CSC358 Week 1

Teaching Team

Instructors:

Larry Zhang (coordinator)
ylzhang@cs.toronto.edu
Office: DH-3070

Michael Liut
michael.liut@utoronto.ca
Office: DH-3017

TAs:

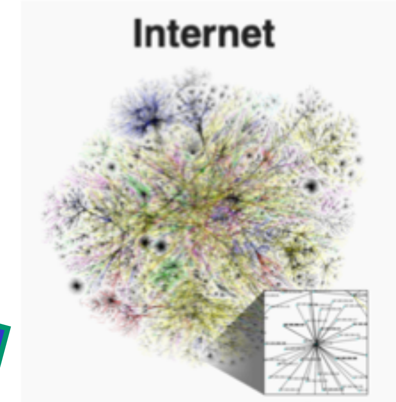
Kyokeun Park
Julian Sequeira

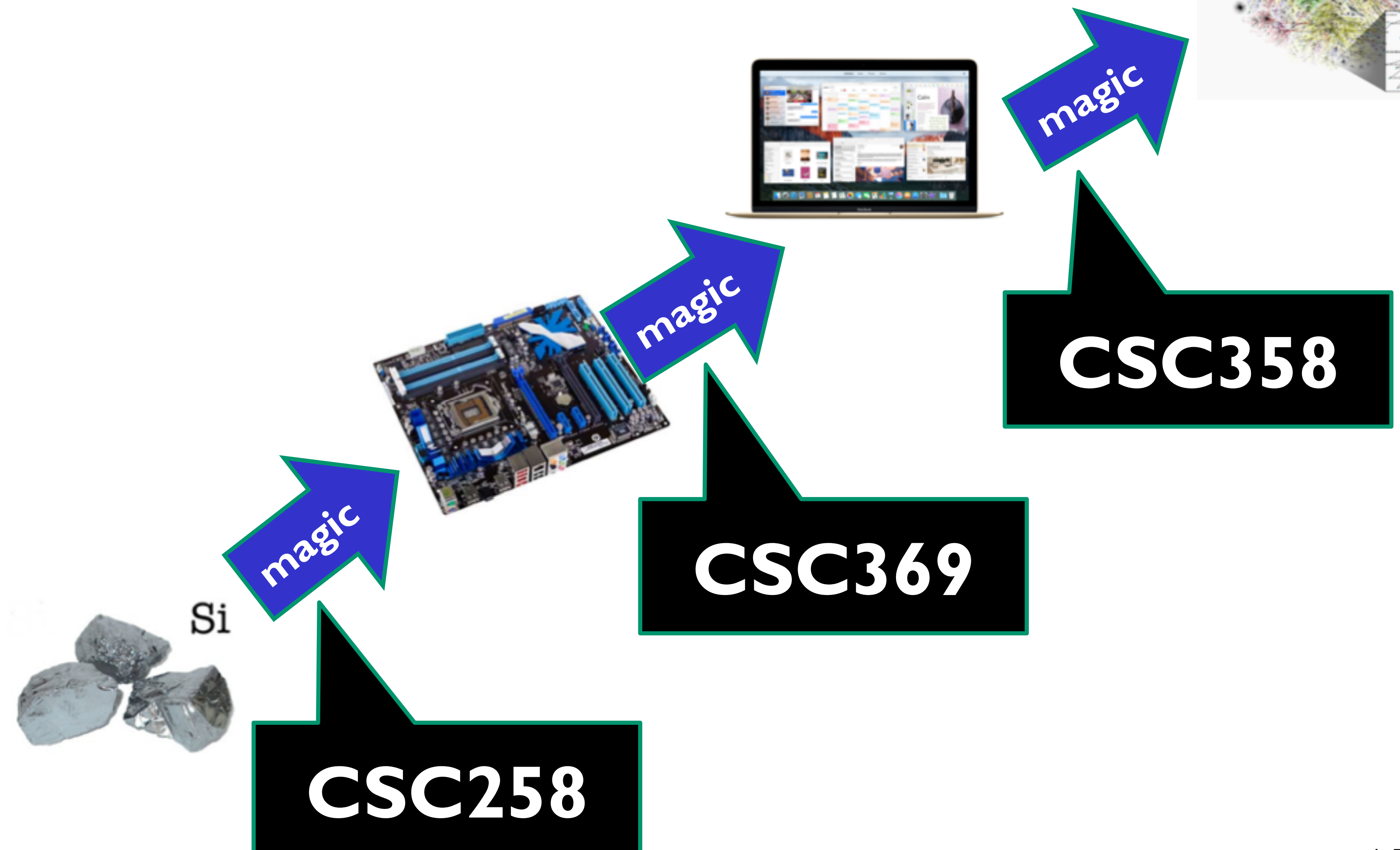
Today's outline

- Logistics (why / what / how)
- Introduction to the Internet

Why CSC358?

Learning the magic





Why CSC358

- How important is the Internet to your life?
- The Internet is the largest engineered system ever created by mankind.
- It’s hard to imagine/believe how people could possibly made it work.
- So let’s learn it.

What's in CSC358

- From top to bottom, the whole design stack of the Internet.
- Before: magic → After: elegant designs that are simple and efficient.
- You will understand how the Internet works.
- You will obtain critical skills to design your own networked system.

How to do well in CSC358

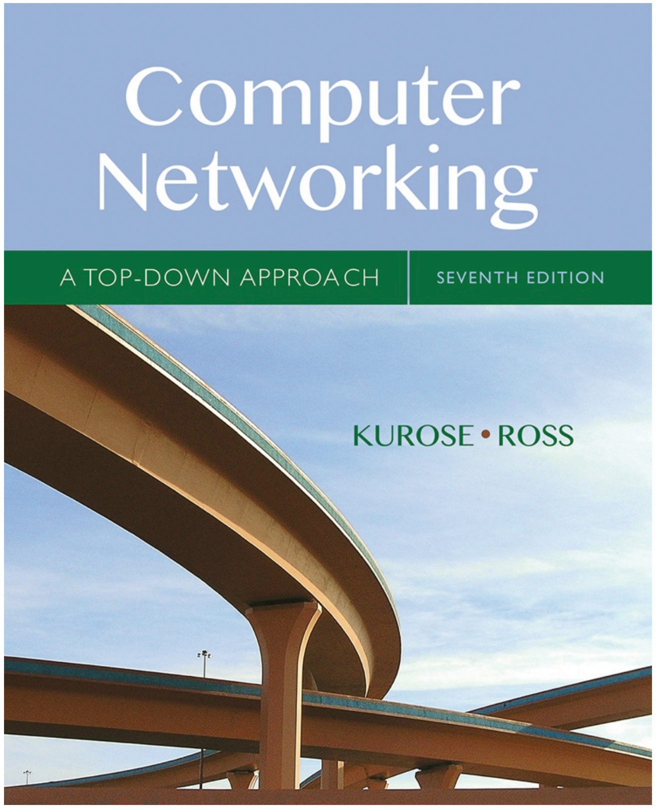
First and foremost

Be Interested.

Course website

https://mcs.utm.utoronto.ca/~358/

Textbook



Computer Networking: A Top Down Approach
7th edition

Jim Kurose, Keith Ross

Pearson/Addison Wesley

April 2016

(any other edition works as well)

Marking Scheme

- 3 assignments: 3 x 10% = 30%
  - Programming assignments in Python and C
- Midterm test: 23%
  - 90 minutes, in class
- Final exam: 47%
  - 3 hours, comprehensive
  - must get >= 40%

Learning components in CSC358

- Lectures
  - Basic concepts and theoretical content
- Assignments
  - Learning by practice, get your hands dirty and play around.
- Tutorials
  - Exercises that prepare you for the test and exam.
  - Help with assignments.
  - As important as lectures to attend!
  - Skipping and just reading solutions will not be even close to the "full experience" of learning.

Prerequisites

- Things we assume you already know
  - Python programming (CSC148)
  - C programming and Unix tools (CSC209)
  - Binary representation and operations (CSC258)
  - Probability and Counting (STA256)
  - Data structures and algorithms (CSC263)

- Review if feeling rusty in any of these!

Get help

- Discussion board
  - link on course website

- Office hours:
  - info on course website

- Get all the help!

Student Feedback

- Give us frequent (e.g., every week) feedbacks on how things are going. It is very useful for improving your learning experience in a timely manner.
- Anonymous feedback form:
  - link on the course website
- Or even better, just drop by and have a chat

Academic Integrity

- **All** assignment submissions will be checked using a plagiarism detection system at the end of the semester.
- Each assignment is 10%, which means any academic offence will be a big deal.
- It's not as simple as "don't cheat/don't copy others"
  - Protect yourself
  - Read this: http://www.cs.toronto.edu/~fpitt/documents/plagiarism.html

Kahoot!

- In-class pop quizzes. To participate, you'll need:
  - be in the class
  - have access to a browser (on a phone, tablet or a laptop), or the Kahoot app
- has nothing (directly) to do with your course grade

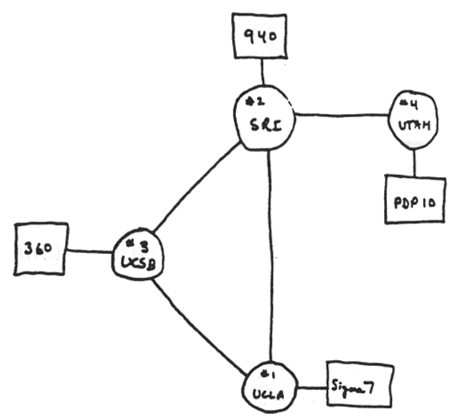
- CSC358 is dense, with a lot of different topics to learn, but it'll be rewarding when you've learned it.

Intro to the Internet
an overview

Internet history

1961-1972: Early packet-switching principles

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational
- 1972:
  - ARPAnet public demo
  - NCP (Network Control Protocol) first host-host protocol
  - first e-mail program
  - ARPAnet has 15 nodes



THE ARPA NETWORK

Internet history

1972-1980: Internetworking, new and proprietary nets

- 1970: ALOHAnet satellite network in Hawaii
- 1974: Cerf and Kahn - architecture for interconnecting networks
- 1976: Ethernet at Xerox PARC
- late70's: proprietary architectures: DECnet, SNA, XNA
- late 70's: switching fixed length packets (ATM precursor)
- 1979: ARPAnet has 200 nodes

Cerf and Kahn's internetworking principles:

- minimalism, autonomy - no internal changes required to interconnect networks
- best effort service model
- stateless routers
- decentralized control

define today's Internet architecture

Internet history

1980-1990: new protocols, a proliferation of networks

- 1983: deployment of TCP/IP
- 1982: smtp e-mail protocol defined
- 1983: DNS defined for name-to-IP-address translation
- 1985: ftp protocol defined
- 1988: TCP congestion control

- new national networks: CSnet, BITnet, NSFnet, Minitel
- 100,000 hosts connected to confederation of networks

Internet history

1990, 2000's: commercialization, the Web, new apps

- early 1990's: ARPAnet decommissioned
- 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- early 1990s: Web
  - hypertext [Bush 1945, Nelson 1960's]
  - HTML, HTTP: Berners-Lee
  - 1994: Mosaic, later Netscape
  - late 1990's: commercialization of the Web

late 1990's – 2000's:

- more killer apps: instant messaging, P2P file sharing
- network security to forefront
- est. 50 million host, 100 million+ users
- backbone links running at Gbps

