Course Introduction

- Introduction
  - Matthew Green, Professor
  - Teaching Assistants: Venkatesh Gopal (head TA), Eyal Foni, Shikha Fadnavis and Praveen Malhan (maybe more soon!)
- Registration: 60-80 students
  - If you're waitlisted, come and see me next Mon
- Prerequisites
  - Intermediate programming
- My teaching style
  - PPT lecture slides
    - Made available after lecture
  - Off script lecturing on whiteboard
    - Just as important towards exams, etc.
  - Do not like late arrivals to class
- A word about academic integrity

Course Introduction

- WireShark labs (15% of course grade)
  - May do with one partner (cannot be same partner for programming assignments)
  - Due at 10pm the night before the first lecture of the week
  - Upload PDF solution via blackboard
- Homework assignments (15% of course grade)
  - Assigned problems from the textbook
  - Also due 10pm the night before the first lecture of the week
  - Upload PDF solution via blackboard
- Programming Projects (20% of grade)
  - May work in groups of 2 students
  - Use Python programming language
- Late assignments, 10% per day, up to 3 days
- Review syllabus

Course intro, cont.

- Course website/syllabus etc.
  - https://isi.jhu.edu/~mgreen/600.444/
  - Piazza Signup: piazza.com/jhu/spring2017/en600344
- Office Hours
  - Mine: Mon 2-4pm (excepting this afternoon)
  - Tues or Weds by appointment
  - TAs will post something

How many of you (show of hands):

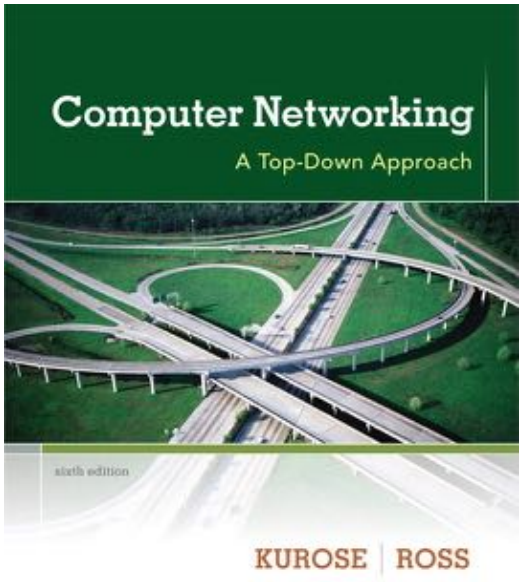
- Understand the difference between TCP and UDP?
- Are familiar with the OSI reference model?
- Understand packet encapsulation?
- Have looked at raw TCPDump output?
- Can analyze raw TCPDump output?
- Have used WireShark before?
- Know what a DNS zone transfer is?
- Could draw an accurate picture of IP header with all fields from memory?
- Have done socket programming before?
- Know the difference between link state and distance vector routing?
- Are family with Scapy?

How many of you (show of hands):

- Are freshman?
- Sophomores?
- Juniors?
- Seniors?
- Graduate students in CS?
- MSSI Graduate students?
- Non-Computer Science majors?

- Someone, please remind me when there are 10 minutes left in class, to go over the Wireshark Lab and the Homework assignment!

Chapter 1 Introduction



Computer Networking: A Top Down Approach
6th edition (or 7th)
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

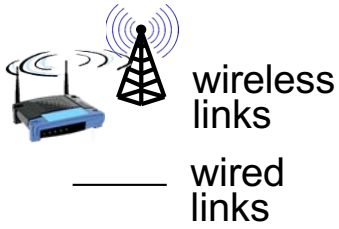
1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

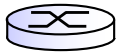
1.6 networks under attack: security

What's the Internet: "nuts and bolts" view

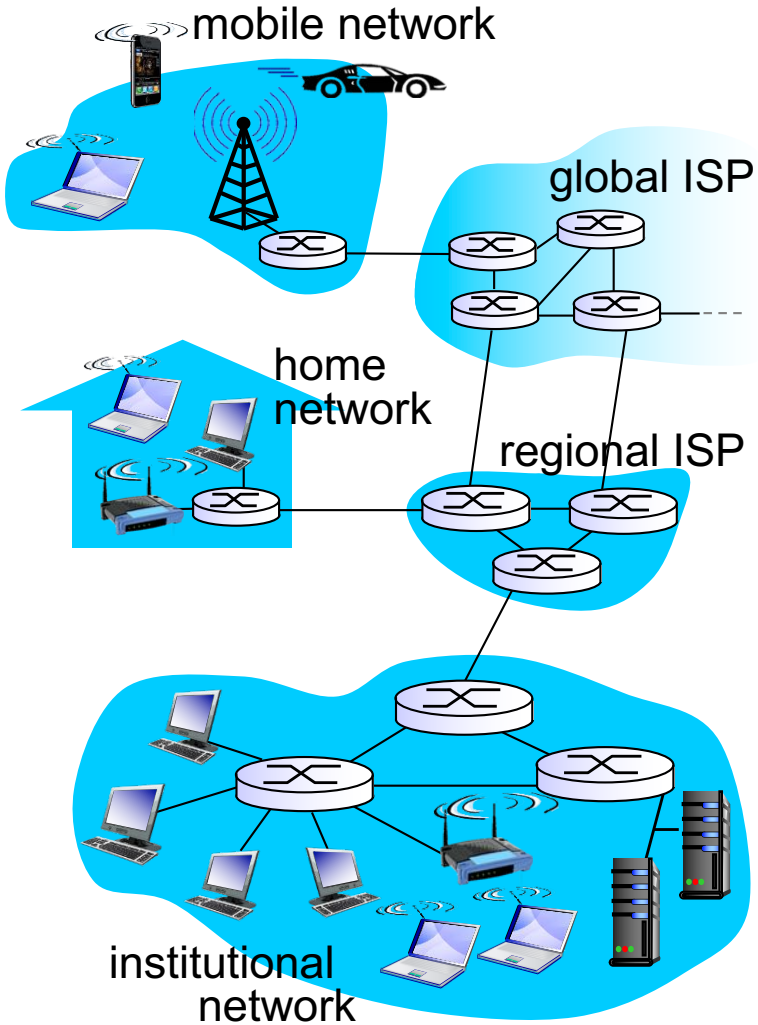
- millions of connected computing devices:
  - *hosts* = *end systems*
  - running *network apps*



- *communication links*
  - fiber, copper, radio, satellite
  - transmission rate: *bandwidth*



- *Packet switches:* forward packets (chunks of data)
  - *routers* and *switches*



