CS4450

Computer Networks:
Architecture and Protocols

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

Spring 2018
Rachit Agarwal

Context for and Goals of Today's Lecture

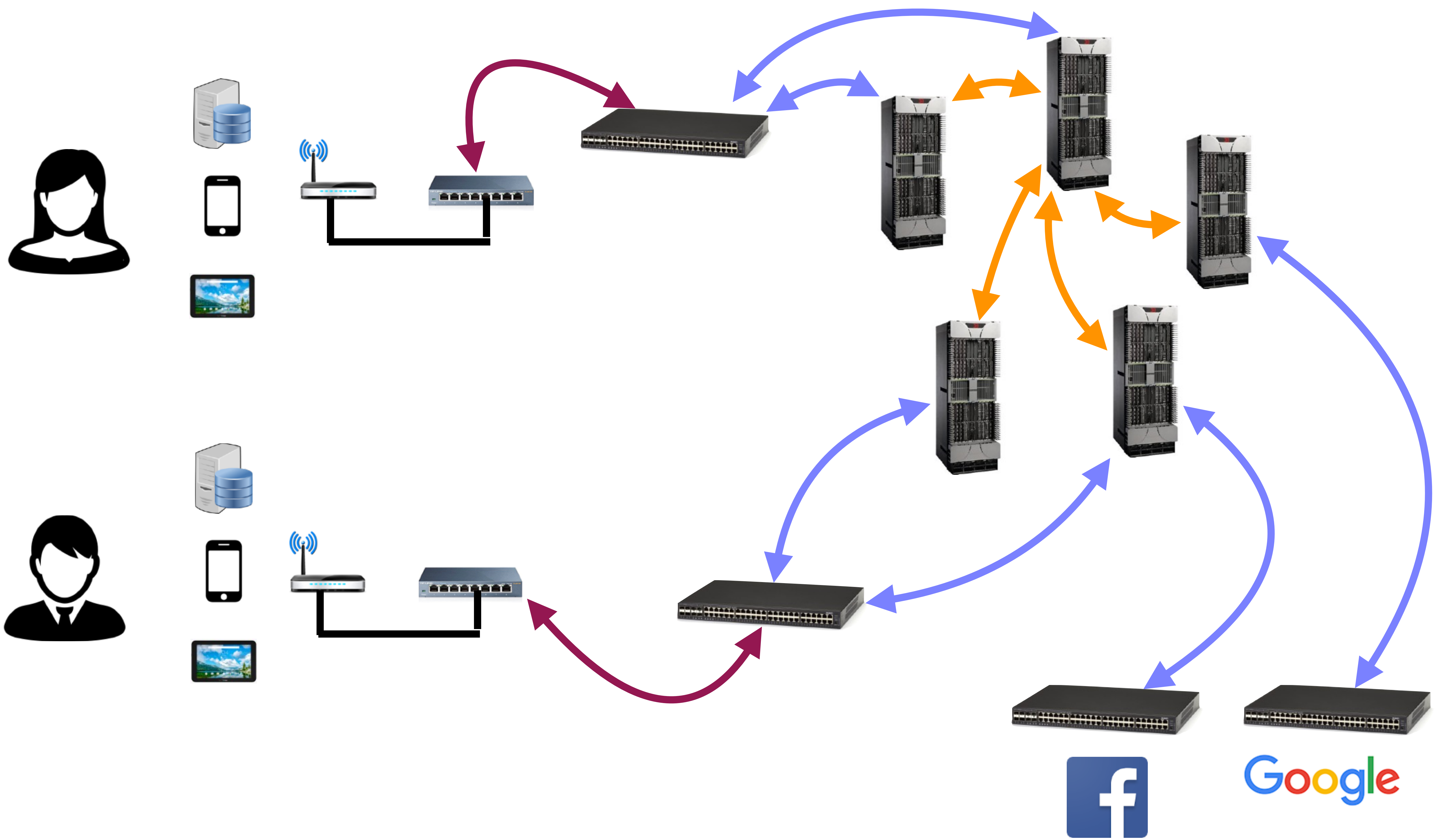
- **Context:**
  - Today's lecture is going to be one of the hardest lectures
  - If you understand everything
    - There is something wrong!
- **Goals:**
  - Wrap up discussion on sharing networks:
    - Packet switching
    - Delay/latency
  - The abstraction of flow:
    - Packets: bags of bits
    - Flows: bags of packets
  - How does the Internet work?
    - An end-to-end view

But, as usual, lets start with:

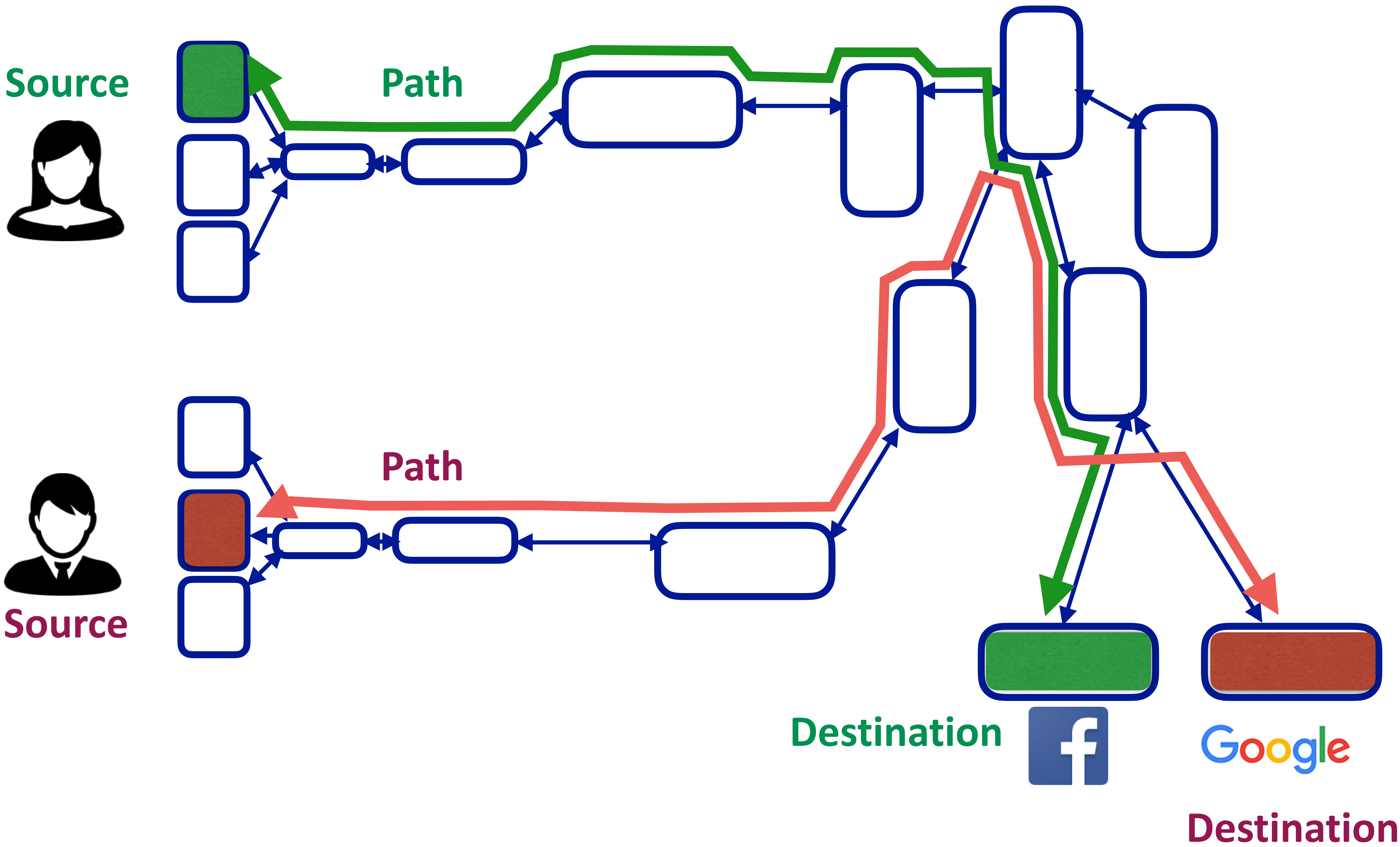
what we learnt last lecture

What is a computer network?