Internet history

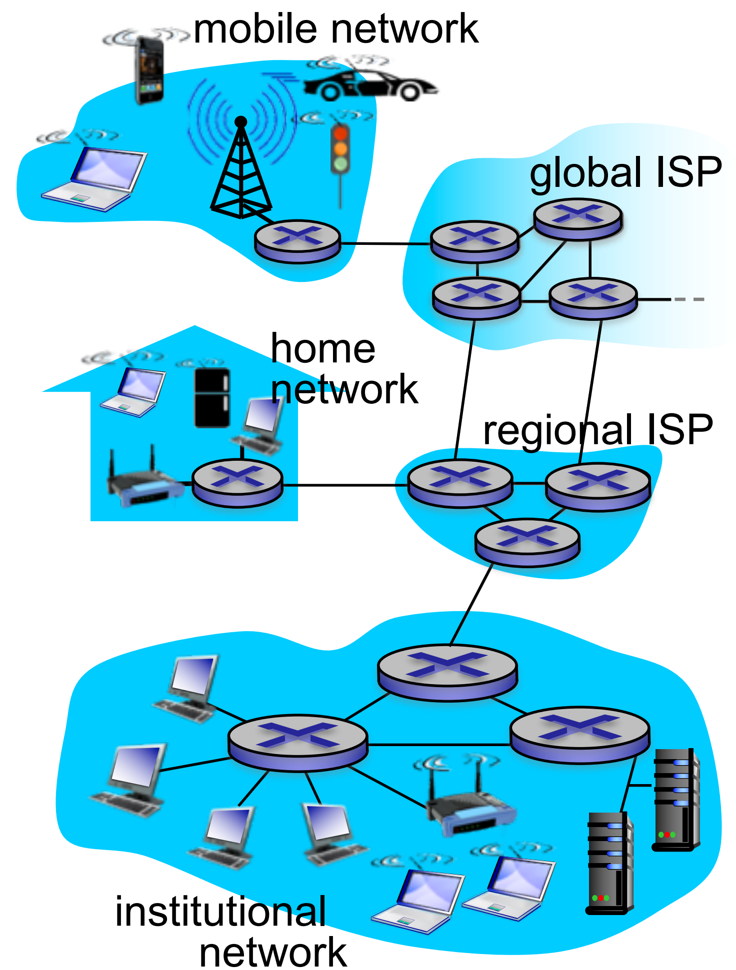
2005-present

- ~5B devices attached to Internet (2016)
  - smartphones and tablets
- aggressive deployment of broadband access
- increasing ubiquity of high-speed wireless access
- emergence of online social networks:
  - Facebook: ~ one billion users
- e-commerce, universities, enterprises running their services in "cloud" (e.g., Amazon EC2)

Structure of the Internet

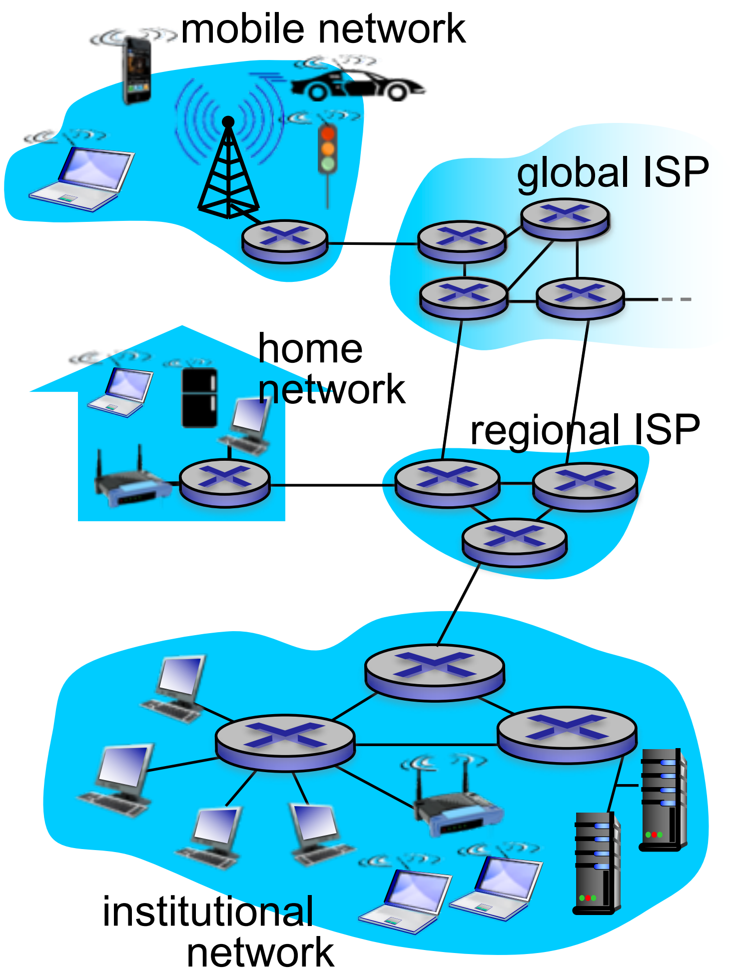
What's the Internet: "nuts and bolts" view

- billions 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*



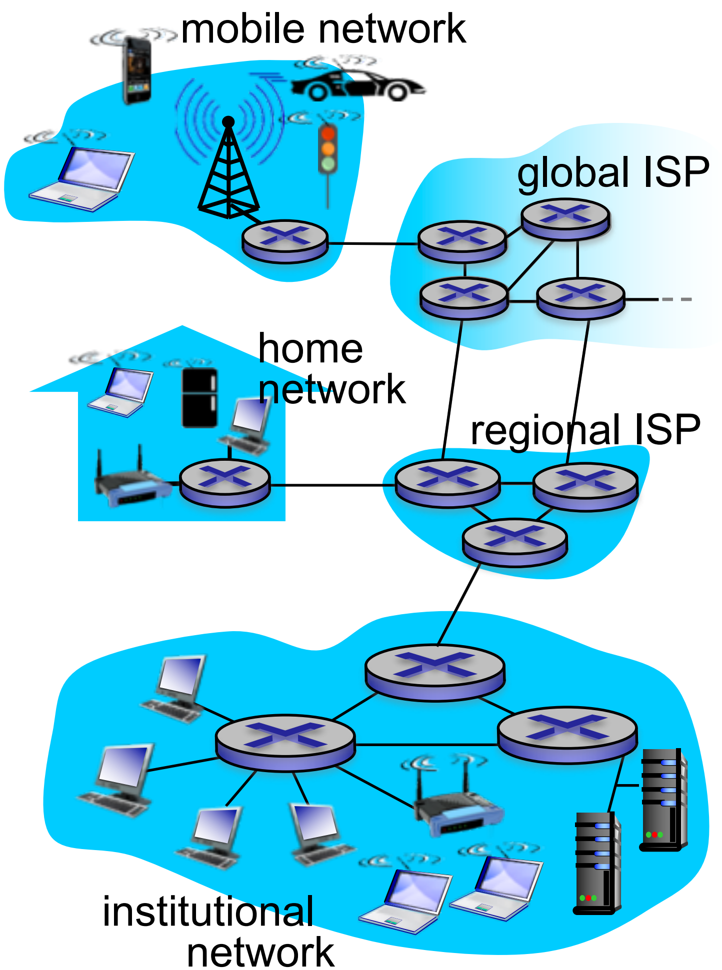
What's the Internet: "nuts and bolts" view

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



Network structure

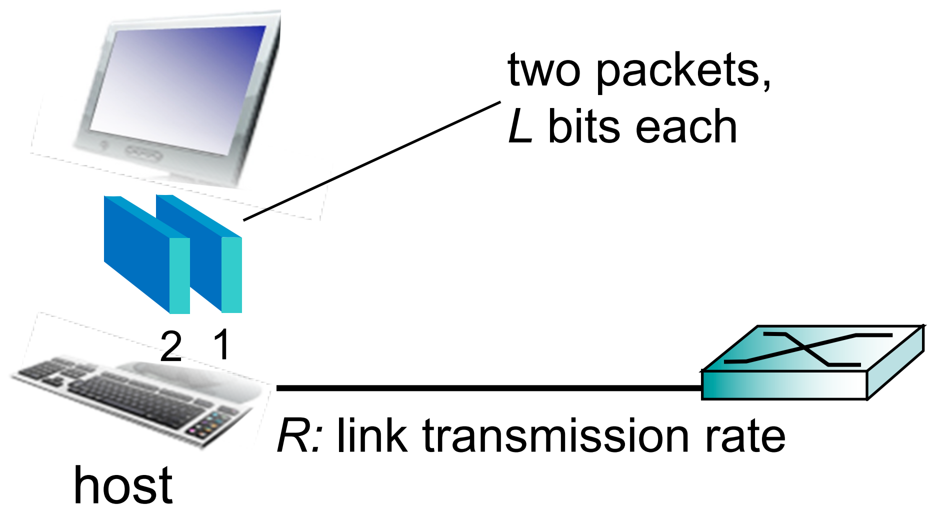
- *network edge:*
  - hosts: clients and servers
  - servers often in data centers
- *access networks, physical media:* wired, wireless communication links
  - DSL, Cable, Wireless LAN, LTE, etc.
- *network core:*
  - interconnected routers
  - network of networks



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*



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

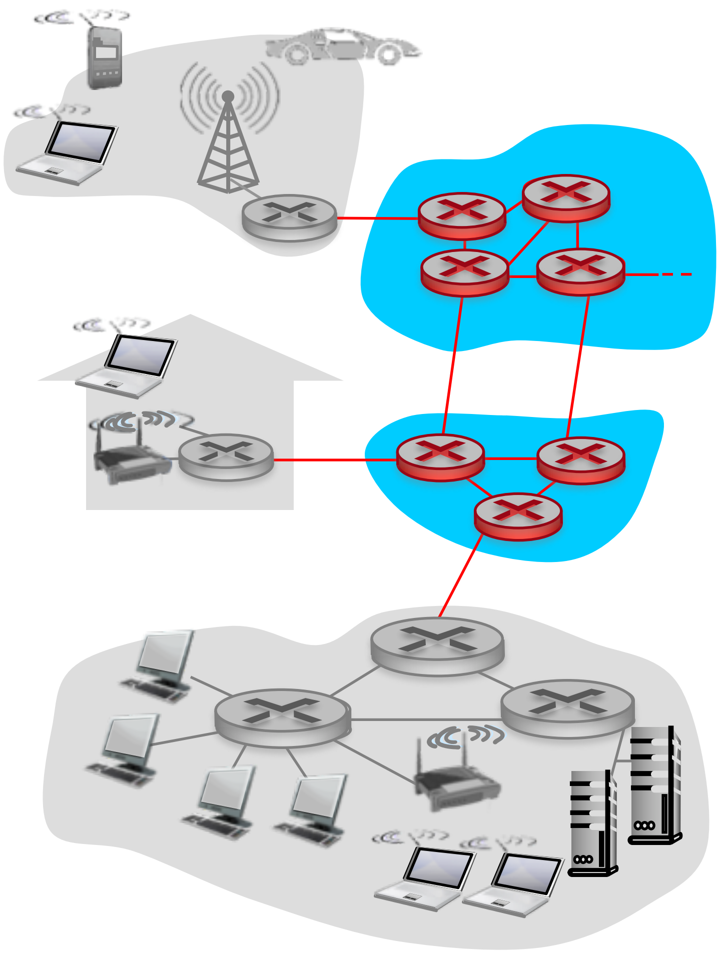
Note on units

- In this course, assume the following conversion

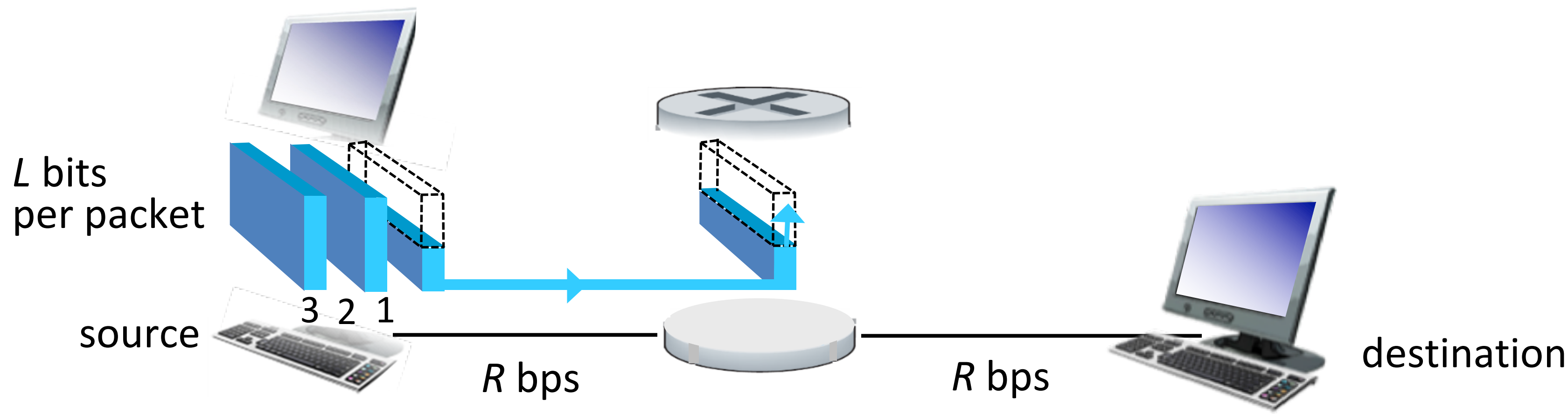
- 1 Gb = 1000 Mb
- 1 Mb = 1000 kb
- 1 kb = 1000 bits

The trip taken by a packet

- mesh of interconnected routers
- A packet may be forwarded by several routers before reaching the destination.

