CS4450

Computer Networks:
Architecture and Protocols

Lecture 3
- "Packets" and Delays
- How the Internet works

Rachit Agarwal

Announcements

- **Ed Discussions has been set up**
- **We will release office hours on Wednesday**
- **Email strategy**
    - This is a large class
    - Hard for me to carefully answer each and every email
    - Please send an email to cs4450questions@gmail.com
    - Direct emails to me only if they are for my eyes only (give reason)

Context for and Goals of Today's Lecture

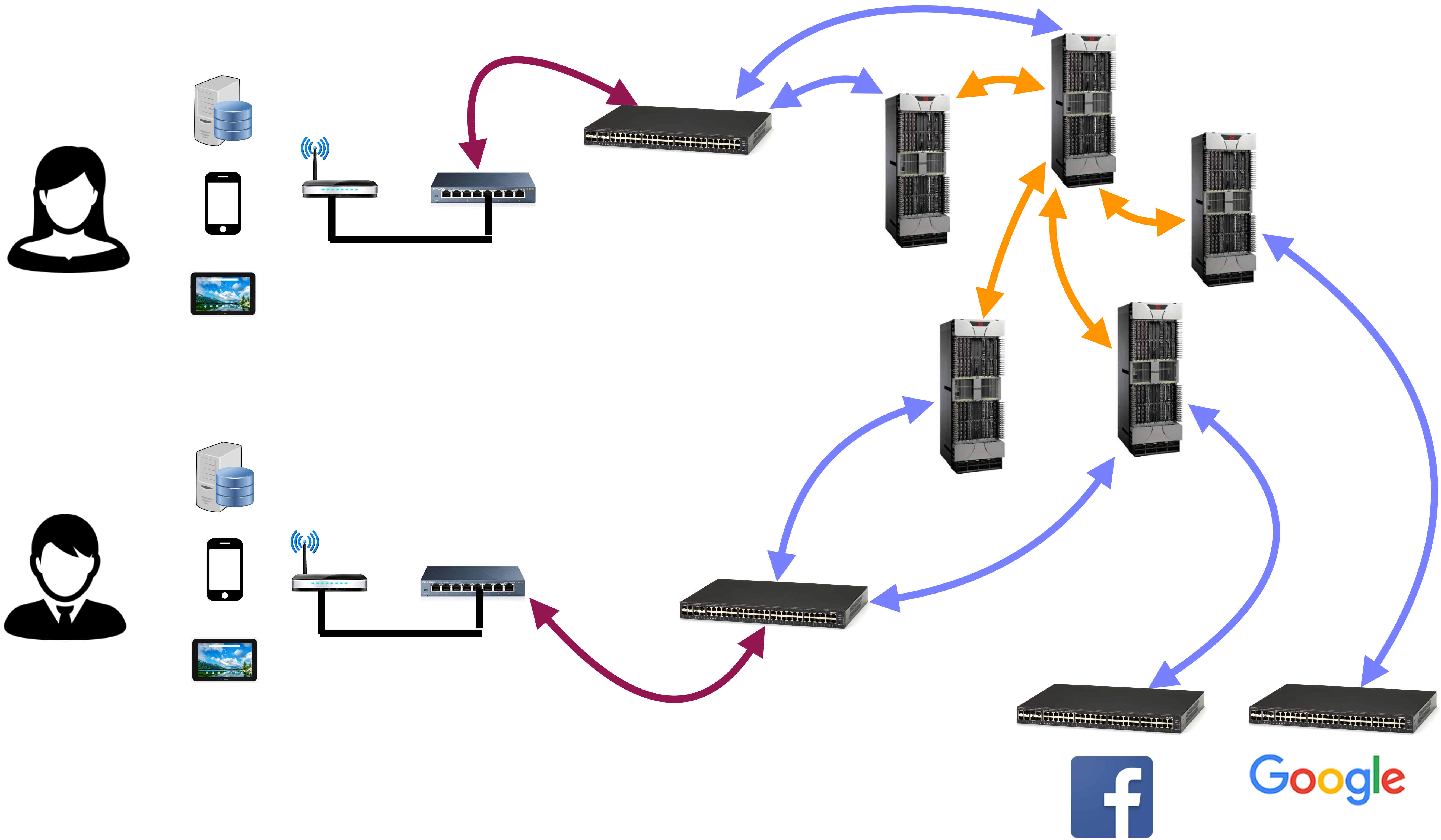
- **Context:**
  - Today's lecture is going to be one of the hardest lectures
  - If you understand everything
    - There is something wrong!
- **Goals:**
  - **Propagation and transmission delays**
  - **How does the Internet work?**
    - An end-to-end view

