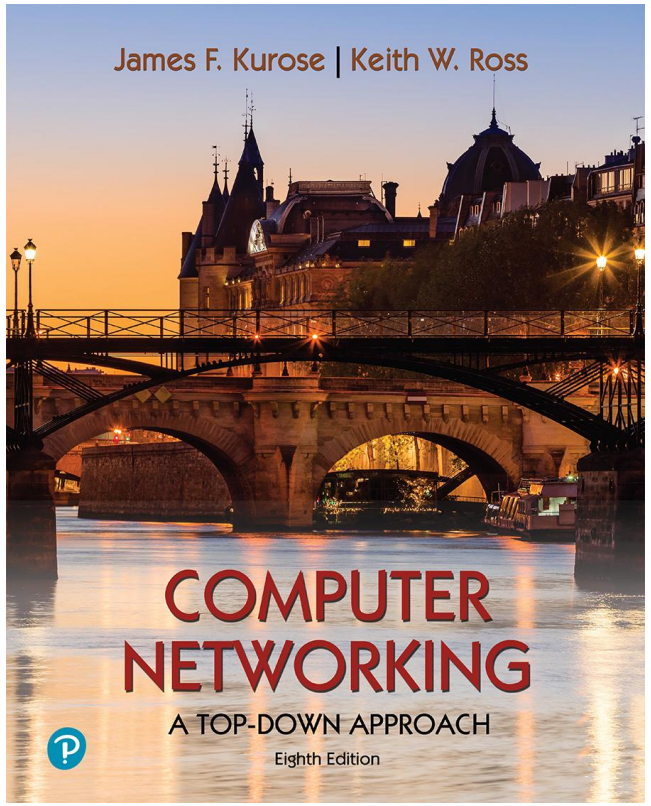
Chapter 1 Introduction



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

A note on the use of these PowerPoint slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

Chapter 1: introduction

Chapter goal:

- Get "feel," "big picture," introduction to terminology
  - more depth, detail *later* in course

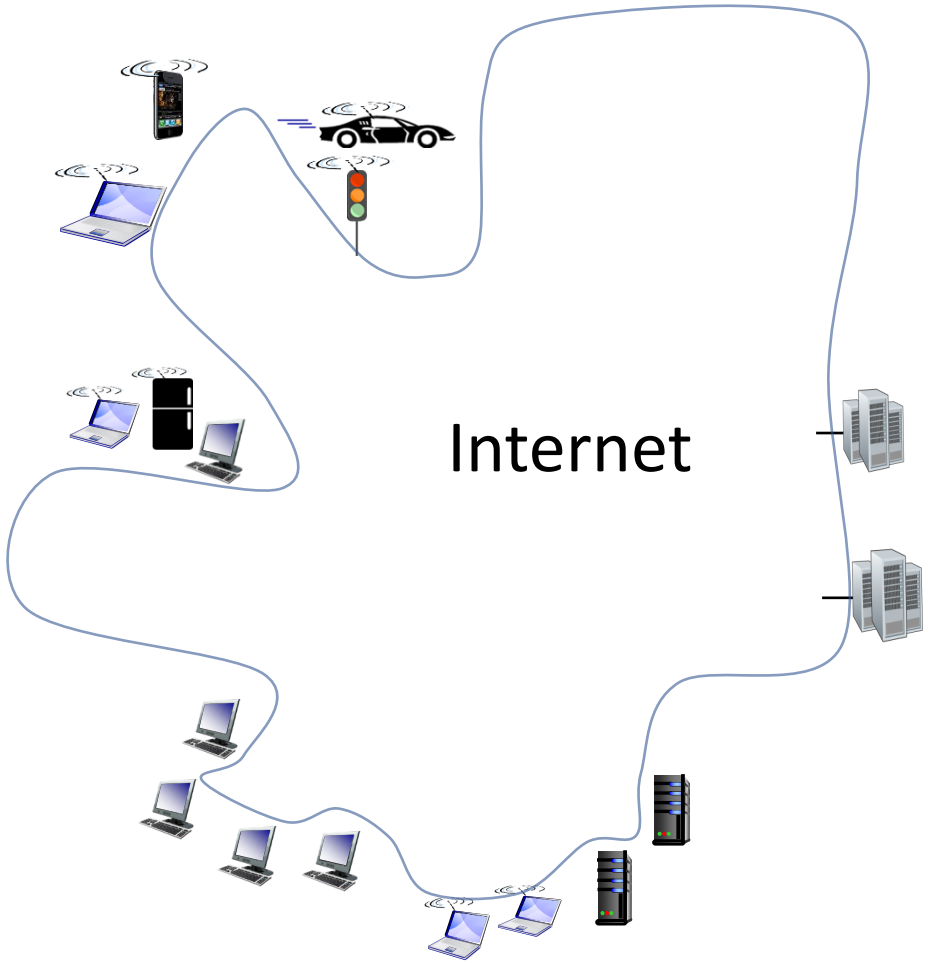
Overview/roadmap:

- What *is* the Internet? What *is* a protocol?
- Packet/circuit switching, internet structure
- Performance: loss, delay, throughput
- Protocol layers, service models

The Internet: a “nuts and bolts” view

Billions of connected computing *devices*:

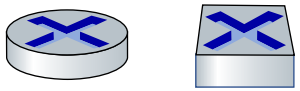
- *hosts = end systems*
- running *network apps* at Internet’s “edge”



The Internet: a "nuts and bolts" view

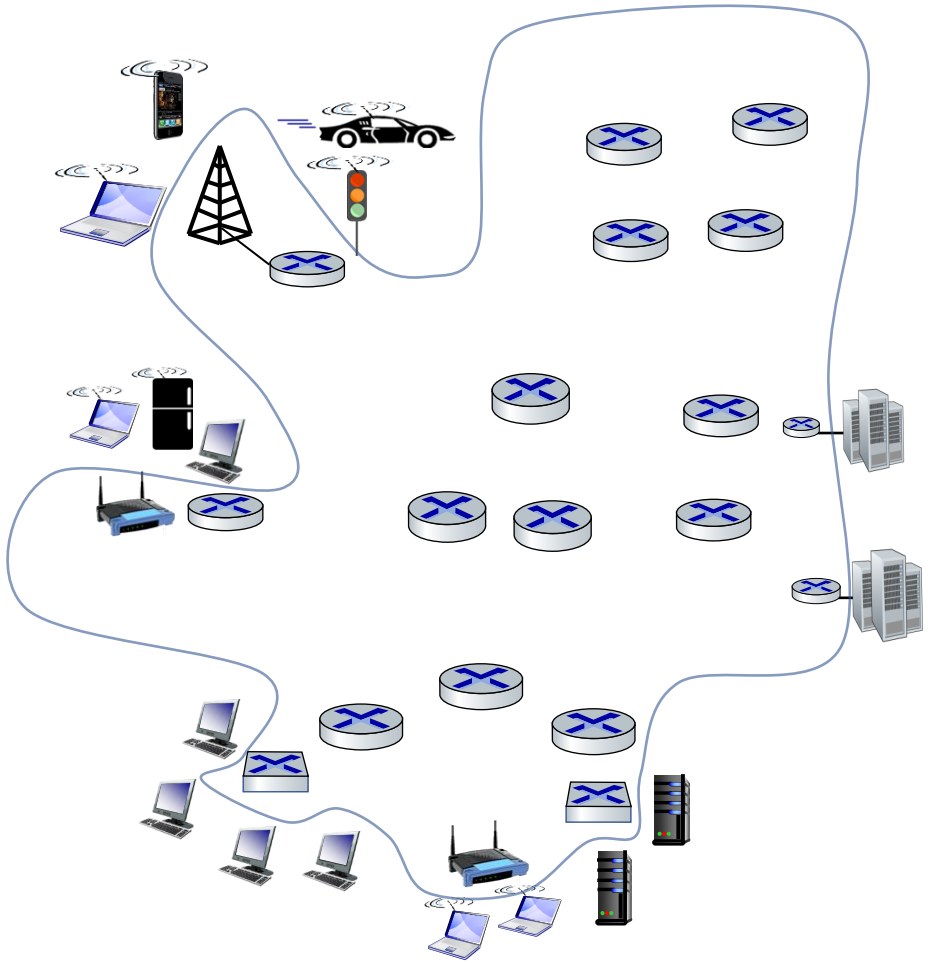
Billions of connected computing *devices*:

- *hosts = end systems*
- running *network apps* at Internet's "edge"



Packet switches: forward packets (chunks of data)

- *routers, switches*



The Internet: a "nuts and bolts" view

Billions of connected computing *devices*:

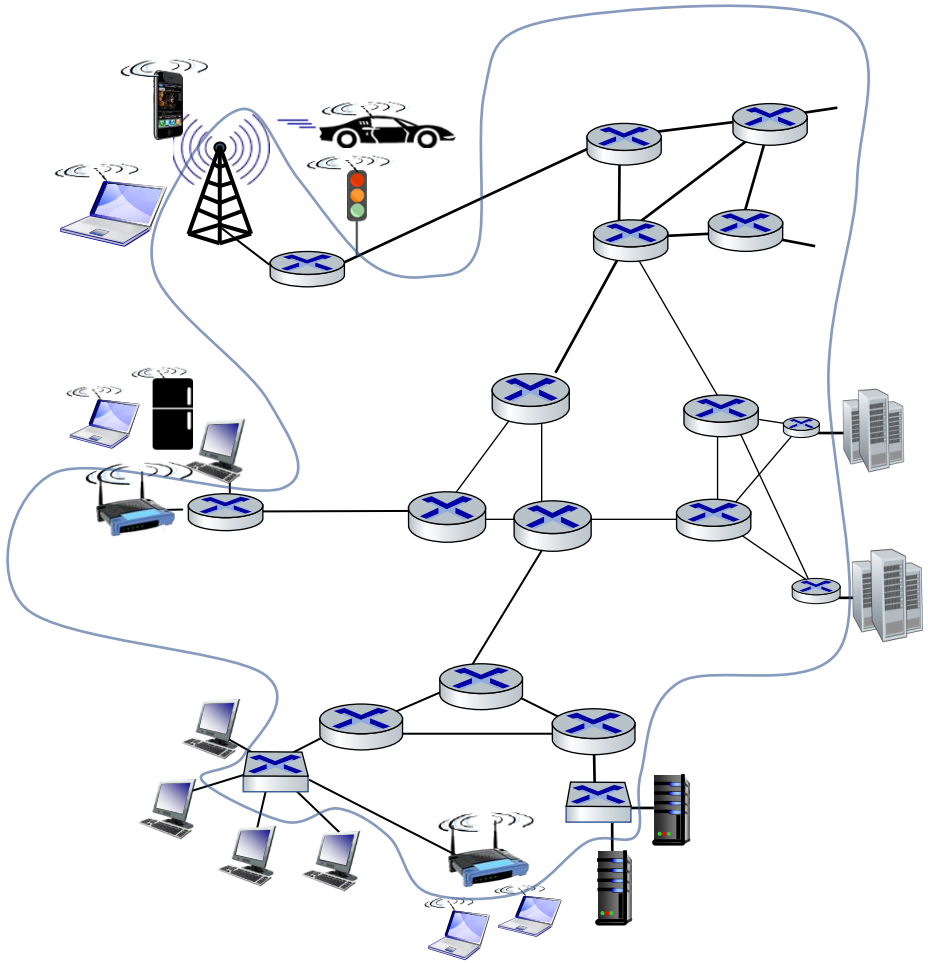
- *hosts = end systems*
- running *network apps* at Internet's "edge"

Packet switches: forward packets (chunks of data)

- *routers, switches*

Communication links

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



The Internet: a "nuts and bolts" view

Billions of connected computing *devices*:

- *hosts = end systems*
- running *network apps* at Internet's "edge"

Packet switches: forward packets (chunks of data)

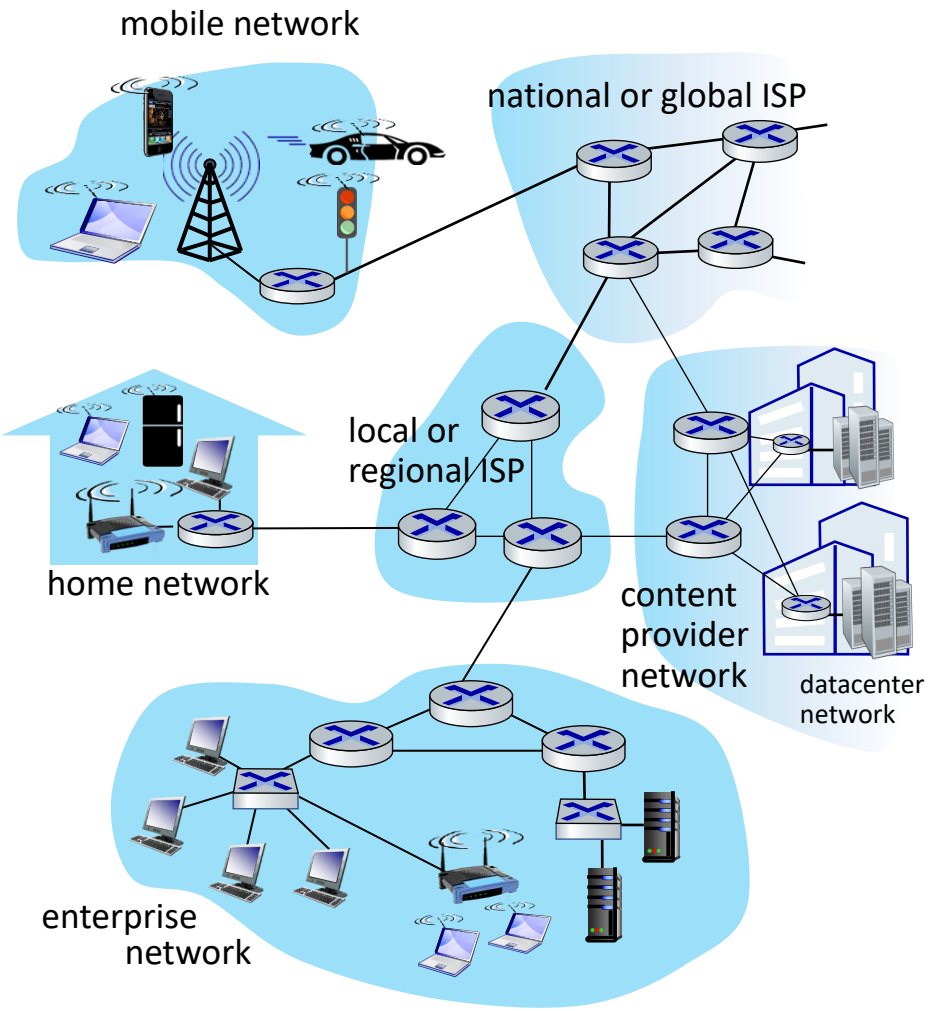
- *routers, switches*

Communication links

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

Networks

- collection of devices, routers, switches, links: managed by an organization



“Fun” Internet-connected devices

Amazon Echo

Internet refrigerator

IP picture frame

Pacemaker & Monitor

Tweet-a-watt: monitor energy use

bikes

cars

scooters

Security Camera

Slingbox: remote control cable TV

Web-enabled toaster + weather forecaster

AR devices

Internet phones

Gaming devices

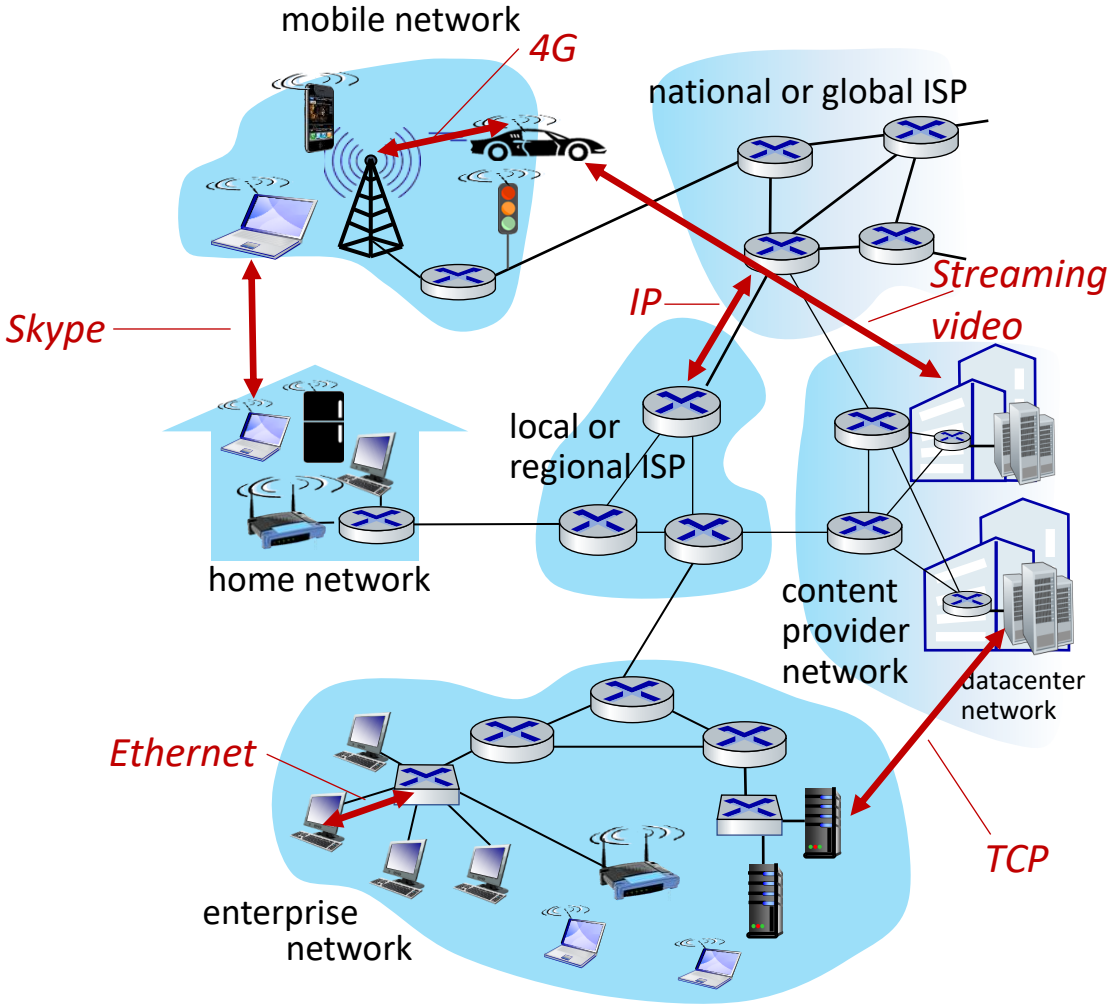
sensorized, bed mattress

Fitbit

Others?

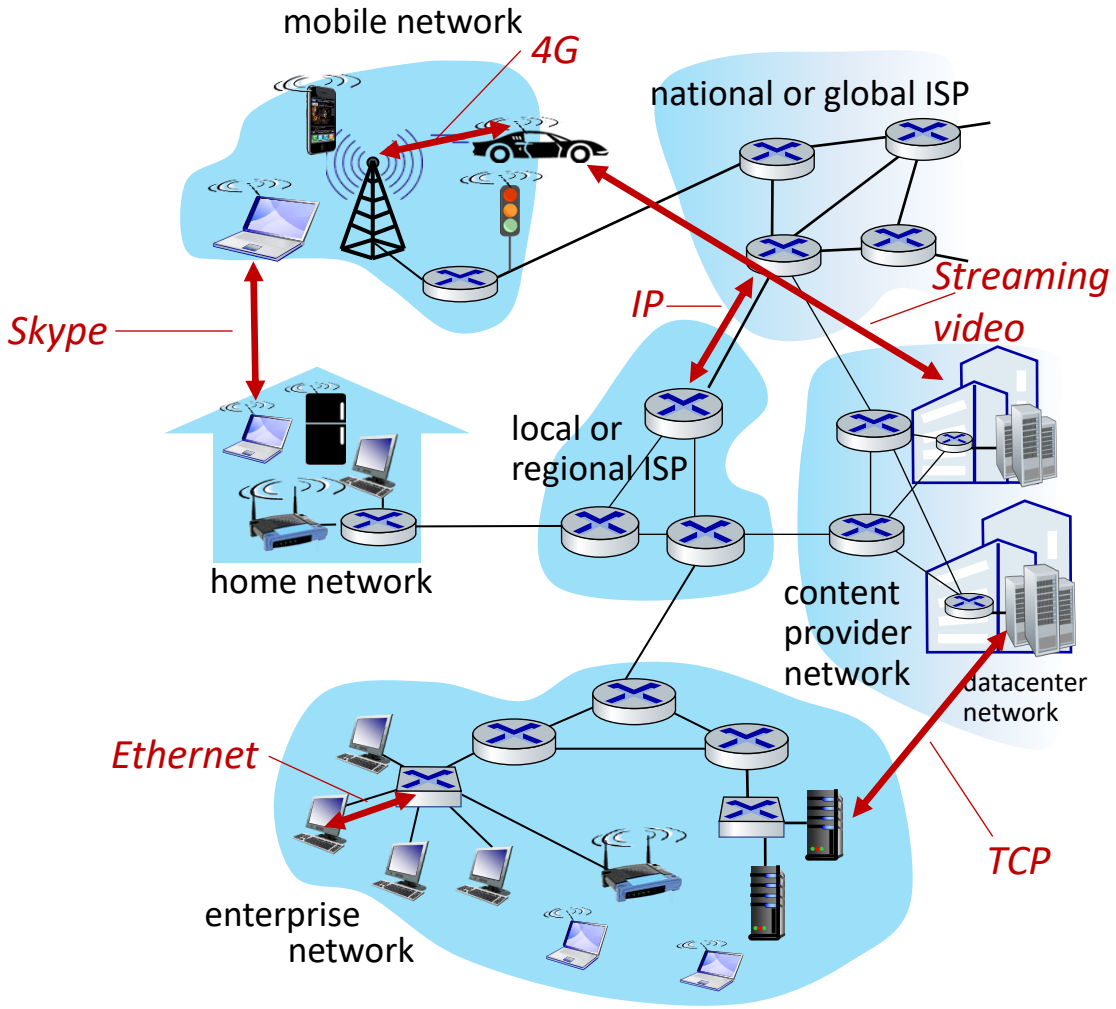
The Internet: a "nuts and bolts" view

- *Internet:* "network of networks"
  - Interconnected ISPs
- *protocols* are *everywhere*
  - control sending, receiving of messages
  - e.g., HTTP (Web), streaming video, Skype, TCP, IP, WiFi, 4G, Ethernet
- *Internet standards*
  - RFC: Request for Comments
  - IETF: Internet Engineering Task Force



The Internet: a "services" view

- *Infrastructure* that provides services to distributed applications:
  - Web, streaming video, multimedia teleconferencing, email, games, e-commerce, social media, inter-connected appliances, …



What's a protocol?

Human protocols:

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

Rules for:

… specific messages sent

… specific actions taken when message received, or other events

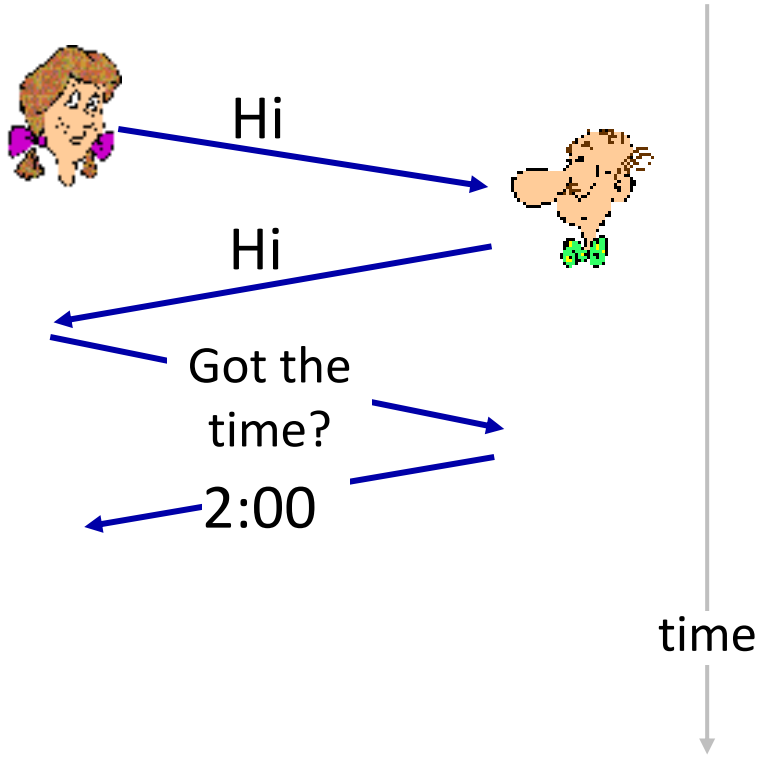
Network protocols:

- computers (devices) rather than humans
- all communication activity in Internet governed by protocols

Protocols define the format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

What's a protocol?

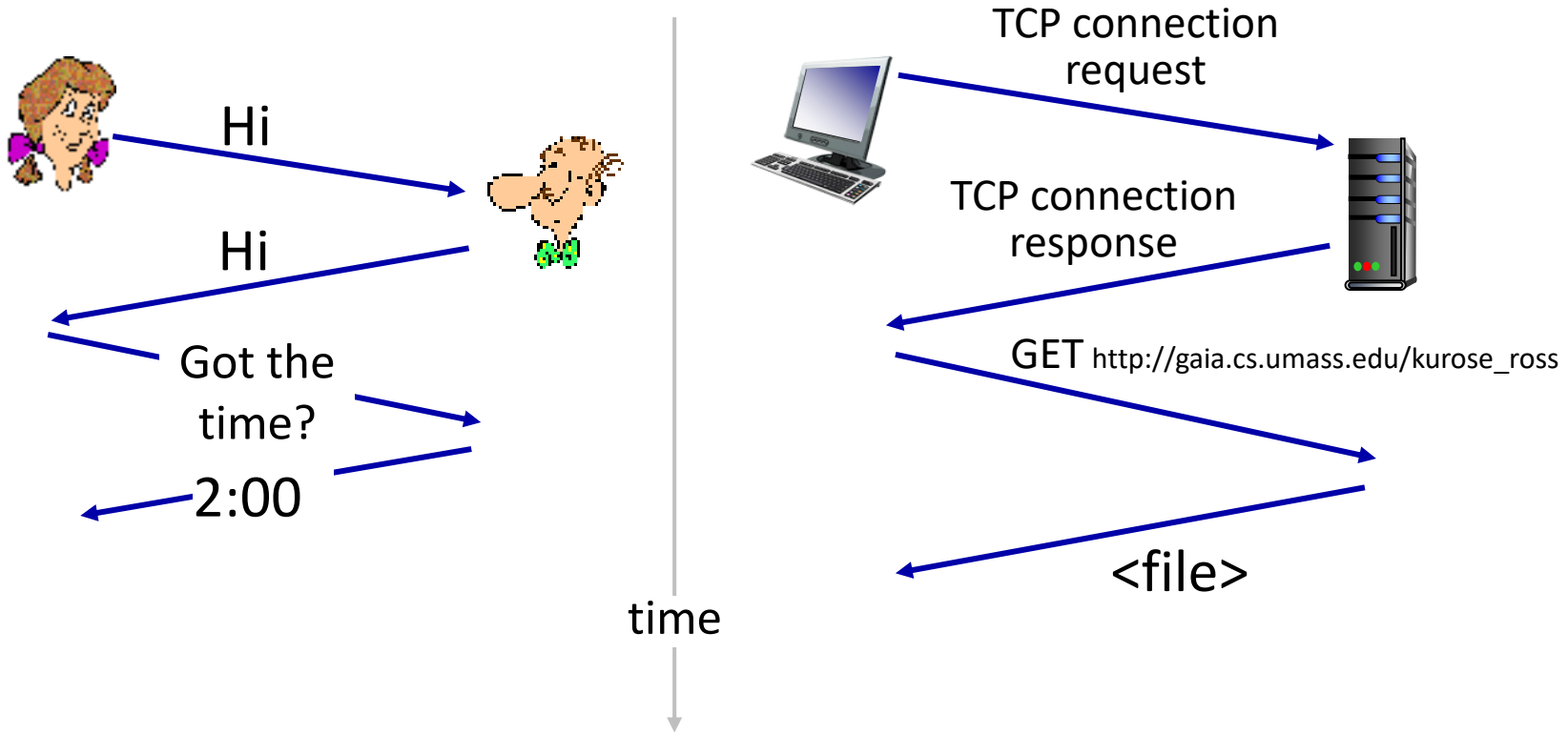
A human protocol and a computer network protocol:



Q: other human protocols?