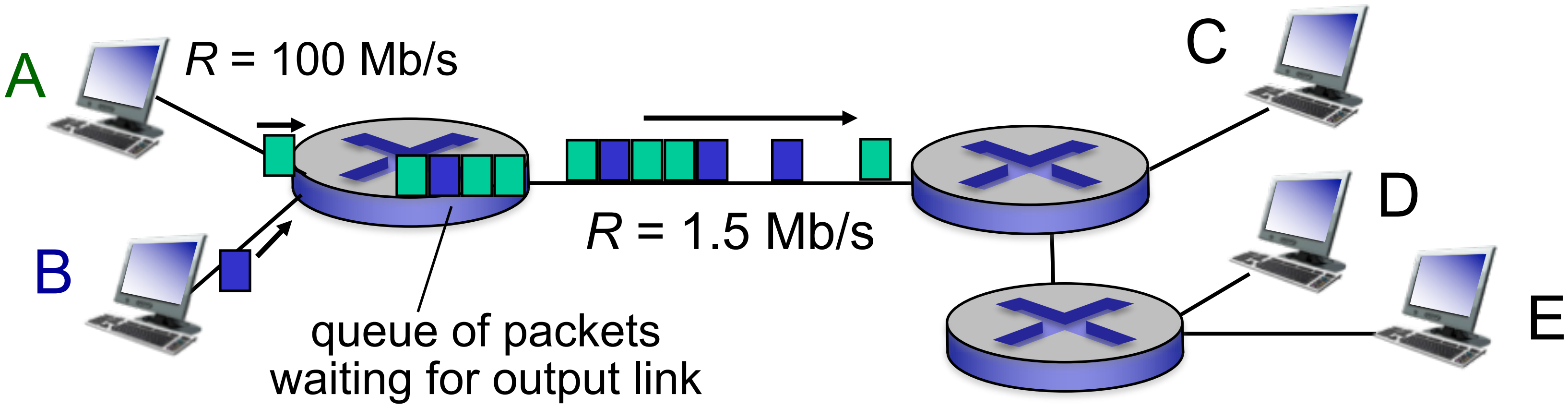


Packet-switching: store-and-forward



- 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 trans. delay = 2*L*/*R*
  - (assuming zero propagation delay, more on this later.)

Packet Switching: queueing delay, loss

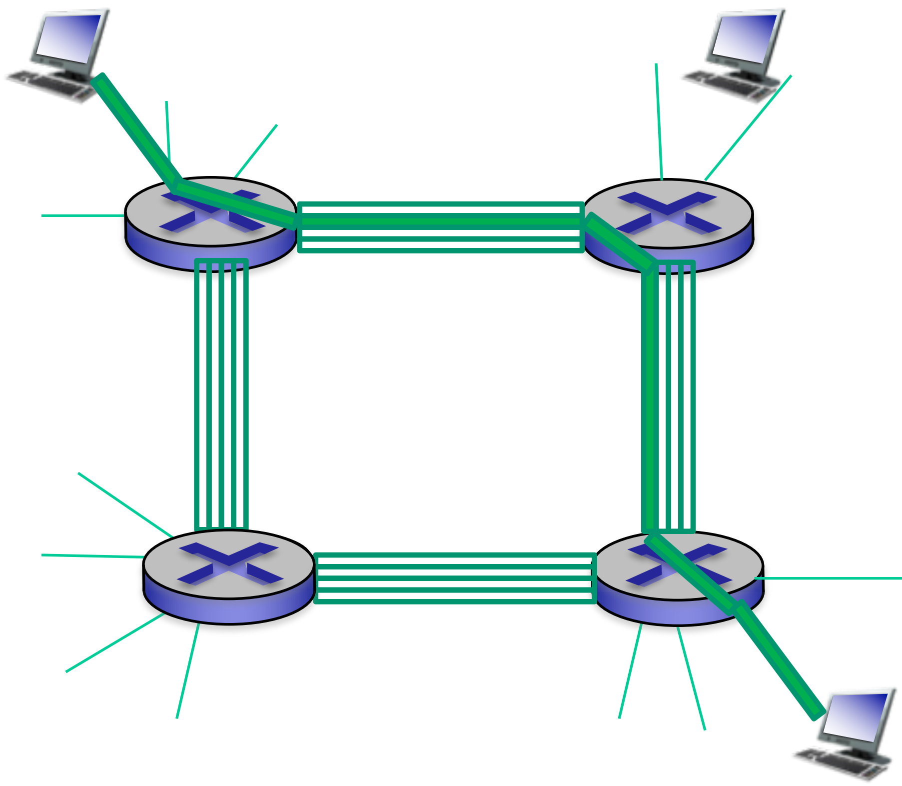


queuing and loss:

- 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

Alternative core: circuit switching

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

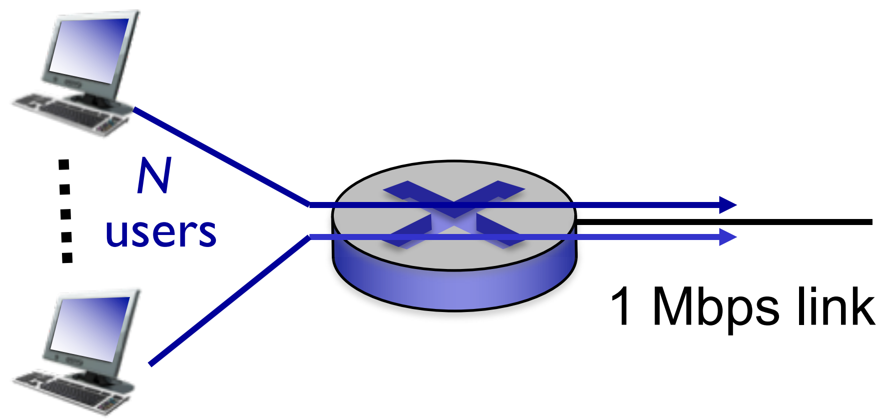


- dedicated resources: no sharing
  - circuit-like (guaranteed) performance
- circuit segment idle if not used by call *(no sharing)*
- commonly used in traditional telephone networks

Packet switching vs circuit switching

example:

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



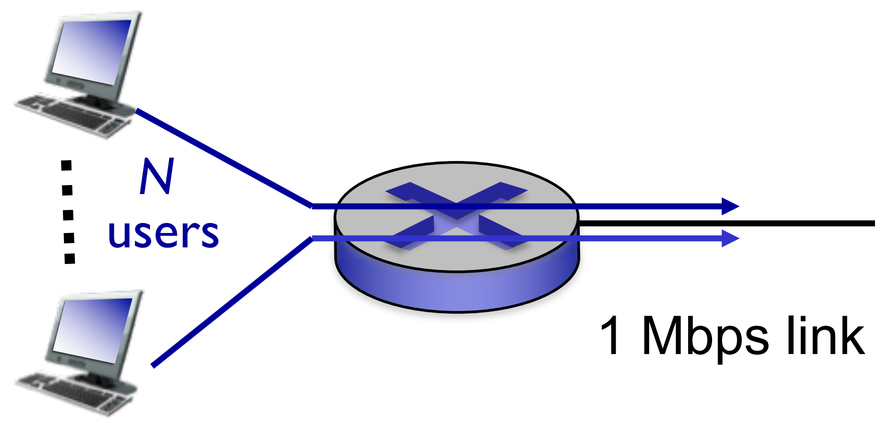
Using Circuit-Switching, how many users are allowed?

Each user reserves 100 Kb/s, no sharing,
so maximum 10 users allowed (1Mbps/100Kbps)

Packet switching vs circuit switching

example:

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



Using Packet Switching, how many users are allowed?

Technically, no limit, as long as <= 10 users are active at the same time. What's the probability of that, assuming we have 35 users in total?

- ~99.96% (we'll do the calculation in the tutorial, try it first!)
- Packet switching allows more users to use network!

Packet switching vs circuit switching

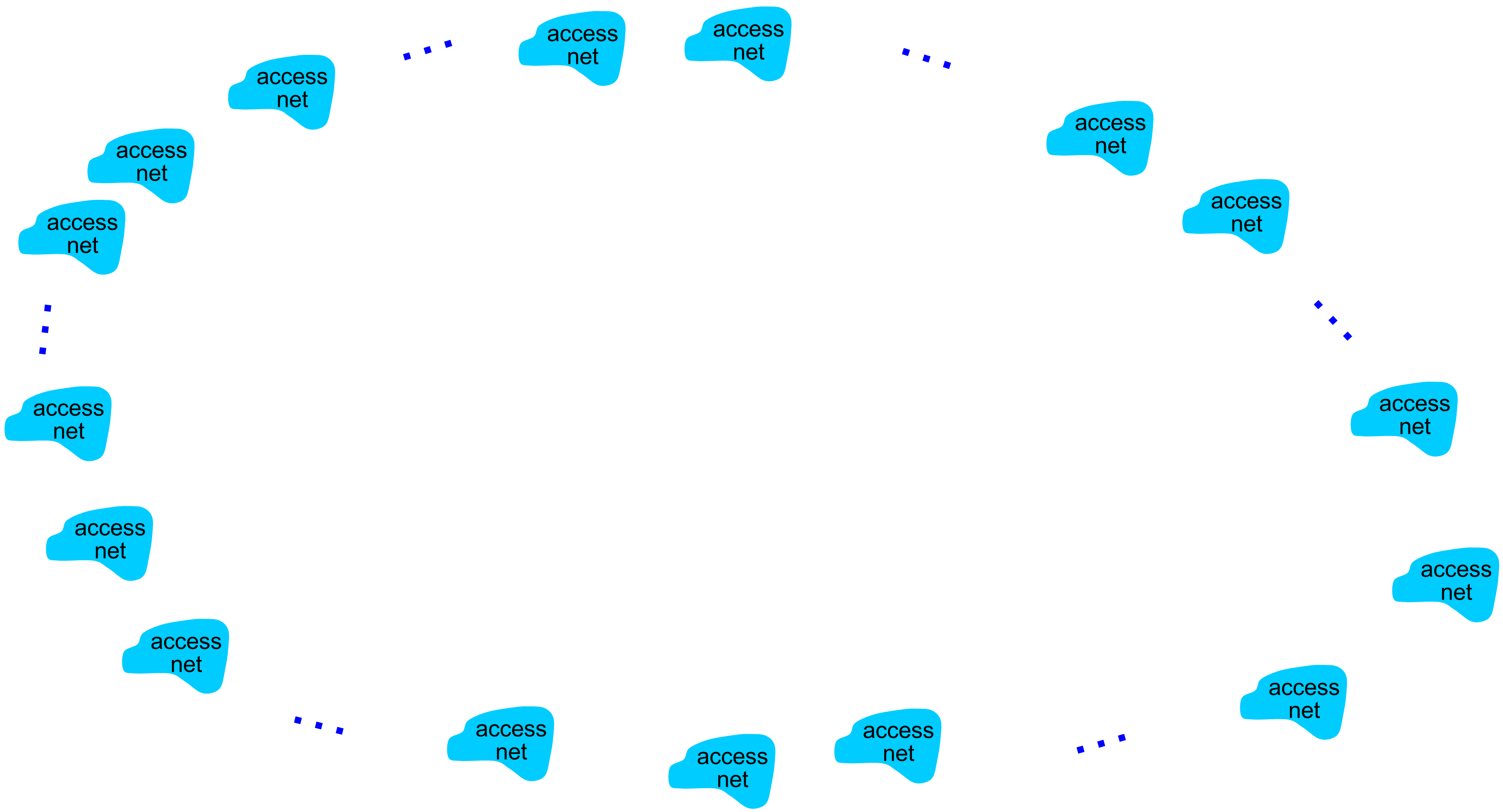
is packet switching a "slam dunk winner?"