But, lets start with:

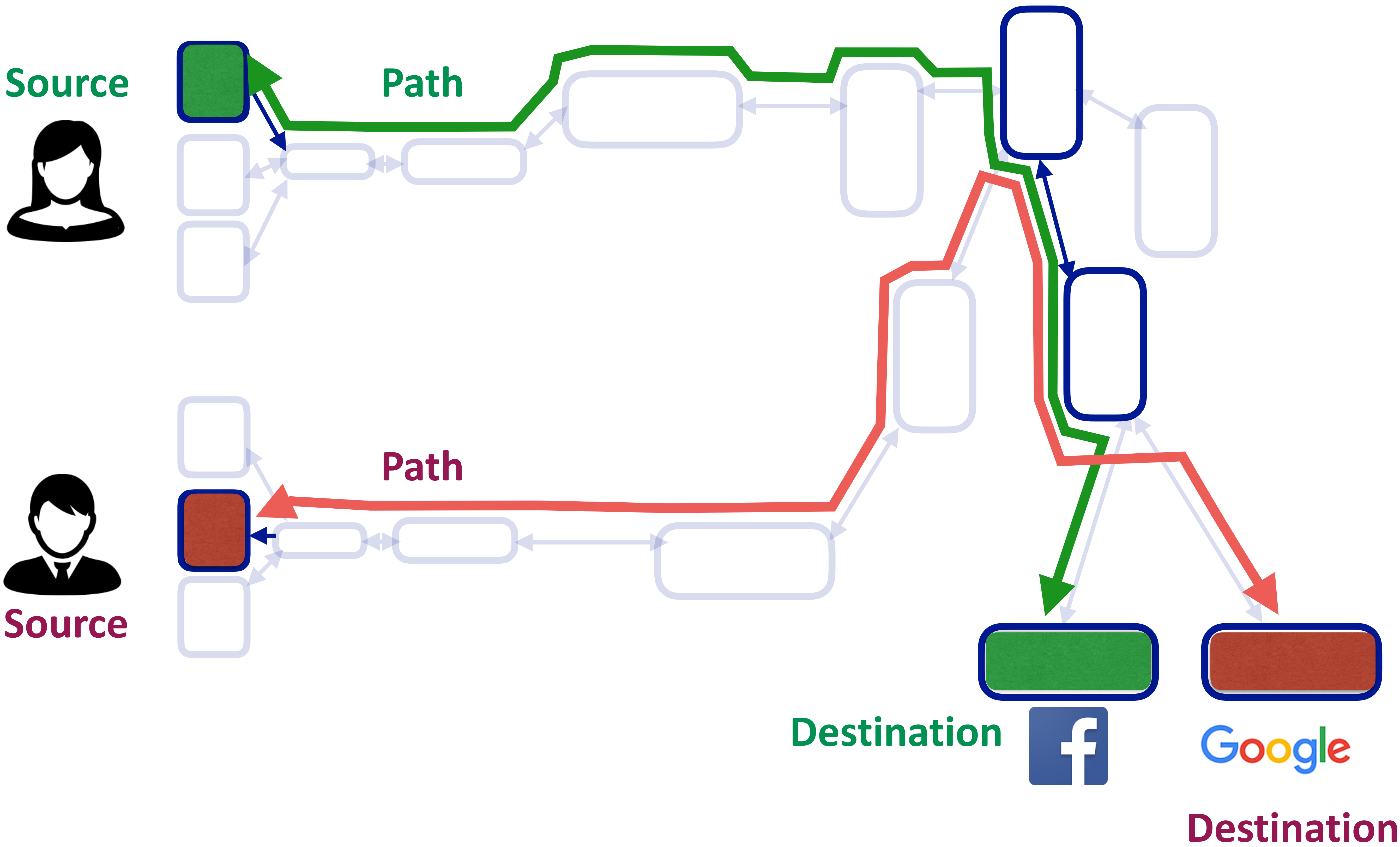
what we learnt last lecture

Recap: What is a computer network?

A set of network elements connected together, that implement a set of protocols for the purpose of sharing resources at the end hosts



Recap: Sharing the network



Recap: Performance metrics in computer networks!