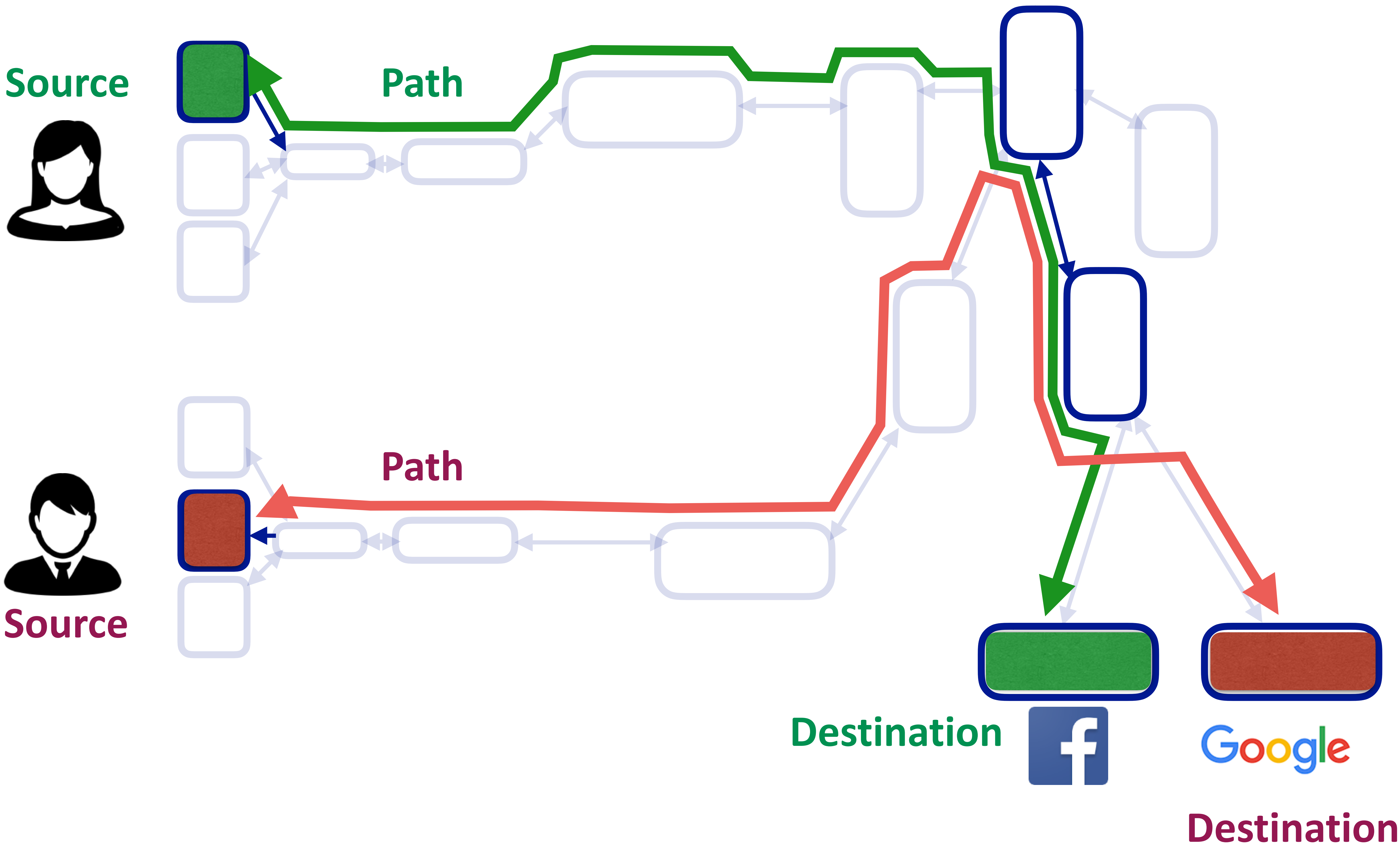
A set of network elements connected together, that implement a set of protocols for the purpose of sharing resources at the end hosts



A computer network can be abstractly represented as a graph

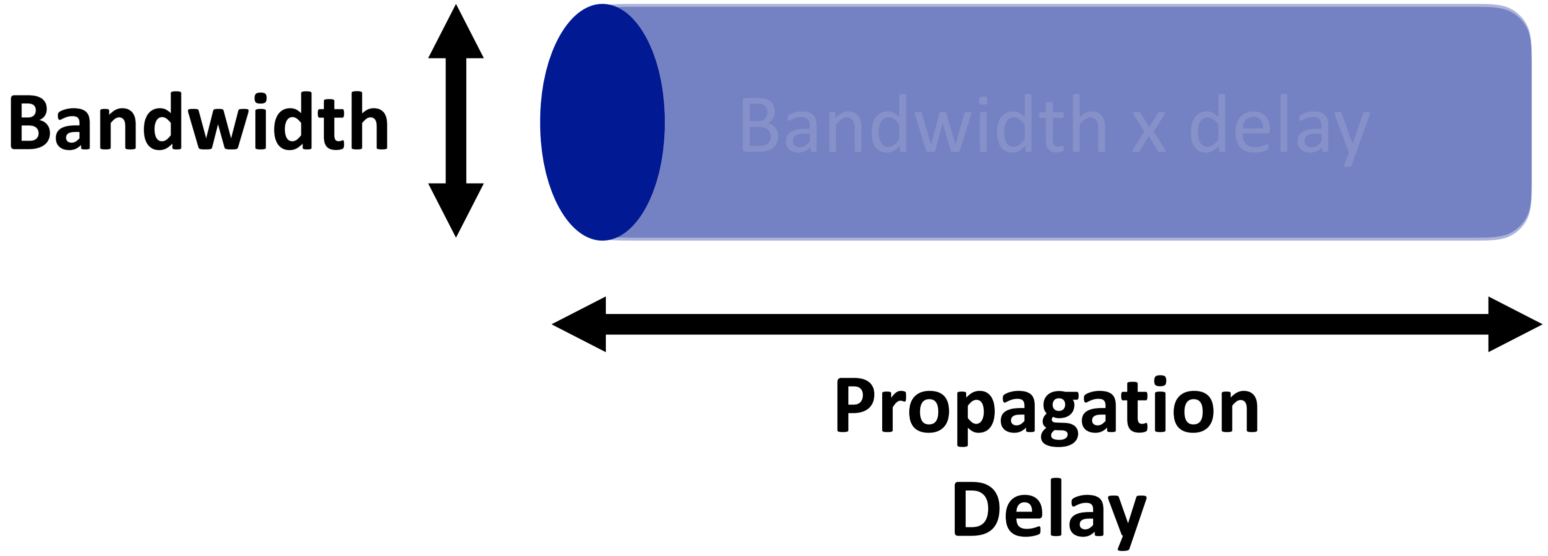


Sharing the network



Performance metrics in computer networks!