- 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
- Important design idea:
  - reserved resource vs on-demand allocation
  - real-life examples?

Network of Networks

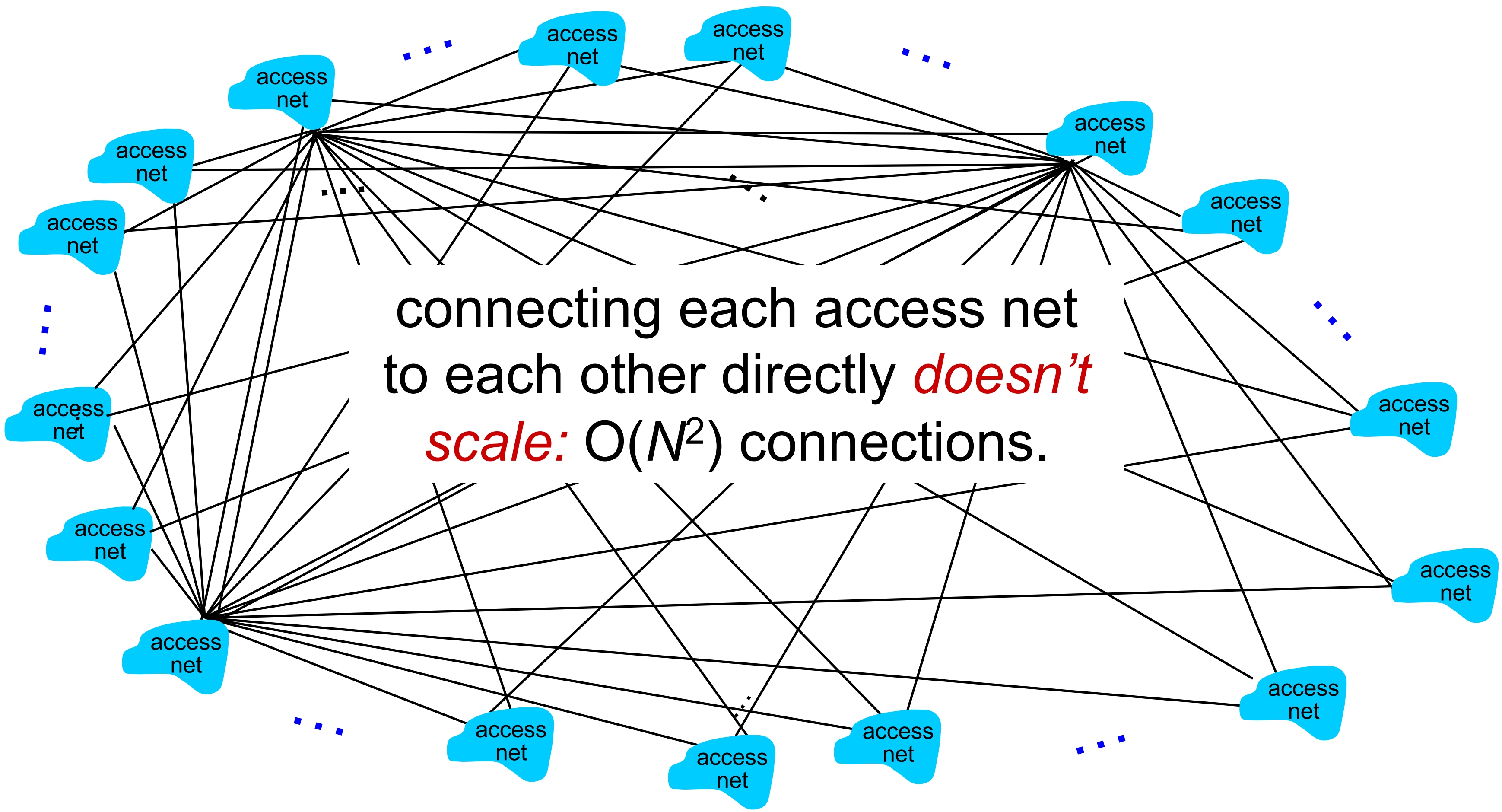
Internet structure: network of networks

Question: given *millions* of access nets, how to connect them together?



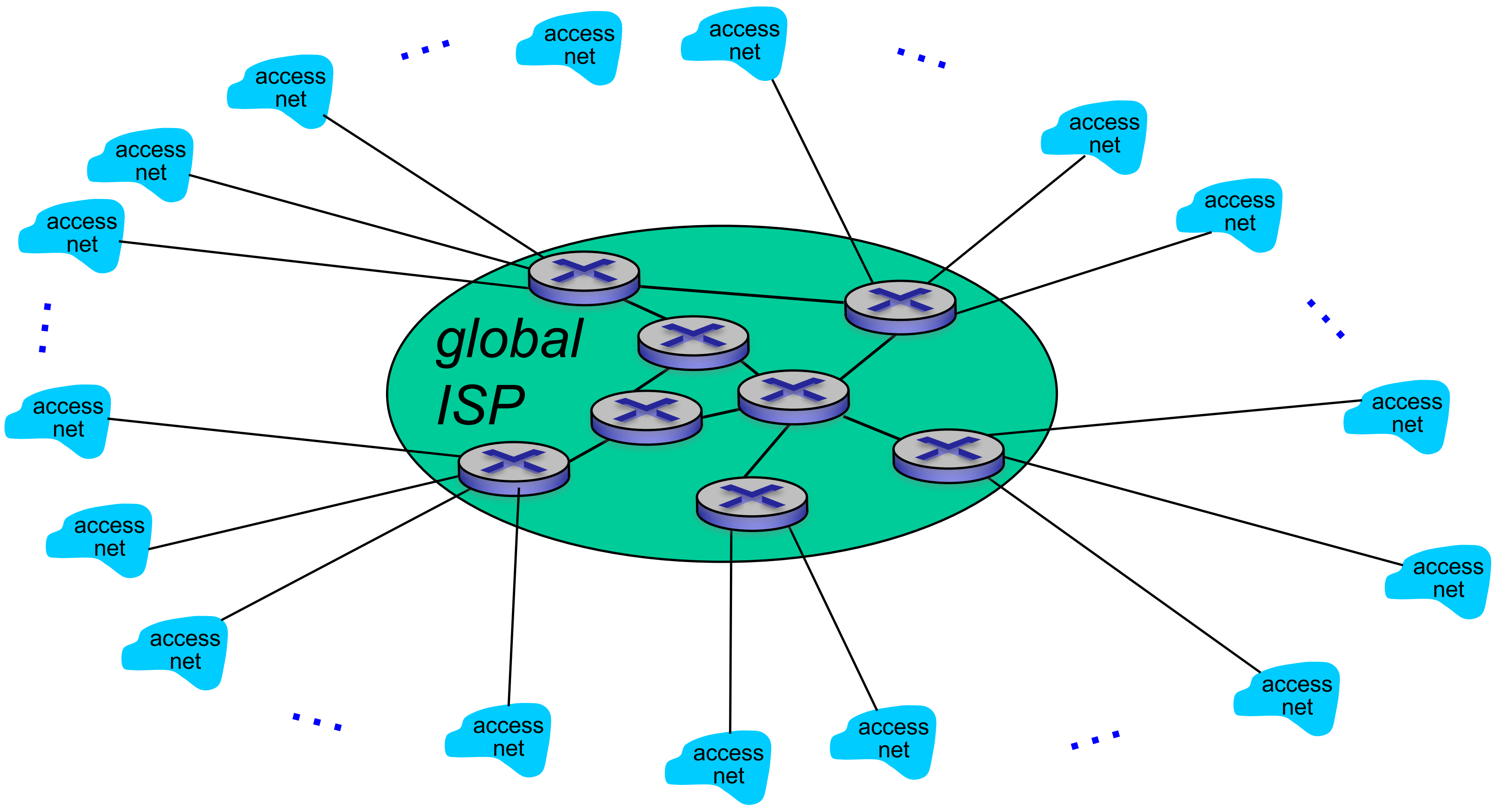
Internet structure: network of networks

Option: connect each access net to every other access net?



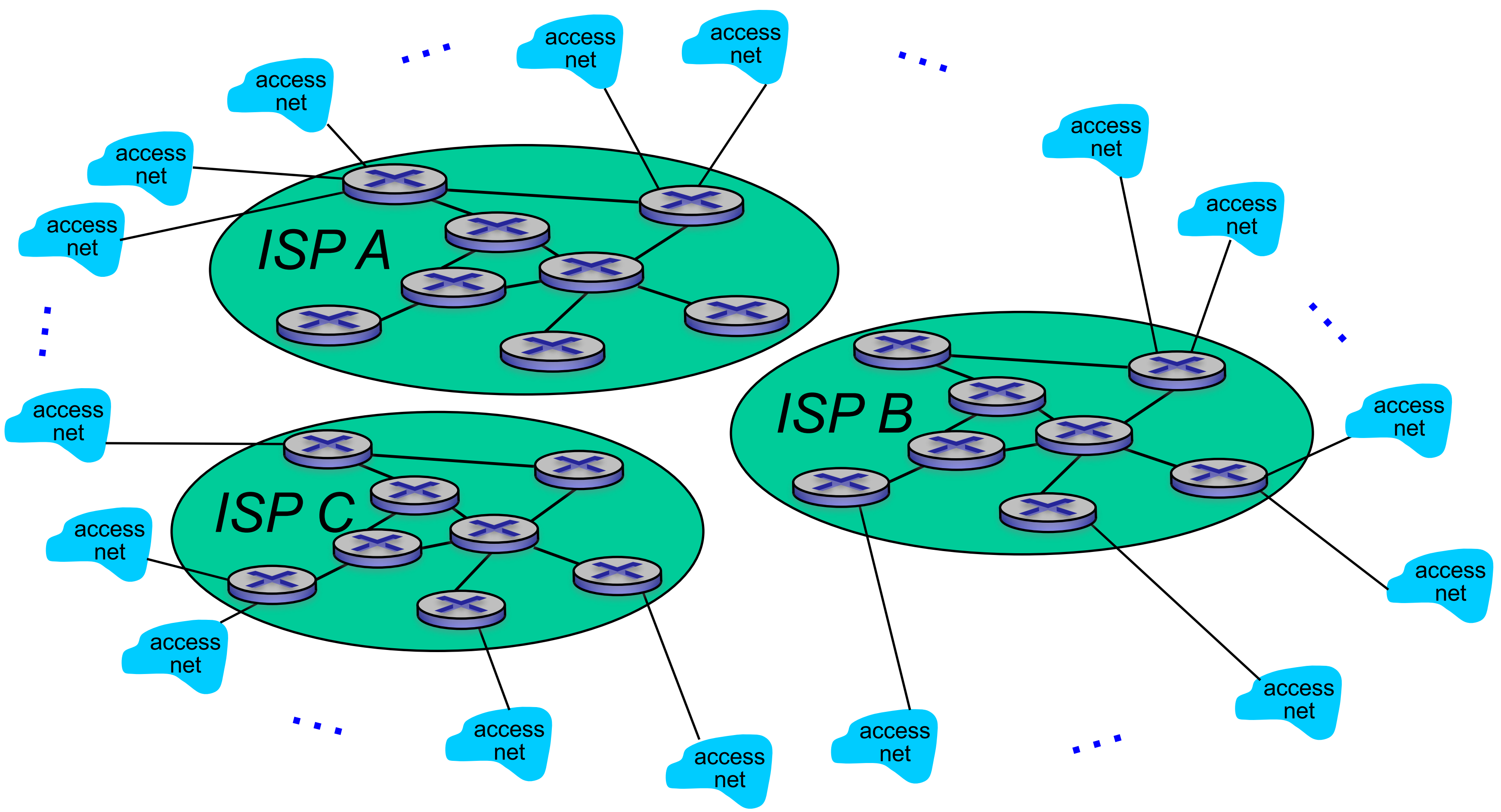
Internet structure: network of networks

Option: connect each access net to one global transit ISP?
Customer and provider of the ISP have economic agreement.