Internet appliances

IP picture frame
http://www.ceiva.com/

Web-enabled toaster +
weather forecaster

Tweet-a-watt:
monitor energy use

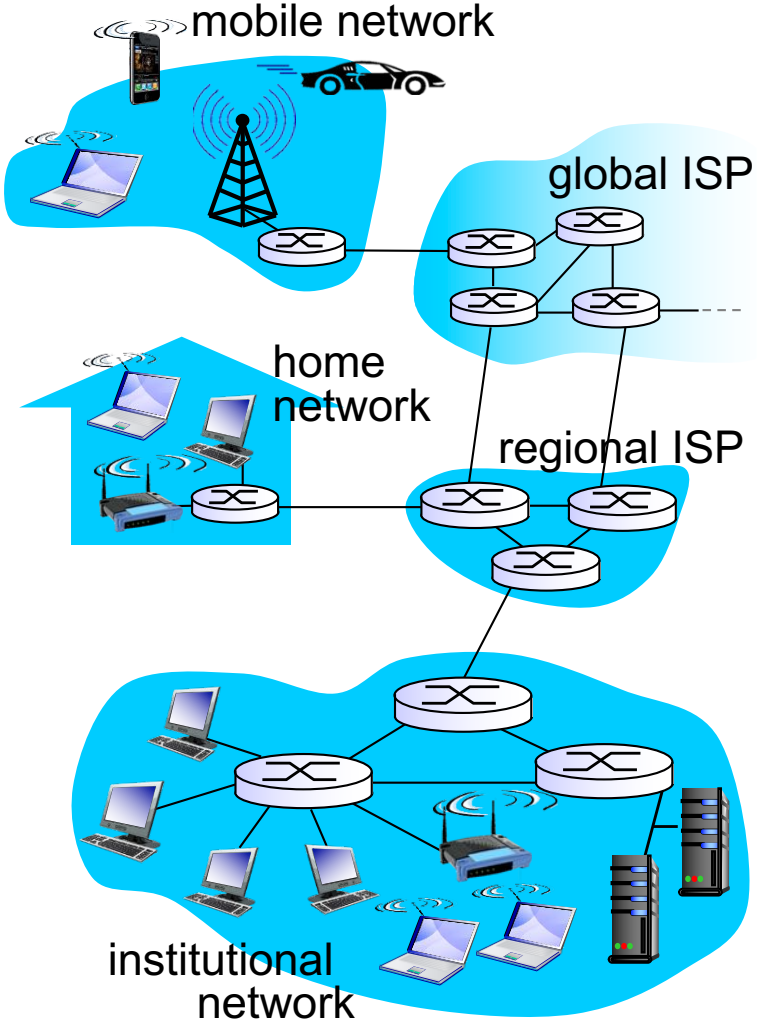
Internet
refrigerator

IP-enabled camera
DDoS your friends for fun

Internet phones

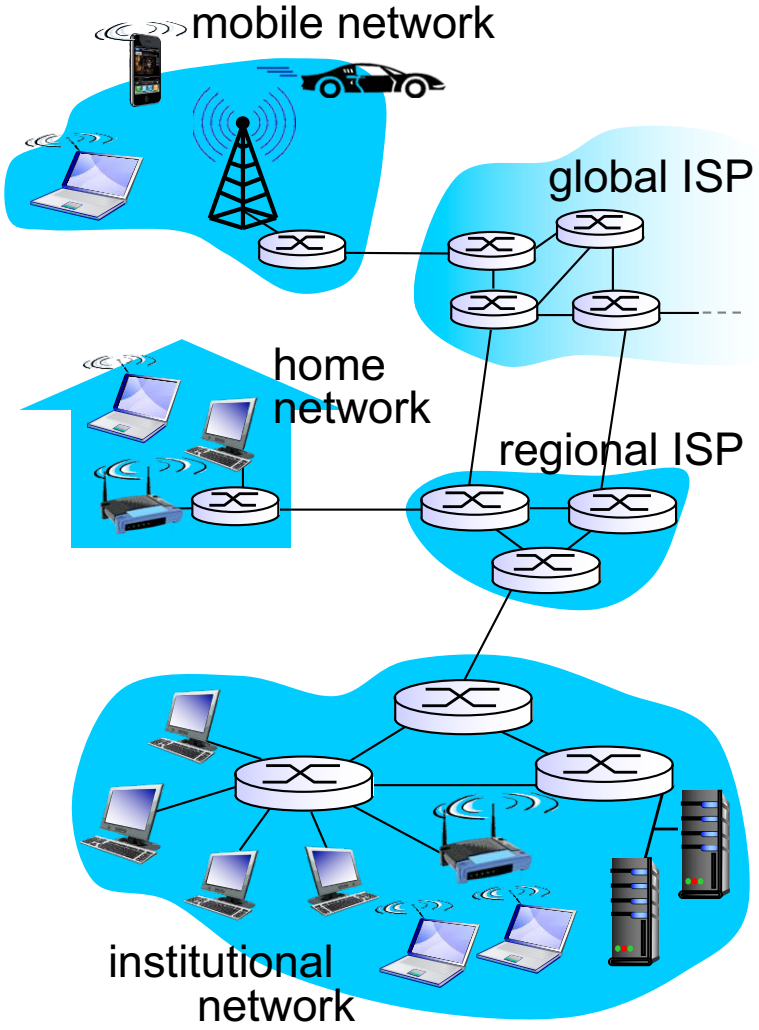
What's the Internet: "nuts and bolts" view

- *Internet:* "network of networks"
  - Interconnected ISPs
- *protocols* control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, 802.11
- *Internet standards*
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force



What's the Internet: a service view

- *Infrastructure that provides services to applications:*
  - Web, VoIP, email, games, e-commerce, social nets, …
- *provides programming interface to apps*
  - hooks that allow sending and receiving app programs to "connect" to Internet
  - provides service options, analogous to postal service



What's a protocol?

What's a protocol?

human protocols:

- "what' s the time?"
- "I have a question"
- introductions

… specific msgs sent

… specific actions taken when msgs received, or other events

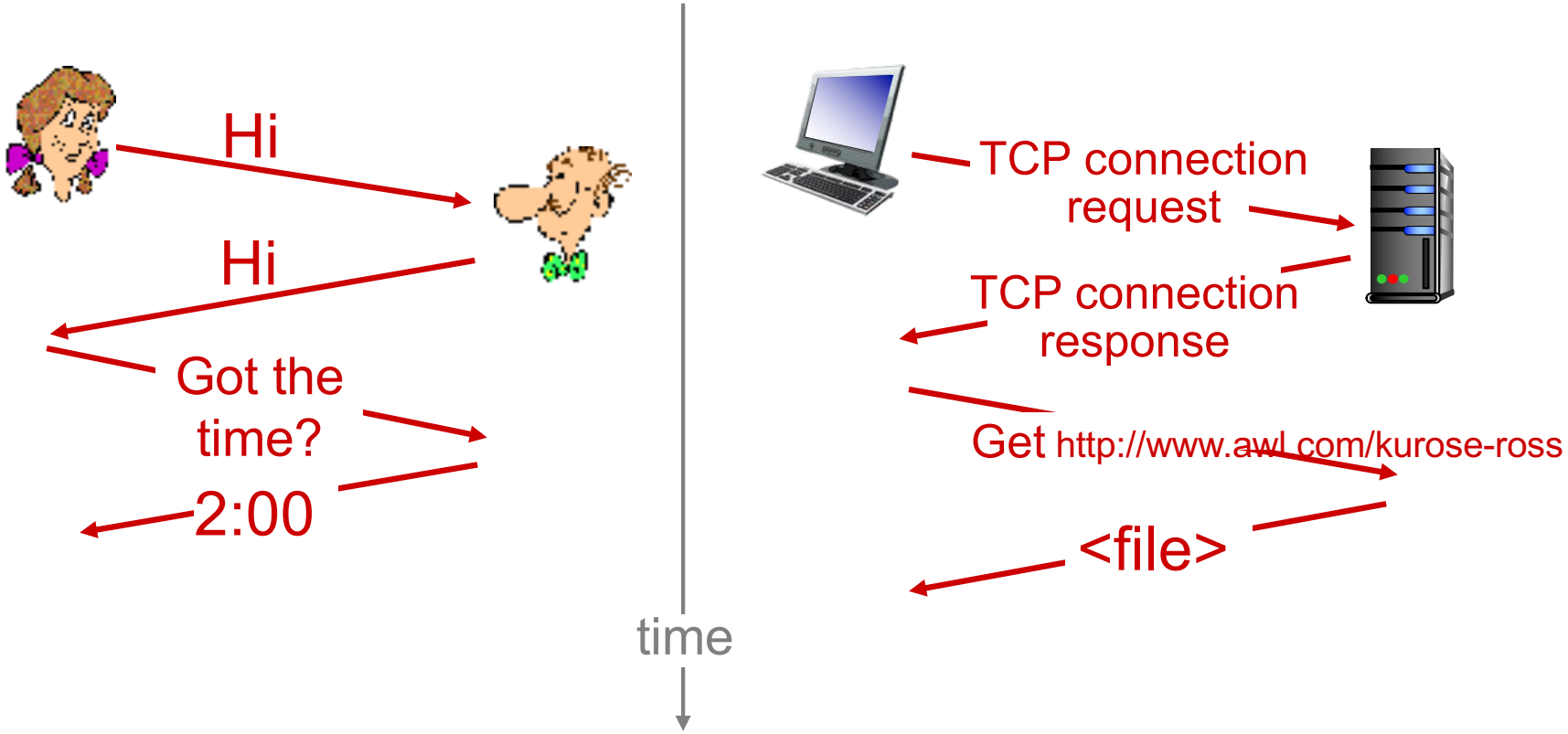
network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt

What's a protocol?