- **Bandwidth: <u>Number of bits</u> sent per unit time** (bits per second, or bps)
  - Capacity: depends on hardware
  - Utilization: depends on network traffic conditions
- **Propagation delay: Time for <u>one</u> bit** to move through the link (seconds)
- **Many other performance metrics**
  - Reliability, fairness, etc.
  - We will come back to these later.

Recap: Two approaches to sharing networks

- **First: Reservations**
    - Reserve (peak) bandwidth needed in advance

- **One way to implement reservations: circuit switching**

Circuit switching summary

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

Recap: Solution: Packet switching

- **Break data into smaller pieces**
  - **Packets!**
- **Transmit the packets without any reservations**
  - And, hope for the best

Recap: Packet switching summary

- **Goods:**
  - Easier to handle failures
  - No resource underutilization
  - No blocked connection problem
  - No per-connection state
  - No set-up cost
- **Not-so-goods:**
  - Unpredictable performance
  - High latency
  - Packet header overhead

Questions?

Summary of network sharing

Statistical multiplexing

- **Statistical multiplexing:** combining demands to share resources efficiently

- Long history in computer science
  - Processes on an OS (vs every process has own core)
  - Cloud computing (vs every one has own datacenter)

- Based on the premise that:
  - **Sum of instantaneous demands << sum of peak demands**

- Therefore, it is better to share resources than to strictly partition them …

Two approaches to sharing networks

Both embody statistical multiplexing

- Reservation: sharing at <u>connection</u> level
  - Resources shared between connections currently in system
  - Reserve the peak demand for a flow

- On-demand: sharing at <u>packet</u> level
  - Resources shared between packets currently in system
  - Resources given out on packet-by-packet basis
  - No reservation of resources

Questions?

Understanding delay/latency

Packet Delay/Latency

- Consists of four components
  - **Propagation delay** (hardware properties, distance)
  - **Transmission delay** (hardware properties)
  - **Queueing delay** (traffic, switch internals)
  - Processing delay (end hosts)
- First, consider propagation and transmission delays
- Then queueing delay
- Ignore processing delays (for now)

Propagation delay

- **How long does it take to move <u>one bit</u> from one end of the link to other?**

- **Link length / Propagation speed of link**
  - Propagation speed ~ some fraction of speed of light

- Example:
  - Length = 30,000 meters
  - Delay = 30*1000/3*100,000,000 second = 100us

Transmission delay

- **How long does it take to push all the bits of a packet into a link?**

- **Packet size / Transmission rate of the link**
  - Transmission rate = Share of Bandwidth

- Example:
  - Packet size = 1000Byte
  - Rate = 100Mbps
  - 1000*8/100*1024*1024 seconds ~76.3us

Questions?

Group Exercise 3:

How long does it take for a *packet* to go from one end of the link to another end?

Constraints:

- Packet size = 1000Byte
- Rate = 100Mbps
- Length = 30,000m

Answer: 176.3us

Why?

Group Exercise 3:

How long does it take for a *packet* to go from one end of the link to another end?

How does the Internet work?

An end-to-end view

The rest of the lecture

- Dive into **end-to-end:** from source to destination

- First look into **switches:** routing, queueing, forwarding

- First look into **network stacks:** sockets, ports, “the stack”

Four fundamental problems!

- **Naming, addressing:** Locating the destination
- **Routing:** Finding a path to the destination
- **Forwarding:** Sending data to the destination
- **Reliability:** Handling failures, packet drops, etc.

Four fundamental problems!

Naming, Routing, Forwarding, Reliability

- Each is motivated by a clear need
- The solutions are not always clean or deep
- **But if you keep in mind what the problem is**
    - You'll be able to understand the solutions
    - When the right time comes :-)

Will take the entire course to learn these:

Lets get an end-to-end picture!

Fundamental problem #1: Naming and Addressing

- **Network Address: where host is located**
  - Requires an address for the destination host
- **Host Name: which host it is**
  - why do we need a name?
- **When you move a host to new building**
  - Address changes
  - Name ***does not*** change
- **Same thing with your own name and address!**
- **Remember the analogy: human names, addresses, post office, letters**

Names versus addresses

- **Consider when you access a web page**
  - Insert URL into browser (eg, www.cornell.edu)
  - Packets sent to web site (reliably)
  - Packet reach application on destination host

- **How do you get to the website?**
  - URL is **user-level name** (eg, www.cornell.edu)
  - Network needs address (eg, where is www.cornell.edu)?