- **Bandwidth:** Number of bits sent per unit time (bits per second, or bps)
- **Propagation delay:** Time for <u>one</u> bit to move through the link (seconds)
- **Bandwidth-delay product:** Number of bits "in flight" at any time (bits)



Two approaches to sharing networks

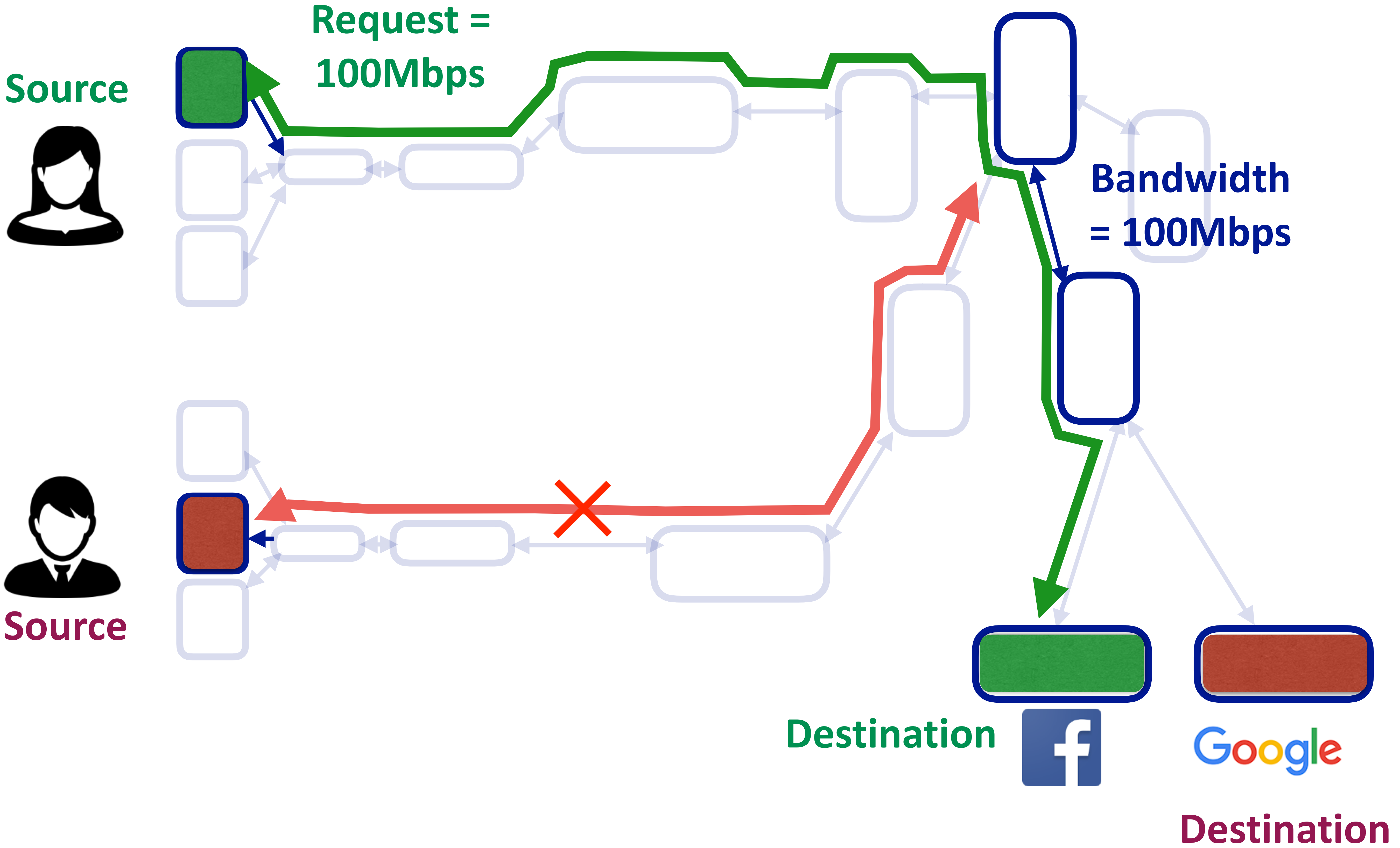
- Reservations
- On demand

Two approaches to sharing networks

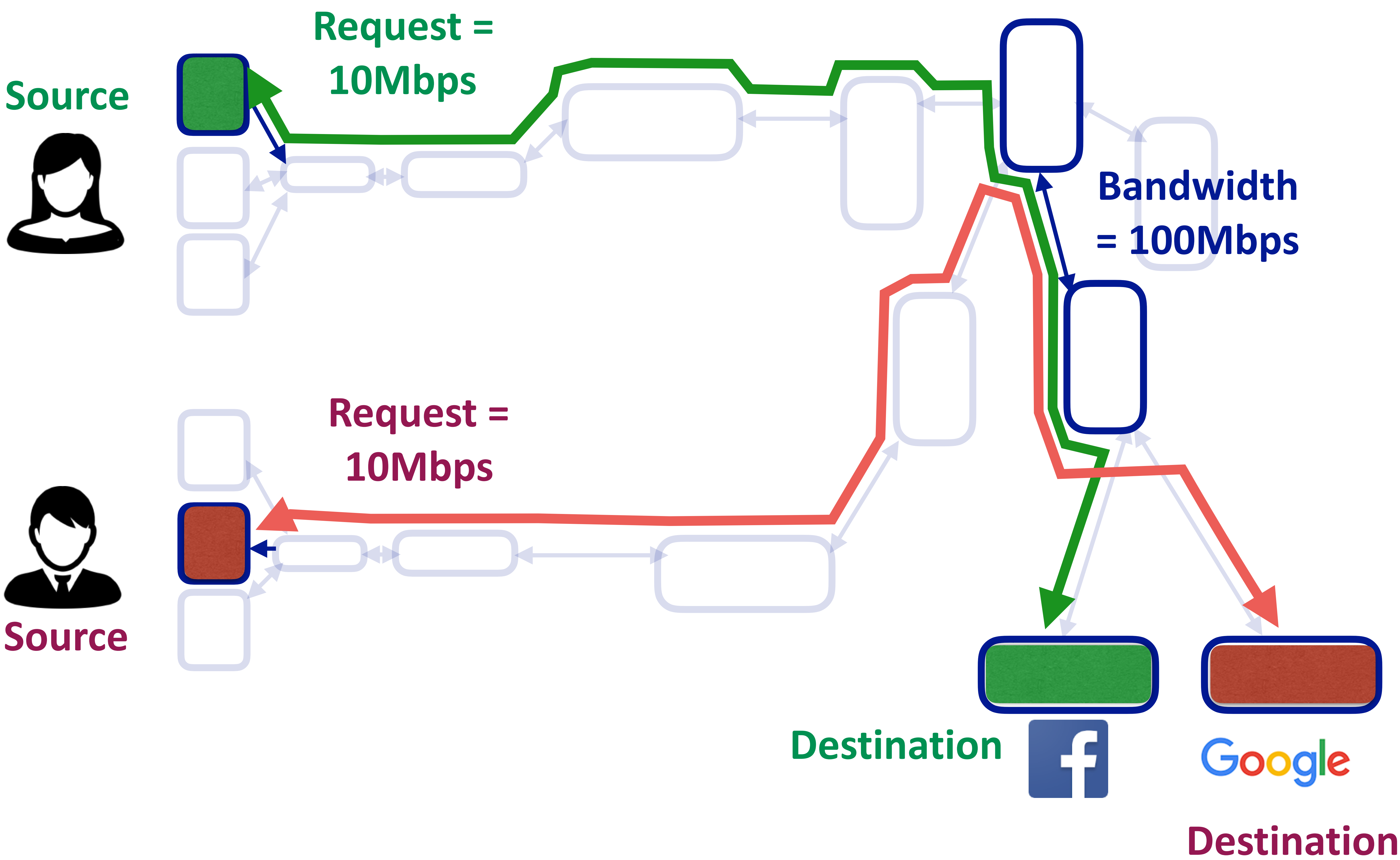
- **First: Reservations**
  - Reserve bandwidth needed in advance
  - Set up <u>circuits</u> and send data over that circuit
  - **Must reserve for peak bandwidth**
    - Applications may generate data at rate varying over time
    - 100MB in first second
    - 10MB in second second …

- One way to implement reservations: circuit switching
  - Source sends a reservation request for peak demand to destination
  - Switches/routers establish a “circuit”
  - Source sends data
  - Source sends a “teardown circuit” message

Circuit switching: an example (red request fails)



Circuit switching: another example (red request succeeds)