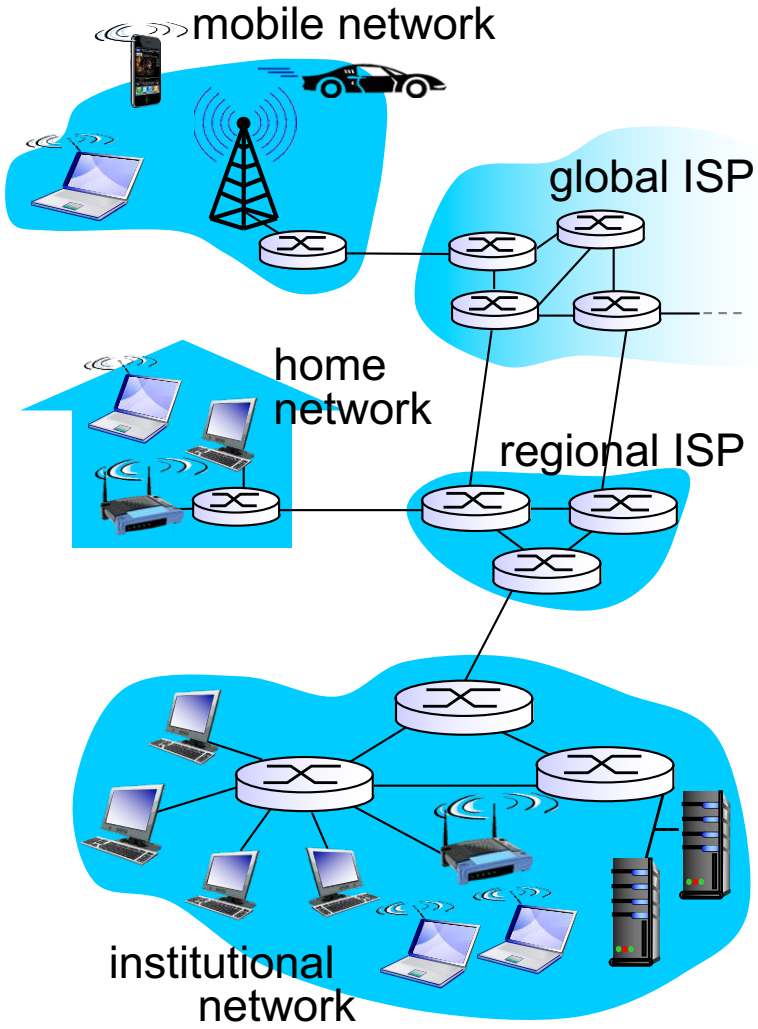
a human protocol and a computer network protocol:



Q: other human protocols?

A closer look at network structure:

- *network edge:*
  - hosts: clients and servers
  - servers often in data centers
- *access networks, physical media:* wired, wireless communication links
- *network core:*
  - interconnected routers
  - network of networks



A little bit of history

POTS

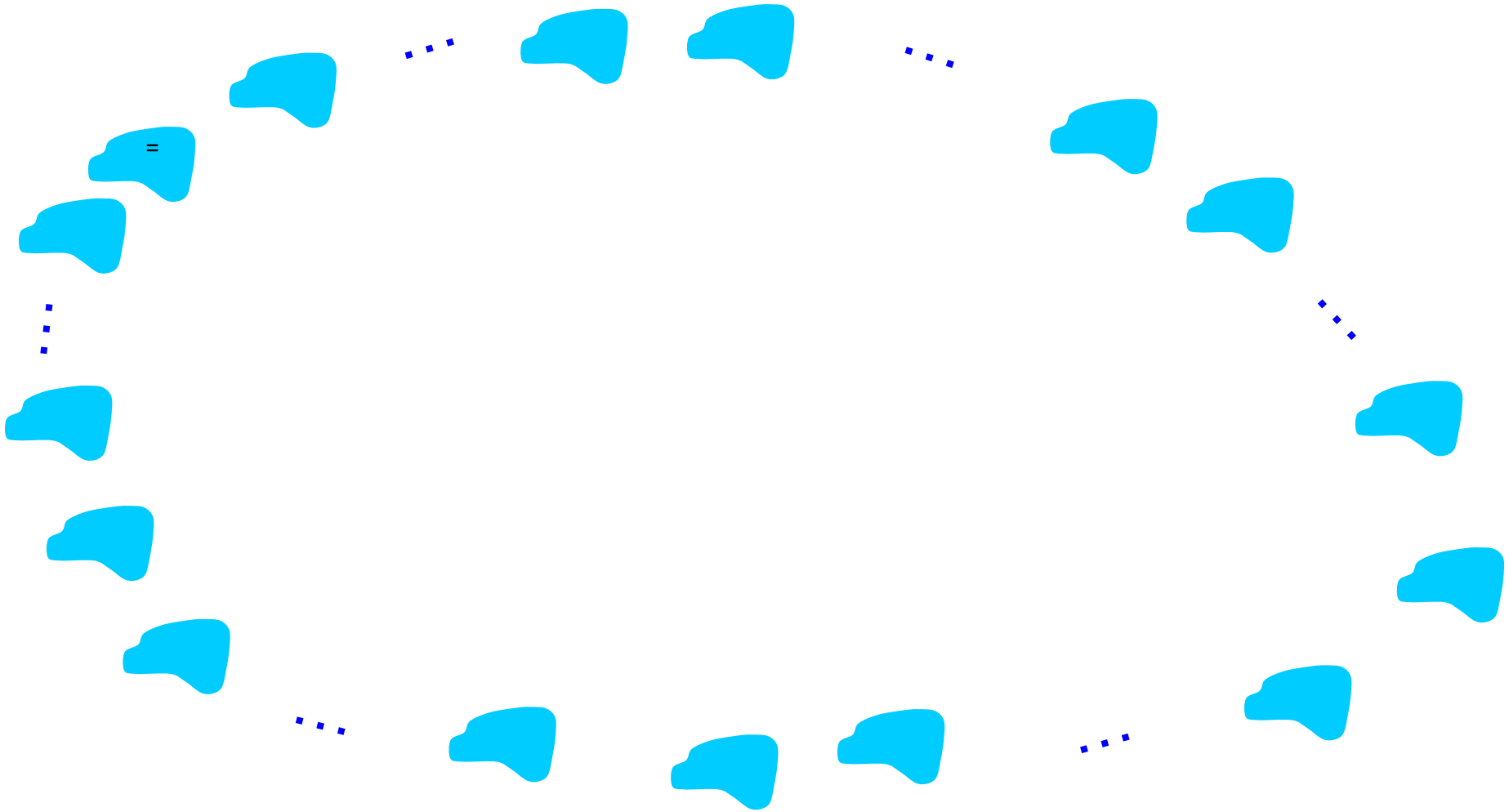
A little bit of history

POTS

(plain old telephone service)