- Must map names to addresses
  - Just like we use an address book to map human names to addresses

Mapping Names to Addresses

- On the Internet, we only name hosts (sort of)
    - URLs are based on the name of the host containing the content (that is, www.cornell.edu names a host)

- Before you can send packets to www.cornell.edu, you must resolve names into the host's address

- Done by the **Domain Name System (DNS)**

The source knows the name;

Maps that name to an address using DNS!

Questions?

Fundamental problem #2

Routing packets through network elements (eg, routers) to destination

- Given destination address (and name), how does each switch/router know where to send the packet so that the packet reaches its destination

- When a packet arrives at a router
  - a **routing table** determines which outgoing link the packet is sent on
  - Computed using **routing protocols**

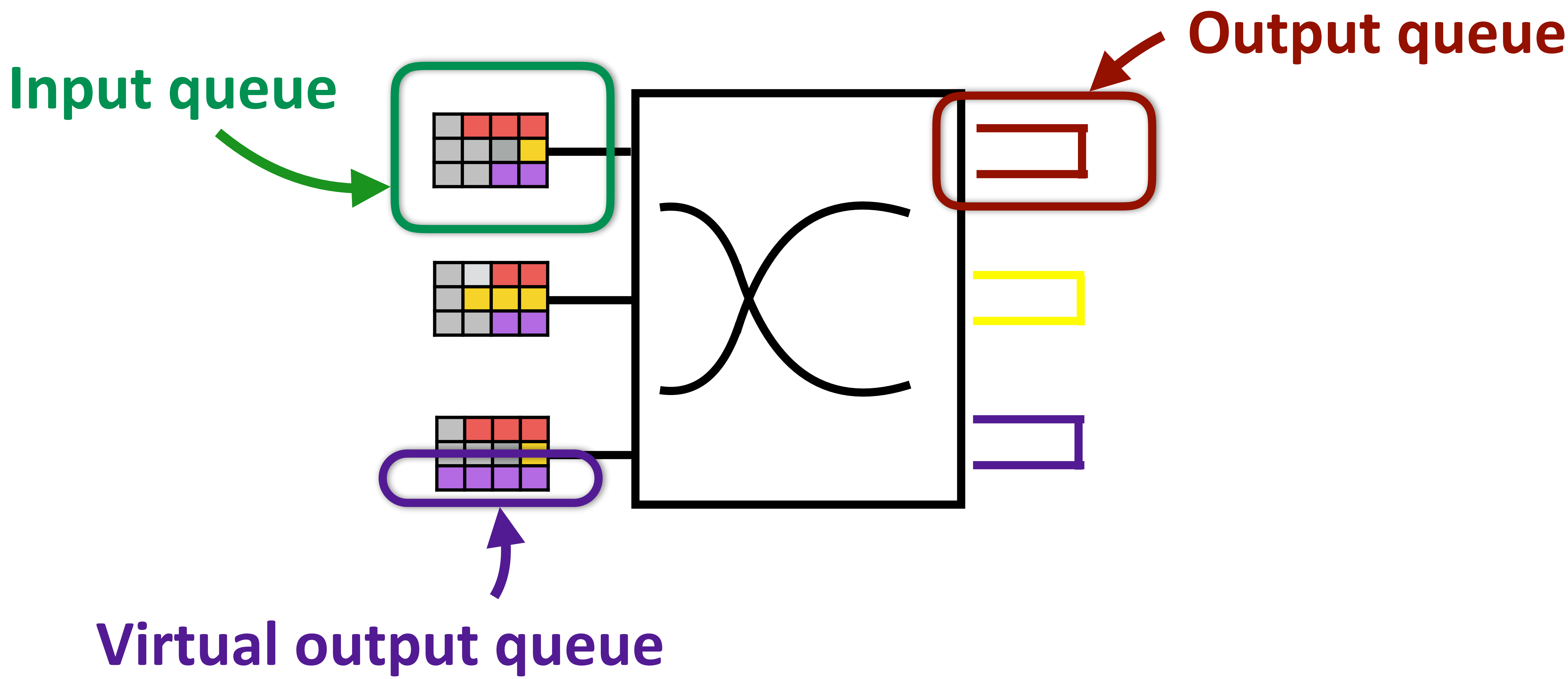
Routing protocols (conceptually)

- Distributed algorithm that runs between routers
  - Distributed means no single router has “full” view of the network
  - Exchange of messages to gather “enough” information …
- … about the network topology
- Compute paths through that topology
- Store forwarding information in each router
  - If packet is destined for X, send out using link l1
  - If packet is destined for Y, send out using link l2
  - Can packets going to different destinations sent out to same link?
- **We call this a routing table**

Questions?