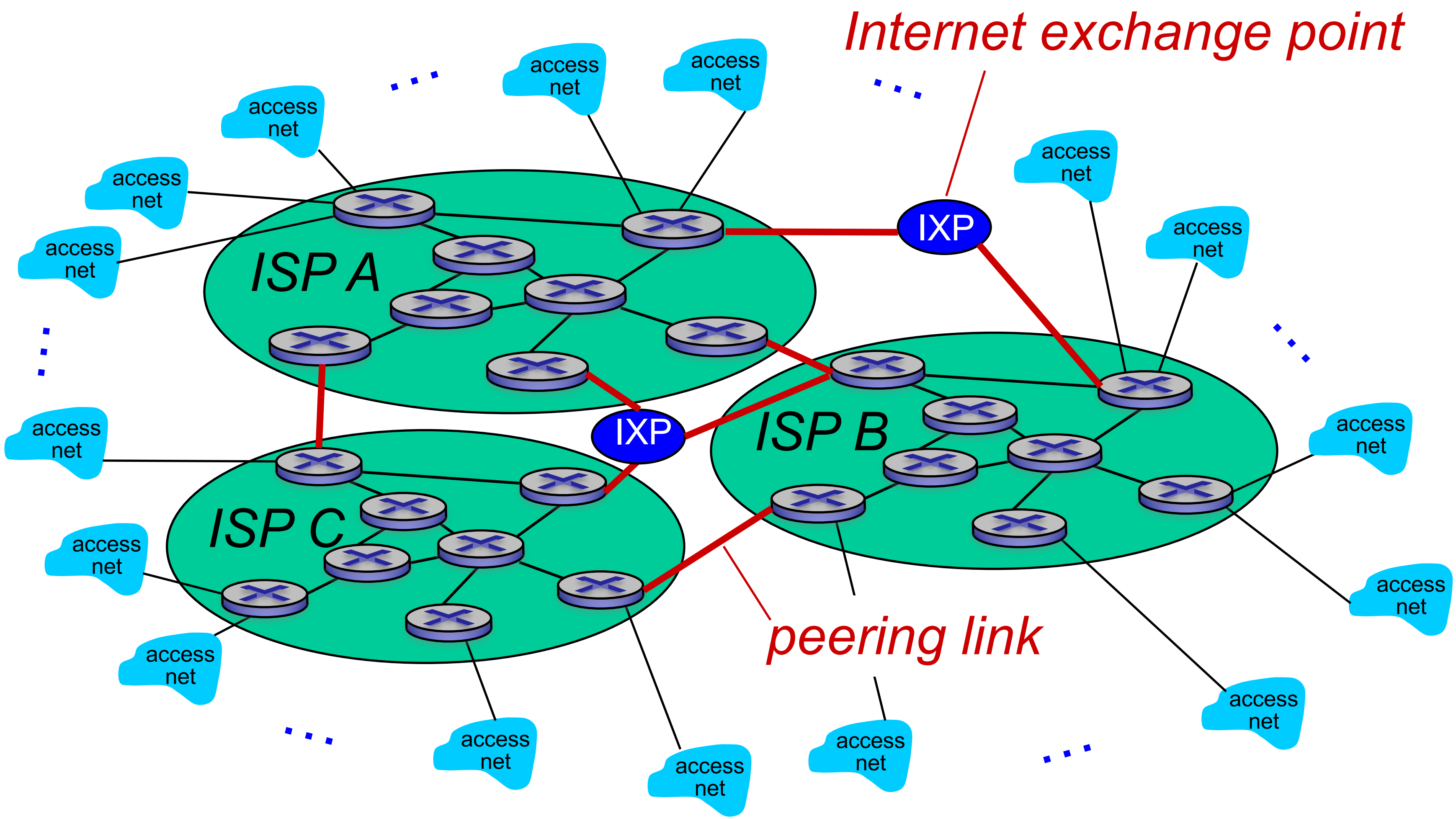
Internet structure: network of networks

But if one global ISP is viable business, there will be competitors ….



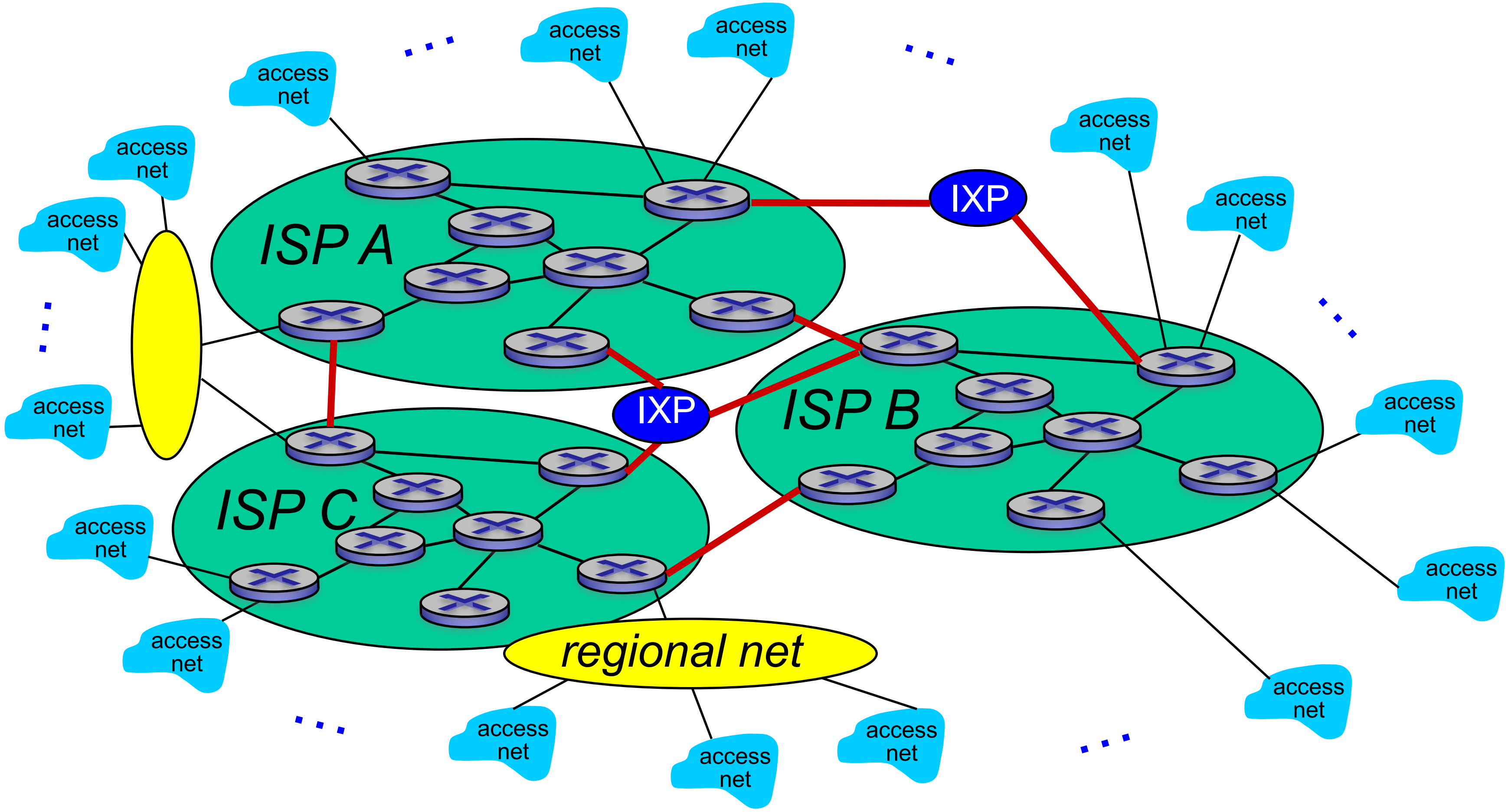
Internet structure: network of networks

But if one global ISP is viable business, there will be competitors …. which must be interconnected