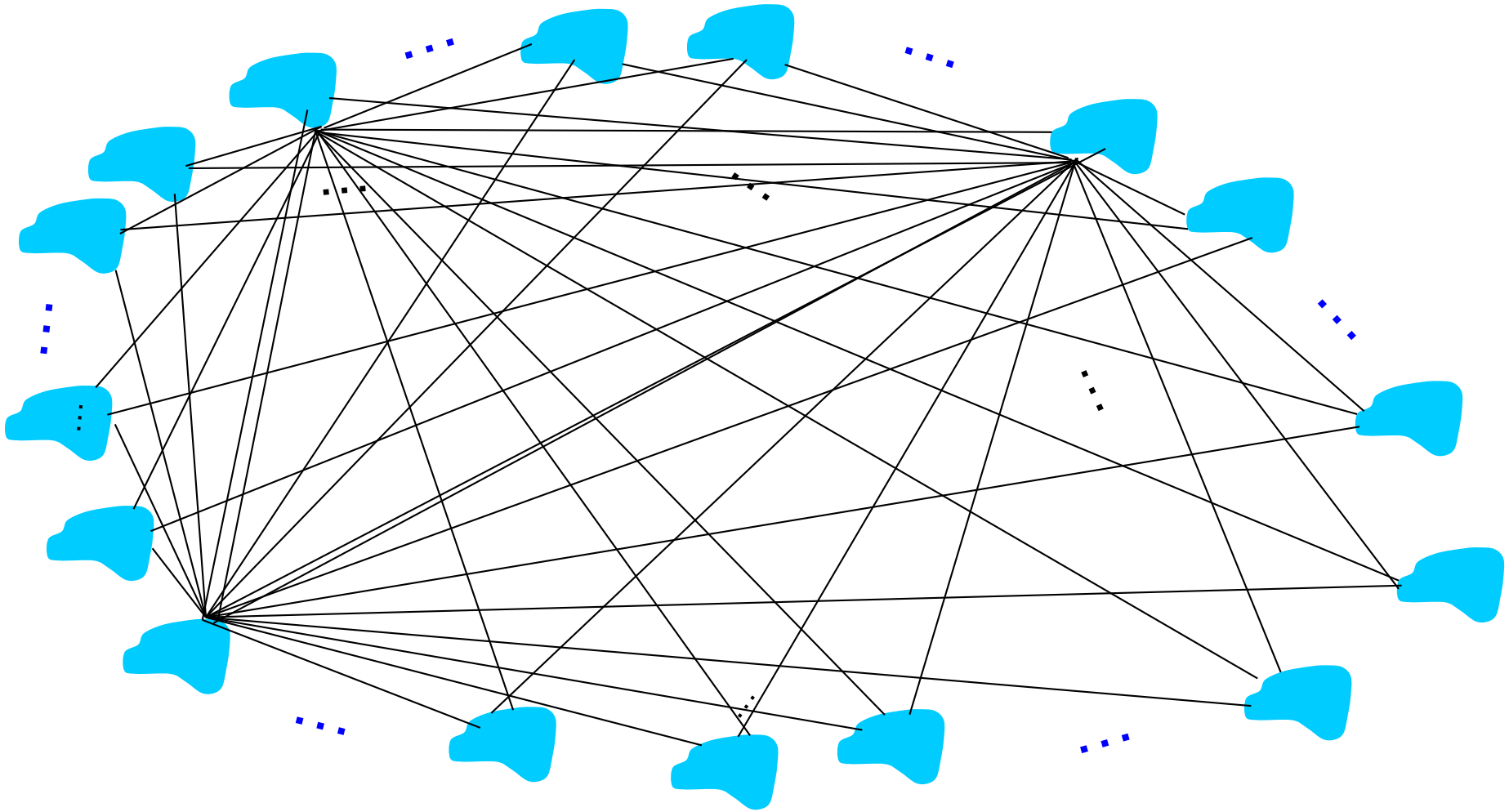
POTS

Question: given *a town of many people, how do we wire them together?*



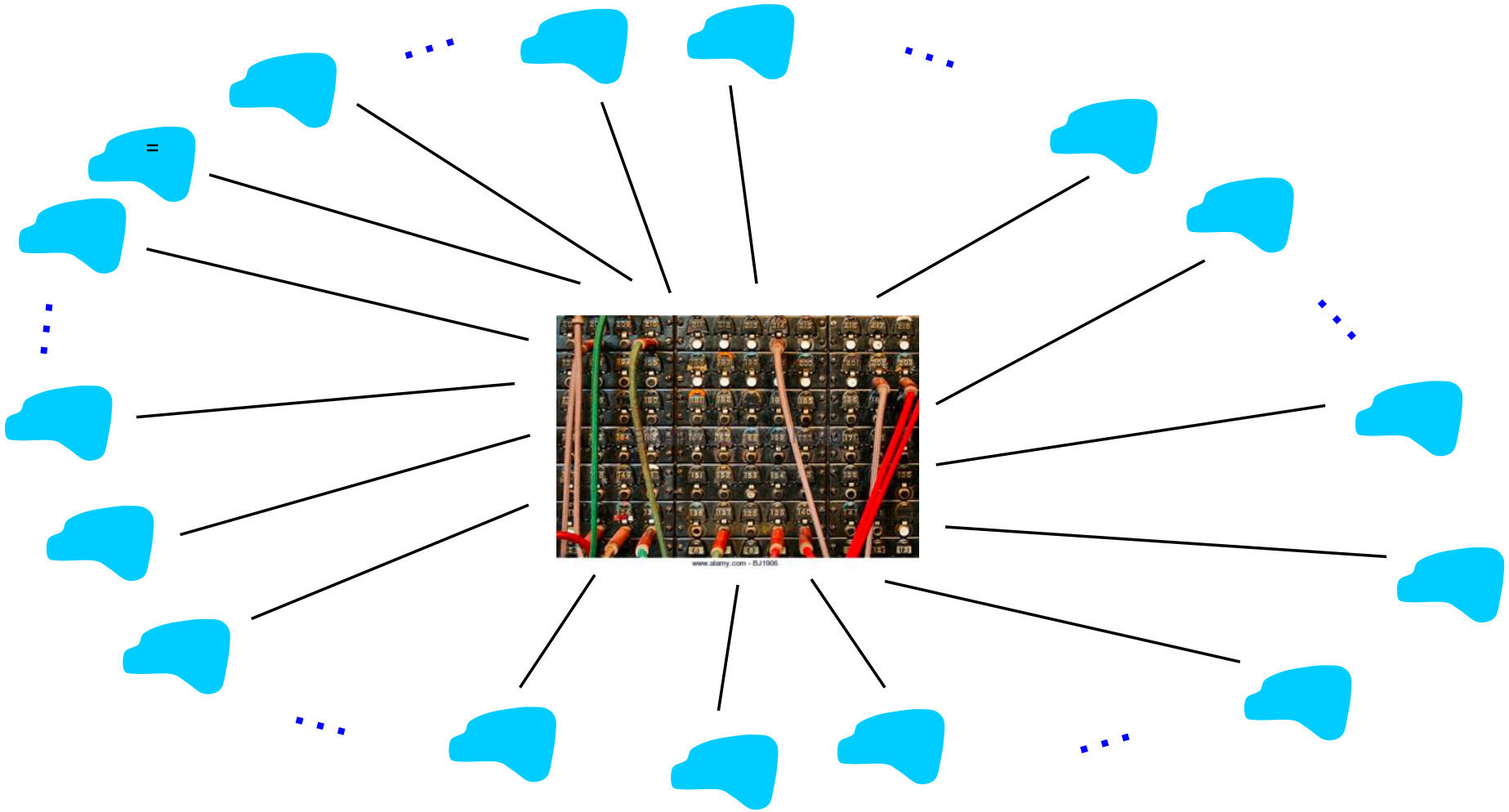
POTS ("fully connected" network)

Option: connect each subscriber to every other subscriber?