Fundamental problem #3

Queueing and Forwarding of packets at switches/routers



Fundamental problem #3

Queueing and Forwarding of packets at switches/routers

- **Queueing:** When a packet arrives, store it in “input queues”
  - Each incoming queue divided into multiple virtual output queues
  - One virtual output queue per outgoing link
  - When a packet arrives:
    - Look up its destination’s address (how?)
    - Find the link on which the packet will be forwarded (how?)
    - Store the packet in corresponding virtual output queue

- **Forwarding:** When the outgoing link free
  - Pick a packet from the corresponding virtual output queue
  - forward the packet!

What must packets carry to enable forwarding?

- **Packets must describe where it should be sent**
  - Requires an address for the destination
- **Packets must describe where its coming from**
  - For handling failures, etc.
  - Requires an address for the source
- **Packets must carry data**
  - can be bits in a file, image, whatever

Switch Processing and Queueing delay

- **Processing delay**
    - Easy; each switch/router needs to decide where to put packet
    - Requires checking header, etc.

- **Queueing delay**
    - Harder; depends on “how many packets are in front of me”
    - Depends on network load
    - As load increases, queueing delay increases

- **In an extreme case, increase in network load**
    - results in packet drops

- We will return to this in much more depth later …

Questions?

Fundamental problem #4

How do you deliver packets reliable?

- Packets can be dropped along the way
  - Buffers in router can overflow
  - Routers can crash while buffering packets
  - Links can garble packets

- How do you make sure packets arrive safely on an unreliable network?
  - Or, at least, know if they are delivered?
  - Want no false positives, and high change of success

Two questions about reliability

- **Who is responsible for this? (architecture)**
    - Network?
    - Host?

- **How is it implemented? (engineering)**

- We will consider both perspectives

Questions?

Finishing our story

- We now have the address of the web site
- And, a route/path to the destination
- And, mechanisms in place to forward the packets at each switch/router
- In a reliable manner
    - So, we can send packets from source to destination
    - Are we done?

- When a packet arrives at a host, what does the host do with it?
    - To which process (application) should the packet be sent?

- If the packet header only has the destination address, how does the host know where to deliver packet?
    - There may be multiple applications on that destination

And while we are finishing our story ….

- Who puts the source address, source port, destination address, destination port in the packet header?

The final piece in the game: End-host stack

Of Sockets and Ports

- When a process wants access to the network, it opens a socket, which is associated with a port

- **Socket:** an OS mechanism that connects processes to the network stack

- **Port:** number that identifies that particular socket

- The port number is used by the OS to direct incoming packets

Implications for Packet Header

- **Packet Header must include:**
  - Destination address (used by network)
  - Destination port (used by network stack)
  - And?
  - Source address (used by network)
  - Source port (used by network stack)

- When a packet arrives at the destination host, packet is delivered to the socket associated with the destination port

- More details later

Separation of concerns

- **Network:** Deliver packets from host to host (based on address)
- **Network stack (OS):** Deliver packets to appropriate socket (based on port)
- **Applications:**
  - Send and receive packets
  - Understand content of packet bodies

Secret of the Internet's success is getting these and other abstractions right

The end-to-end story

- Application opens a **socket** that allows it to connect to the **network stack**
- Maps **name** of the web site to its **address** using **DNS**
- The network stack at the source embeds the address and **port** for both the source and the destination in **packet header**
- Each **router** constructs a **routing table** using a distributed algorithm
- Each router uses destination address in the packet header to look up the **outgoing link** in the routing table
    - And when the link is free, forwards the packet
- When a packet arrives the destination:
    - The network stack at the destination uses the port to forward the packet to the right application

Today's lecture

- The Internet is a huge, complicated system
- One can study the parts in isolation
  - Forwarding
  - Routing
  - Reliability
  - Ports, sockets
  - Network stack
  - …
- But the pieces all fit together in a particular way
- Today was quick overview of how pieces fit…
  - Don't worry if you didn't understand much of it
  - **You probably absorbed more than you realize**