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)**

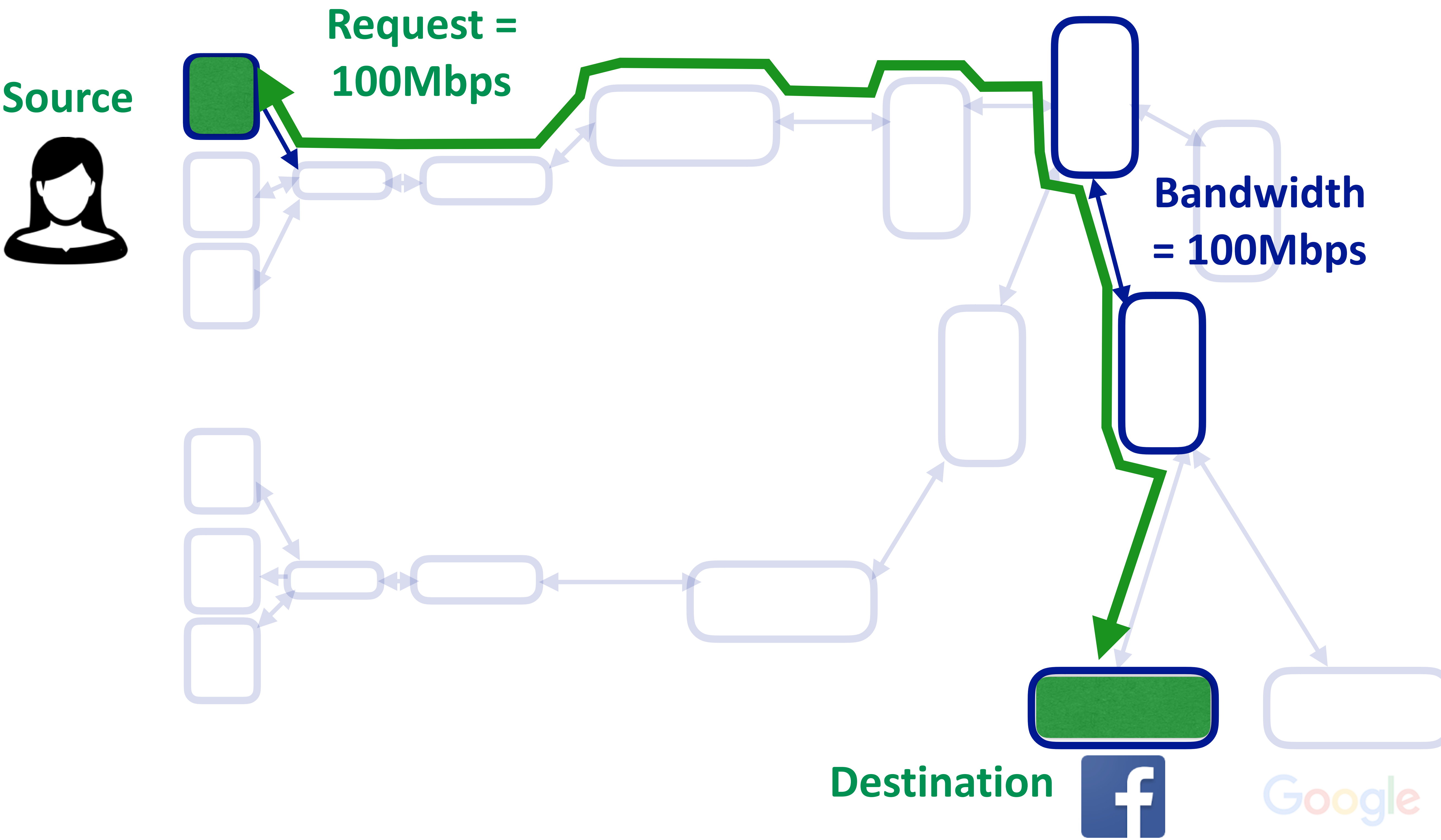
Today's lecture

1. Packet switching for sharing networks
2. **Why "packets" and "flows"?**
3. Understanding bandwidth and latency for packets
4. How does Internet work?

Lets dive deeper into not-so-goods for circuit switching

- Not-so-goods
    - Handling failures
    - Resource underutilization
    - Blocked connections
    - Connection set up overheads
    - Per-connection state in switches (scalability problem)