What's a protocol?

A human protocol and a computer network protocol:

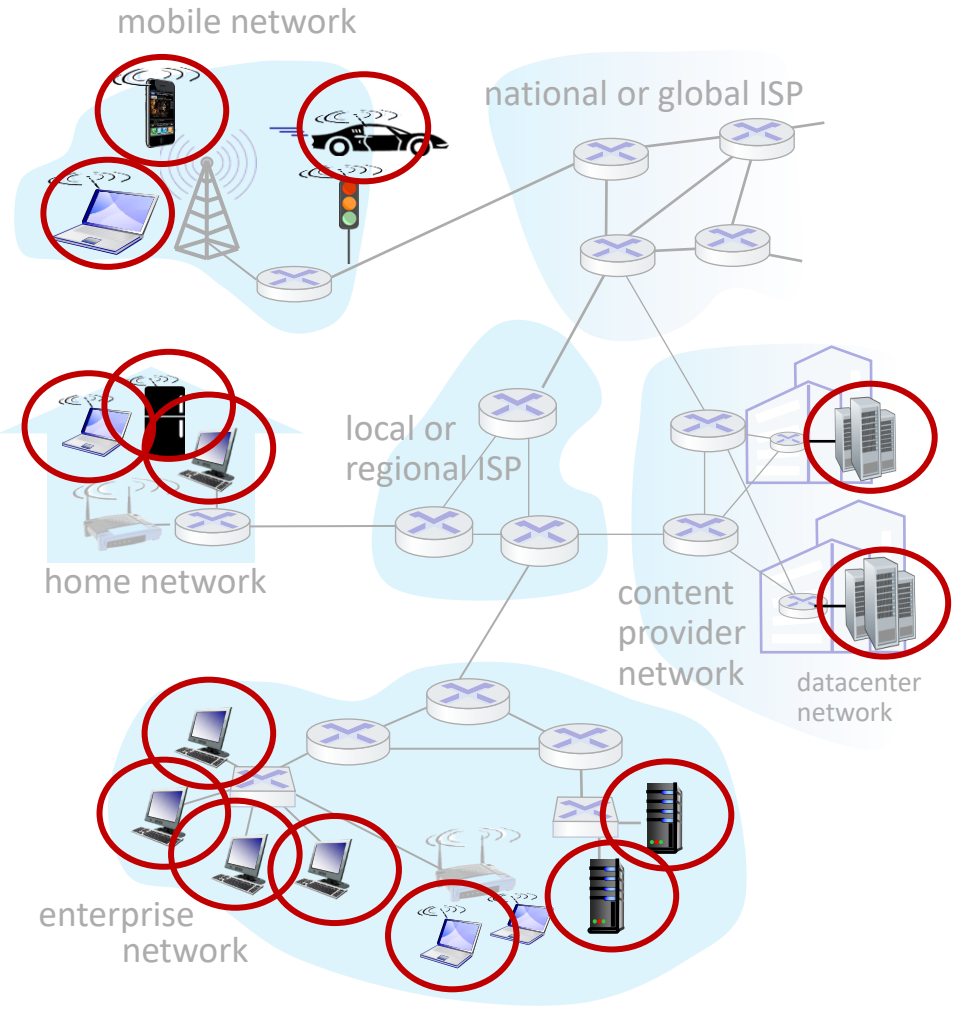


Q: other human protocols?

A closer look at Internet structure

Network edge:

- hosts: clients and servers
- servers often in data centers



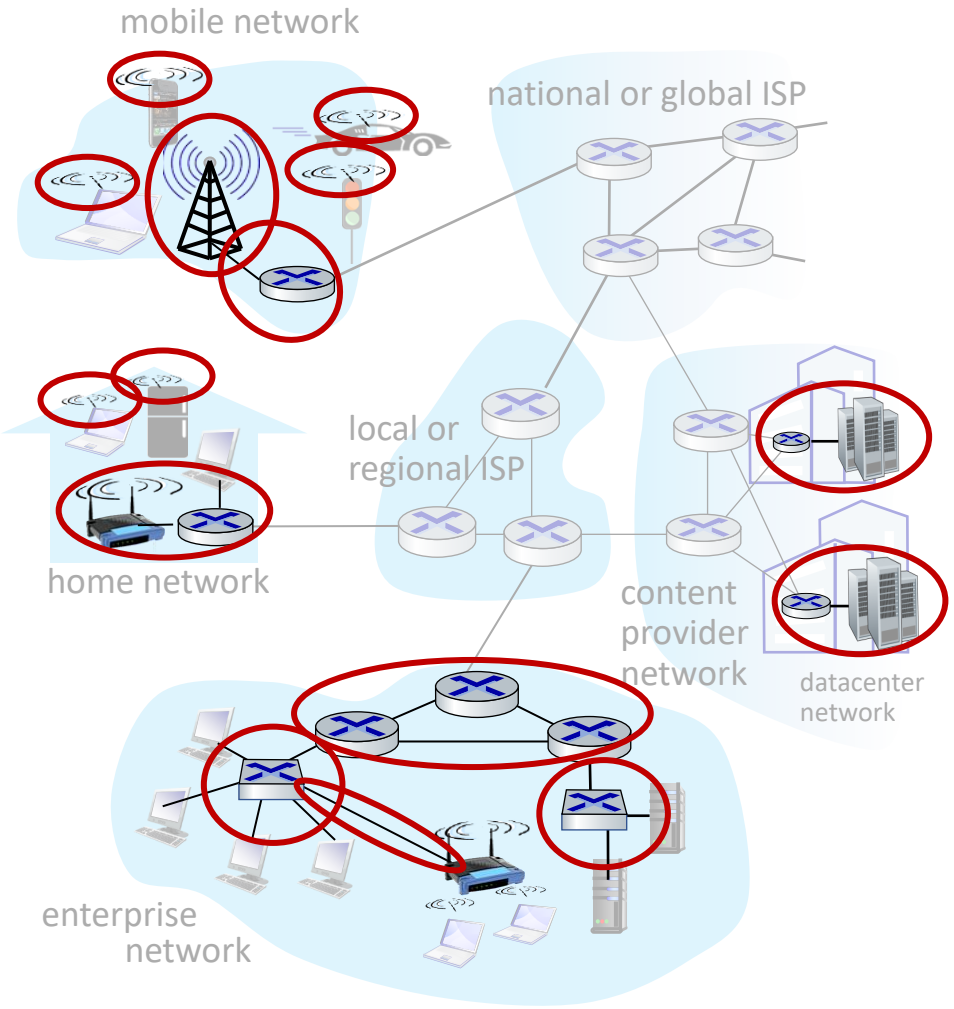
A closer look at Internet structure

Network edge:

- hosts: clients and servers
- servers often in data centers

Access networks:

- wired, wireless communication links



A closer look at Internet structure

Network edge:

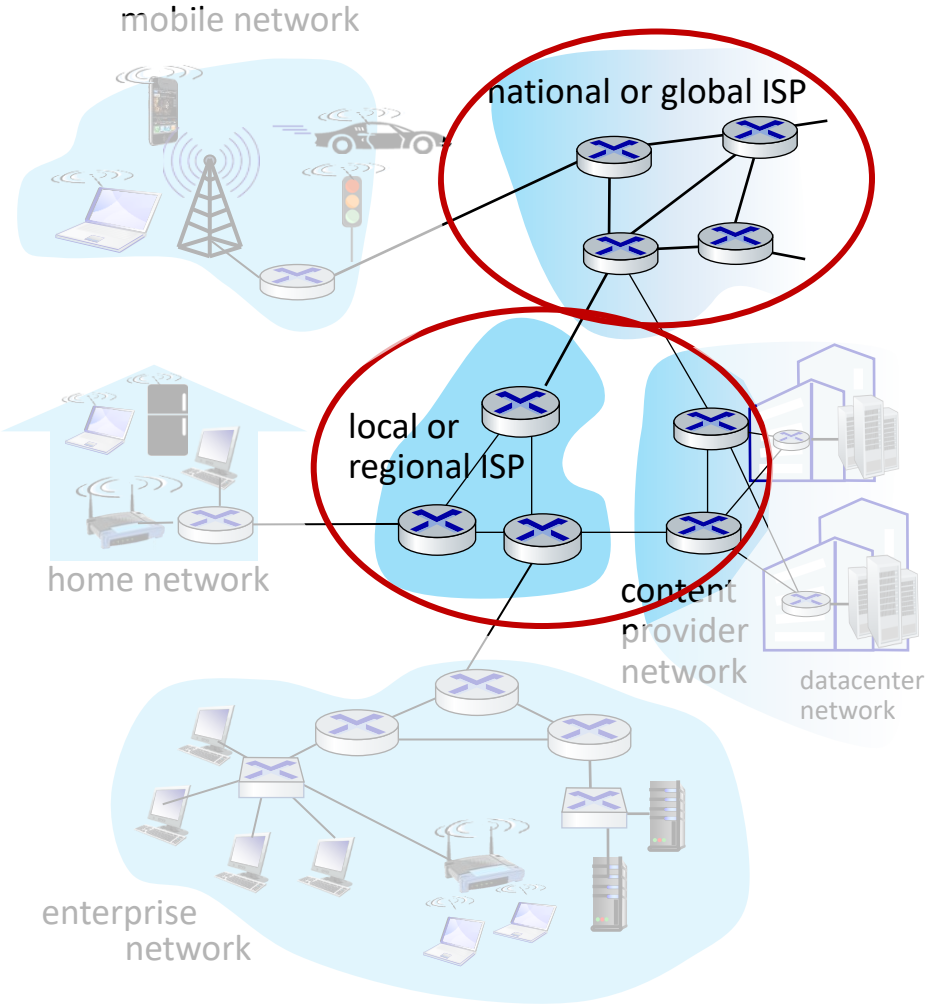
- hosts: clients and servers
- servers often in data centers

Access networks:

- wired, wireless communication links

Network core:

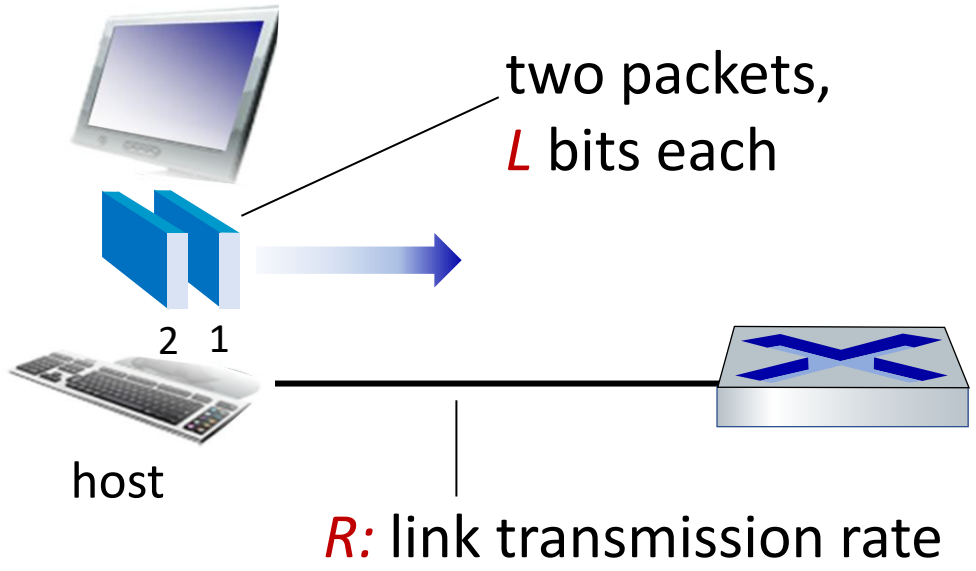
- interconnected routers
- network of networks



Network Edge
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*

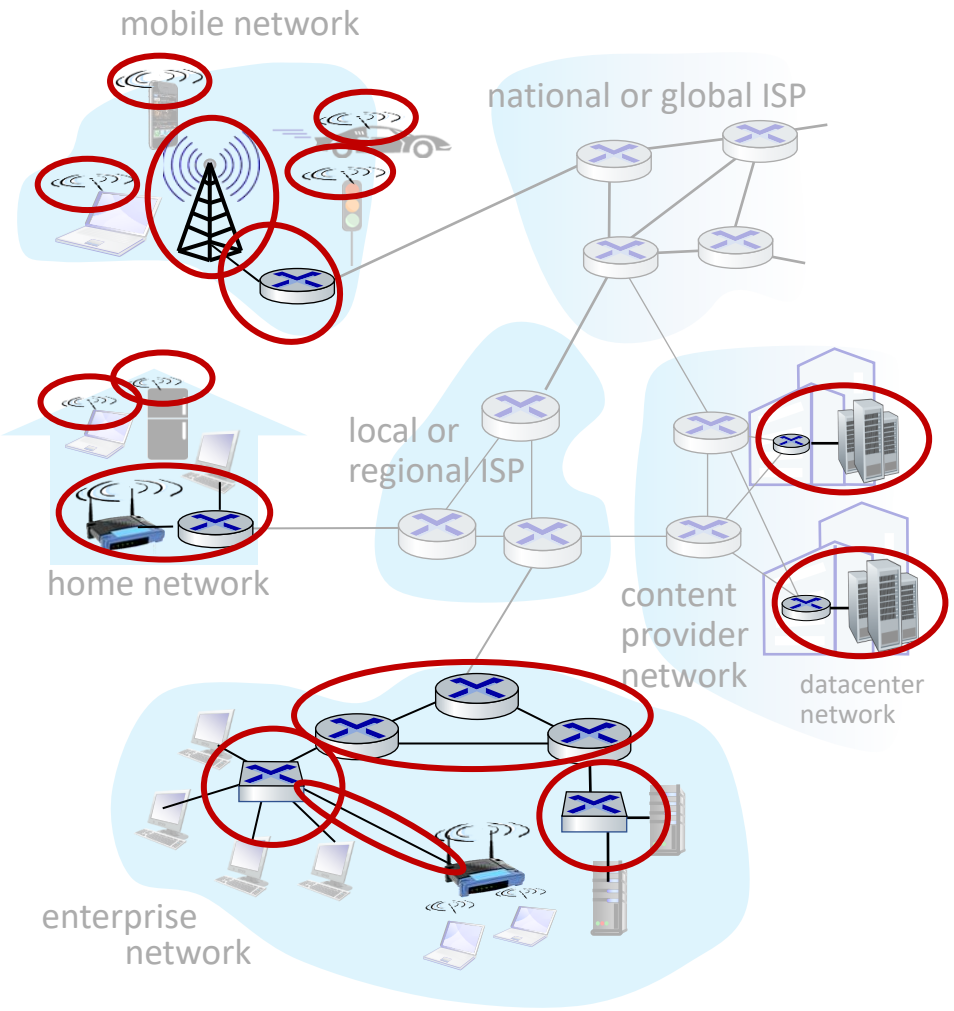


$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

Access networks

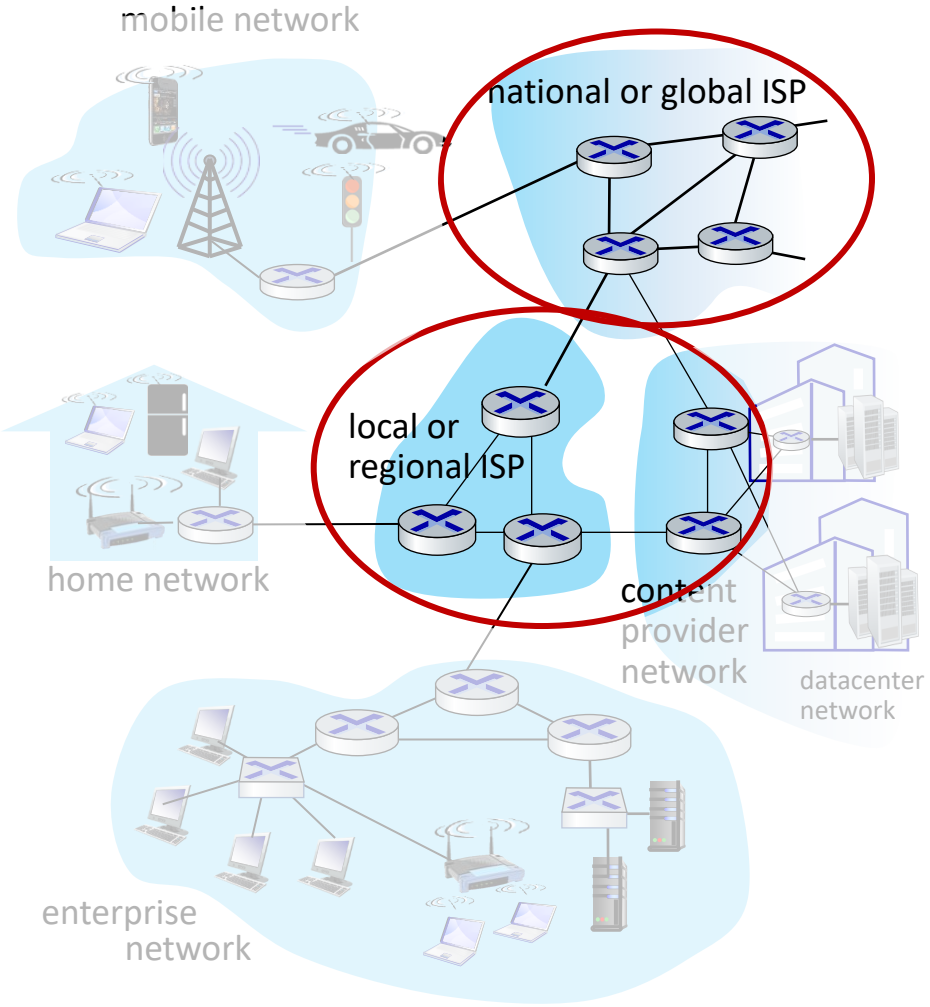
Q: How to connect end systems to "Internet"?

- residential access nets
- institutional access networks (school, company)
- mobile access networks (WiFi, 4G/5G)

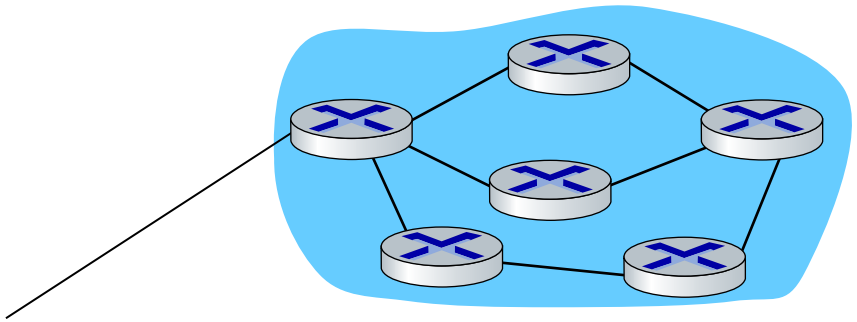


Network core

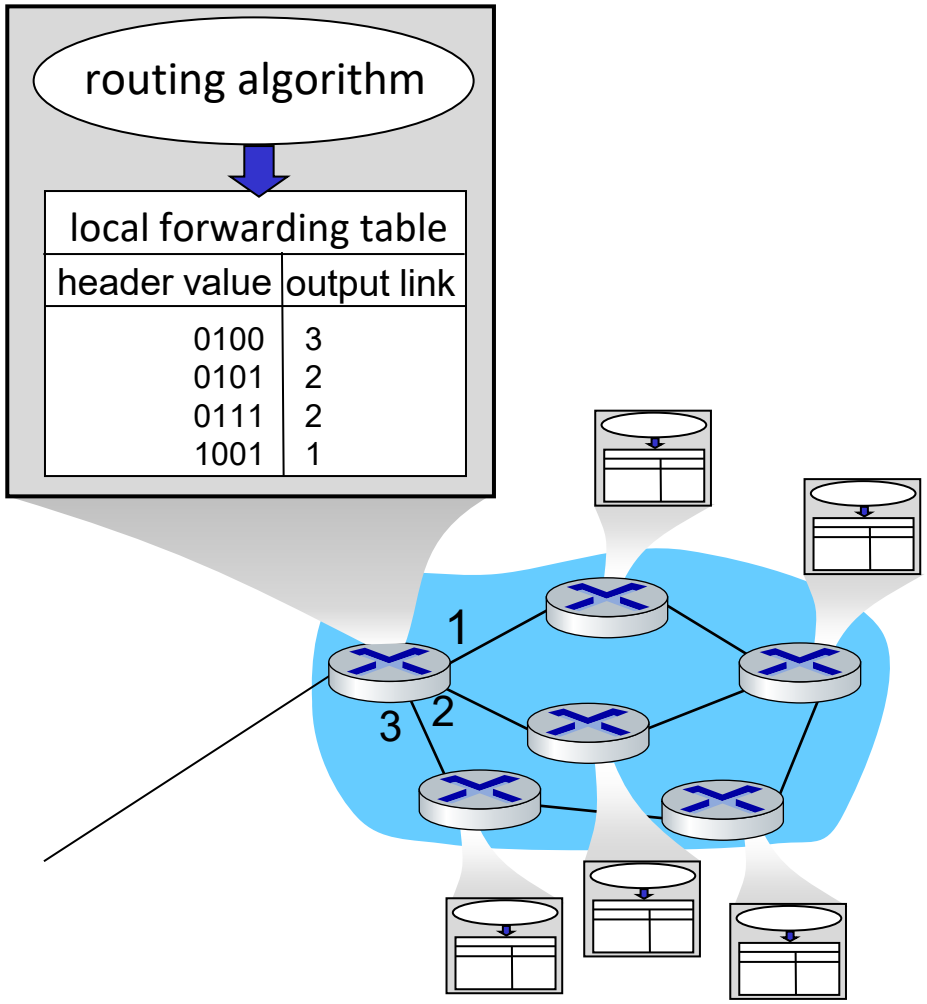
- mesh of interconnected routers
- packet-switching: hosts break application-layer messages into *packets*
  - network forwards packets from one router to the next, across links on path from source to destination



Two key network-core functions



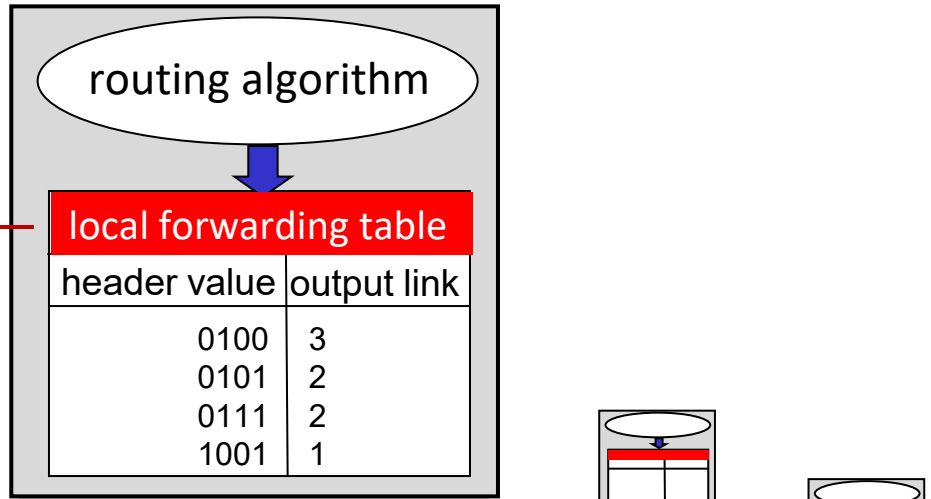
Two key network-core functions



Two key network-core functions

Forwarding:

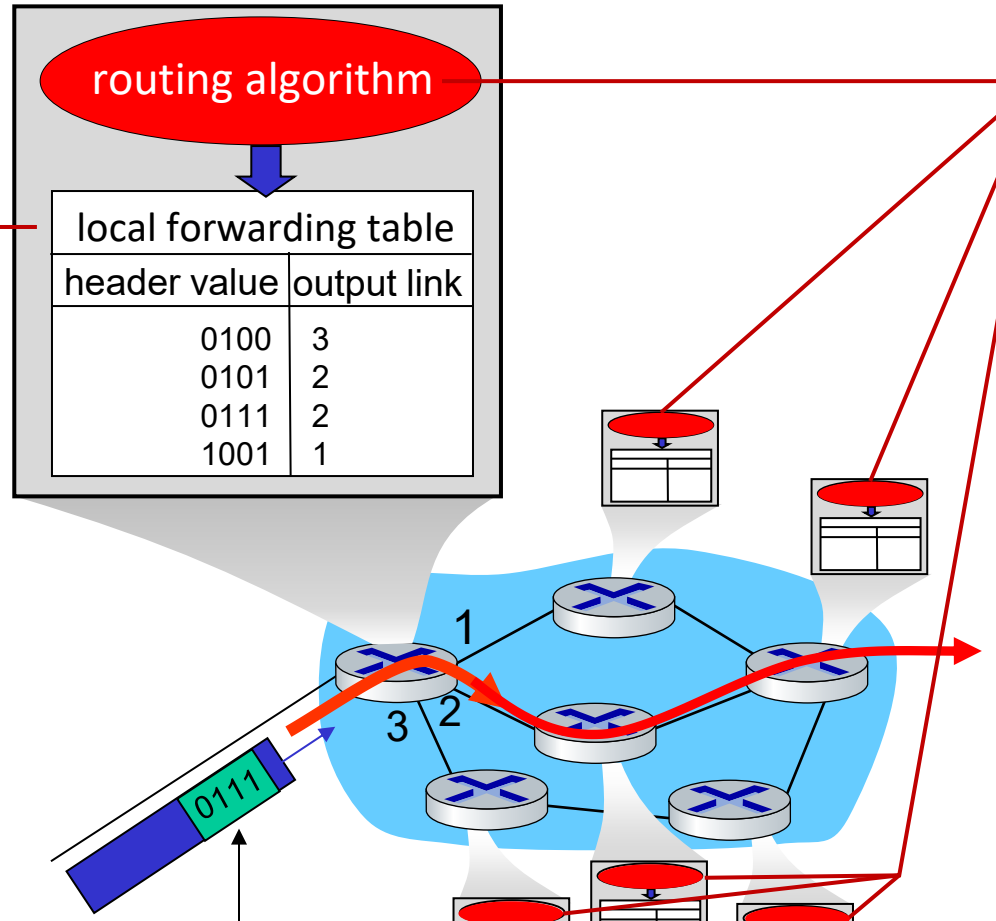
- aka "switching"
- *local* action: move arriving packets from router's input link to appropriate router output link