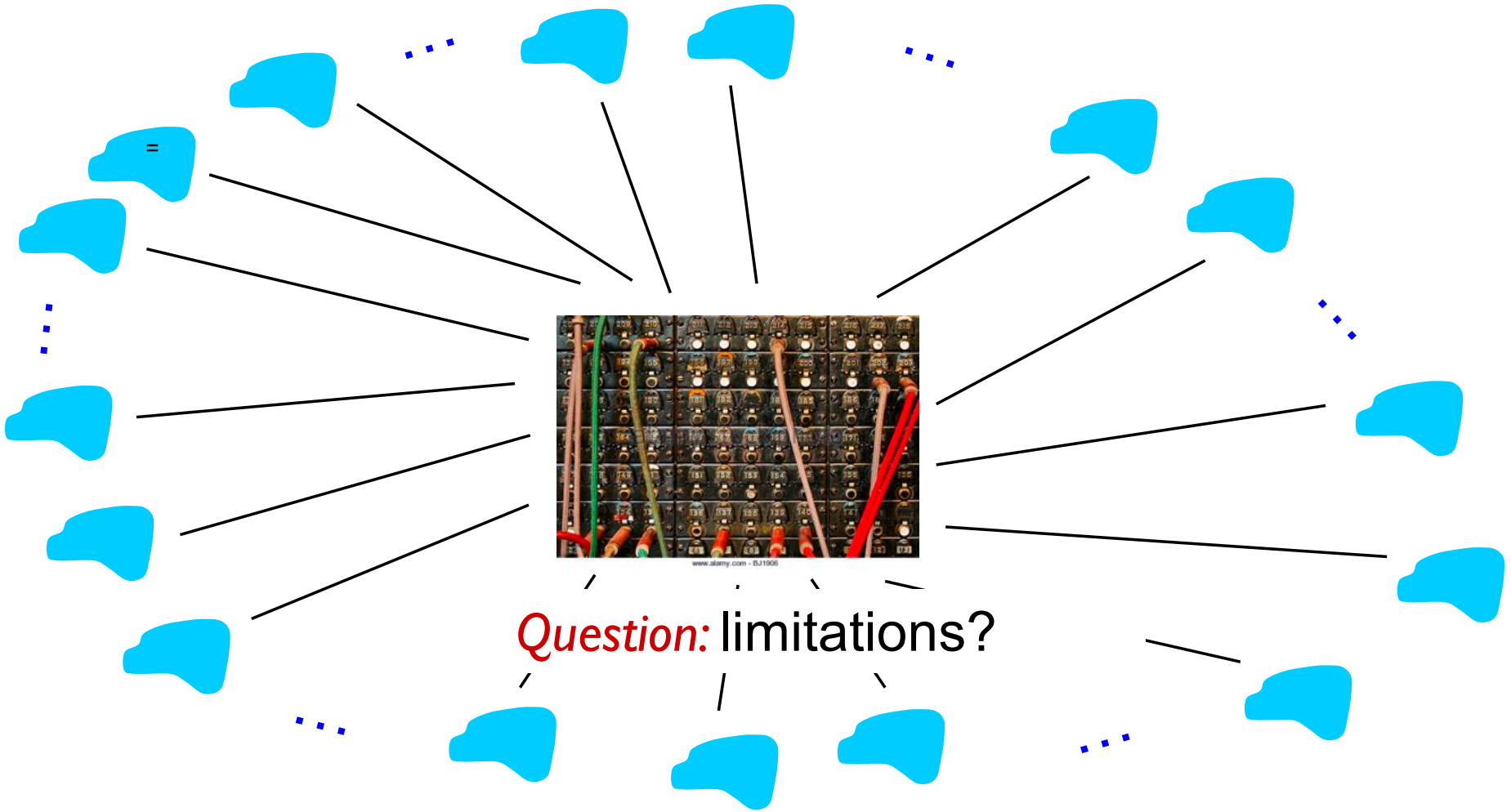
Circuit switching

Option: connect each subscriber to a central switchboard



Circuit switching

Option: *connect each subscriber to a central switchboard*

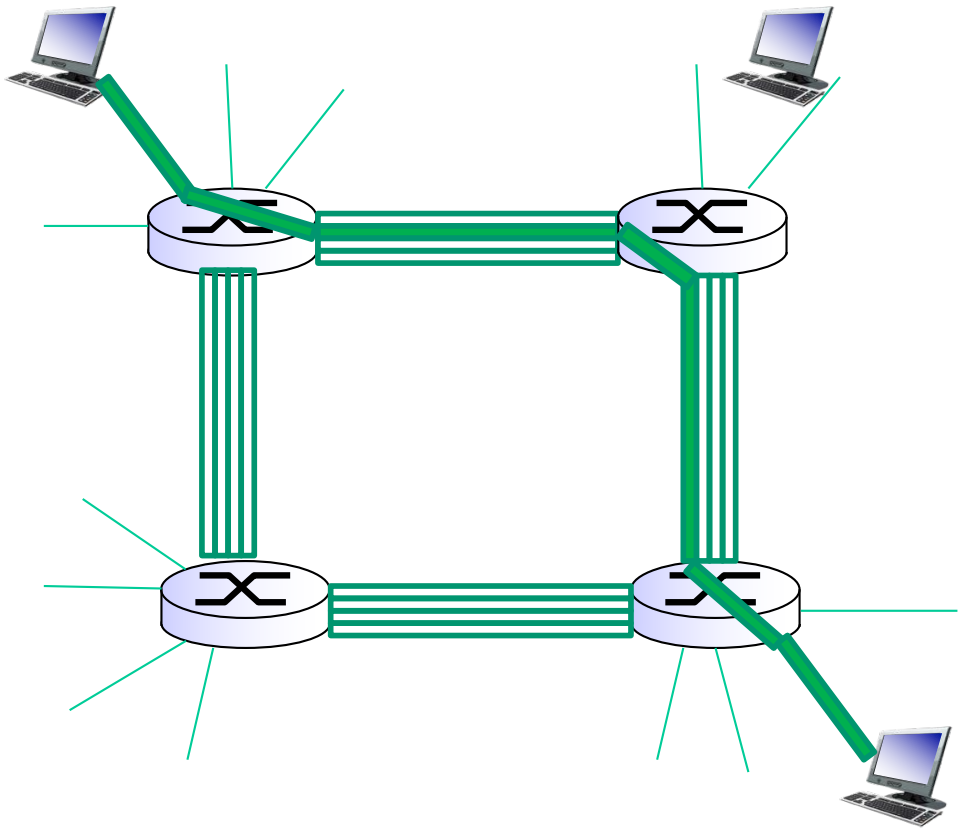


Question: limitations?

Circuit switching

end-end resources allocated to, reserved for "call" between source & dest:

- In diagram, each link has four circuits.
  - call gets 2nd circuit in top link and 1st circuit in right link.
- dedicated resources: no sharing
  - circuit-like (guaranteed) performance
- circuit segment idle if not used by call *(no sharing)*
- Commonly used in traditional telephone networks

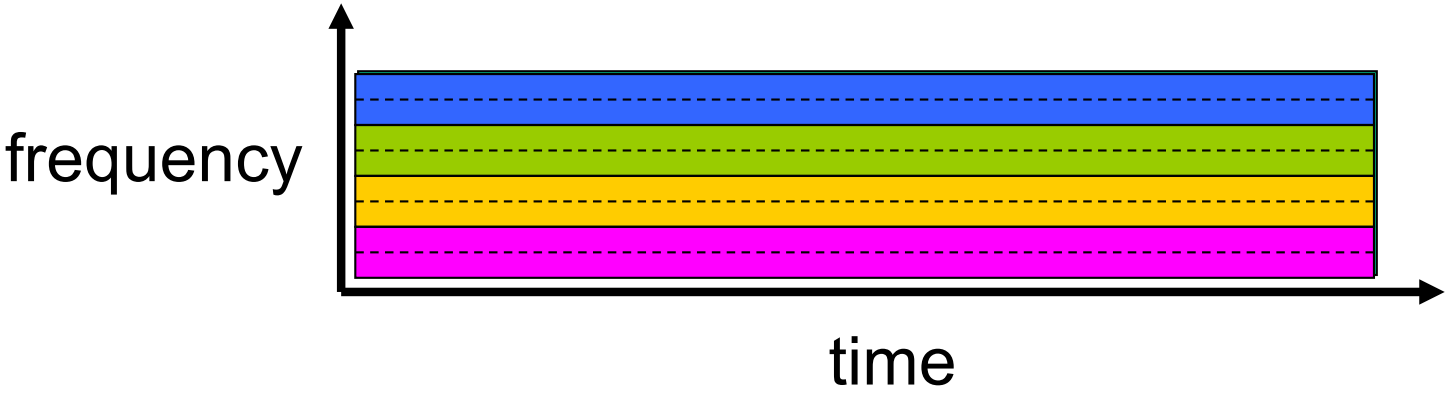


Circuit switching: FDM versus TDM

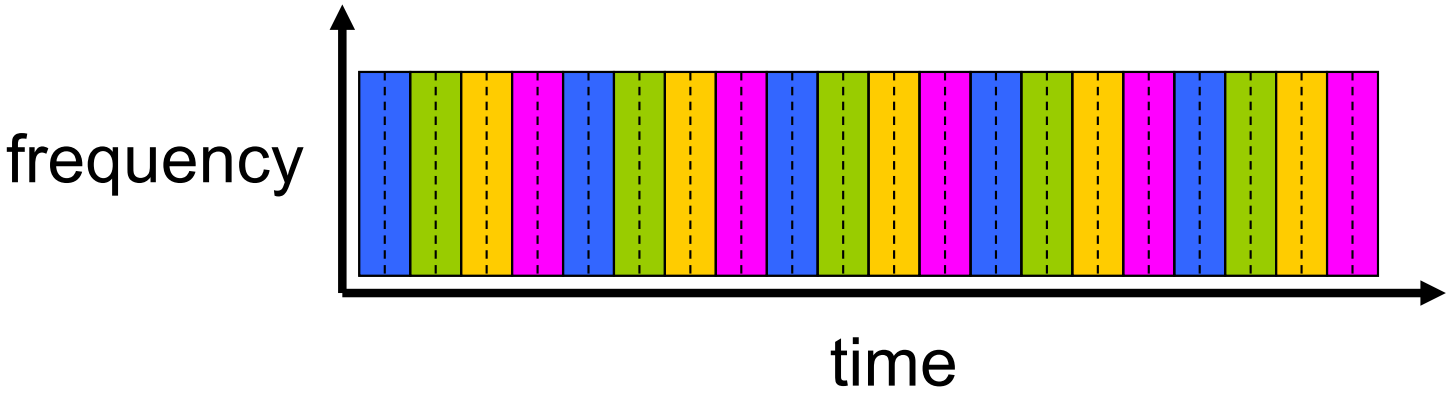
Example:

4 users

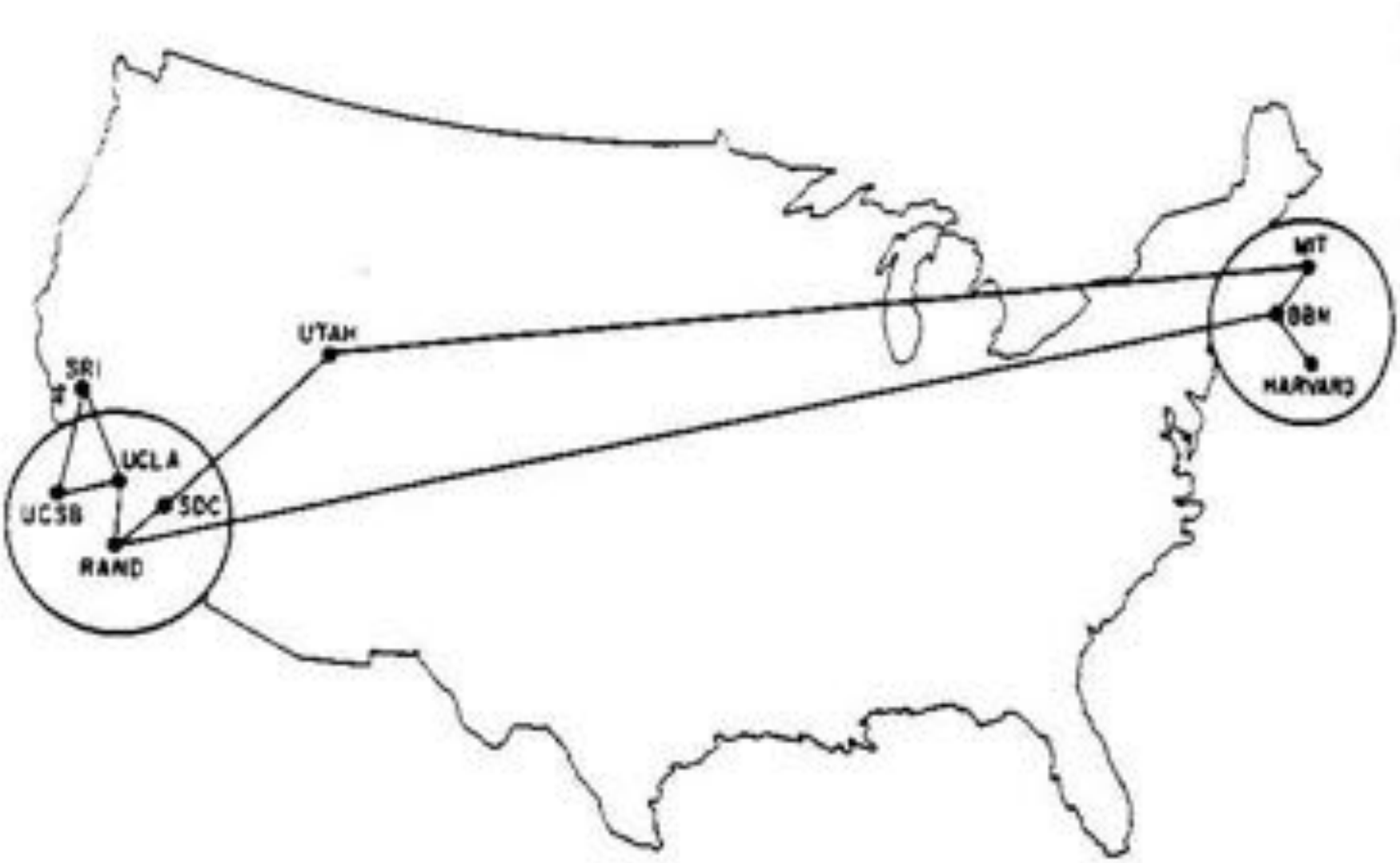
FDM



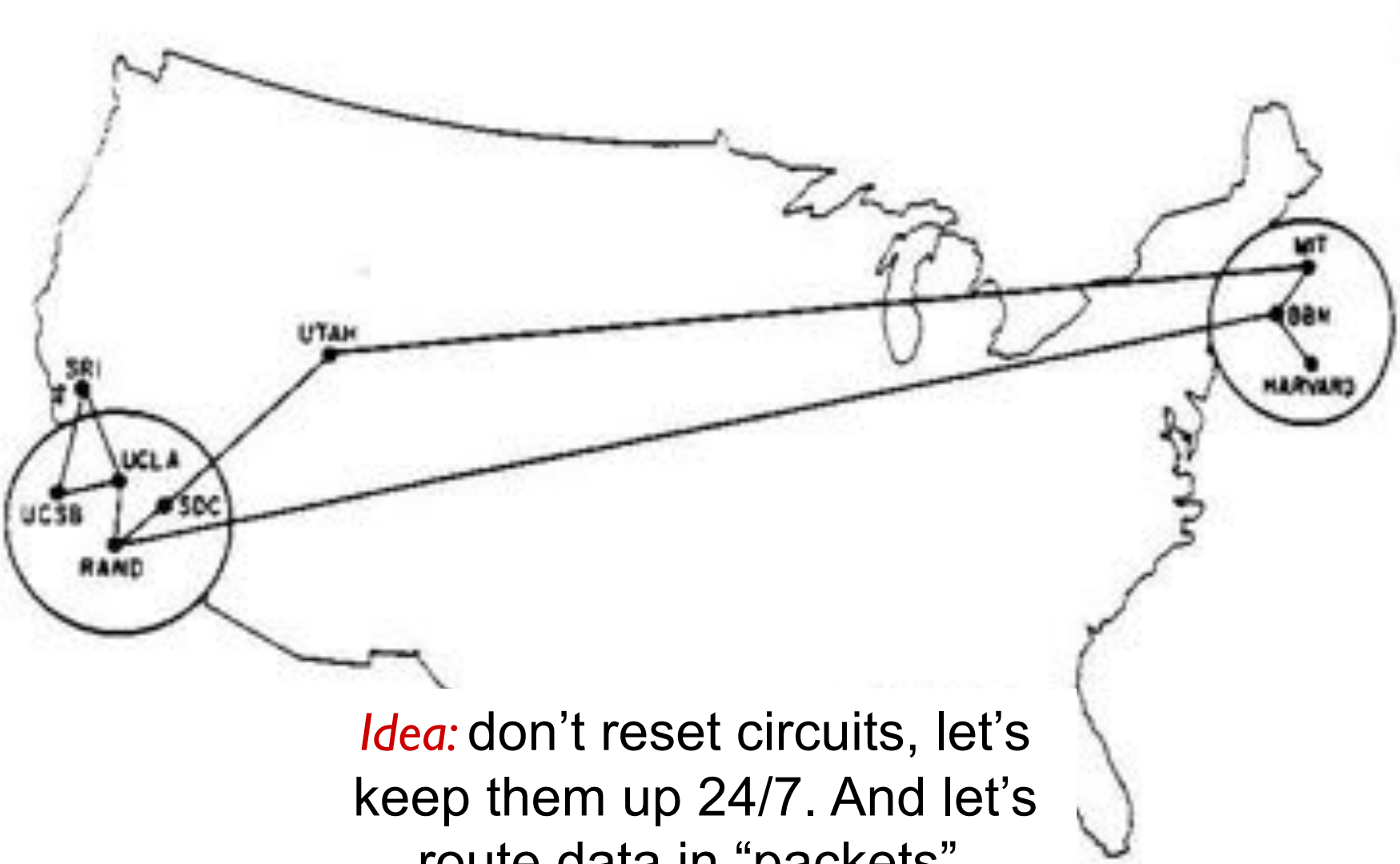
TDM



A bit more history



A bit more history

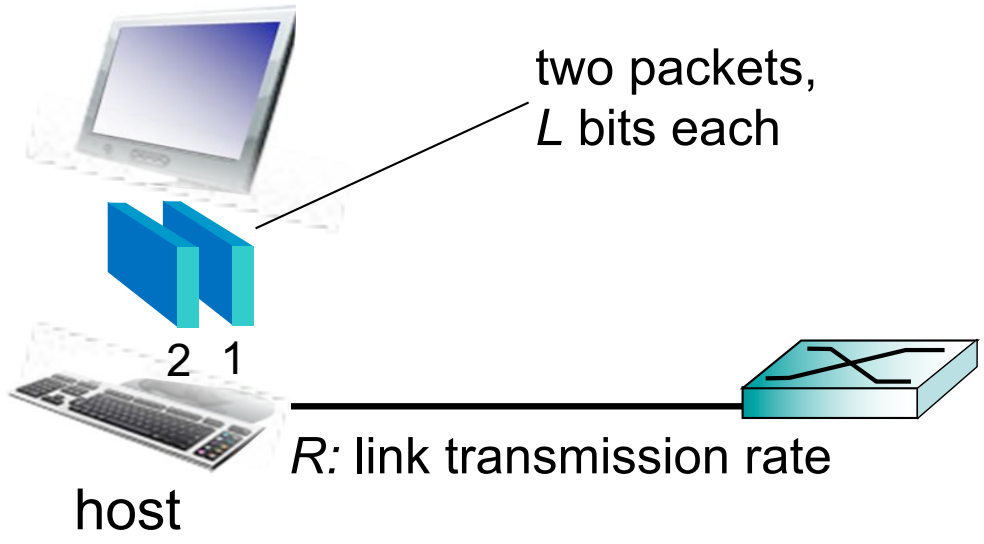


Idea: don't reset circuits, let's keep them up 24/7. And let's route data in "packets".