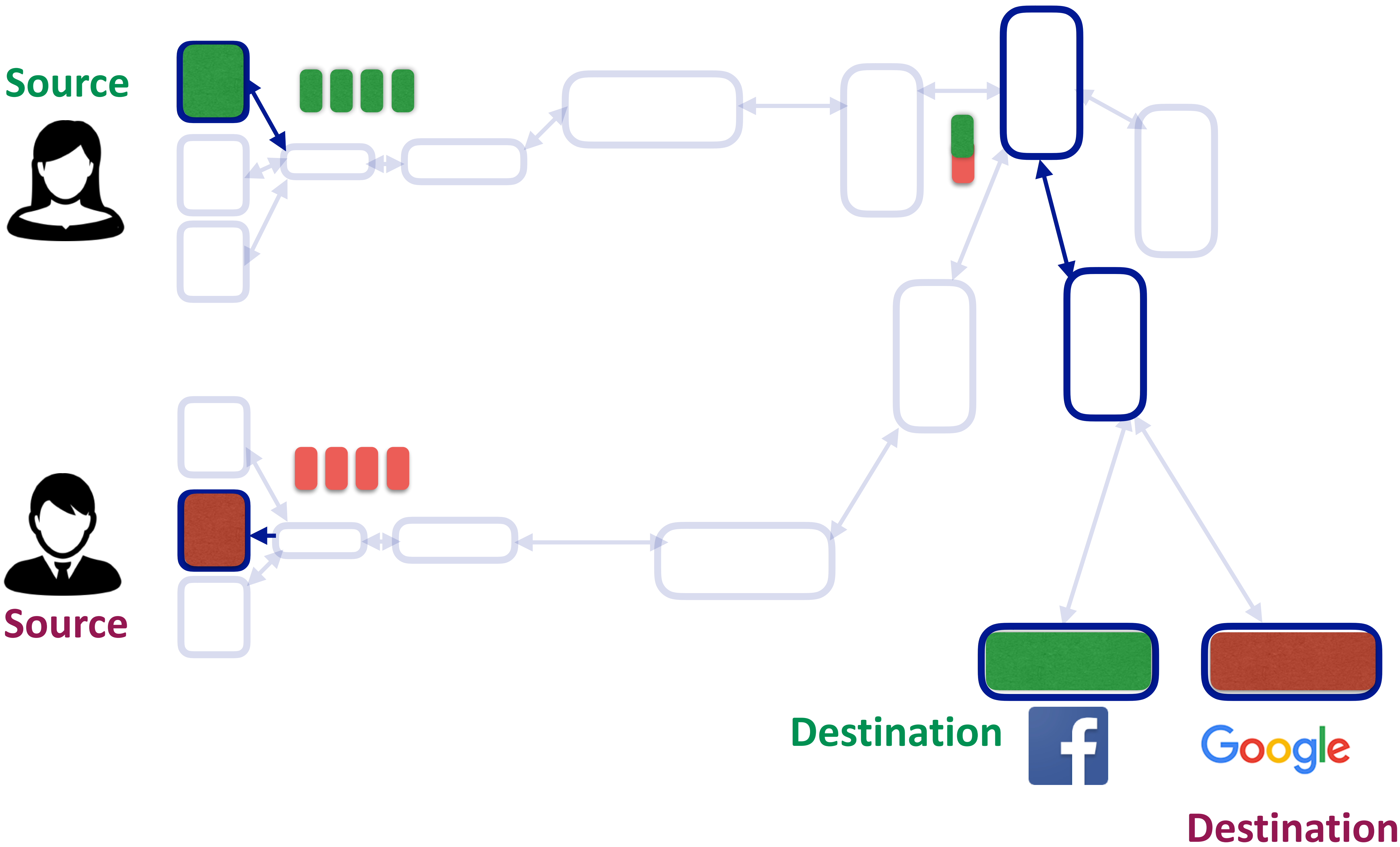
Circuit switching: challenges



Getting rid of the challenges

- **Break data into smaller pieces**
  - **Packets!**

Packet switching: an example



Group Exercise 2:

How do *packets* solve problems with reservations?

- **Handling failures**
- **Resource underutilization**
- **Blocked connections**
- **Connection set up overheads**
- **Per-connection state in switches (scalability problem)**

Packet switching summary

- **Goods:**
  - With proper mechanisms in place
    - Easier to handle failures
  - 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:**
  - Unpredictable performance
  - High latency
  - Packet header overhead

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:
  - **Peak of aggregate load is << aggregate of peak load**

- 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 six components**
  - Link properties:
    - Transmission delay
    - Propagation delay
  - OS internals:
    - Processing delay
    - Queueing delay
  - Traffic matrix and switch internals:
    - Processing delay
    - Queueing delay
- First, consider transmission, propagation delays
- Queueing delay and processing delays later in the course

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

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

Questions?

Group Exercise 3:

How long does it take for a *packet* on a link?

Constraints:

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

Solution to Group Exercise 3:

How long does it take for a *packet* on a link?

Constraints:

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

Solution to Group Exercise 2:

How long does it take for a *packet* on a link?

176.3us

Why?

How does the Internet work?

Many mechanisms!

- **Locating the destination**: Naming, addressing
- **Finding a path to the destination**: Routing
- **Sending data to the destination:** Forwarding
- **Failures, reliability, etc.:** Distributed routing and congestion control

Will take the entire course to learn these:

Lets get an end-to-end picture!

What do computer networks look like?

Three Basic components

- **End hosts**: they send/receive packets
    - Require a “network stack” — networking software/hardware
    - stack replicates some router/switch functionality …
    - … before handing data to application
    - More discussion in next lecture

- **Switches/Routers:** they forward packets

- **Links:** connect end hosts to switches, and switches to each other

What must packets carry?