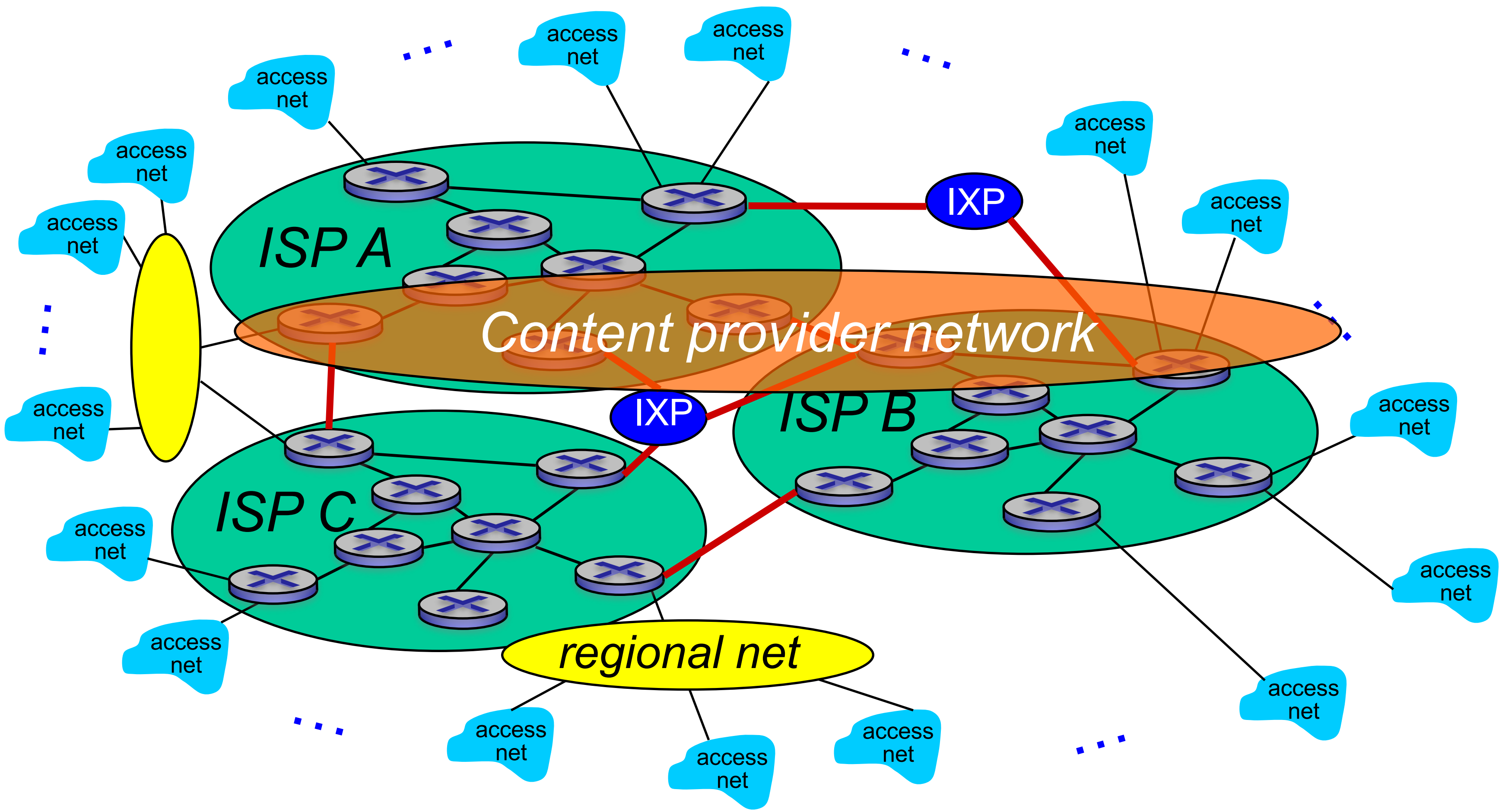
Internet structure: network of networks

… and regional networks may arise to connect access nets to ISPs

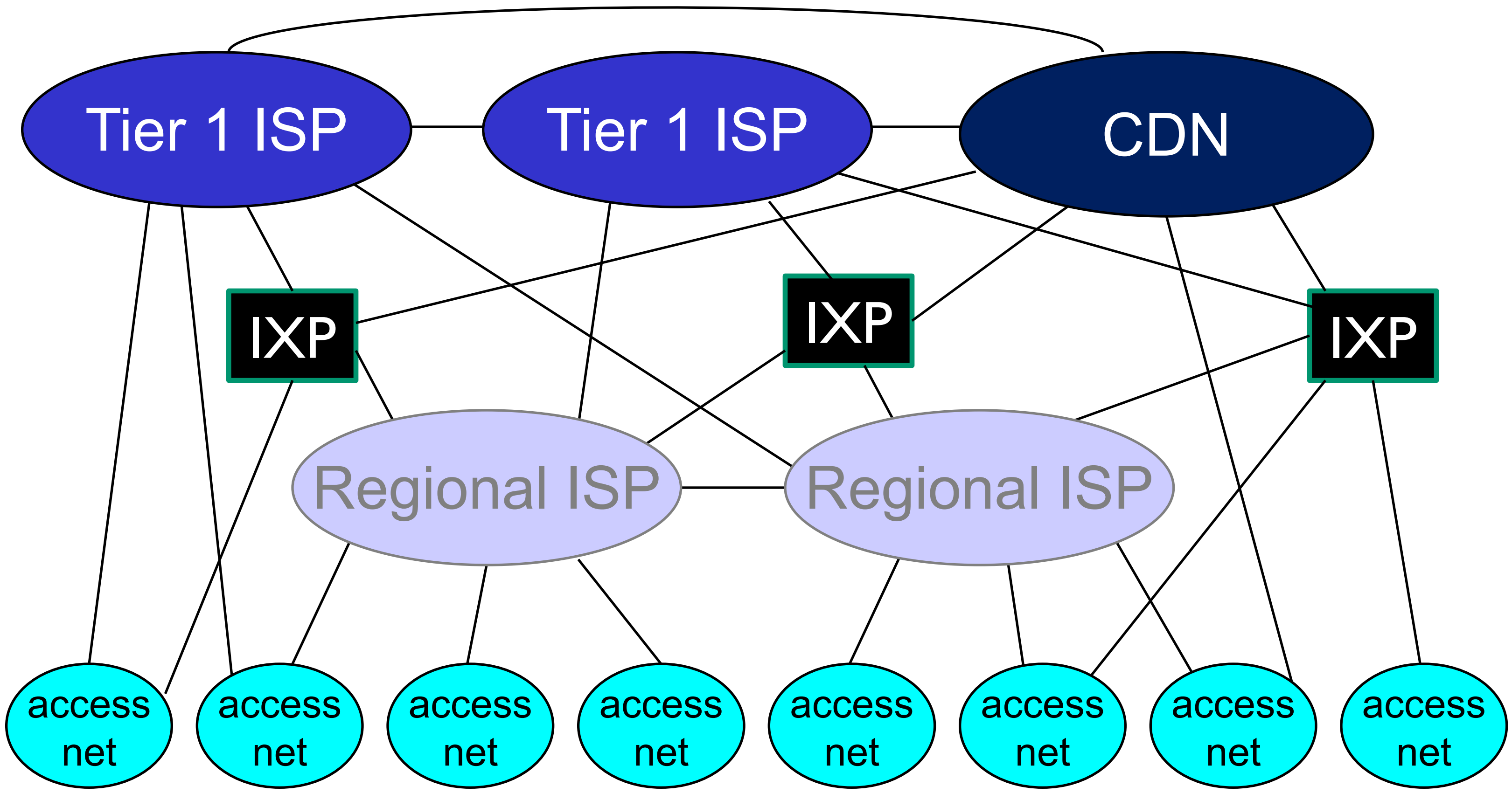


Internet structure: network of networks

… and content provider networks (e.g., Google, Microsoft, Netflix) may run their own network, to bring services, content close to end users



Internet structure: network of networks



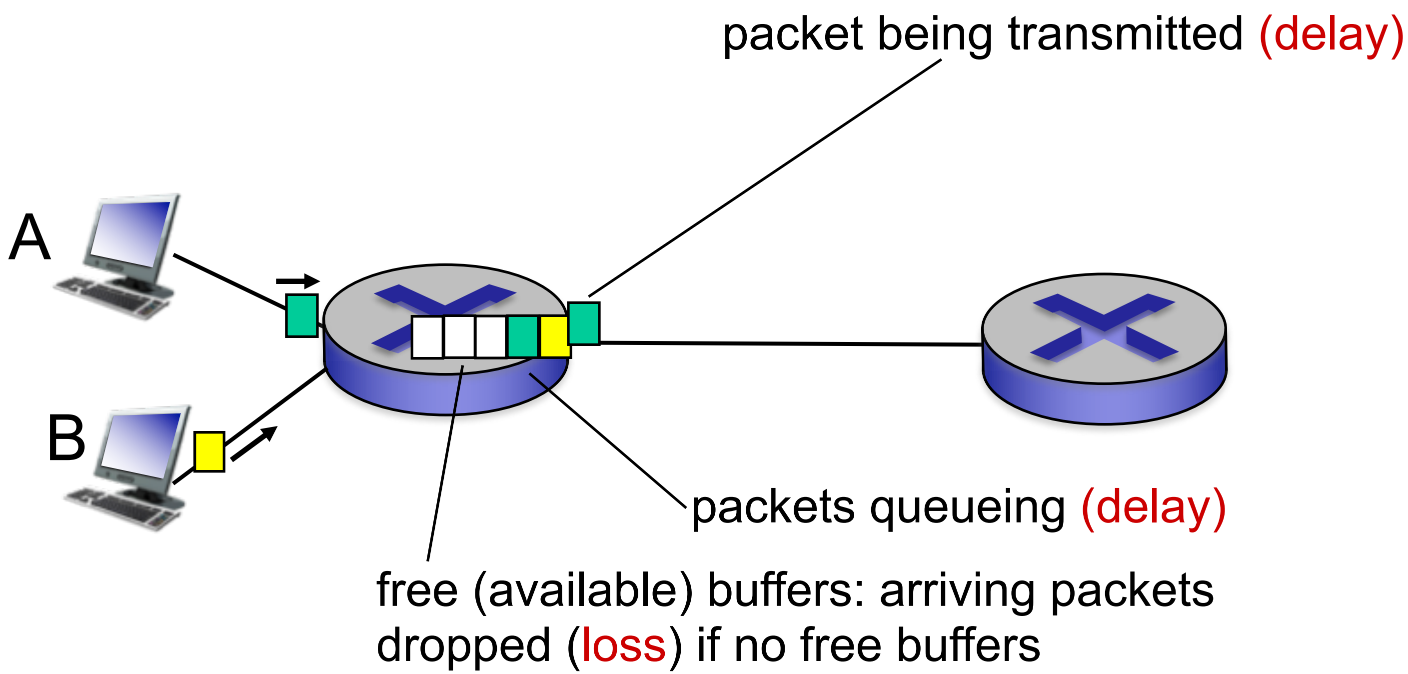
- at center: small # of well-connected large networks
  - "tier-1" commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT), national & international coverage
  - content provider network (e.g., Google): private network that connects it data centers to Internet, often bypassing tier-1, regional ISPs

Delay, Loss, Throughput

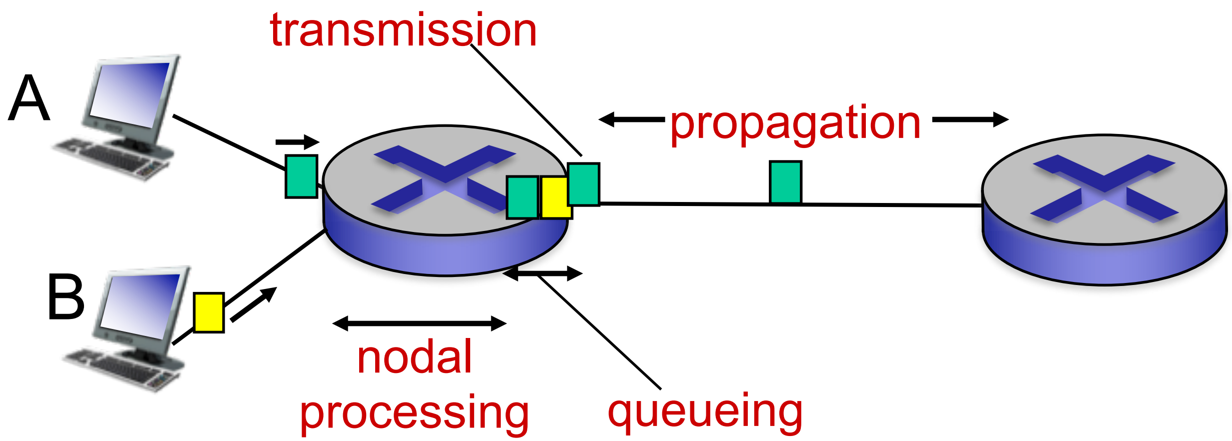
How do loss and delay occur?

packets *queue* in router buffers

- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn



Four sources of packet delay

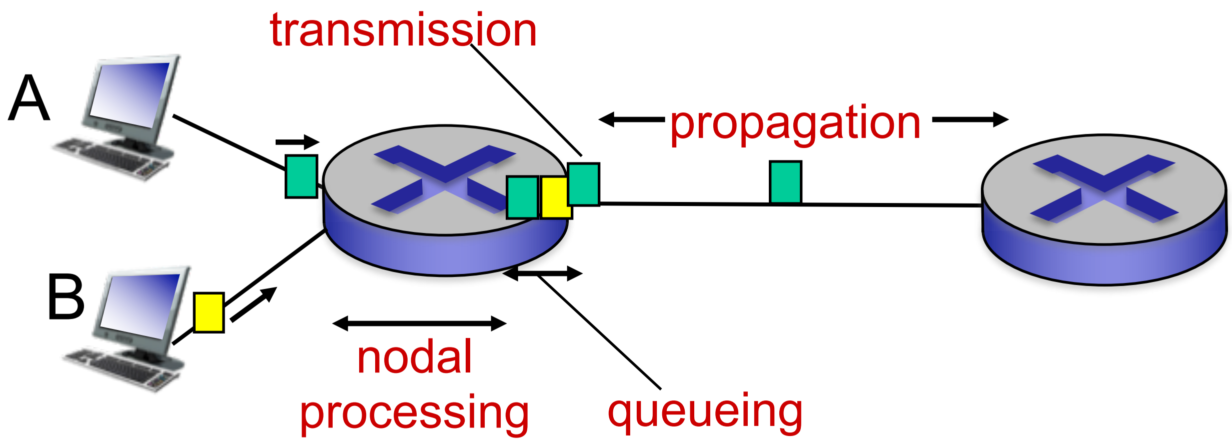


$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{proc}: nodal processing
- check bit errors
- determine output link
- typically < msec

d_{queue}: queueing delay
- time waiting at output link for transmission
- depends on congestion level of router

Four sources of packet delay



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{trans}: transmission delay:

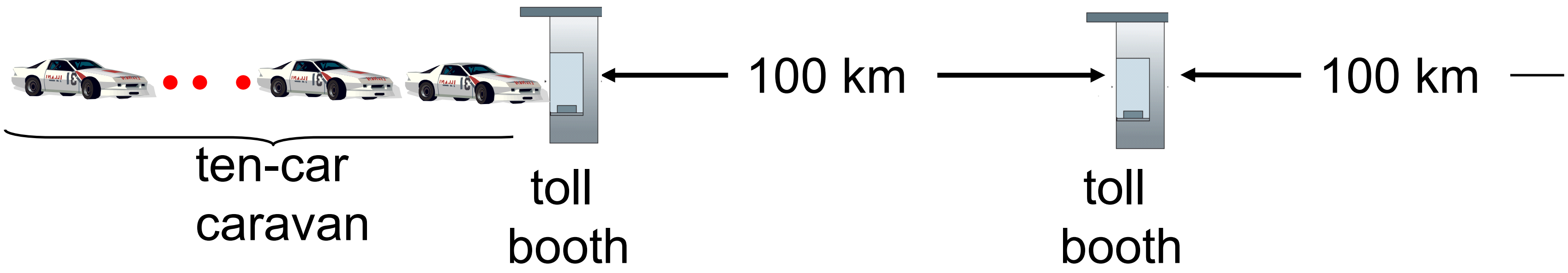
- *L*: packet length (bits)
- *R*: link *bandwidth (bps)*
- $d_{trans} = L/R$

d_{prop}: propagation delay:

- *d*: length of physical link
- *s*: propagation speed (~2×10^8 m/sec)
- $d_{prop} = d/s$

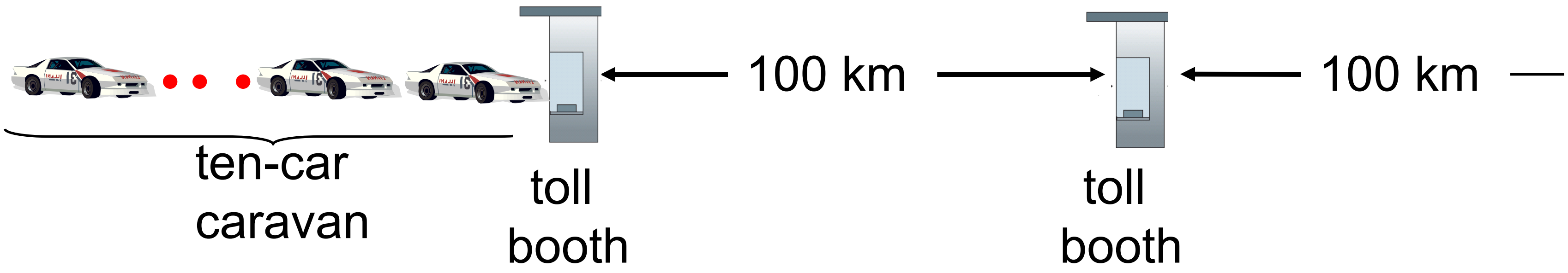
d_{trans} and d_{prop} *very* different

Caravan analogy



- cars "propagate" at 100 km/hr
- toll booth takes 12 sec to service car (bit transmission time)
- car ~ bit; caravan ~ packet
- *Q*: How long until caravan is lined up before 2nd toll booth?
- time to "push" entire caravan through toll booth onto highway = 12*10 = 120 sec
- time for last car to propagate from 1st to 2nd toll both: 100km/(100km/hr)= 1 hr
- *A*: 62 minutes
- Propagation is bottle neck

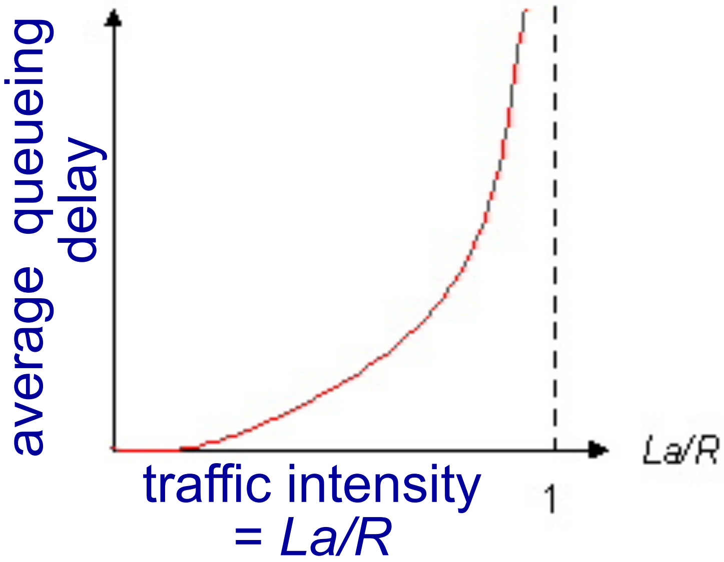
Caravan analogy (more)



- suppose cars now "propagate" at 1000 km/hr
- and suppose toll booth now takes 1 minute to service a car
- *Q:* Will cars arrive to 2nd booth before all cars serviced at first booth?
  - *A: Yes!* after 7 min, first car arrives at second booth; three cars still at first booth
  - Transmission becomes bottleneck.

Queueing delay (revisited)

- *R:* link bandwidth (bps)
- *L:* packet length (bits)
- a: average packet arrival rate



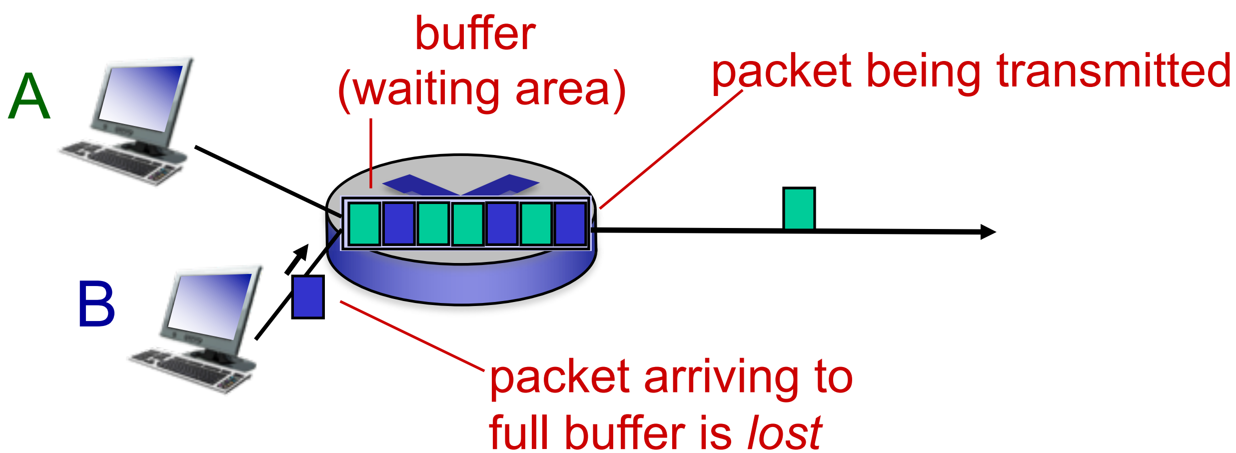
- *La/R ~ 0:*
  - avg. queueing delay small
- La/R → 1:
  - avg. queueing delay large
- *La/R > 1:*
  - more "work" arriving than can be serviced, average delay infinite!

La/R ~ 0

La/R → 1

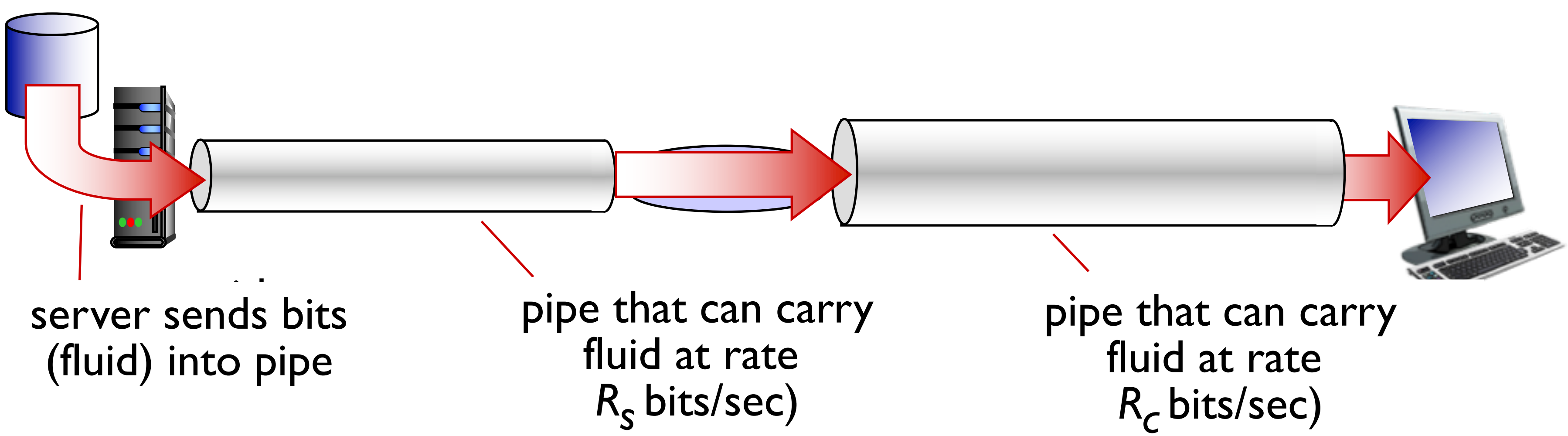
Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



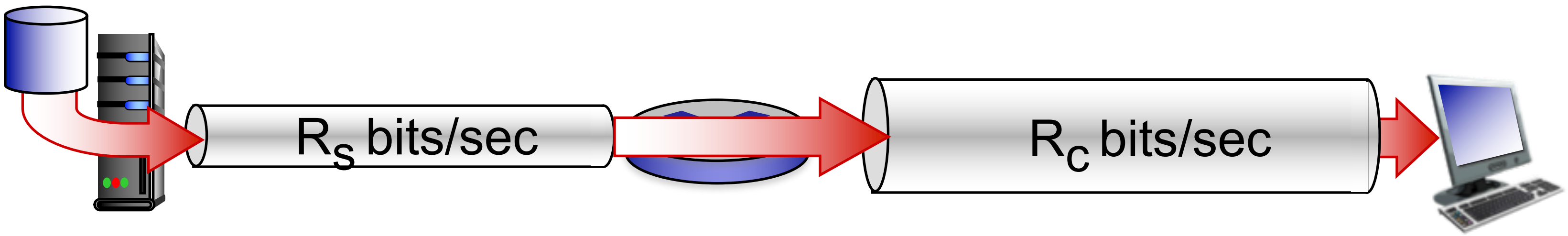
Throughput

- *throughput:* rate (bits/time unit) at which bits transferred between sender/receiver
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time

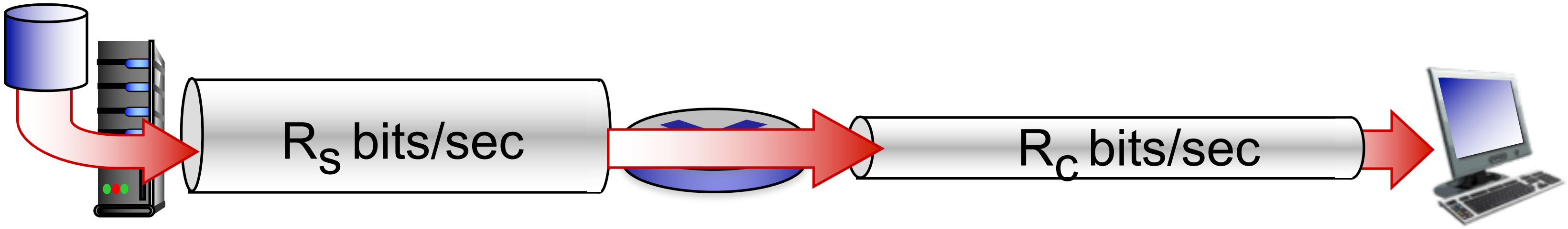


Throughput (more)

- $R_s < R_c$ What is average end-end throughput?