Two key network-core functions

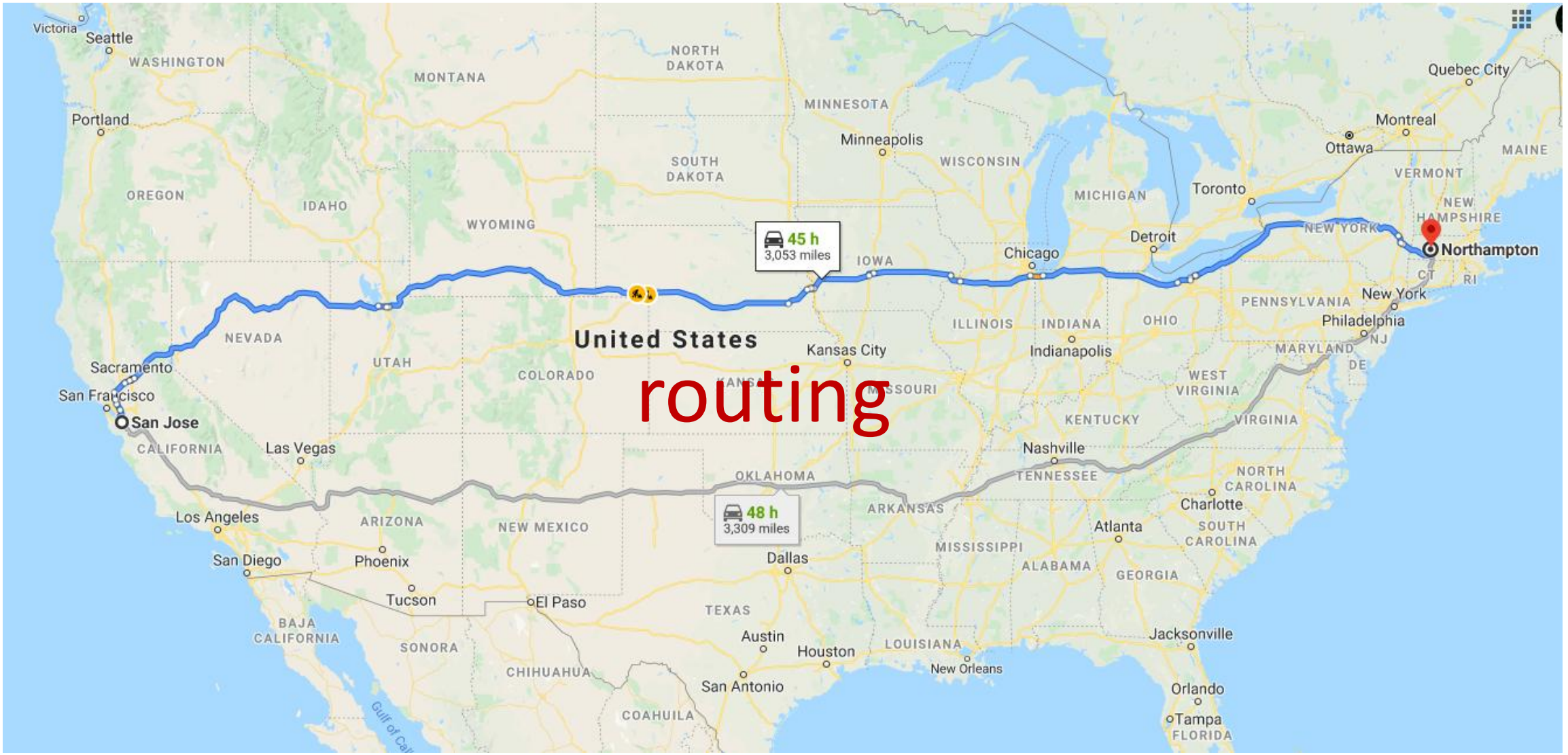
Forwarding:

- aka "switching"
- *local* action: move arriving packets from router's input link to appropriate router output link

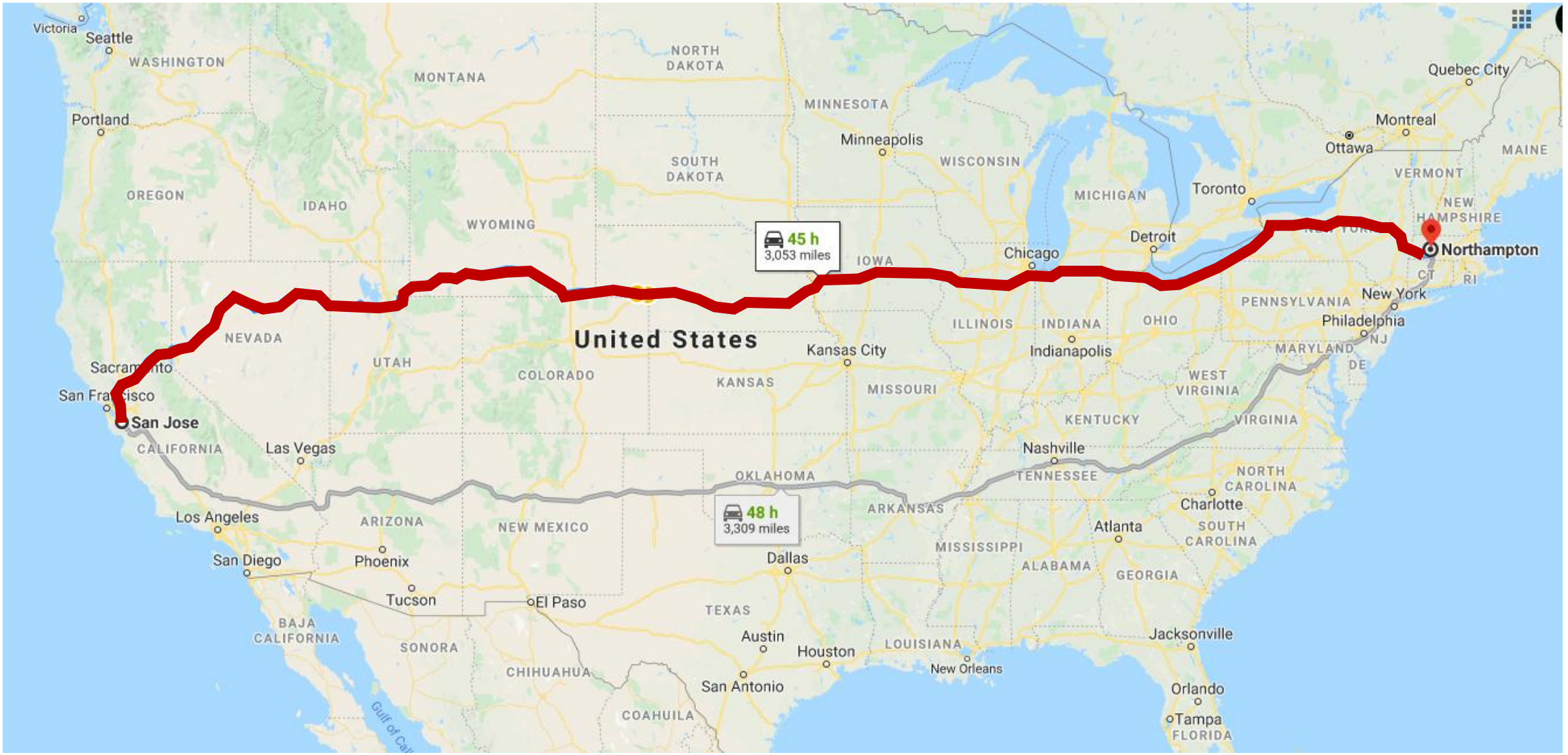


Routing:

- *global* action: determine source-destination paths taken by packets
- routing algorithms


routing
45 h
3,053 miles
48 h
3,309 miles
United States
Victoria
Seattle
WASHINGTON
Portland
OREGON
NORTH DAKOTA
MONTANA
MINNESOTA
SOUTH DAKOTA
IDAHO
WYOMING
Minneapolis
WISCONSIN
MICHIGAN
Toronto
Quebec City
Montreal
Ottawa
MAINE
VERMONT
NEW HAMPSHIRE
NEW YORK
Detroit
Chicago
IOWA
Northampton
CT
RI
PENNSYLVANIA
New York
Philadelphia
NJ
MARYLAND
DE
ILLINOIS
INDIANA
OHIO
Indianapolis
Kansas City
NEVADA
UTAH
COLORADO
Sacramento
San Francisco
San Jose
CALIFORNIA
Las Vegas
MISSOURI
WEST VIRGINIA
KENTUCKY
VIRGINIA
Nashville
TENNESSEE
OKLAHOMA
ARKANSAS
NORTH CAROLINA
Charlotte
Atlanta
SOUTH CAROLINA
Los Angeles
ARIZONA
NEW MEXICO
San Diego
Phoenix
Dallas
MISSISSIPPI
ALABAMA
GEORGIA
Tucson
El Paso
TEXAS
BAJA CALIFORNIA
SONORA
CHIHUAHUA
Austin
Houston
LOUISIANA
New Orleans
Jacksonville
San Antonio
COAHUILA
Orlando
Tampa
FLORIDA


routing
45 h
3,053 miles
48 h
3,309 miles
United States
Victoria
Seattle
WASHINGTON
Portland
OREGON
NORTH DAKOTA
MONTANA
MINNESOTA
SOUTH DAKOTA
IDAHO
WYOMING
Minneapolis
WISCONSIN
MICHIGAN
Toronto
Quebec City
Montreal
Ottawa
MAINE
VERMONT
NEW HAMPSHIRE
Detroit
Chicago
IOWA
Northampton
CT
RI
NEVADA
UTAH
COLORADO
Kansas City
ILLINOIS
INDIANA
OHIO
PENNSYLVANIA
New York
Philadelphia
NJ
MARYLAND
DE
Indianapolis
WEST VIRGINIA
KENTUCKY
VIRGINIA
Sacramento
San Francisco
San Jose
CALIFORNIA
Las Vegas
OKLAHOMA
TENNESSEE
Nashville
NORTH CAROLINA
Charlotte
ARKANSAS
Los Angeles
ARIZONA
NEW MEXICO
Atlanta
SOUTH CAROLINA
MISSISSIPPI
ALABAMA
GEORGIA
San Diego
Phoenix
Dallas
Tucson
El Paso
TEXAS
BAJA CALIFORNIA
SONORA
Austin
Houston
LOUISIANA
New Orleans
Jacksonville
CHIHUAHUA
San Antonio
Orlando
Tampa
FLORIDA
COAHUILA

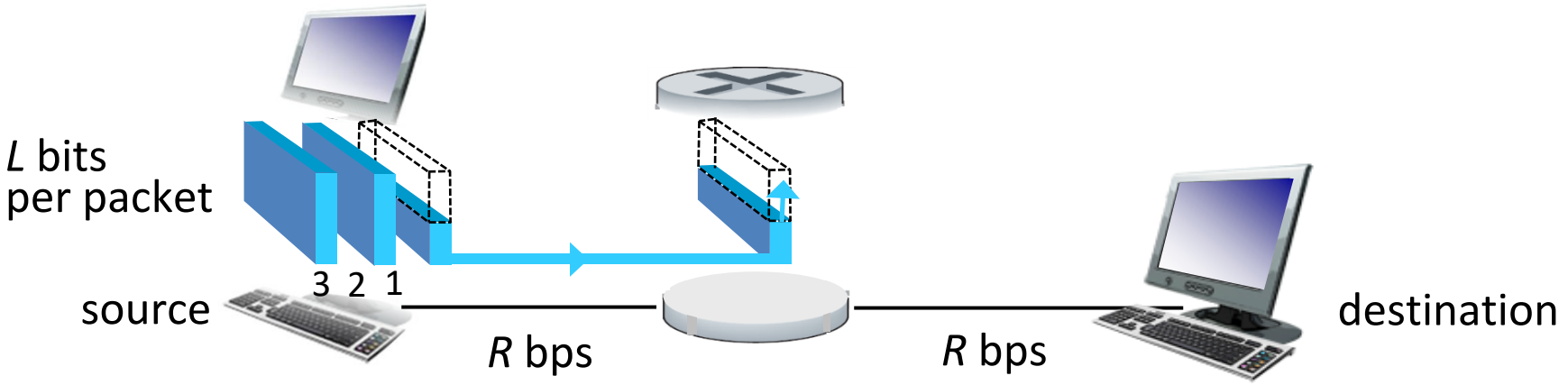

45 h
3,053 miles
48 h
3,309 miles
United States
San Jose
Northampton
Victoria
Seattle
Portland
San Francisco
Sacramento
Las Vegas
Los Angeles
San Diego
Phoenix
Tucson
El Paso
Dallas
Austin
Houston
San Antonio
New Orleans
Kansas City
Minneapolis
Chicago
Detroit
Toronto
Ottawa
Montreal
Quebec City
Indianapolis
Nashville
Atlanta
Charlotte
Jacksonville
Orlando
Tampa
New York
Philadelphia
WASHINGTON
OREGON
CALIFORNIA
NEVADA
IDAHO
MONTANA
WYOMING
UTAH
COLORADO
ARIZONA
NEW MEXICO
NORTH DAKOTA
SOUTH DAKOTA
NEBRASKA
KANSAS
OKLAHOMA
TEXAS
MINNESOTA
IOWA
MISSOURI
ARKANSAS
LOUISIANA
WISCONSIN
ILLINOIS
MICHIGAN
INDIANA
OHIO
KENTUCKY
TENNESSEE
MISSISSIPPI
ALABAMA
GEORGIA
FLORIDA
SOUTH CAROLINA
NORTH CAROLINA
VIRGINIA
WEST VIRGINIA
PENNSYLVANIA
NEW YORK
MARYLAND
DE
NJ
CT
RI
VERMONT
NEW HAMPSHIRE
MAINE
BAJA CALIFORNIA
SONORA
CHIHUAHUA
COAHUILA


Victoria
Seattle
WASHINGTON
Portland
NORTH DAKOTA
MINNESOTA
Minneapolis
SOUTH DAKOTA
WISCONSIN
MICHIGAN
Toronto
Quebec City
Montreal
Ottawa
MAINE
VERMONT
NEW HAMPSHIRE
WYOMING
45 h
3,053 miles
IOWA
Chicago
Detroit
Northampton
CT
RI
PENNSYLVANIA
New York
Philadelphia
NJ
MARYLAND
DE
ILLINOIS
INDIANA
OHIO
Indianapolis
United States
Kansas City
UTAH
COLORADO
KANSAS
MISSOURI
WEST VIRGINIA
KENTUCKY
VIRGINIA
San Francisco
San Jose
CALIFORNIA
Las Vegas
Nashville
TENNESSEE
OKLAHOMA
NORTH CAROLINA
Charlotte
ARKANSAS
Los Angeles
ARIZONA
NEW MEXICO
48 h
3,309 miles
Atlanta
SOUTH CAROLINA
MISSISSIPPI
ALABAMA
GEORGIA
San Diego
Phoenix
Dallas
Tucson
El Paso
TEXAS
BAJA CALIFORNIA
SONORA
Austin
Houston
LOUISIANA
Jacksonville
New Orleans
CHIHUAHUA
San Antonio
Orlando
COAHUILA
Tampa
FLORIDA
Sacramento


Victoria
Seattle
WASHINGTON
Portland
NORTH DAKOTA
MINNESOTA
Minneapolis
SOUTH DAKOTA
WISCONSIN
MICHIGAN
Toronto
Quebec City
Montreal
Ottawa
MAINE
VERMONT
NEW HAMPSHIRE
WYOMING
45 h
3,053 miles
IOWA
Chicago
Detroit
Northampton
New York
United States
ILLINOIS
INDIANA
OHIO
Kansas City
Indianapolis
UTAH
COLORADO
KANSAS
MISSOURI
San Francisco
San Jose
KENTUCKY
CALIFORNIA
Las Vegas
Nashville
TENNESSEE
OKLAHOMA
48 h
3,309 miles
ARKANSAS
Los Angeles
ARIZONA
NEW MEXICO
Atlanta
MISSISSIPPI
ALABAMA
San Diego
Phoenix
Dallas
Tucson
El Paso
TEXAS
BAJA CALIFORNIA
SONORA
Austin
Houston
LOUISIANA
New Orleans
CHIHUAHUA
San Antonio
COAHUILA
Tampa
FLORIDA
Cleveland
DOWNTOWN
CENTRAL
OHIO CITY


forwarding
forwarding
45 h
3,053 miles
48 h
3,309 miles
United States
Victoria
Seattle
WASHINGTON
Portland
NORTH DAKOTA
MINNESOTA
Minneapolis
SOUTH DAKOTA
WISCONSIN
MICHIGAN
Toronto
Quebec City
Montreal
Ottawa
MAINE
VERMONT
NEW HAMPSHIRE
Northampton
WYOMING
IOWA
Chicago
Detroit
ILLINOIS
INDIANA
OHIO
Indianapolis
Kansas City
COLORADO
KANSAS
MISSOURI
KENTUCKY
Nashville
TENNESSEE
San Francisco
San Jose
CALIFORNIA
Las Vegas
OKLAHOMA
ARKANSAS
Atlanta
Los Angeles
ARIZONA
NEW MEXICO
MISSISSIPPI
ALABAMA
San Diego
Phoenix
Dallas
Tucson
El Paso
TEXAS
BAJA CALIFORNIA
SONORA
Austin
Houston
LOUISIANA
New Orleans
CHIHUAHUA
San Antonio
COAHUILA
Tampa
FLORIDA
Cleveland
DOWNTOWN
OHIO CITY
CENTRAL

Packet-switching: store-and-forward

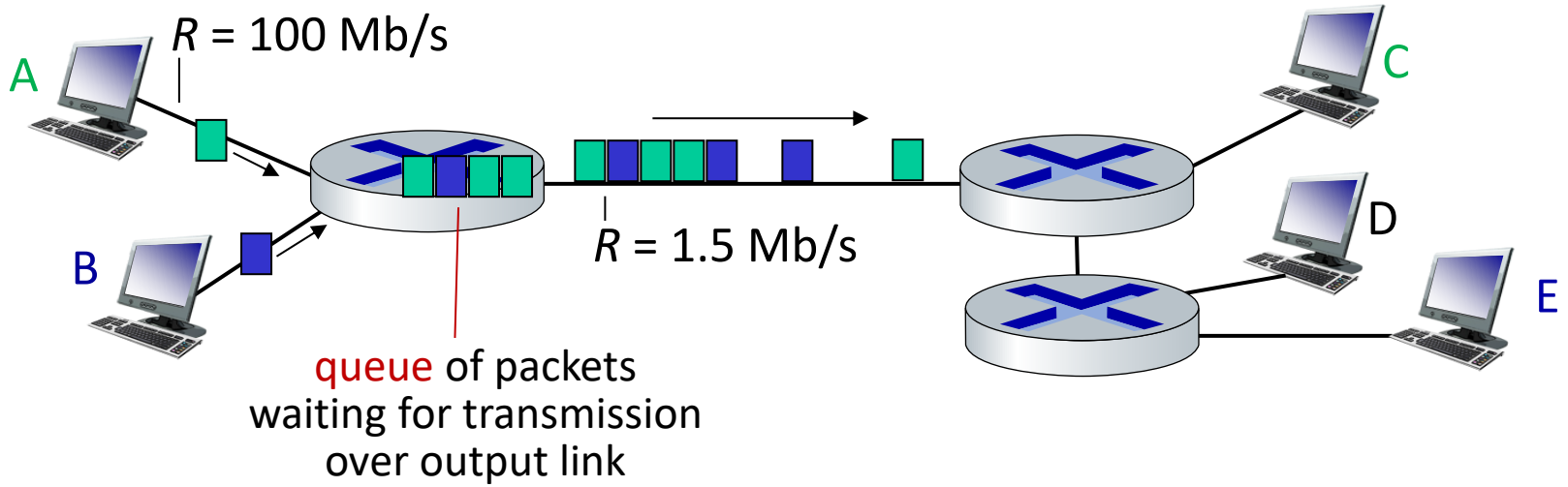


- packet transmission delay: takes L/R seconds to transmit (push out) L-bit packet into link at R bps
- *store and forward: entire* packet must arrive at router before it can be transmitted on next link

One-hop numerical example:

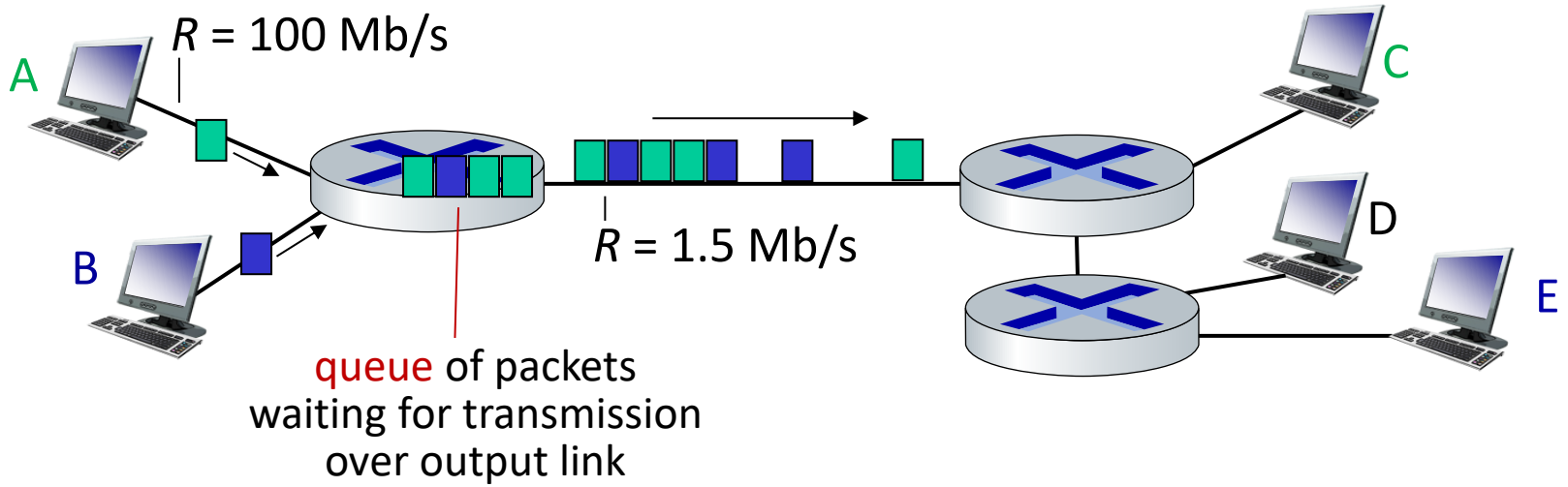
- L = 10 Kbits
- R = 100 Mbps
- one-hop transmission delay = 0.1 msec

Packet-switching: queueing



Queueing occurs when work arrives faster than it can be serviced:

Packet-switching: queueing



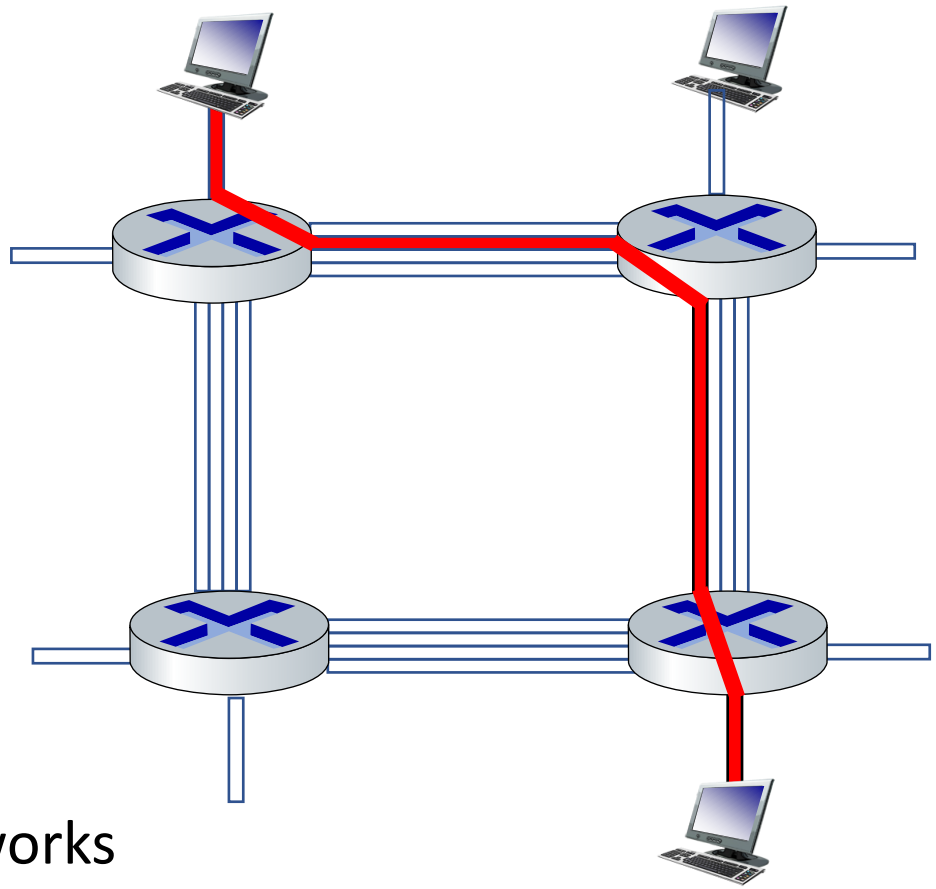
Packet queuing and loss: if arrival rate (in bps) to link exceeds transmission rate (bps) of link for some period of time:

- packets will queue, waiting to be transmitted on output link
- packets can be dropped (lost) if memory (buffer) in router fills up

Alternative to packet switching: circuit switching

end-end resources allocated to, reserved for "call" between source and destination

- 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

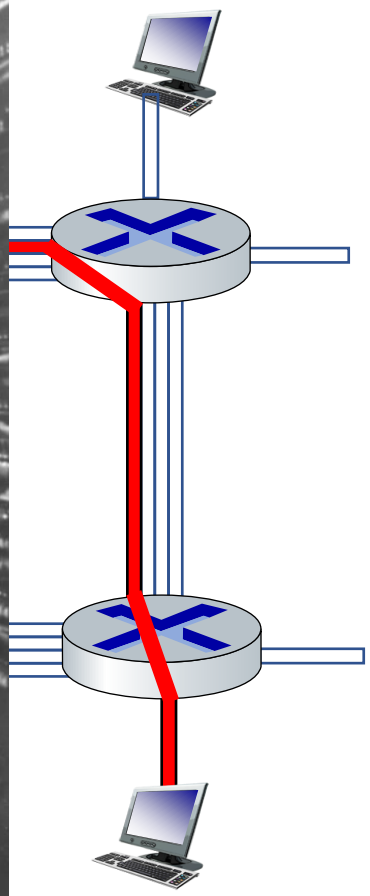


* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive

Alternative to packet switching: circuit switching

end-er
reserv
and de

- in dia
  - ca
    cir
- dedi
  - cir
- circu
  shari
- comr

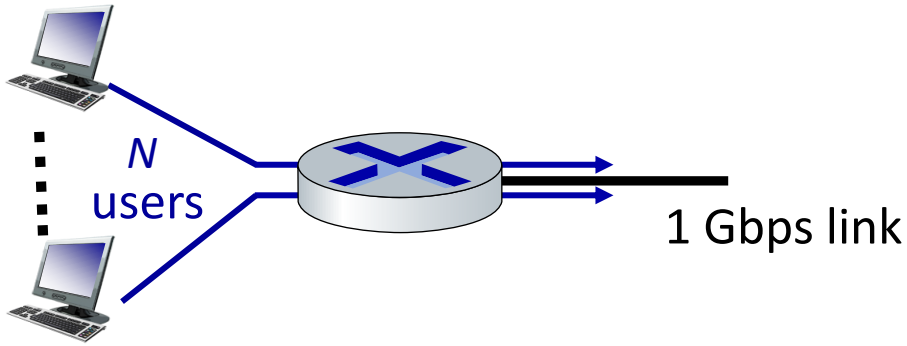


* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive

Packet switching versus circuit switching

example:

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



Q: how many users can use this network under circuit-switching and packet switching?