Host: sends *packets* of data

host sending function:

- takes application message
- breaks into smaller chunks, known as *packets*, of length *L* bits
- transmits packet into access network at *transmission rate R*
  - link transmission rate, aka link *capacity, aka link bandwidth*



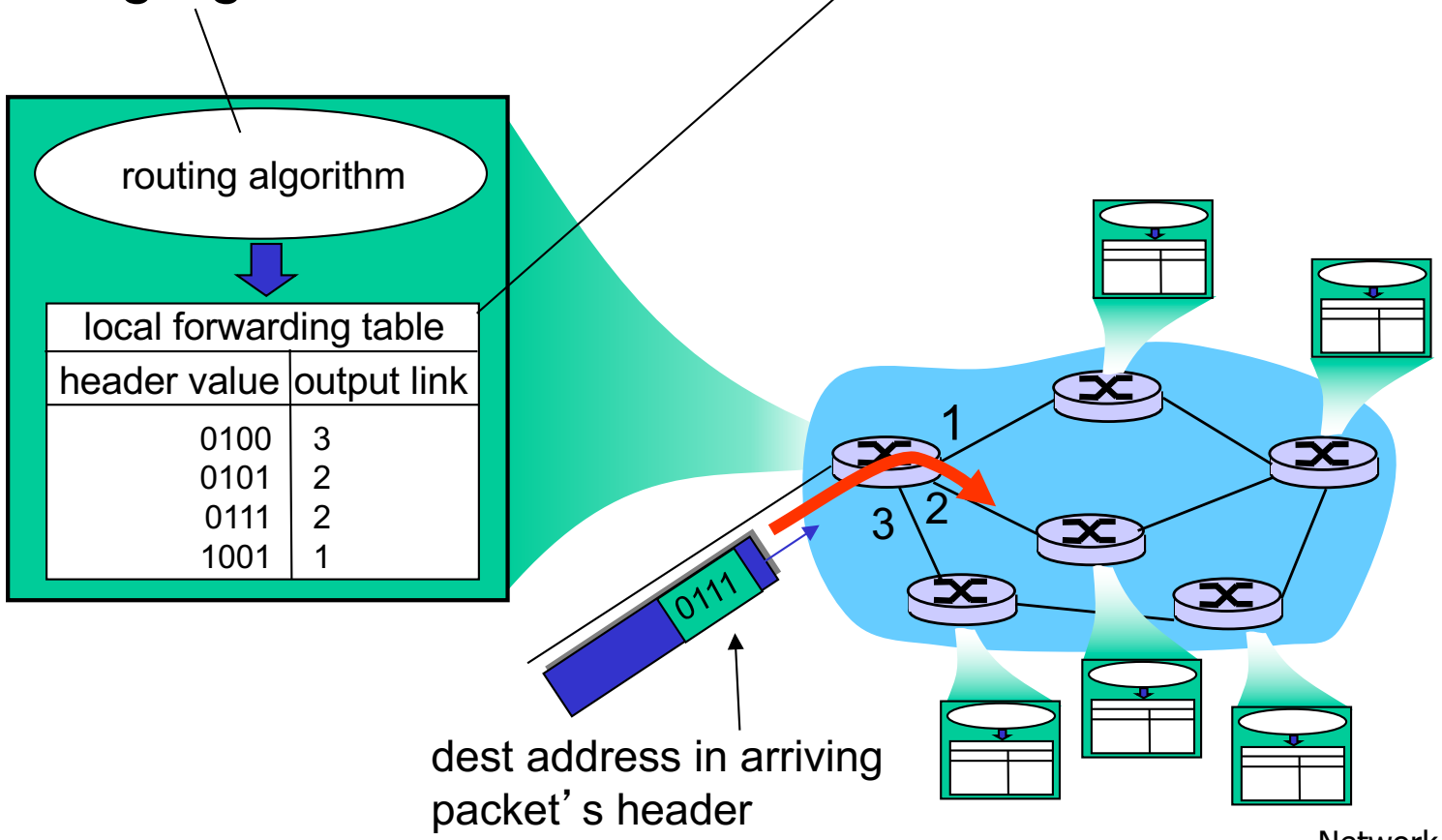
$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

Packet switching

routing: determines source-destination route taken by packets

- *routing algorithms*

forwarding: move packets from router's input to appropriate router output

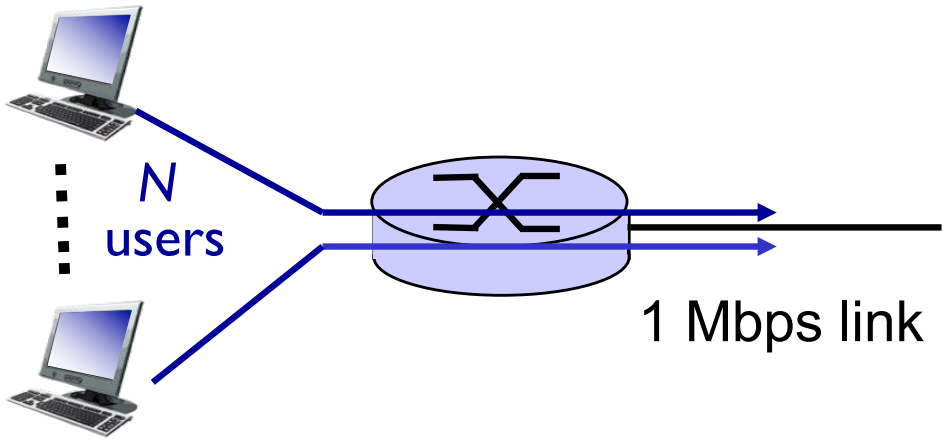


Packet switching versus circuit switching

packet switching allows more users to use network!

example:

- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time



❖ *circuit-switching:*

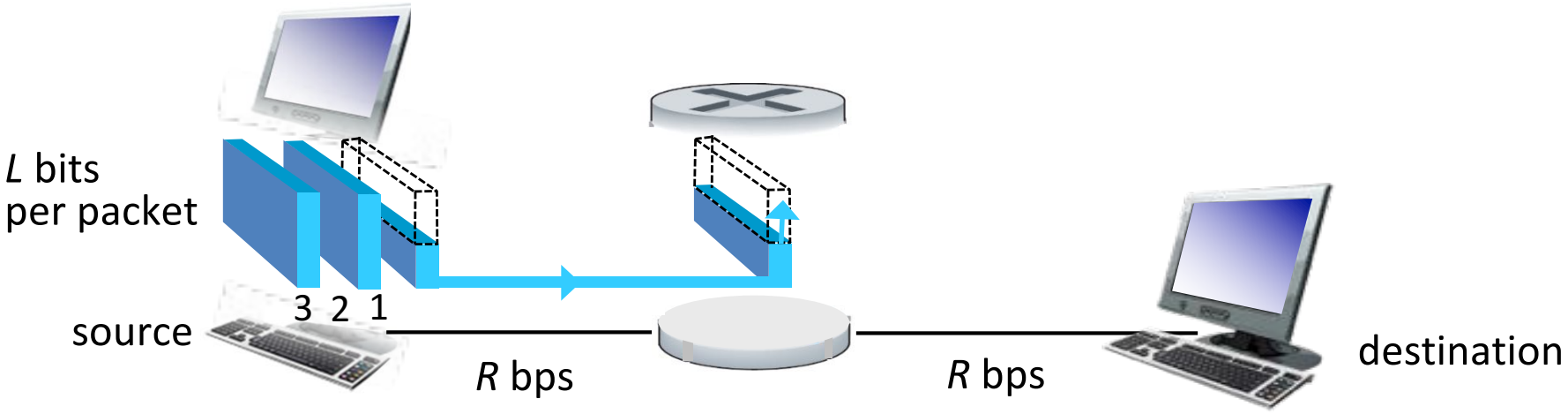
- 10 users

❖ *packet switching:*

- with 35 users, probability > 10 active at same time is less than .0004

Q: what happens if > 35 users?

