00000-101
128	84	139	5



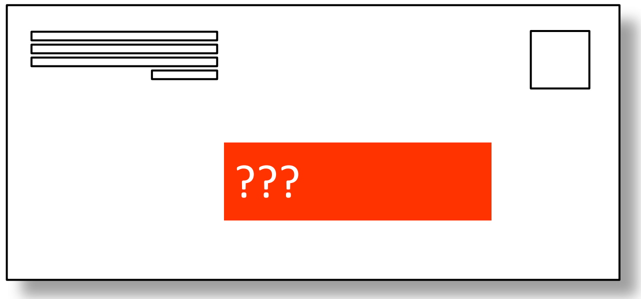
Network

Machine

IP address: 128.84.139.5

Networking — Network layer

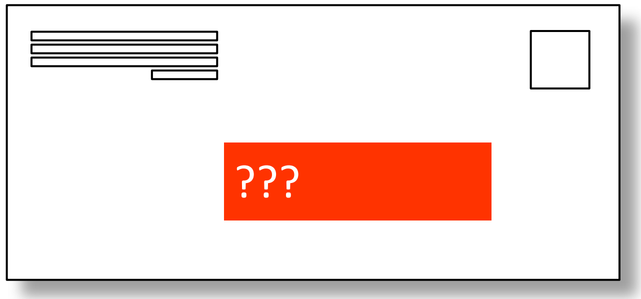
Hierarchical addressing



- **Why is it more scalable?**
  - Need to keep track of next step only!
  - **Flight to:** USA
  - **Truck to:** Ithaca, NY
  - **Direction to**: 306 State St.
  - **Mailbox:** Rachit Agarwal

Networking — Network layer

Hierarchical addressing



- **Why is it easier to assign?**
    - Just assign a new machine a “local” address!
    - **E.g.,** adding a new machine to Cornell network
    - **If last local address:** 128.84.139.5
    - **New machine gets**: 128.84.139.6

Networking — Network layer

Three concepts

- Naming
  - A way to identify the source/destination
  - E.g., house address
- Routing
  - Finding “how to” move towards the destination
  - E.g., which airplane should the stuff go on
- **Forwarding**
  - Actually “moving” towards the destination
  - E.g., Using airplane/truck/rail

Network layer — Forwarding

Lets come up with an approach? Generalize Ethernet ideas?

Network layer — Forwarding

Attempt 1: Broadcast

- **Send to everybody**
- **Goods**
    - Oh, well, simplicity
- **Not-so-goods**
    - Oh, well, everything else
    - Bandwidth overheads

Network layer — Forwarding

Attempt 2: Time division Multiplexing

- **Each source-destination pair assigned a time slot**
  - Can send data only during that slot
- **Goods**
  - No collisions
- **Not-so-goods**
  - Underutilization of resources

Network layer — Forwarding

Attempt 3: Frequency division Multiplexing

- **Each source-destination pair assigned a subset of resources**
  - Can use only “assigned” resources (e.g., bandwidth)
- **Goods**
  - Predictable performance
- **Not-so-goods**
  - Underutilization of resources

Network layer — Forwarding

Attempt 2 and 3: Circuit Switching

- **Source establishes connection**
    - Resources along the path are reserved
- **Source sends data**
    - Transmit data using the reserved resources
- **Source tears down connection**
    - Free resources for others to use

Network layer — Forwarding

Circuit Switching

- **Goods:**
  - Predictable performance
  - Reliable delivery
  - Simple forwarding mechanism
- **Not-so-goods**
  - Resource underutilization
  - Blocked connections
  - Connection set up overheads
  - Per-connection state in switches (scalability problem)

Network layer — Forwarding

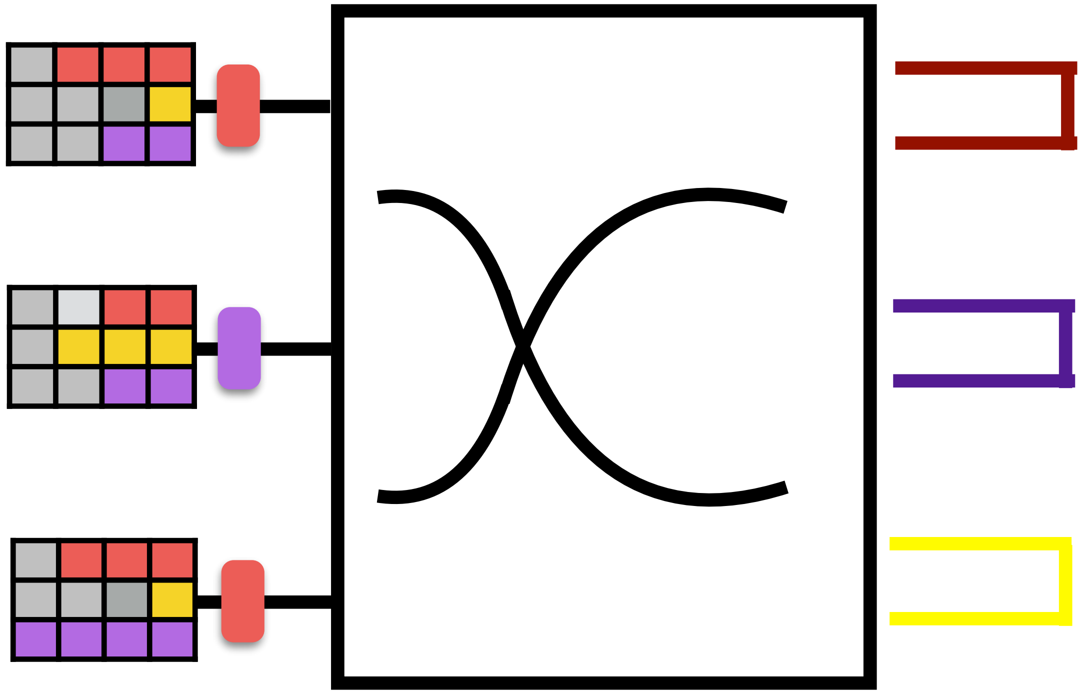
Attempt 4: Packet Switching

- **Divide the message into packets**
- **Put destination address in the header of each packet**
    - Just like shipping stuff
- **Each device stores a “look-up table”**
    - Whats the next hop towards the destination?
- **Destination receives the packet(s)**
    - And reconstructs the message

Network layer — Forwarding

Packet Switched forwarding

- **Hop-by-hop forwarding**
- **Each router has a "look-up table" (forwarding information base)**
    - What should be stored in this table?
    - Prefix-based forwarding **(longest-prefix matching)**
        - Maps **prefixes** to the next-hop



Network layer — Forwarding

Packet Switching

- **Goods:**
    - No resource underutilization
        - A source can send more if others don’t use resources
    - No blocked connection problem
    - No per-connection state
    - No set-up cost
- **Not-so-goods:**
    - Packet header overhead
    - Network failures become a problem