- *circuit-switching:* 10 users
- *packet switching:* with 35 users, probability > 10 active at same time is less than 0.0004 *

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive

Packet switching versus circuit switching

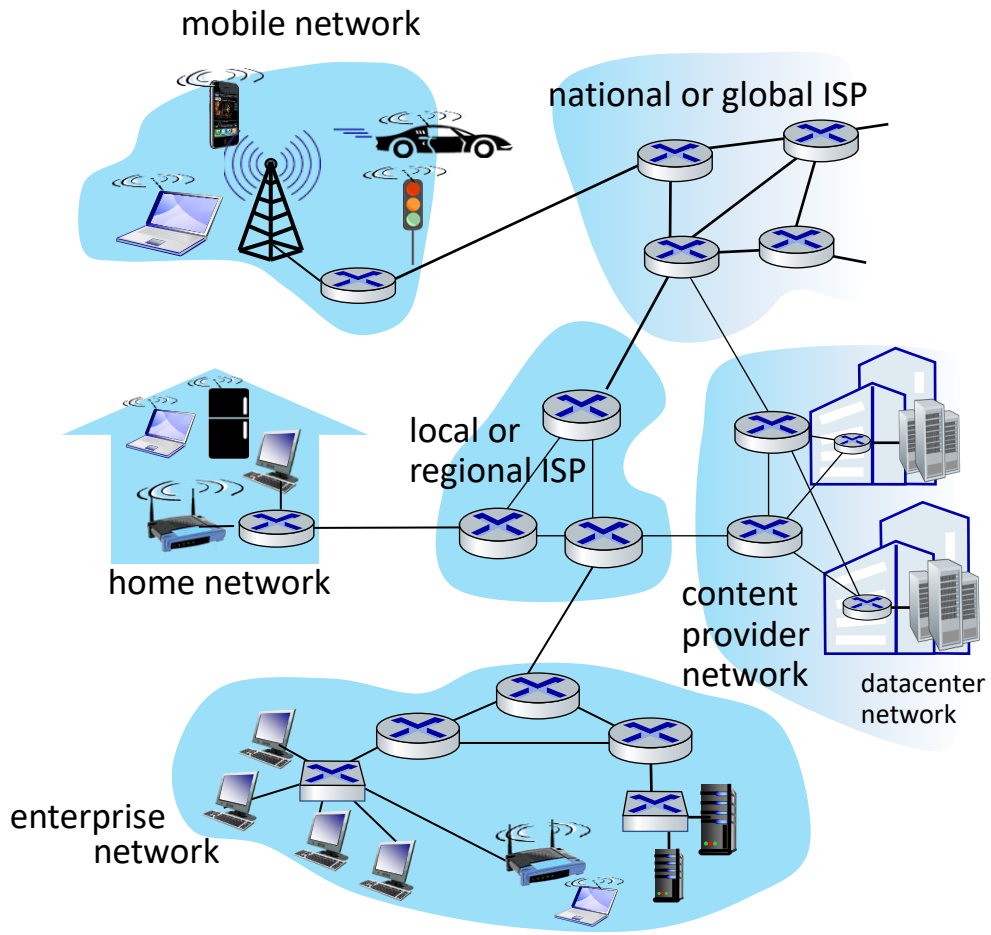
Is packet switching a "slam dunk winner"?

- great for "bursty" data – sometimes has data to send, but at other times not
  - resource sharing
  - simpler, no call setup
- excessive congestion possible: packet delay and loss due to buffer overflow
  - protocols needed for reliable data transfer, congestion control
- *Q:* How to provide circuit-like behavior with packet-switching?
  - "It's complicated." Various techniques that try to make packet switching as "circuit-like" as possible are possible (but we do not see them in this course)

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

Internet structure: a "network of networks"

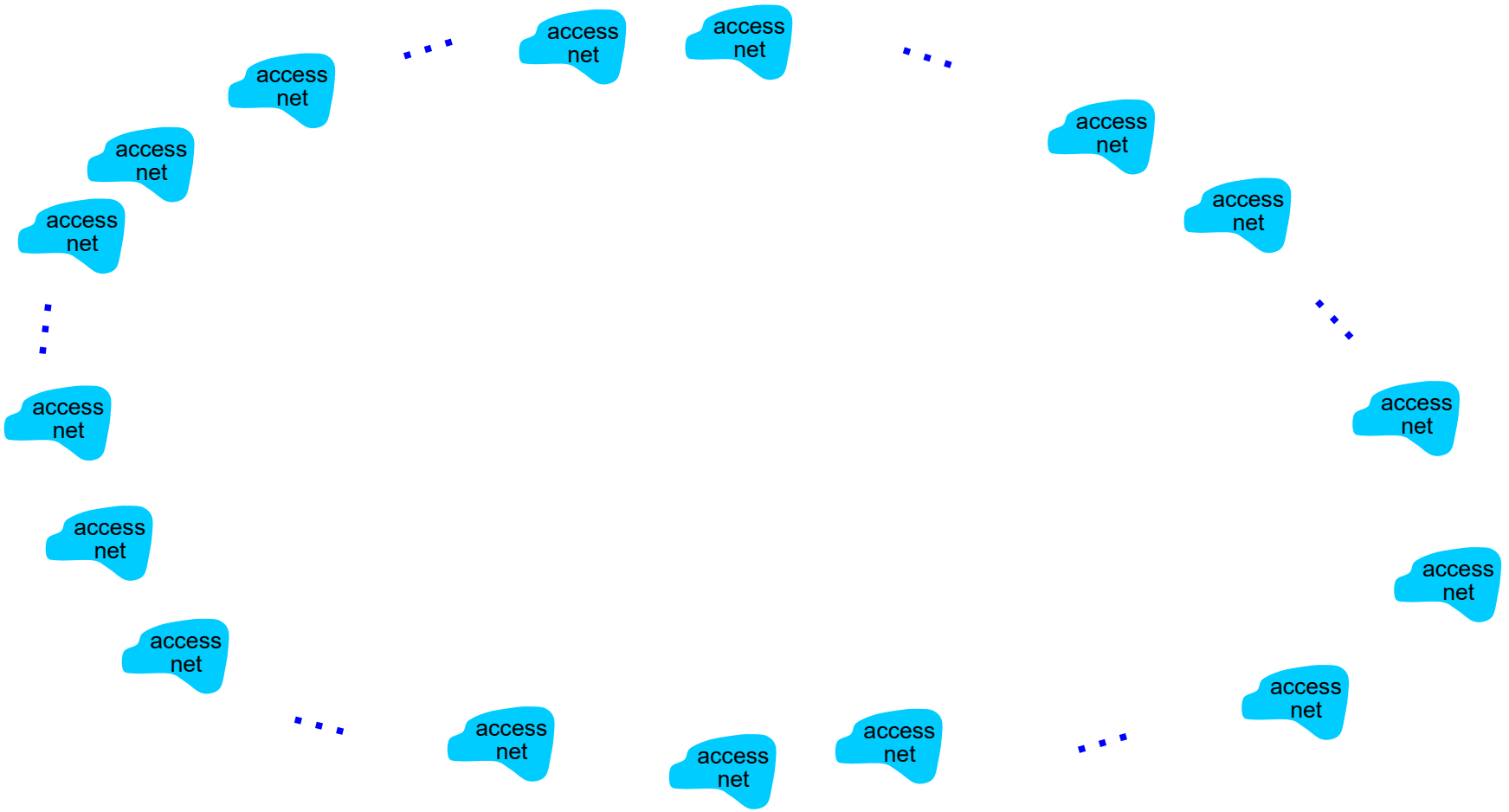
- hosts connect to Internet via access Internet Service Providers (ISPs)
- access ISPs in turn must be interconnected
  - so that *any* two hosts *(anywhere!)* can send packets to each other
- resulting network of networks is very complex
  - evolution driven by economics, national policies



Let's take a stepwise approach to describe current Internet structure

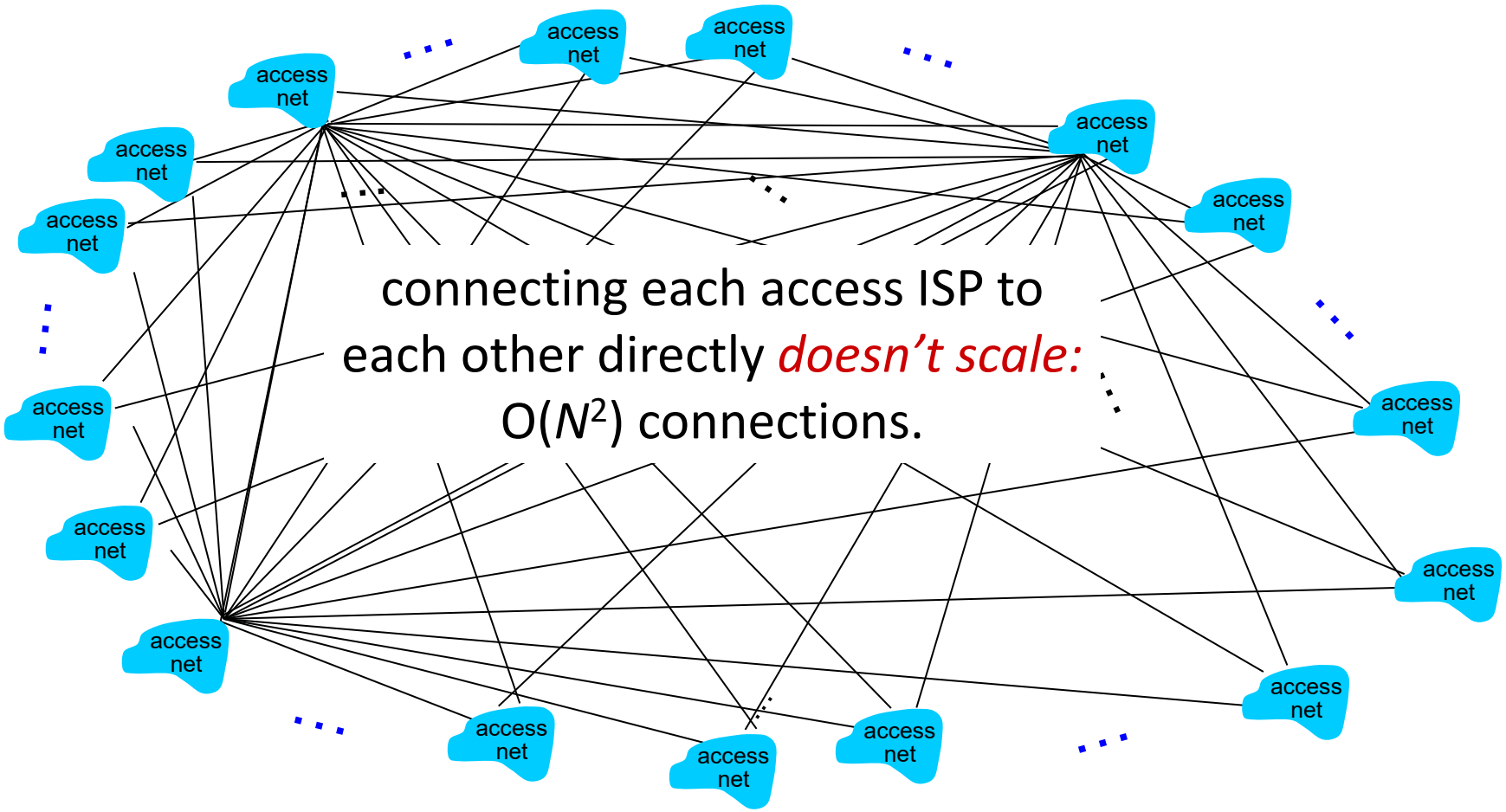
Internet structure: a "network of networks"

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



Internet structure: a "network of networks"

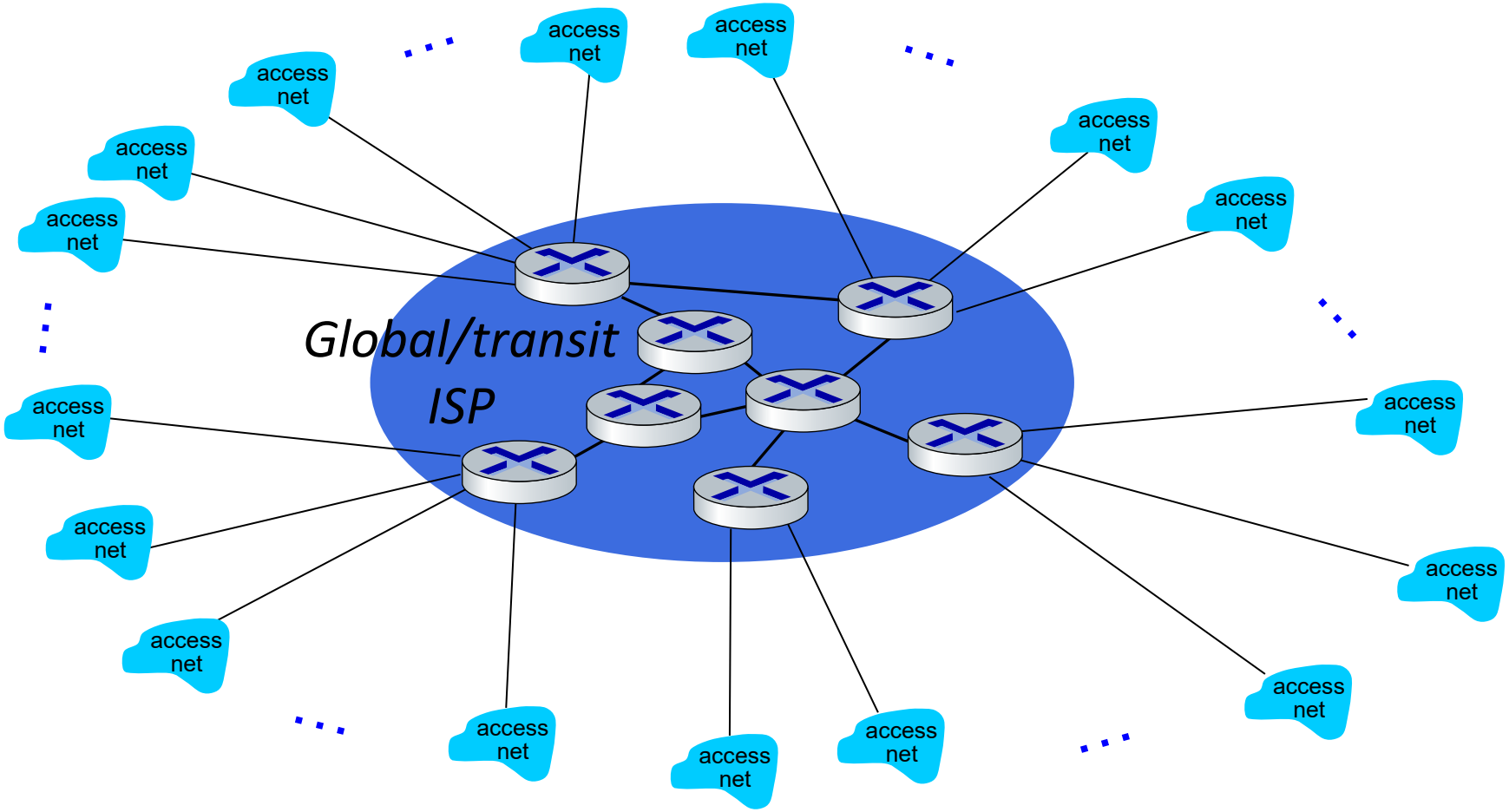
Question: given *millions* of access ISPs, how to connect them together?



Internet structure: a "network of networks"

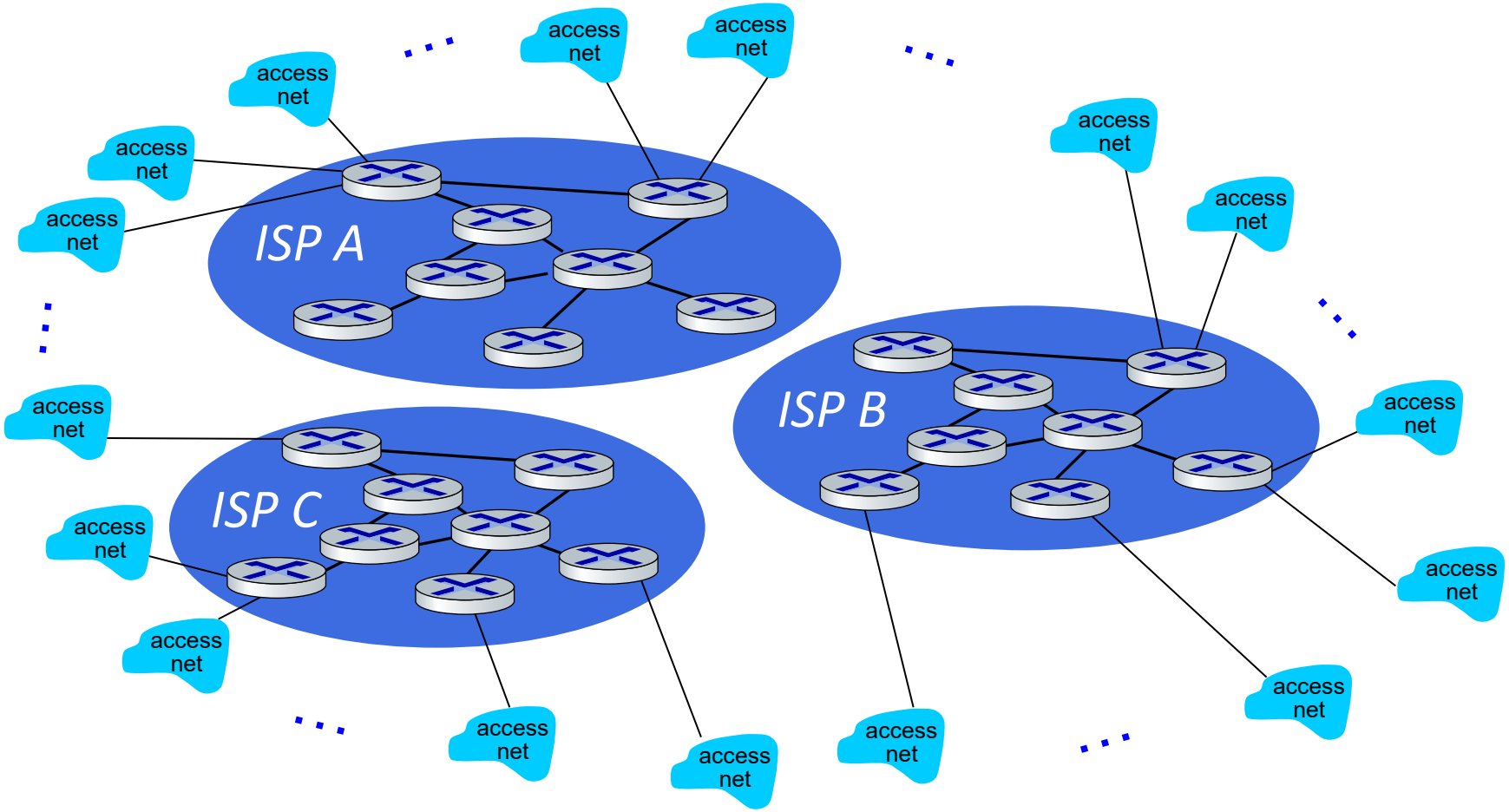
Option: connect each access ISP to one global transit ISP?

Customer and provider ISPs have economic agreement.



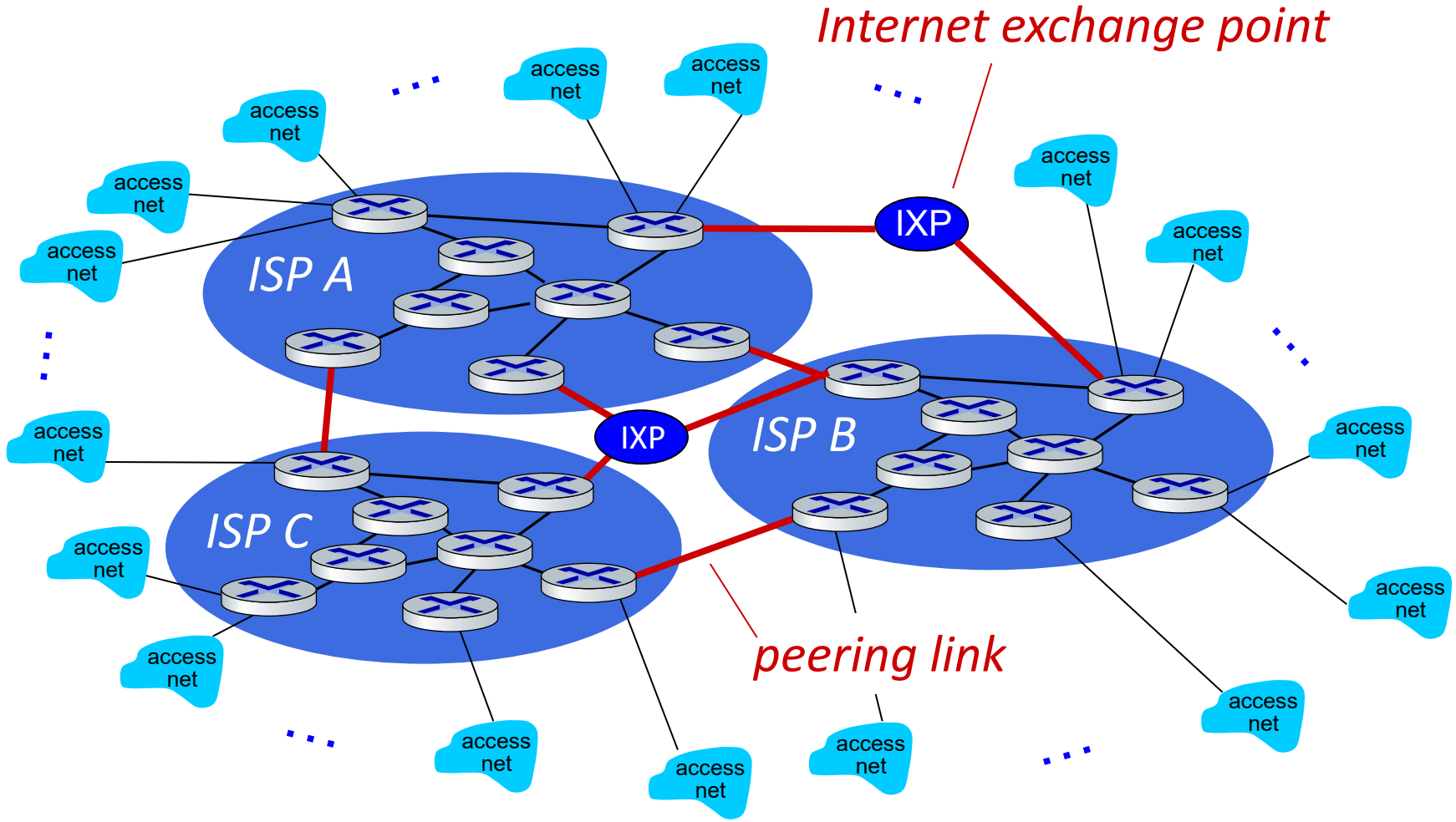
Internet structure: a "network of networks"

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



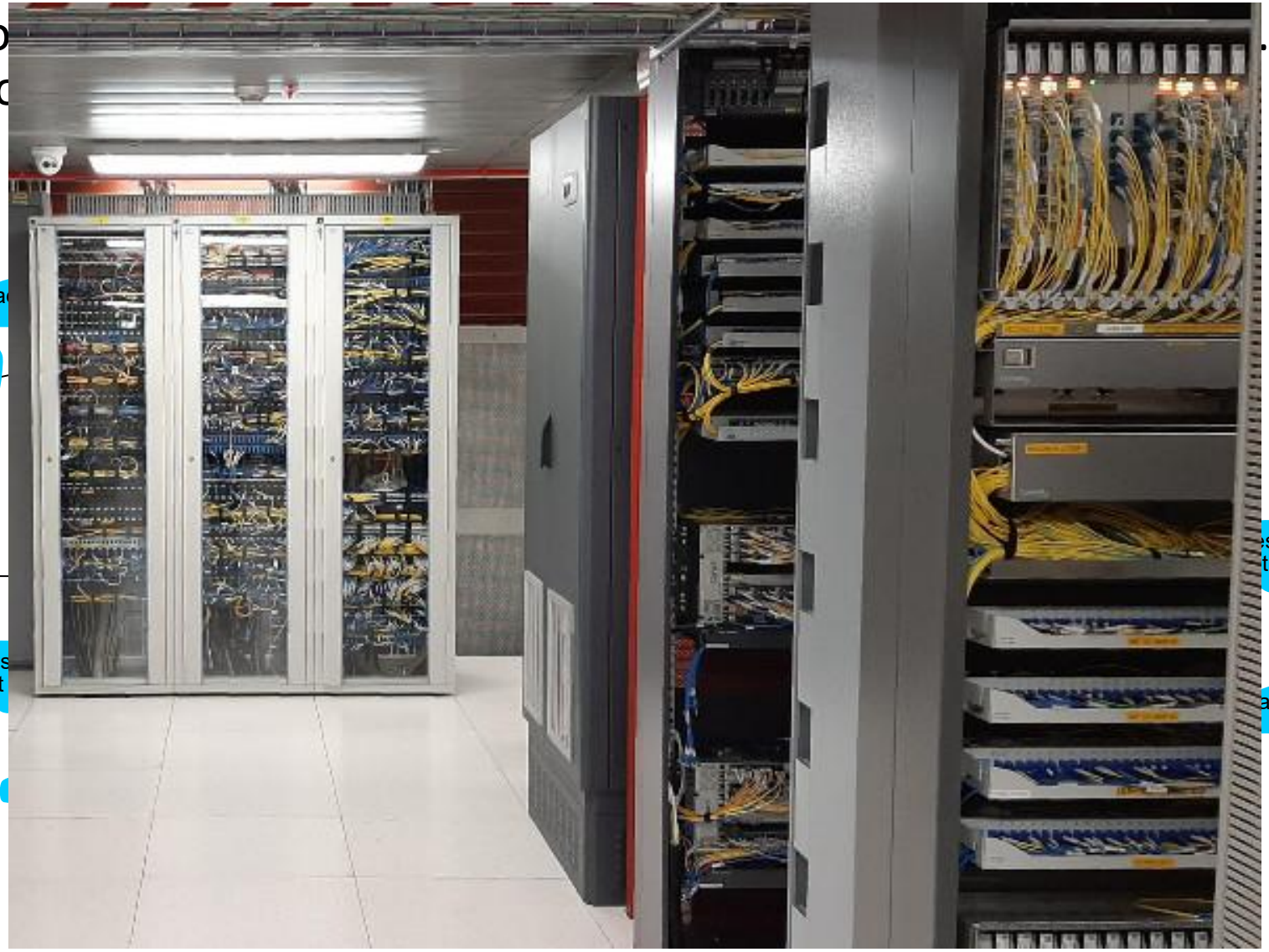
Internet structure: a "network of networks"