Packet-switching: store-and-forward



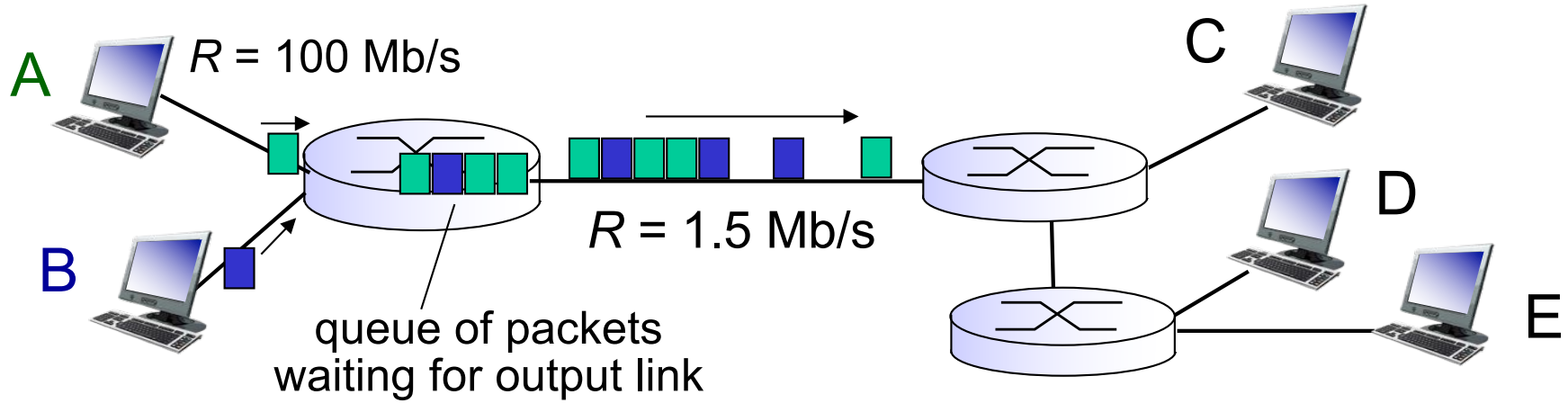
- takes *L*/*R* seconds to transmit (push out) *L*-bit packet into link at *R* bps
- *store and forward:* entire packet must arrive at router before it can be transmitted on next link
- end-end delay = 2*L*/*R* (assuming zero propagation delay)

one-hop numerical example:

- *L* = 7.5 Mbits
- *R* = 1.5 Mbps
- one-hop transmission delay = 5 sec

more on delay shortly …

Packet Switching: queueing delay, loss



queuing and loss:

- If arrival rate (in bits) to link exceeds transmission rate of link for a period of time:
  - packets will queue, wait to be transmitted on link
  - packets can be dropped (lost) if memory (buffer) fills up

Packet switching versus circuit switching

is packet switching a "slam dunk winner?"

- great for bursty data
  - resource sharing
  - simpler, no call setup
- excessive congestion possible: packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- *Q:* How to provide circuit-like behavior?
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem (chapter 7)

Q: human analogies of reserved resources (circuit switching) versus on-demand allocation (packet-switching)?

Protocol "layers"

Networks are complex, with many "pieces":

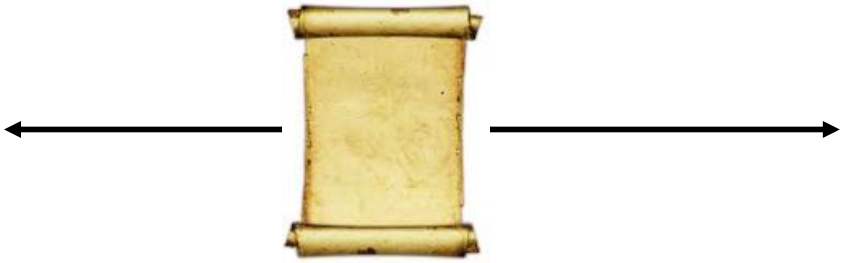
- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:

is there any hope of *organizing* structure of network?

…. or at least our discussion of networks?

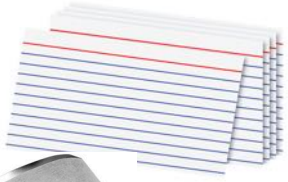
An analogy



An analogy

Writing

Transcription



Coding

Boxing

Delivery

An analogy

Reading

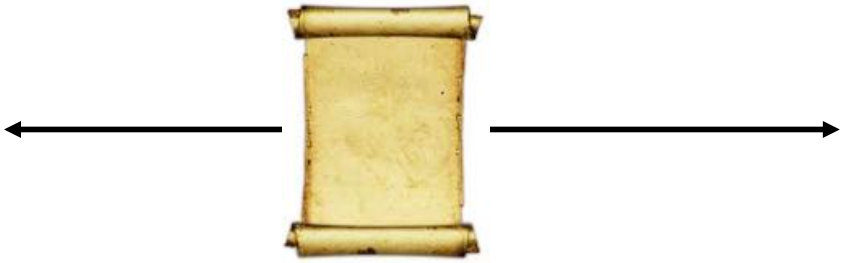
Transcription

Decoding

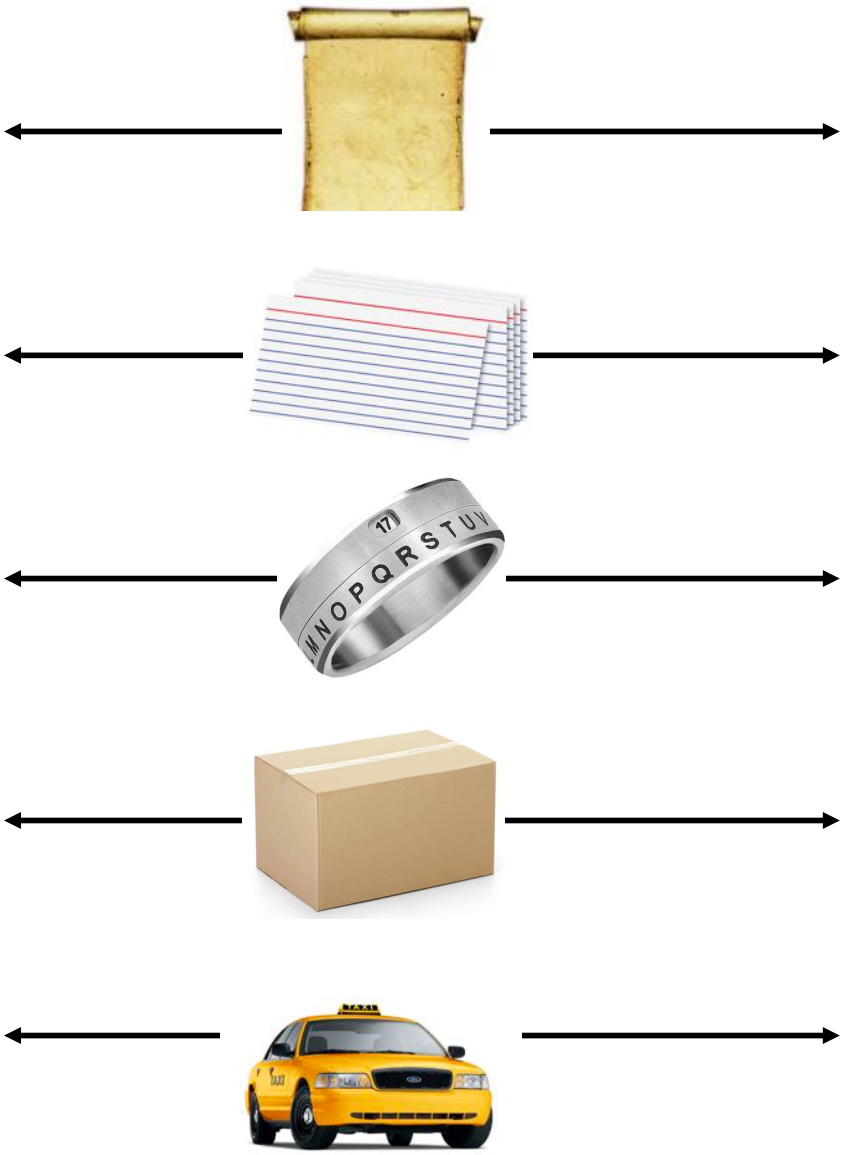
Unboxing

Receiving

An analogy



An analogy



An analogy

Why layering?