- $R_s > R_c$ What is average end-end throughput?



bottleneck link

link on end-end path that constrains end-end throughput

Protocol Layers

Protocol “layers”

Networks are complex,
with many “pieces”:

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

Question:

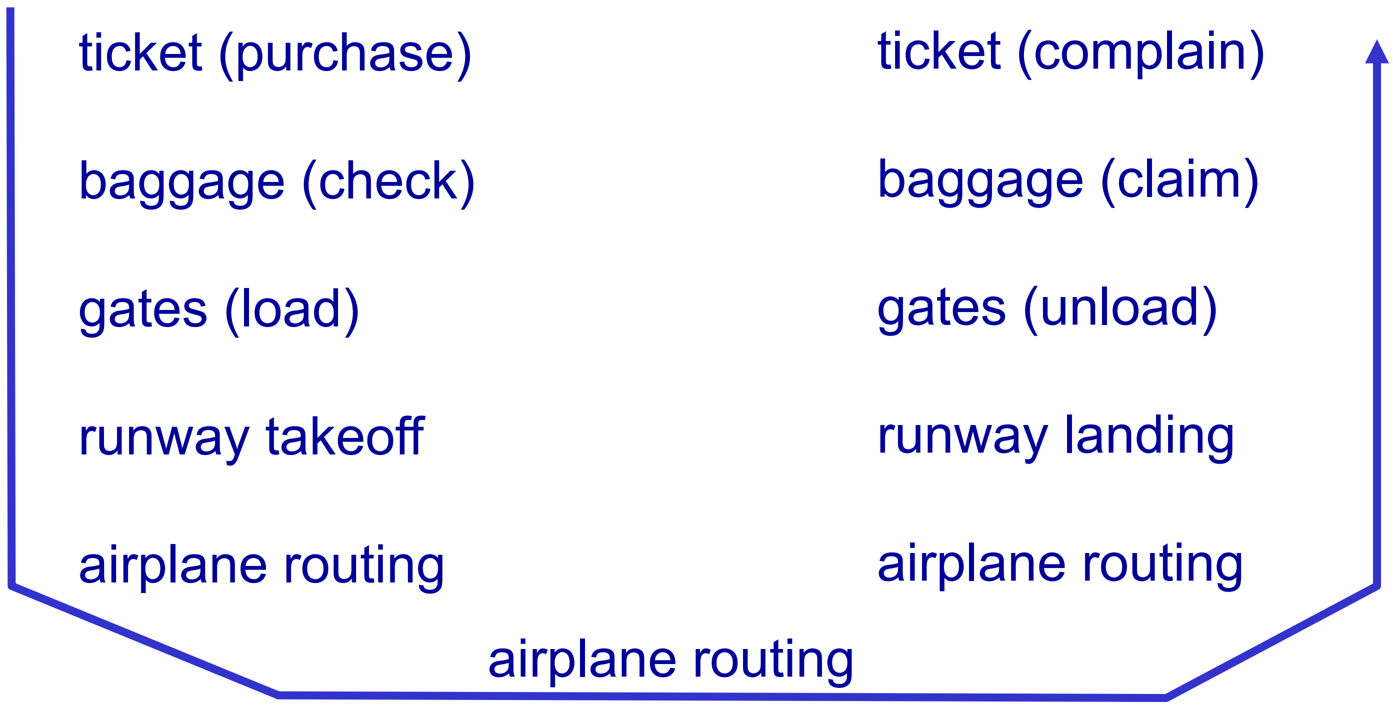
is there any hope of *organizing* structure of network?

…. or at least our discussion of networks?

Try this first:

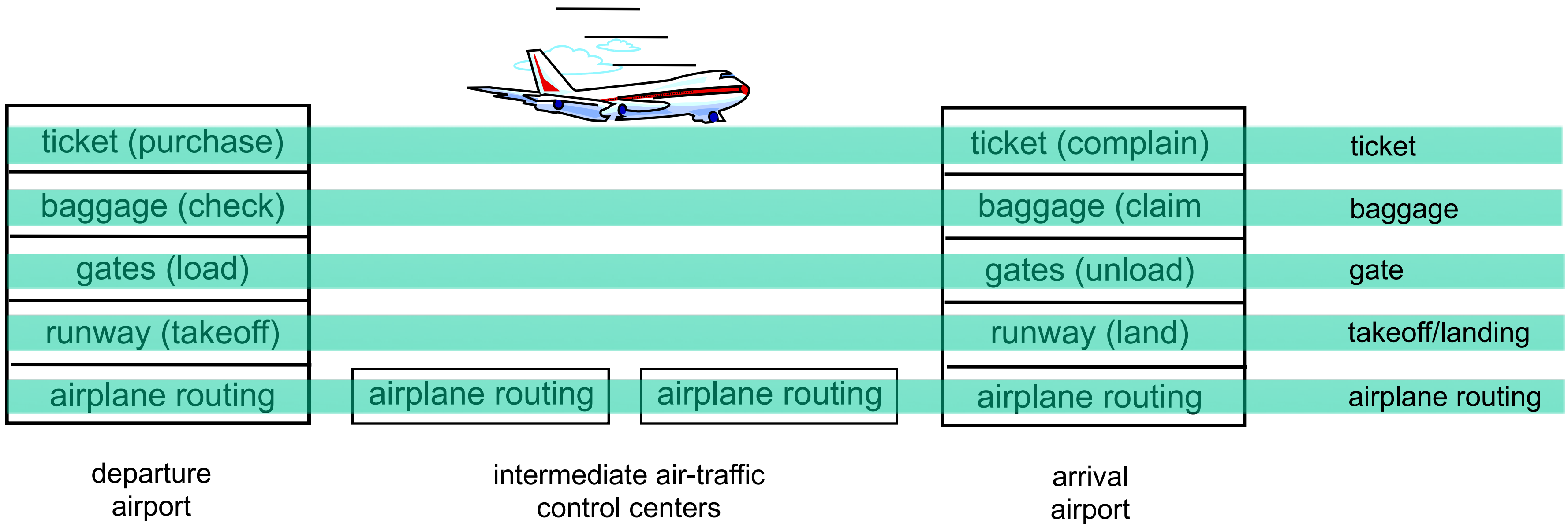
- Describe the organization of the whole air travel system, in an organized way.

Organization of air travel



- a series of steps

Layering of airline functionality



layers: each layer implements a service

- via its own internal-layer actions
- relying on services provided by layer below

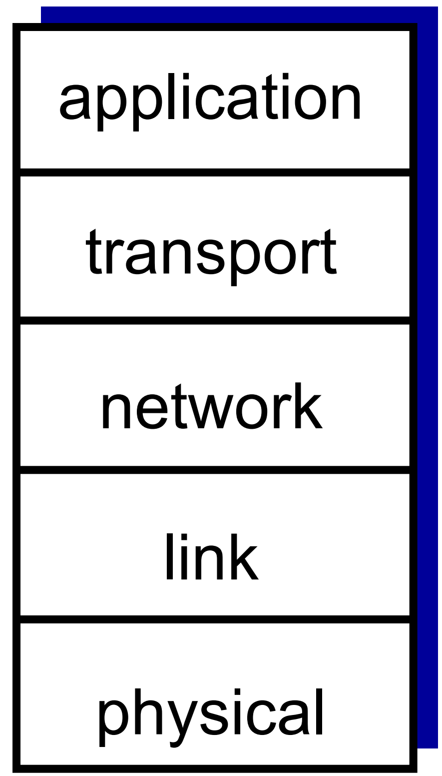
Why layering?

dealing with complex systems:

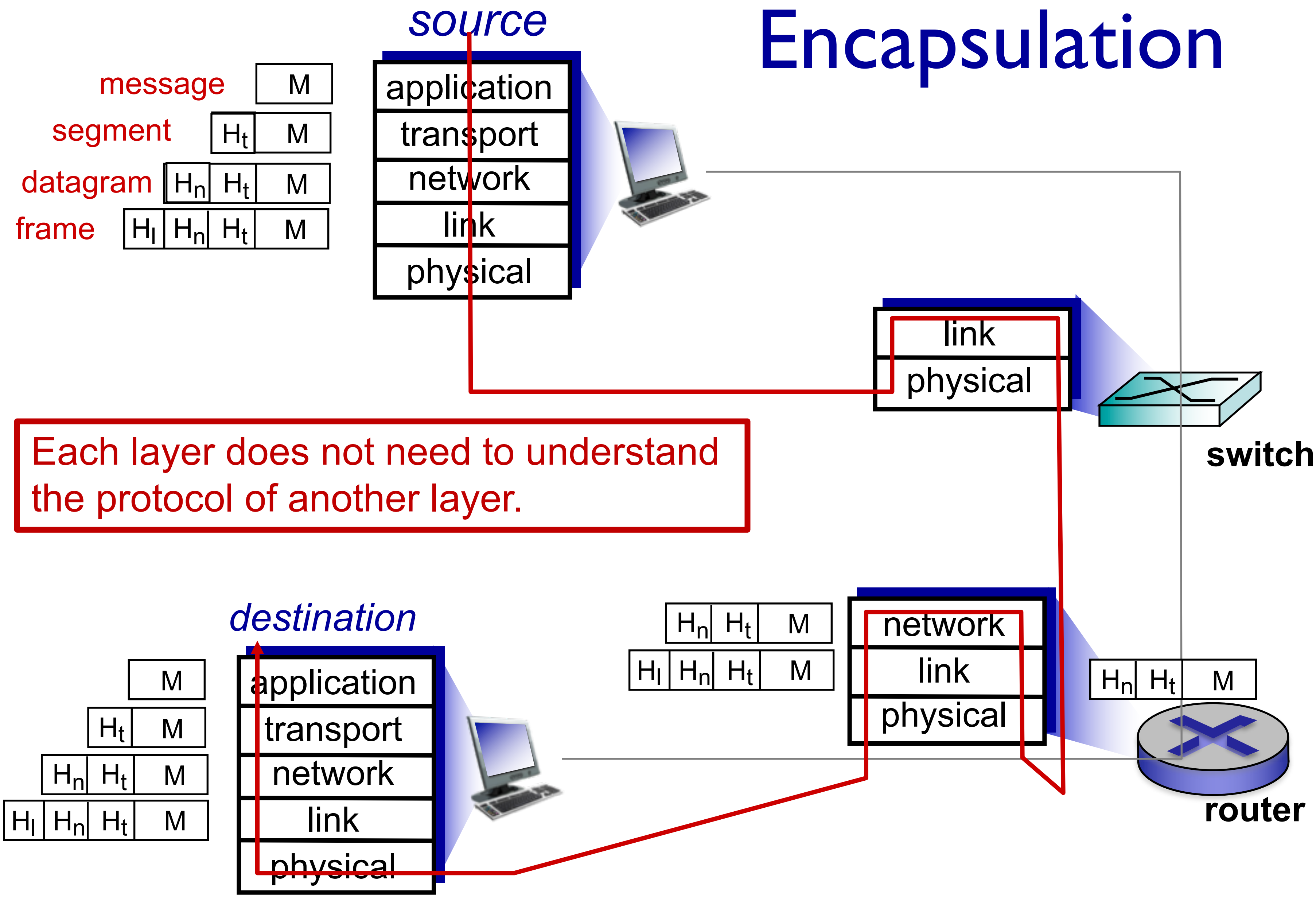
- explicit structure allows identification, relationship of complex system's pieces
  - layered *reference model* for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system

Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



Encapsulation



Each layer does not need to understand the protocol of another layer.

A top-down approach

- In this course, we will learn the Internet protocol stack layer by layer, starting from the top.
- Next week: application layer.