But if one global ISP is viable business, there will be competitors… who will want to be connected



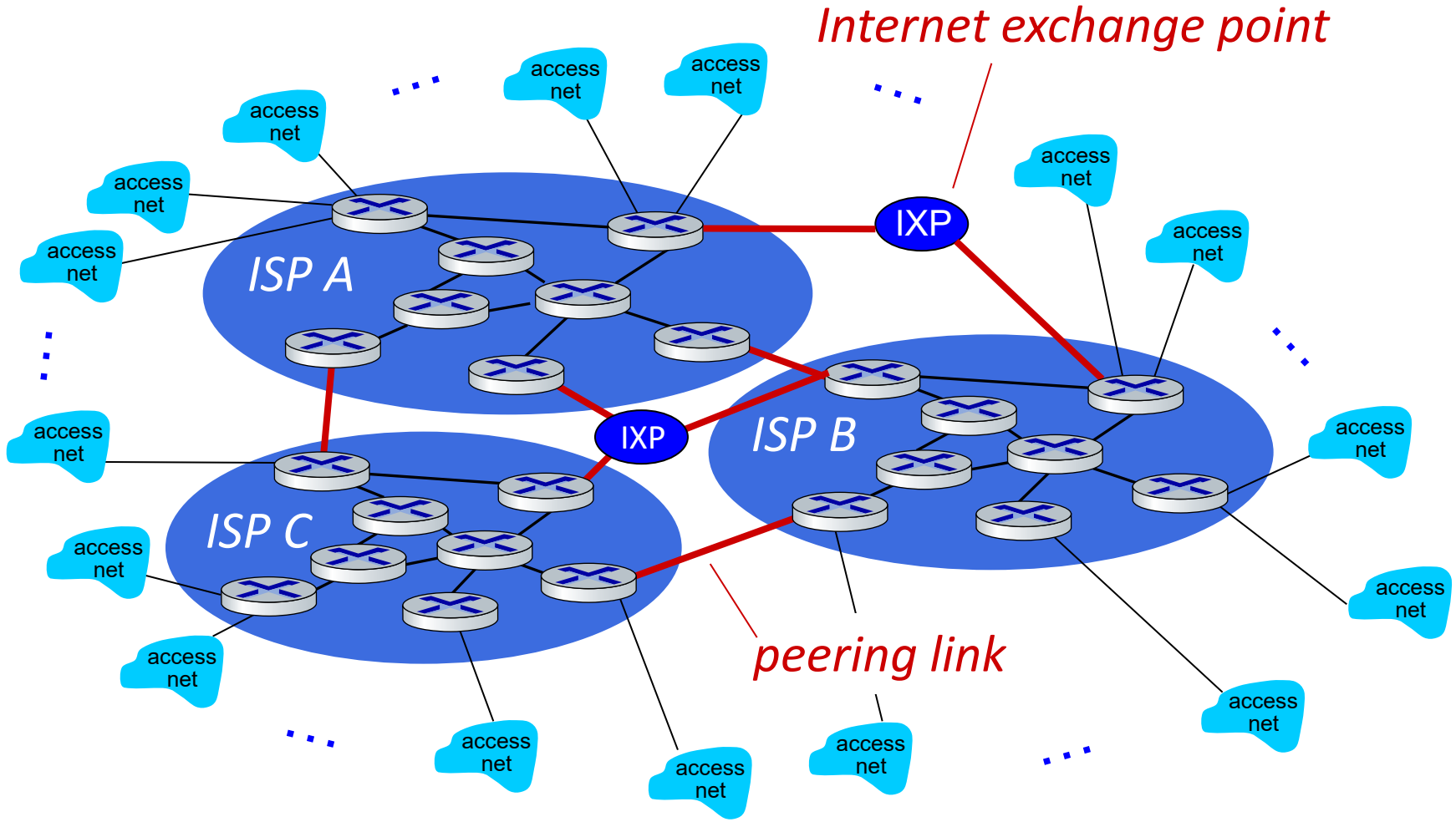
Internet structure: a "network of networks"

But if o … who will want to



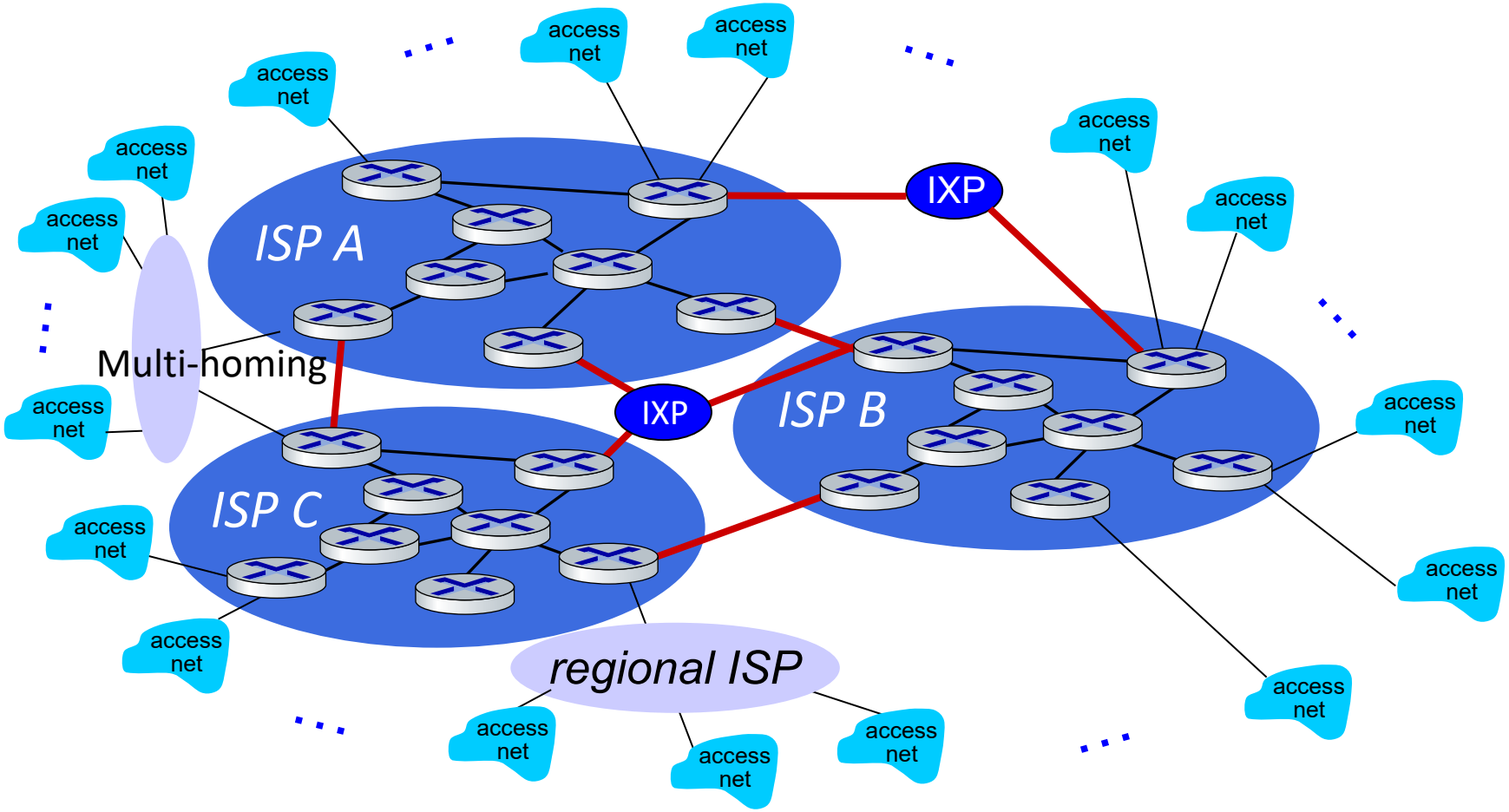
Internet structure: a "network of networks"

But if one global ISP is viable business, there will be competitors… who will want to be connected



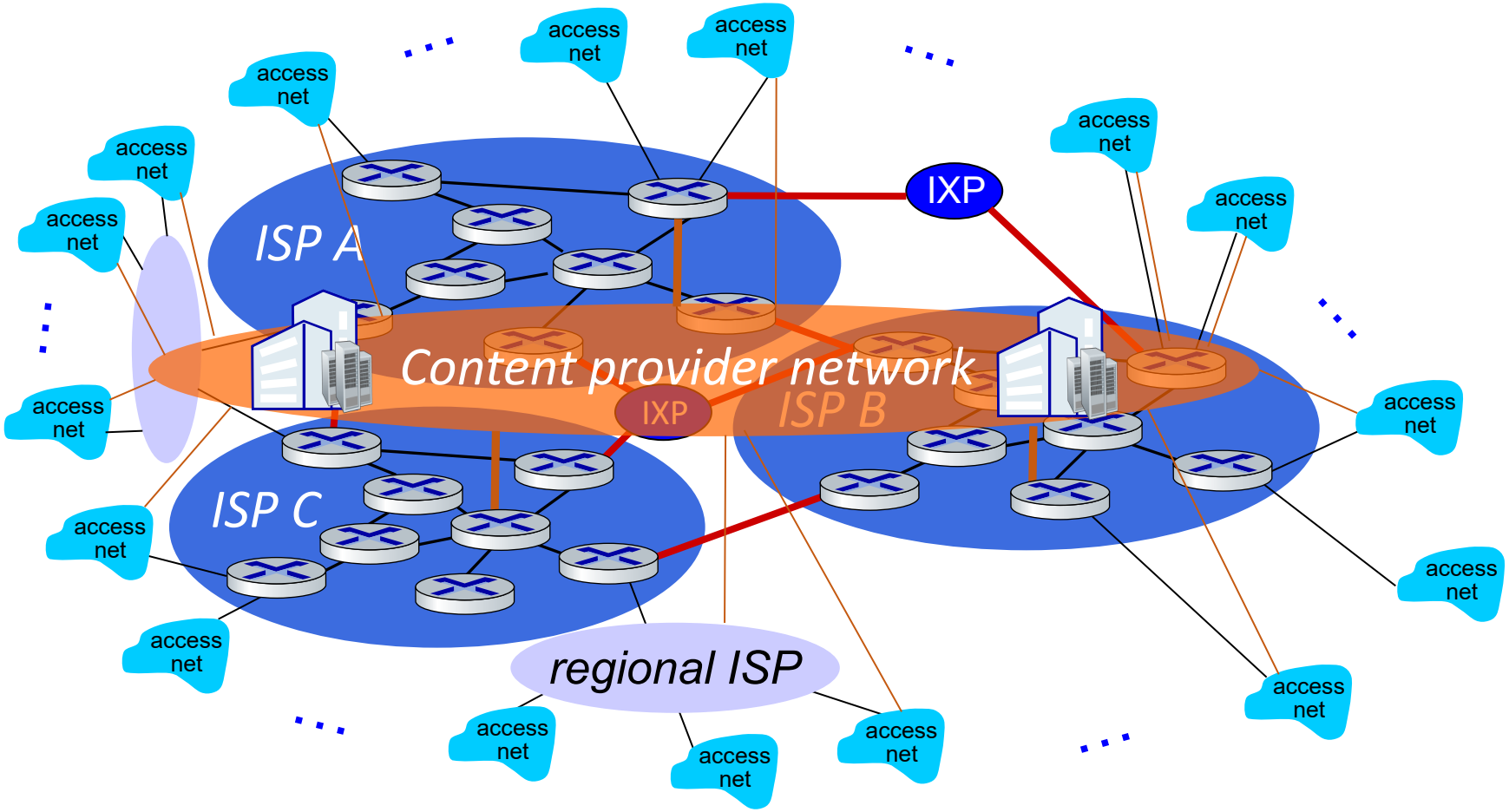
Internet structure: a "network of networks"

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

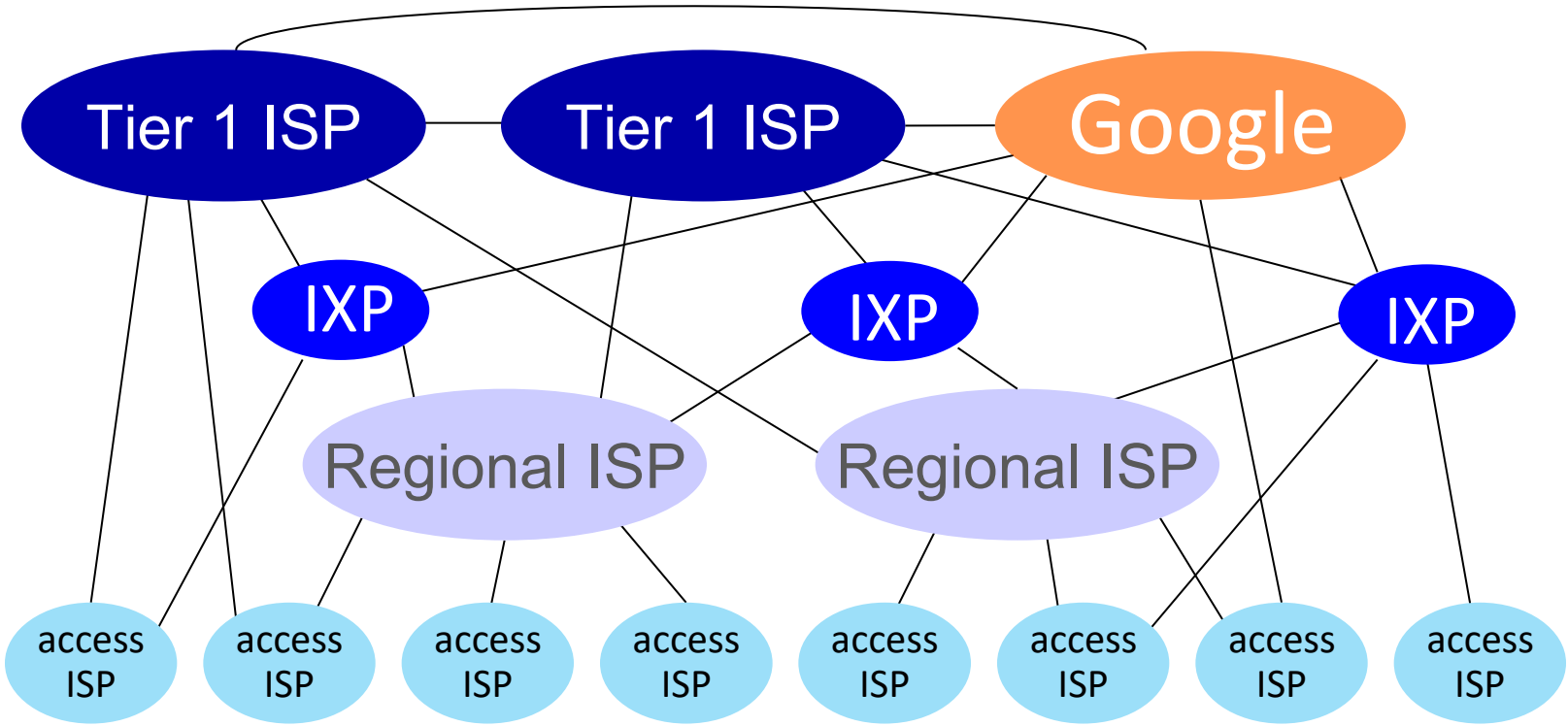


Internet structure: a "network of networks"

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



Internet structure: a "network of networks"

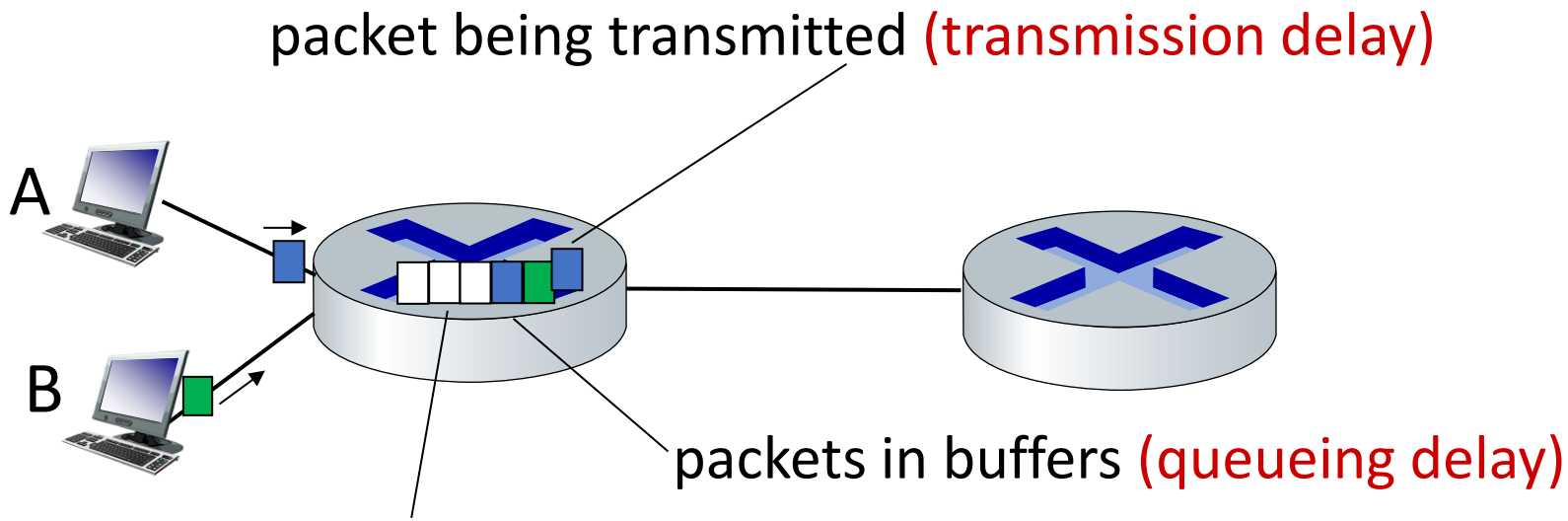


At "center": small # of well-connected large networks

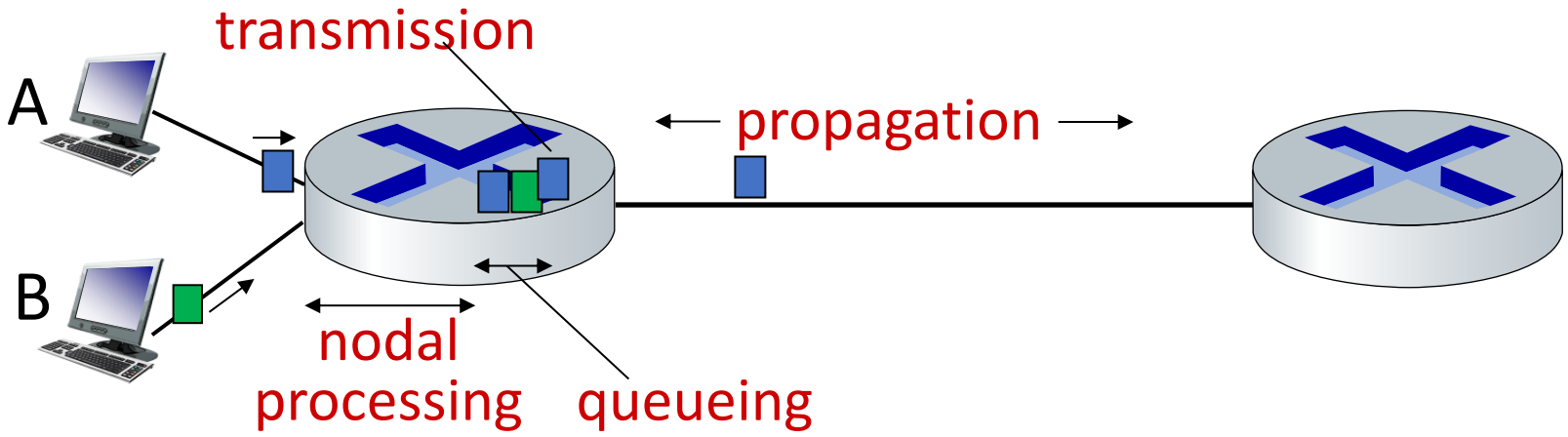
- "tier-1" commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT), national & international coverage
- content provider networks (e.g., Google, Facebook): private network that connects its data centers to Internet, often bypassing tier-1, regional ISPs

How do packet delay and loss occur?

- packets *queue* in router buffers, waiting for turn for transmission
  - queue length grows when arrival rate to link (temporarily) exceeds output link capacity
- packet *loss* occurs when memory to hold queued packets fills up



Packet delay: four sources



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

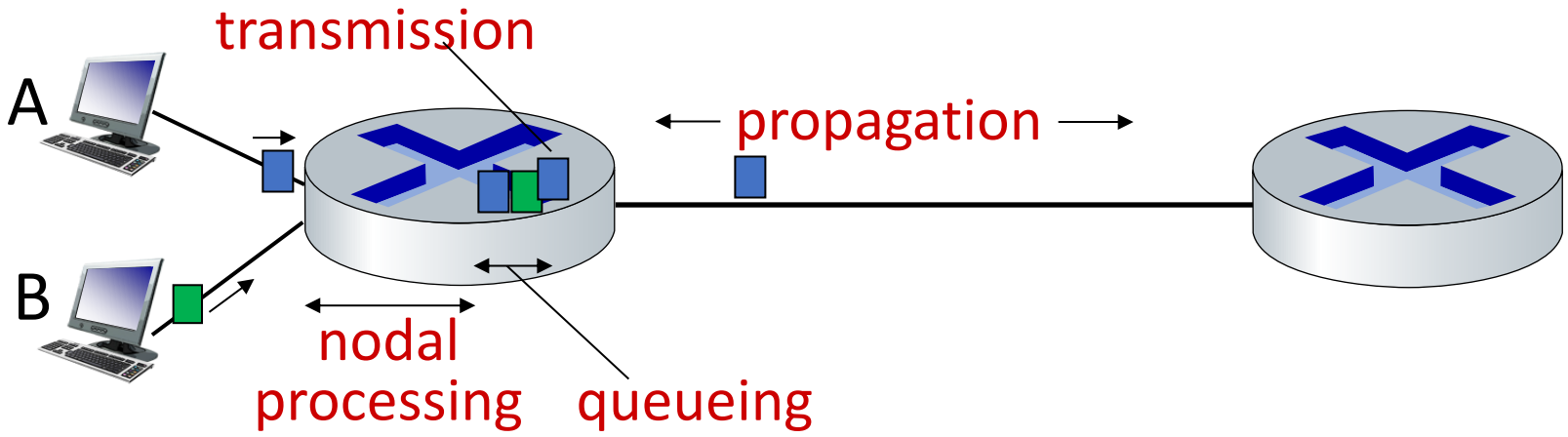
d_{proc}: nodal processing

- check bit errors
- determine output link
- typically < microsecs

d_{queue}: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router
- typically in the order of millisecs

Packet delay: four sources



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

d_{trans}: transmission delay:

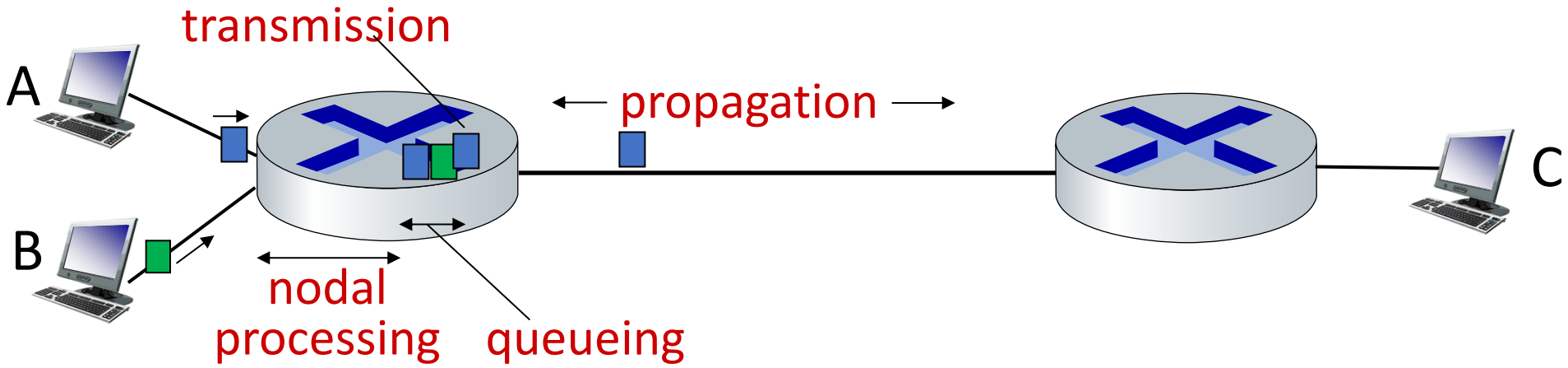
- L: packet length (bits)
- R: link transmission rate (bps)
- $d_{trans} = L/R$

d_{prop}: propagation delay:

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

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

Packet delay: end-to-end delay



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

End-to-end delay:

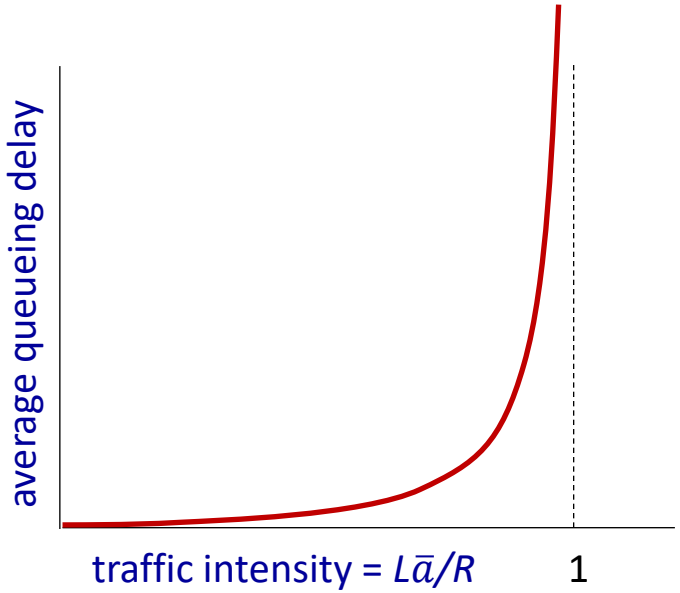
- Sum of:
  - $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$ introduced by all the packet switches on the path between involved end systems (source and destination)
  - delays introduced by the end systems themselves

Packet queueing delay (revisited)

- $\bar{a}$: average packet arrival rate (packet/s)
- L: packet length (bit/packet)
- R: link bandwidth (bit transmission rate, bit/s)

$$\frac{L \cdot \bar{a}}{R} : \frac{\text{average arrival rate of bits (bit/s)}}{\text{service rate of bits (bit/s)}}$$

"traffic intensity"



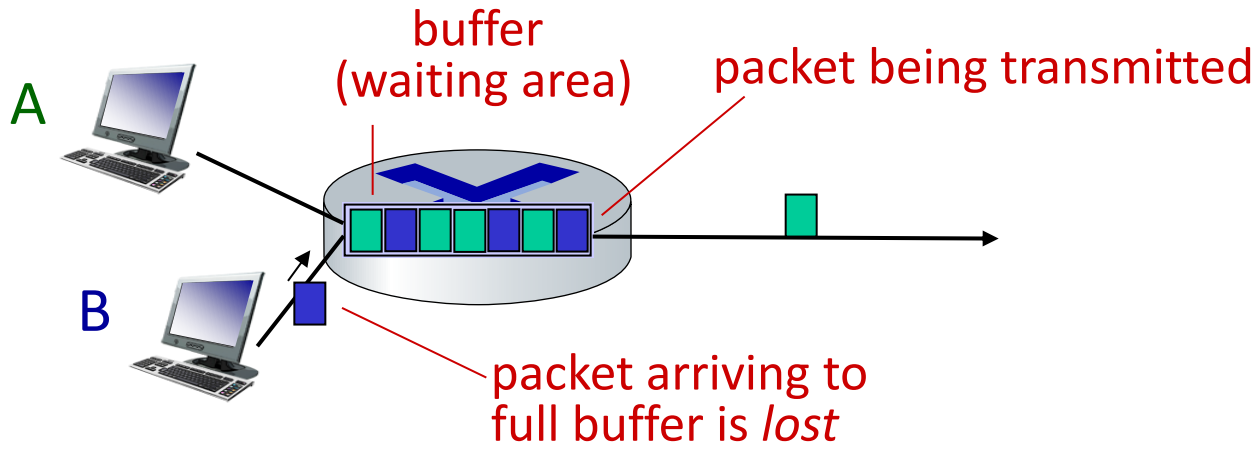
- $L\bar{a}/R \sim 0$: avg. queueing delay small
- $L\bar{a}/R \rightarrow 1$: avg. queueing delay large
- $L\bar{a}/R > 1$: "work" arriving is more than it can be serviced - average delay infinite (assuming a queue of infinite size!)