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

Name versus Addresses

- Network **Address: where host is located**
  - Requires an address for the destination host
    - can be multiple headers
- Network **Name: which host it is**
  - why?
- **When you move server to new building**
  - Name doesn't change
  - Address does change
- **Same thing with your own name and address!**
- **Lets get back to packet delivery….**

Questions?

Fundamental Challenge #1

Routing packets through the network elements to destination

- Given destination address, how does each switch/router forward packets so that packet reaches destination

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

  - **outgoing link is often referred to as a port**
  - **The word port has two meanings in this lecture (later)**

Routing protocols (conceptually)

- Distributed algorithm run between routers
- Gather information about the network topology
- Compute paths through that topology
- Store forwarding information in each router
    - If packet is destined for X, send out port p1
    - If packet is destined for Y, send out port p2
    - Can packets going to different destinations sent out to same port?
- **We call this a routing table**

Control plane vs data plane

- **Control plane: mechanisms used to compute routing tables (and other forwarding information)**
    - Inherently global: must know topology to compute
    - Routing algorithm is part of the control plane
    - Time scale: per network event
- **Data plane: using those tables to actually forward packets**
    - Inherently local: depends only on arriving packet and local routing table
    - Forwarding mechanism is part of the data plane
    - Time scale: per packet arrival

Questions?

Fundamental challenge #2

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

What challenges have we missed?

- **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

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)**

Finishing our story

- We now have the address of the web site
  - So, we can send packets to host
  - 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

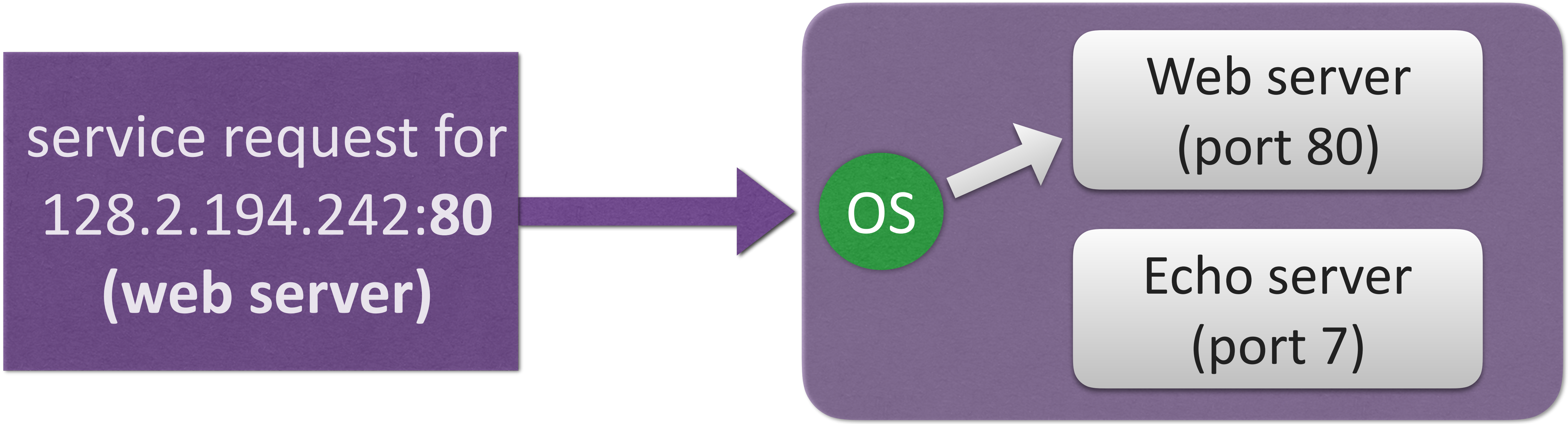
Two Meanings of "Port"

- Switches/routers have "physical ports":
    - Places where links connect to switches

- Network stacks have "logical ports"
    - Logical places where applications connect to stack

Of Sockets and Ports

- When a process wants access to the network, it open a socket, which is associated with a port
    - This is not a physical port, just a logical one
- **Socket:** an OS mechanism that connects processes to the networking stack
- **Port:** number that identifies that particular socket
- The port number is used by the OS to direct incoming packets


service request for
128.2.194.242:**80**
(web server)
OS
Web server
(port 80)
Echo server
(port 7)

Implications for Packet Header

- **Packet Header must include:**
    - Destination address (used by network)
    - Destination 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

Who cares?

- **Why is separation of concerns important?**
  - Separation of concerns ~ Modularity

- Because if each component has a well-defined task, you can focus design on that task
  - And replace it with any other implementation that does that task, without changing anything else
- When you don’t have separation of concerns, then you have one big pile of code that does everything …
  - Very hard to modify, or understand

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

What else are we missing?

- How do hosts get their addresses?
- What else happens to packets on path?
- What about security?
- What about specialized networks?

These are our topics

- Each is motivated by a clear problem
- 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 solution

Today's lecture

- The Internet is a huge, complicated system
- One can study the parts in isolation
  - Routing
  - 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**