$L\bar{a}/R \sim 0$

$L\bar{a}/R \rightarrow 1$

Packet loss

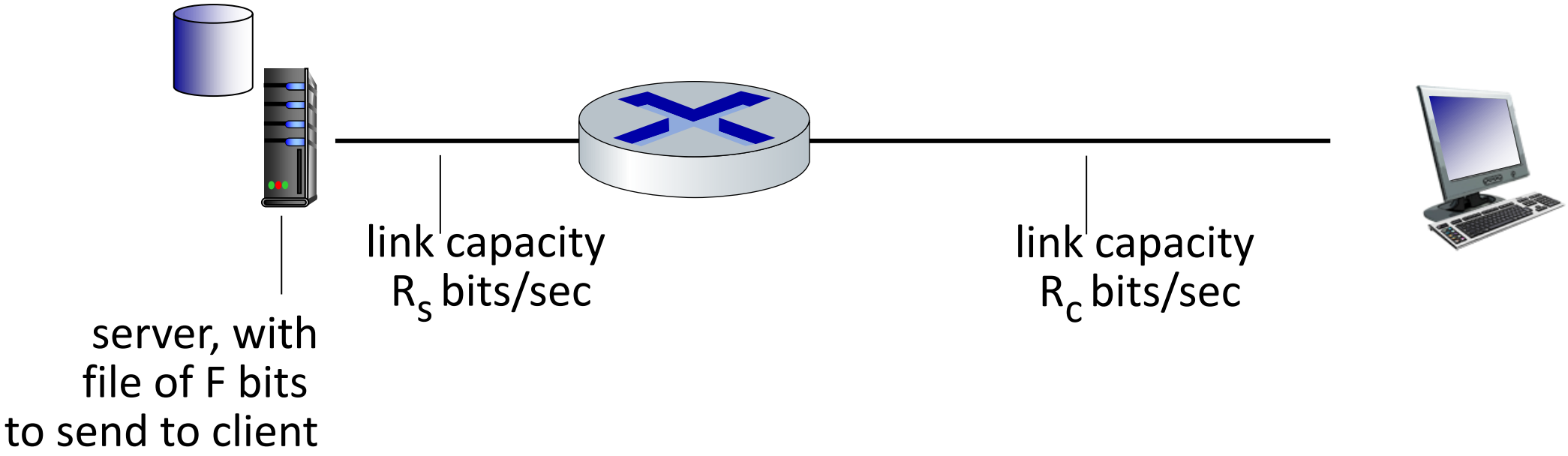
- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- packets can be discarded, and thus lost, also if transmission errors occur



- lost packets may be retransmitted by previous node, by source end system, or not at all

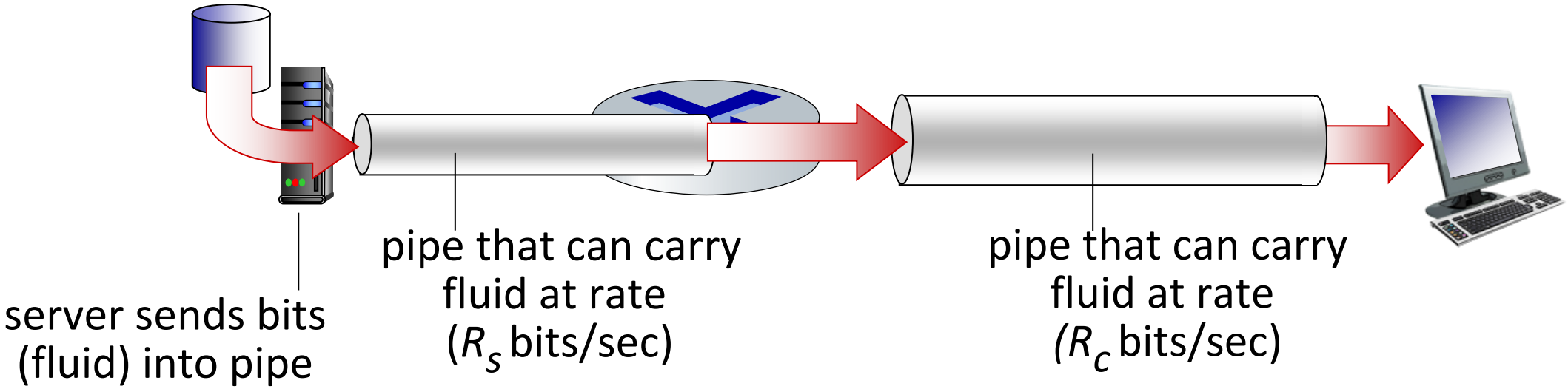
Throughput

- *throughput:* rate (bits/time unit) at which bits are being sent from sender to receiver (also often called *end-to-end throughput*)
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time



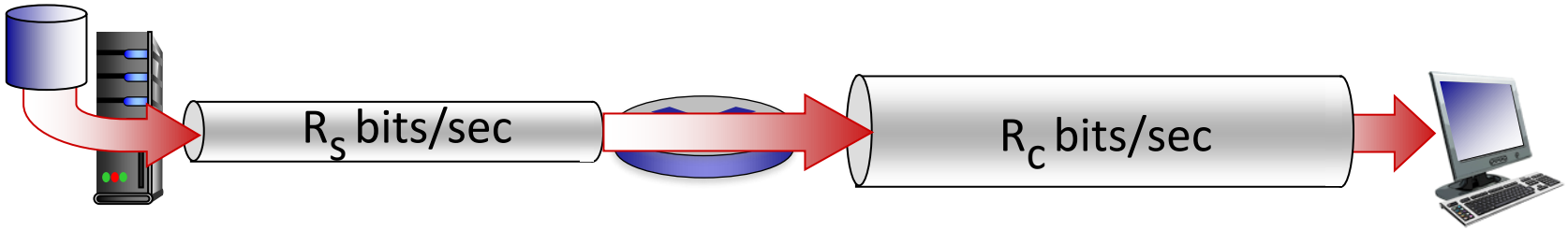
Throughput

- *throughput:* rate (bits/time unit) at which bits are being sent from sender to receiver (also often called *end-to-end throughput*)
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time



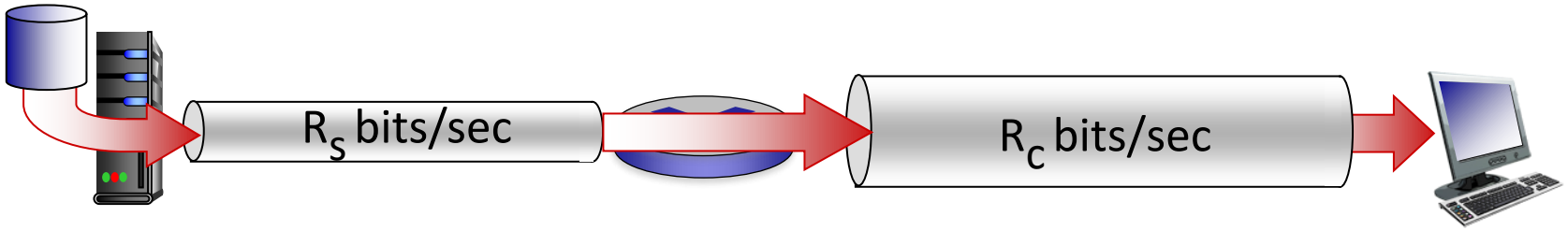
Throughput

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

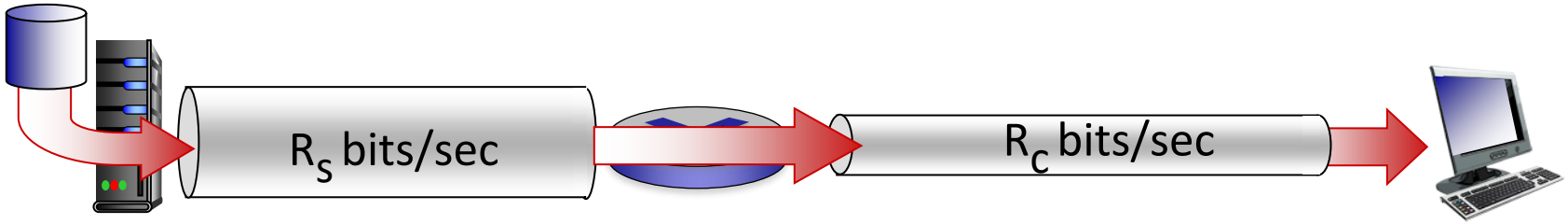


Throughput

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

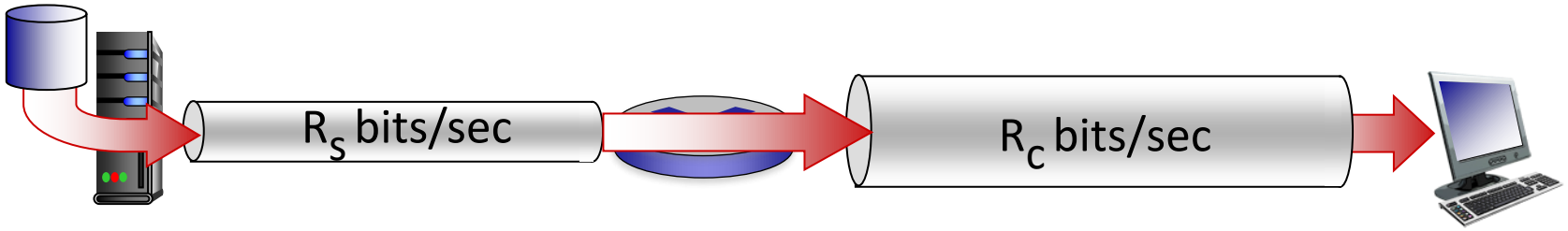


$R_s > R_c$ What is average end-to-end throughput?

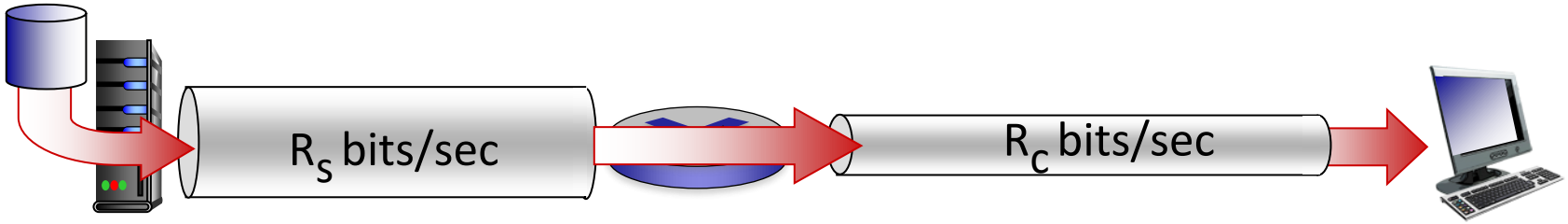


Throughput

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



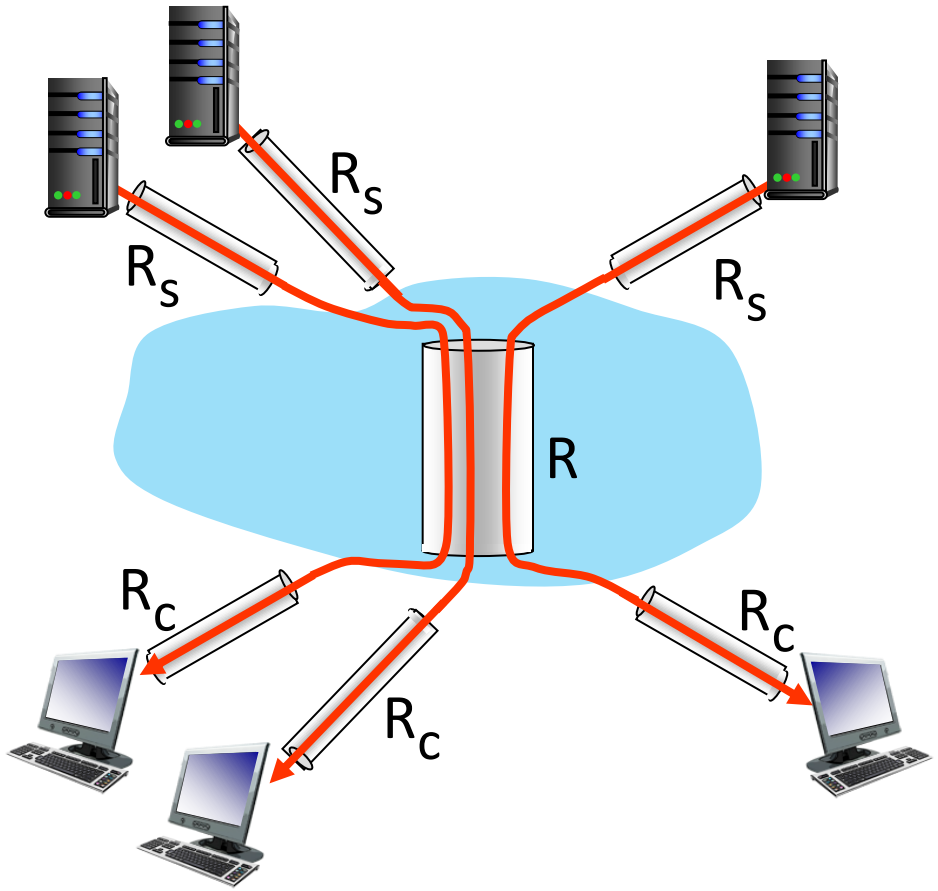
$R_s > R_c$ What is average end-to-end throughput?



bottleneck link

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

Throughput: network scenario



10 connections (fairly) share bottleneck link *R* bits/sec

- per-connection end-end throughput: *min(R_c,R_s,R/10)*
- in practice: R_c or R_s is often bottleneck

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/

Protocol “layers” and reference models

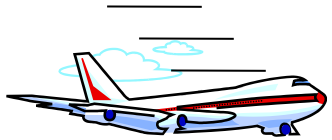
Networks are complex systems, with many “pieces”:

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

Question: is there any hope of *organizing* the architecture of networks?

- and/or our *discussion* of networks?

Example: organization of air travel

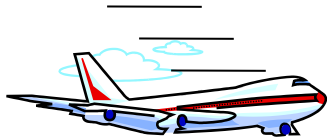


end-to-end transfer of person plus baggage

How would you *define/discuss* the *system* of airline travel?

- a series of steps, involving many services

Example: organization of air travel



end-to-end transfer of person plus baggage

ticket (purchase)

baggage (check)

gates (load)

runway takeoff

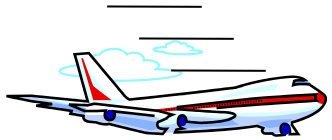
airplane routing

airplane routing

How would you *define/discuss* the *system* of airline travel?

- a series of steps, involving many services

Example: organization of air travel



end-to-end transfer of person plus baggage

ticket (purchase)

baggage (check)

gates (load)

runway takeoff

airplane routing

ticket (complain)

baggage (claim)

gates (unload)

runway landing

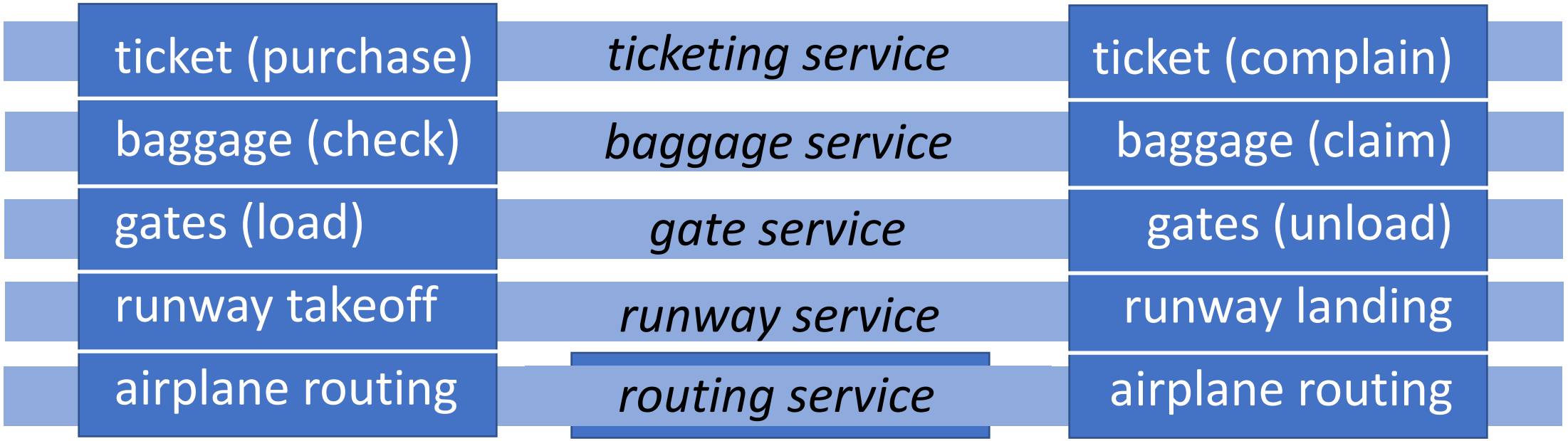
airplane routing

airplane routing

How would you *define/discuss* the *system* of airline travel?

- a series of steps, involving many services

Example: organization of air travel



layers: each layer implements a service

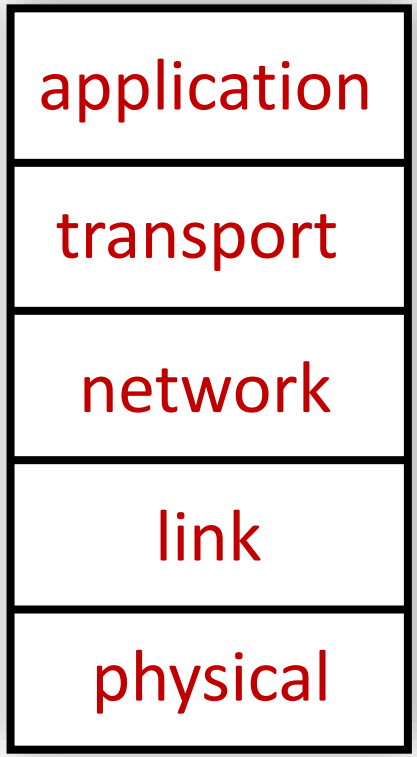
- via its own internal-layer actions
- relying on services provided by layer below

Why layering?

Approach to designing/discussing complex systems:

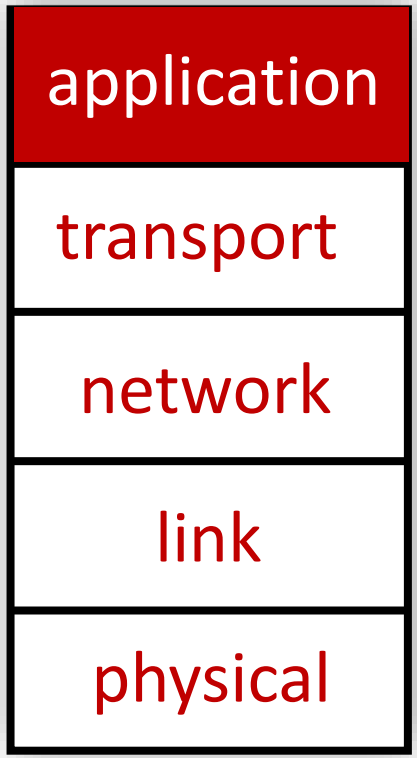
- explicit structure allows identification, relationship of system's pieces
  - layered *reference model* for discussion
- modularization eases maintenance, updating of system
  - change in layer's service *implementation*: transparent to rest of system
  - e.g., change in gate procedure doesn't affect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

Layered Internet protocol stack



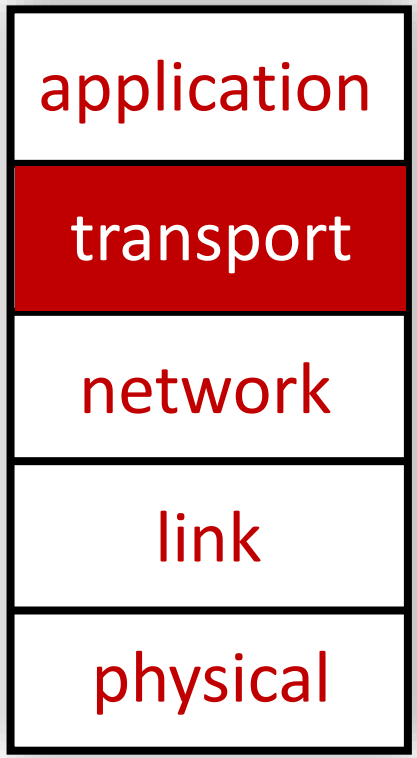
Layered Internet protocol stack

- *application:* supporting network applications
  - HTTP, SMTP, DNS



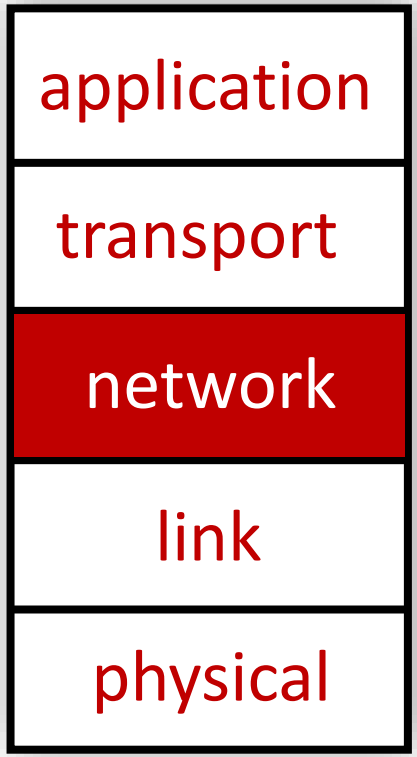
Layered Internet protocol stack

- *application:* supporting network applications
  - HTTP, SMTP, DNS
- *transport:* process-process data transfer
  - TCP, UDP



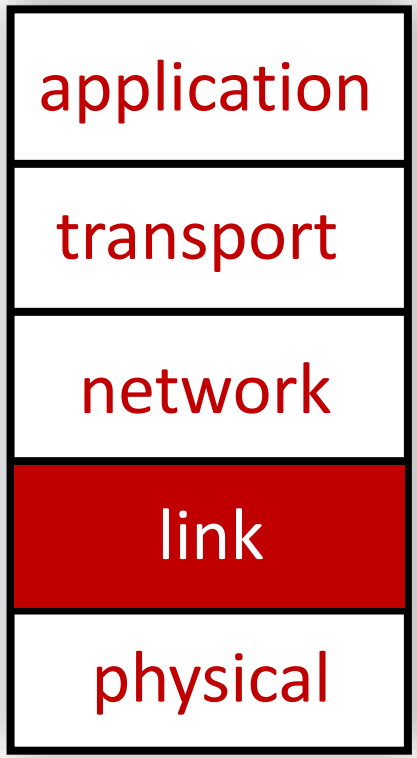
Layered Internet protocol stack

- *application:* supporting network applications
  - HTTP, SMTP, DNS
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols



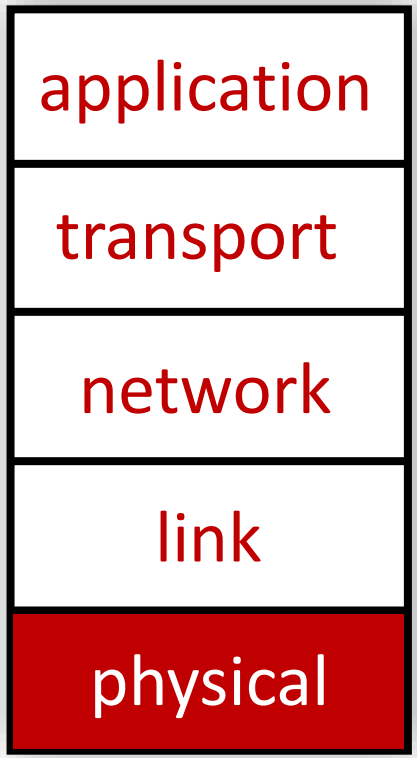
Layered Internet protocol stack

- *application:* supporting network applications
  - HTTP, SMTP, DNS
- *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, 802.11 (WiFi)



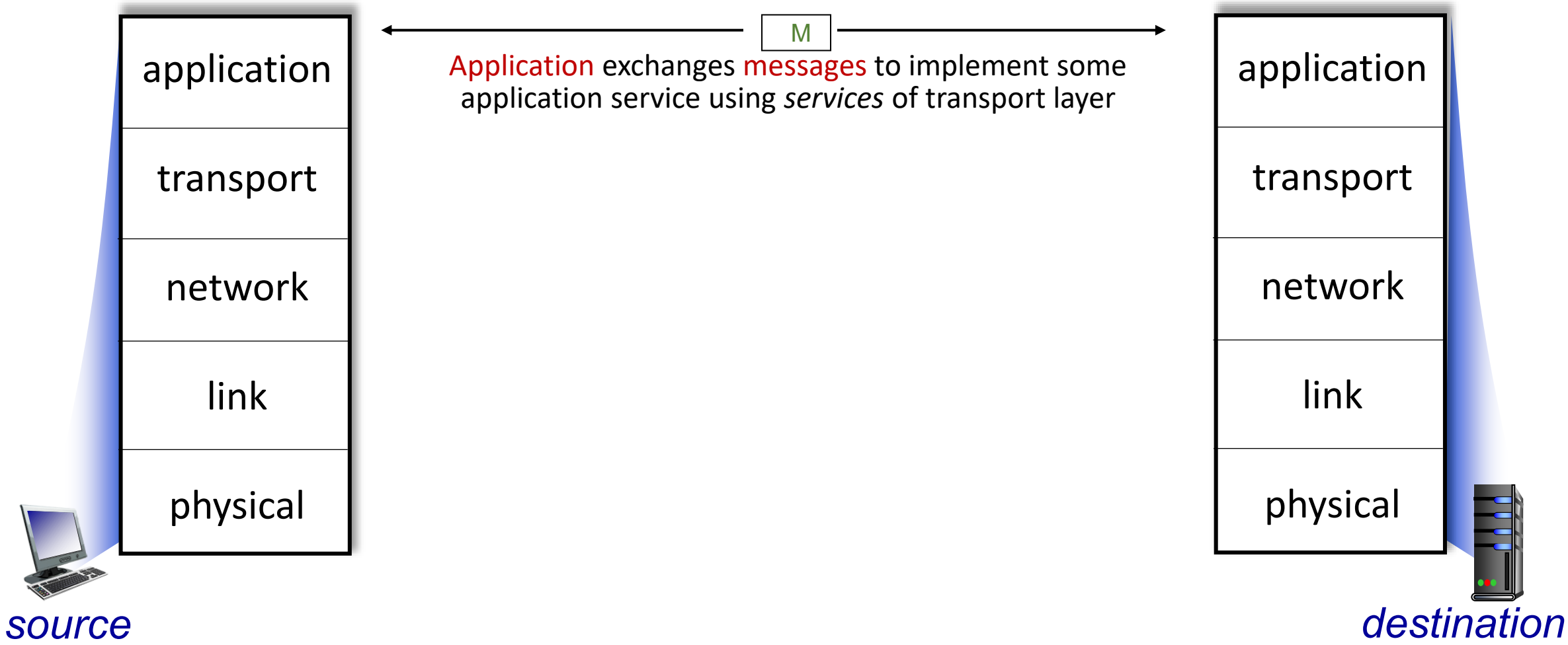
Layered Internet protocol stack

- *application:* supporting network applications
  - HTTP, SMTP, DNS
- *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, 802.11 (WiFi)
- *physical:* bits “on the wire”

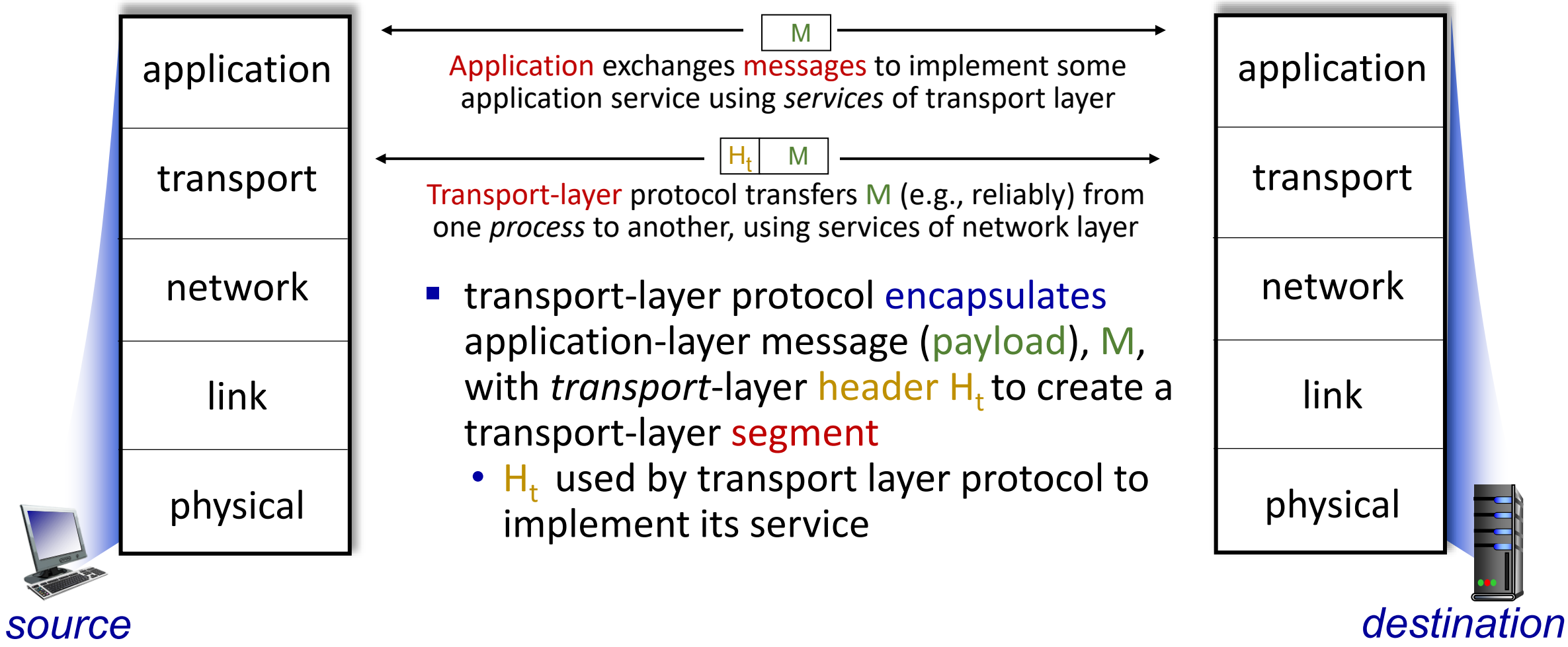


Services, Layering and Encapsulation

Services, Layering and Encapsulation

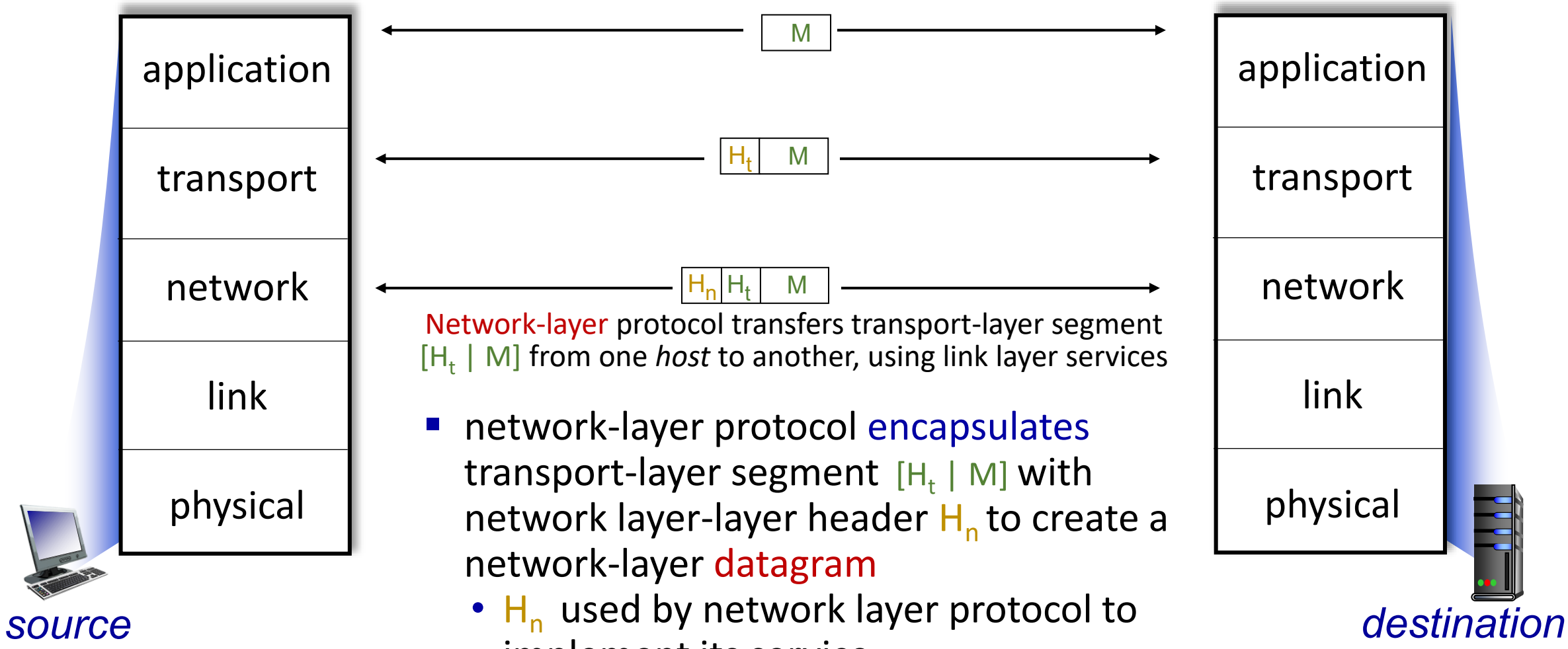


Services, Layering and Encapsulation



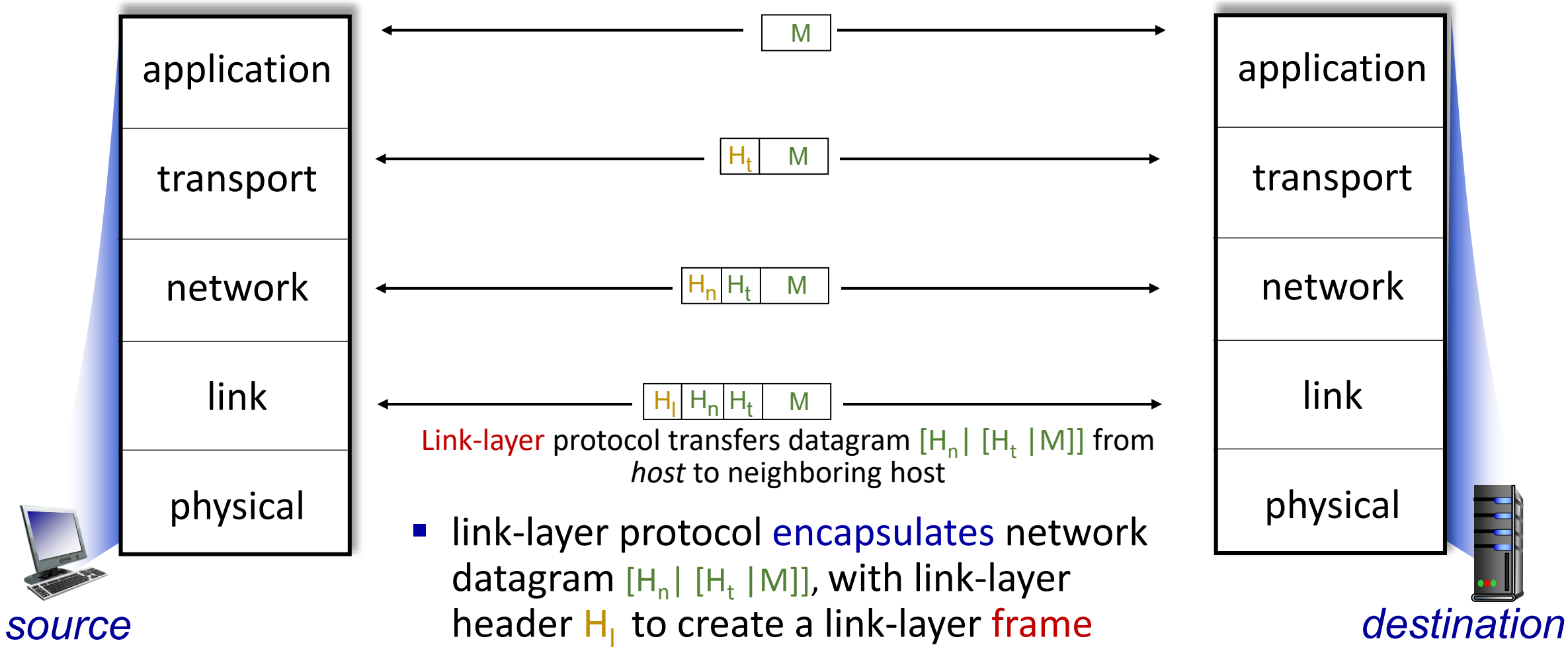
- transport-layer protocol encapsulates application-layer message (payload), M, with *transport*-layer header H_t to create a transport-layer segment
  - H_t used by transport layer protocol to implement its service

Services, Layering and Encapsulation



- network-layer protocol encapsulates transport-layer segment [H_t | M] with network layer-layer header H_n to create a network-layer datagram
  - H_n used by network layer protocol to implement its service

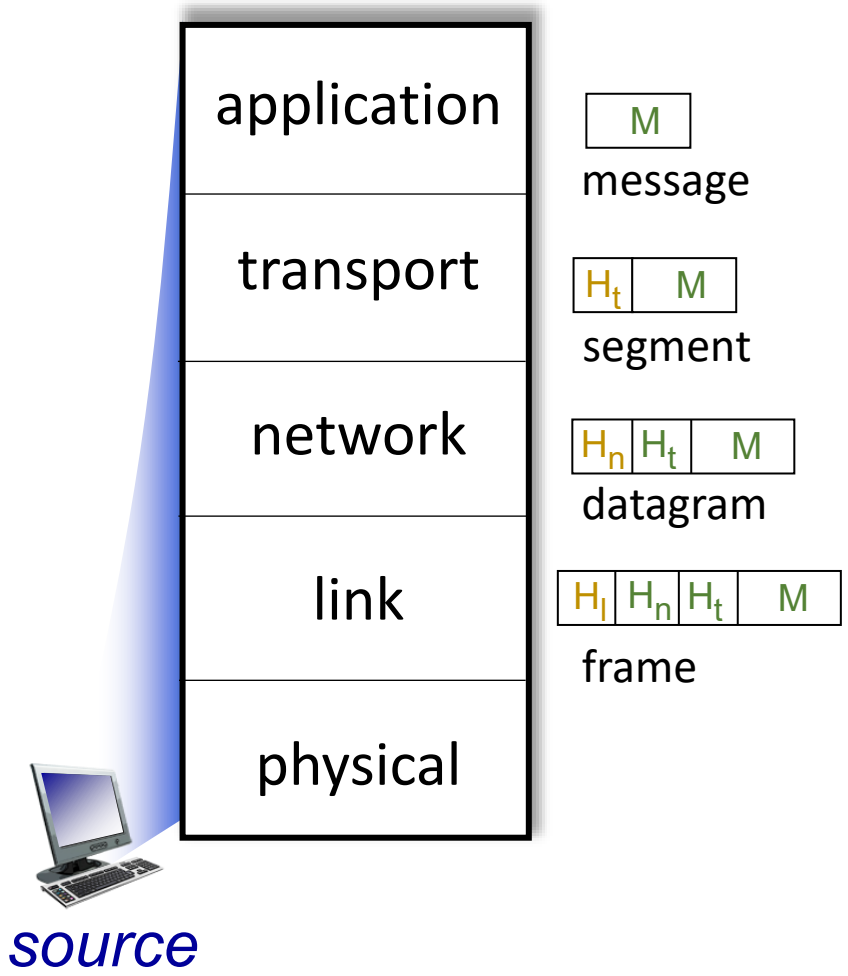
Services, Layering and Encapsulation

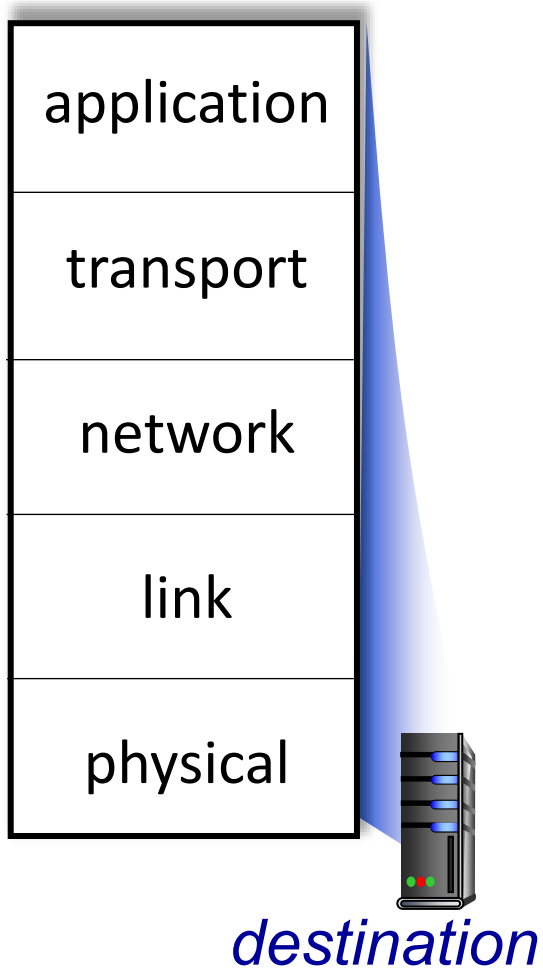


Link-layer protocol transfers datagram $[H_n|\ [H_t\ |M]]$ from *host* to neighboring host

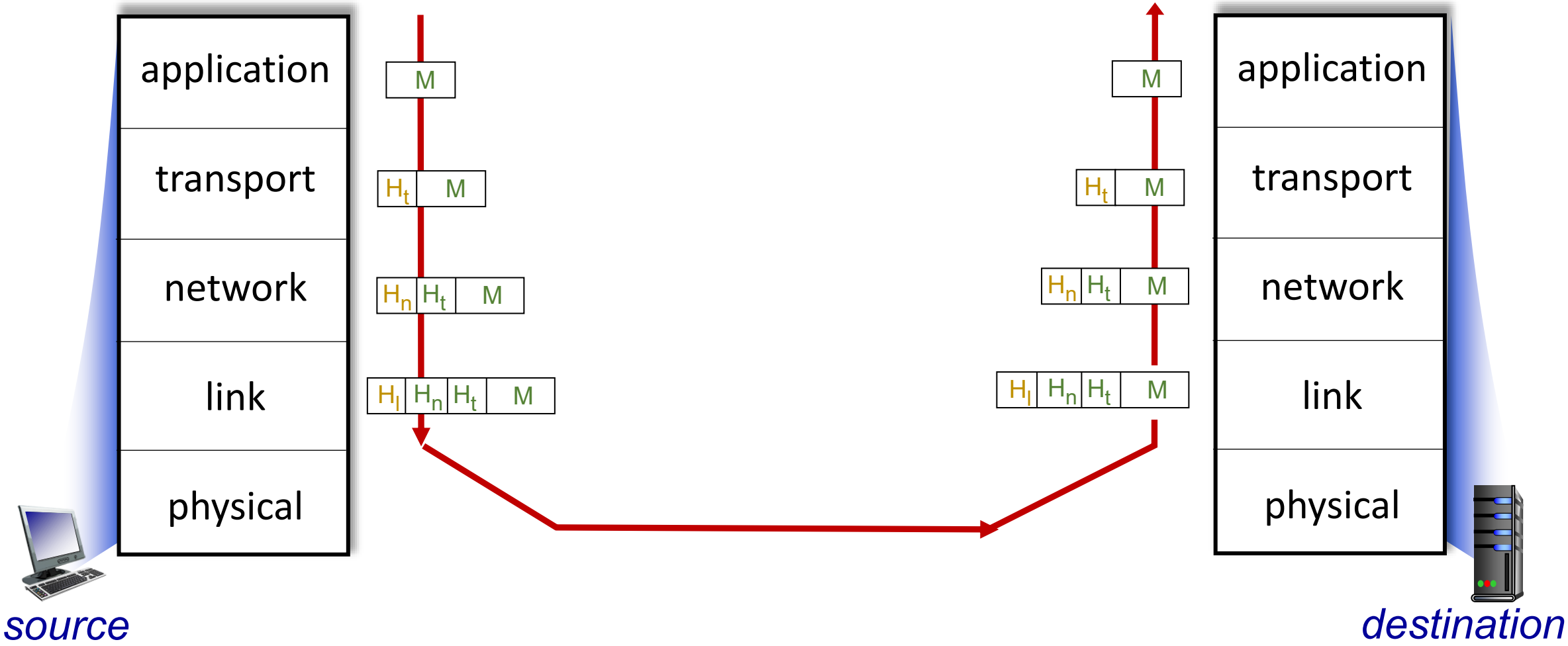
- link-layer protocol encapsulates network datagram $[H_n|\ [H_t\ |M]]$, with link-layer header H_l to create a link-layer frame

Services, Layering and Encapsulation

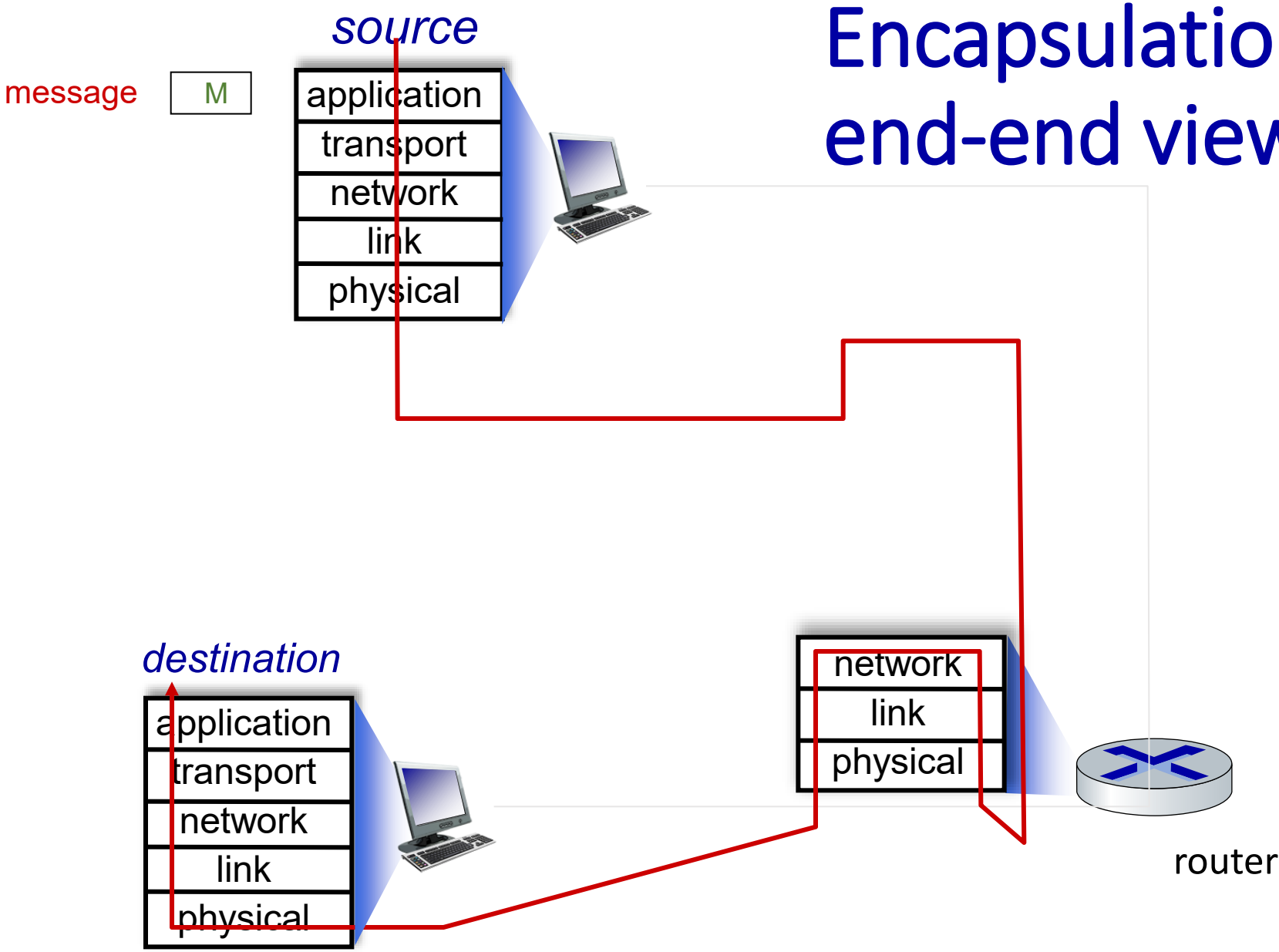




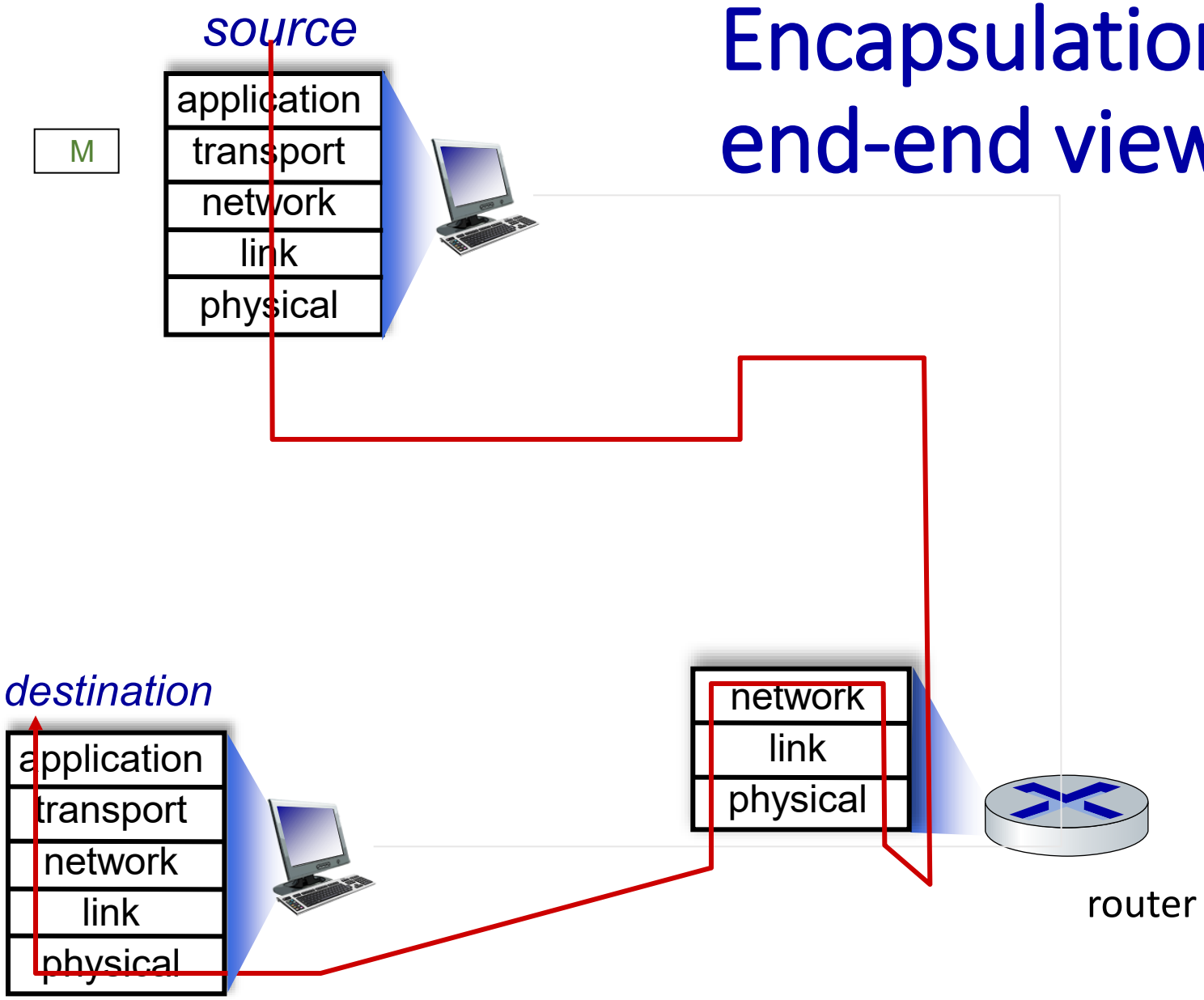
Services, Layering and Encapsulation



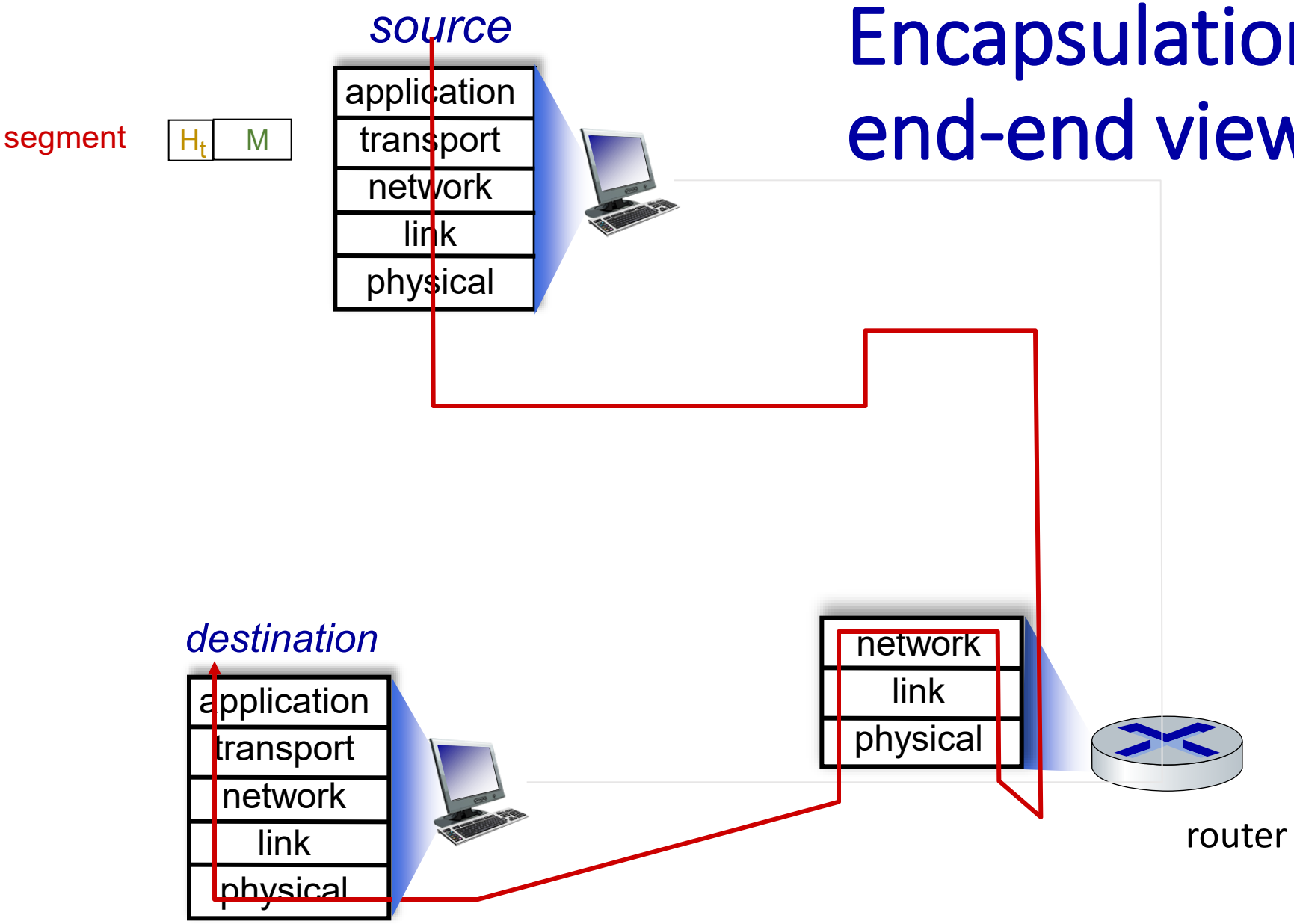
Encapsulation: an end-end view



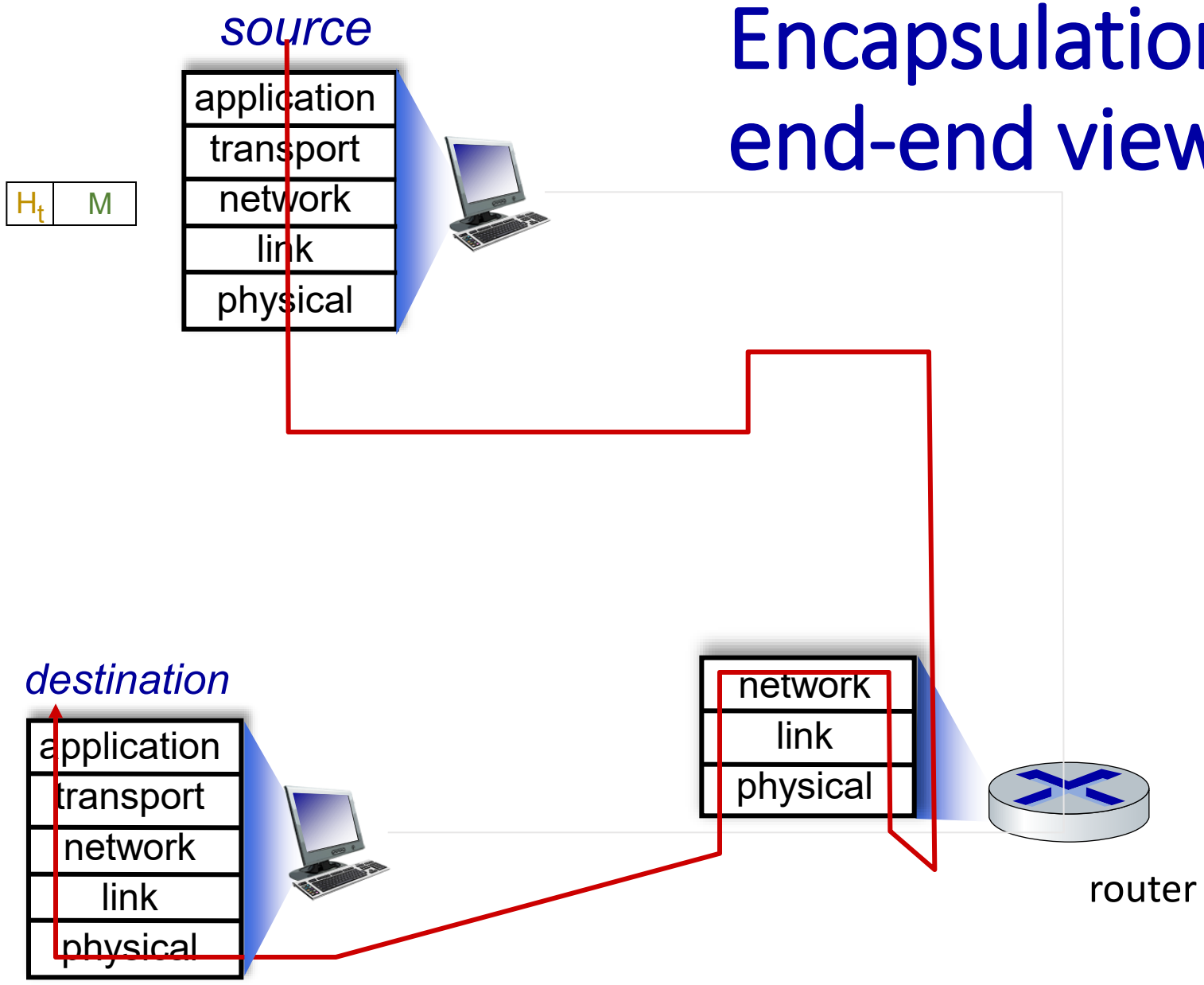
Encapsulation: an end-end view



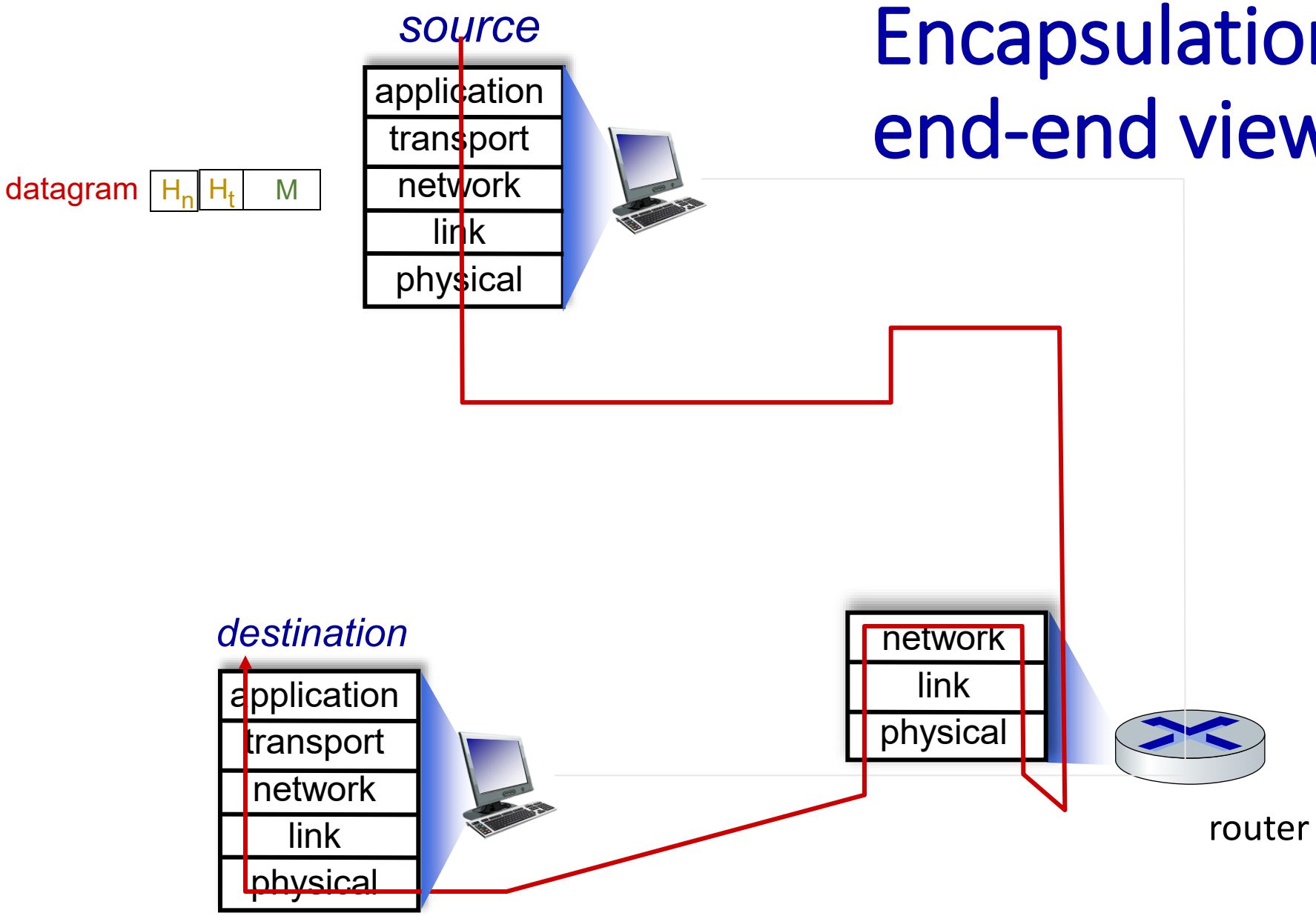
Encapsulation: an end-end view



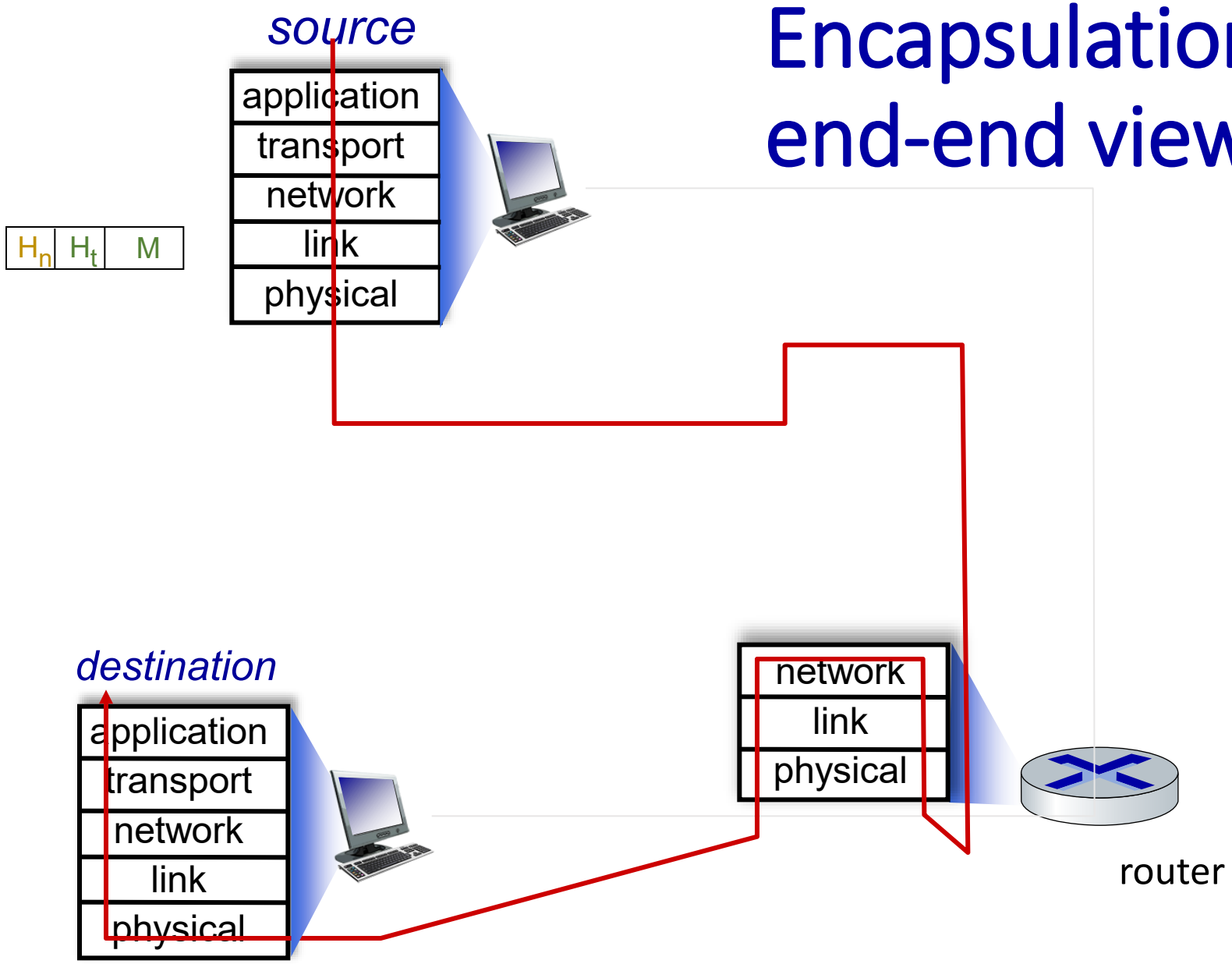
Encapsulation: an end-end view



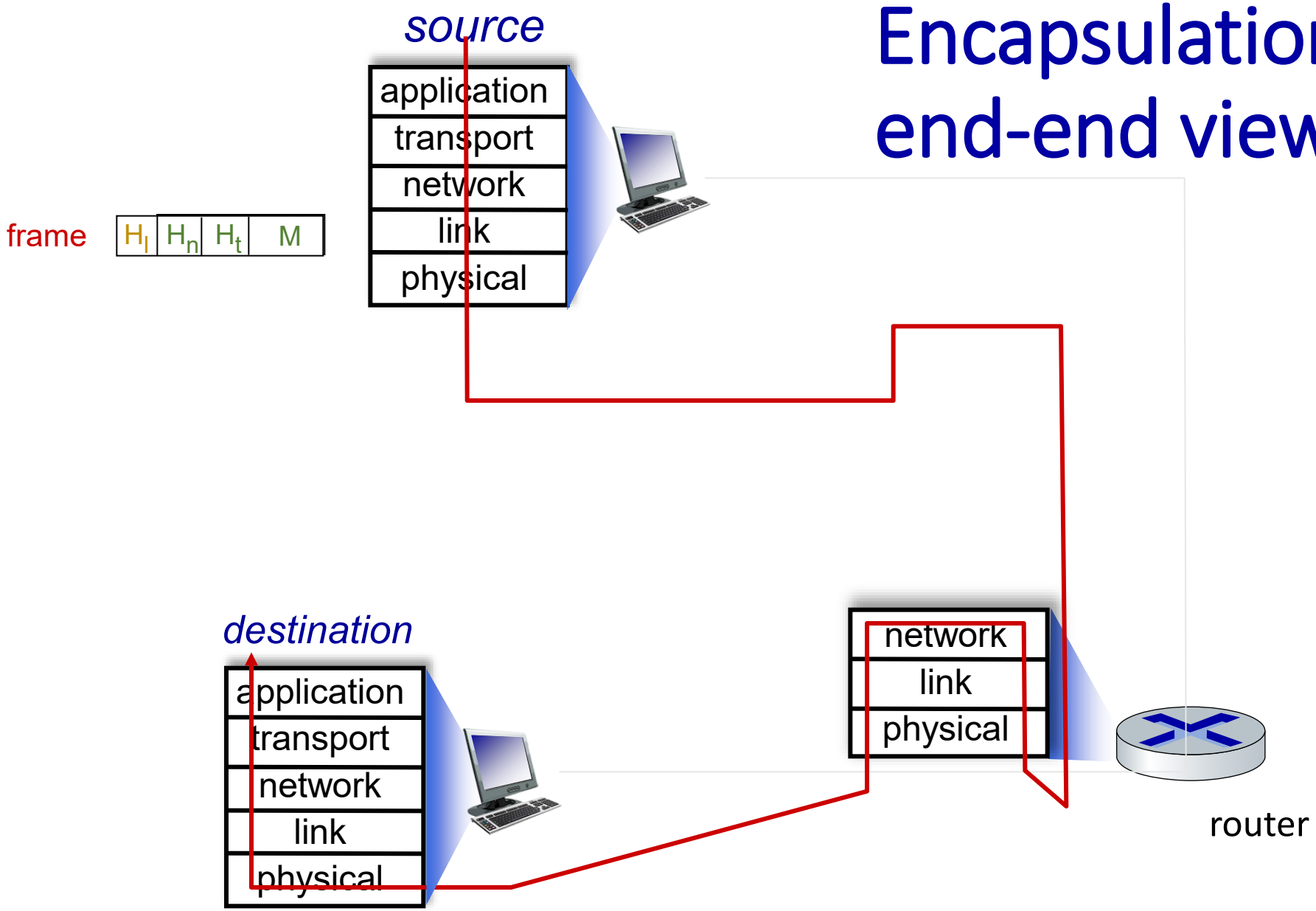
Encapsulation: an end-end view



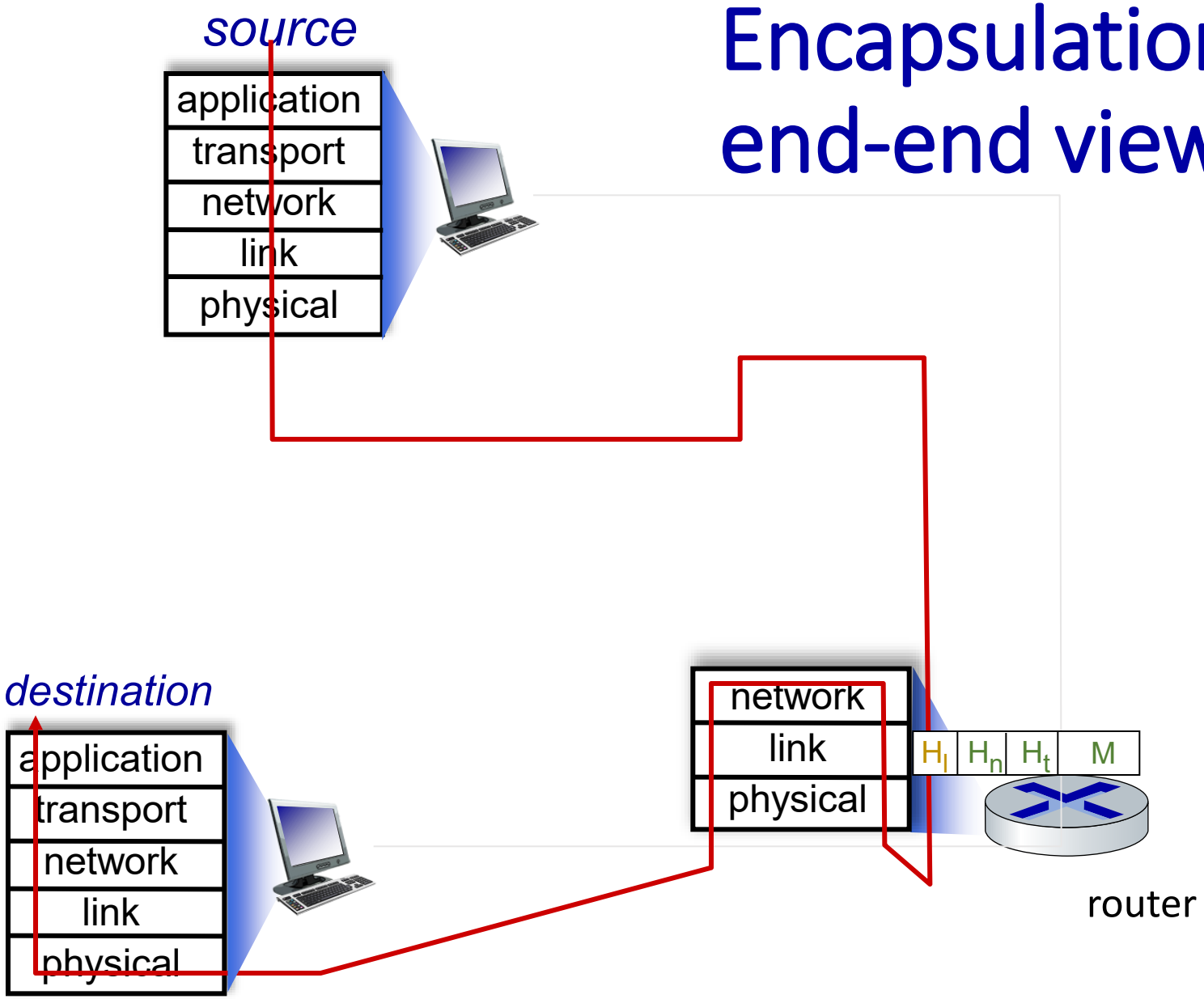
Encapsulation: an end-end view



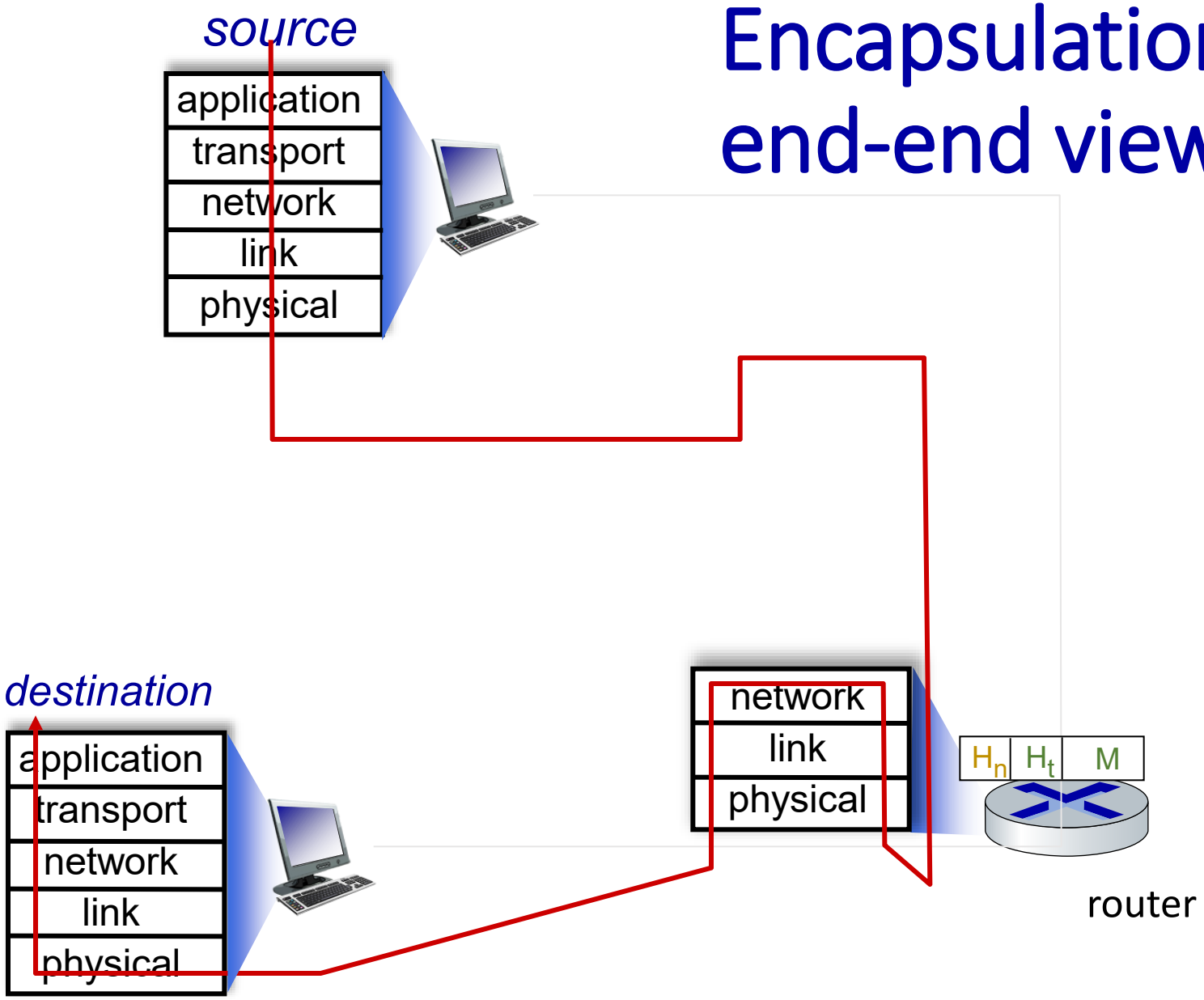
Encapsulation: an end-end view



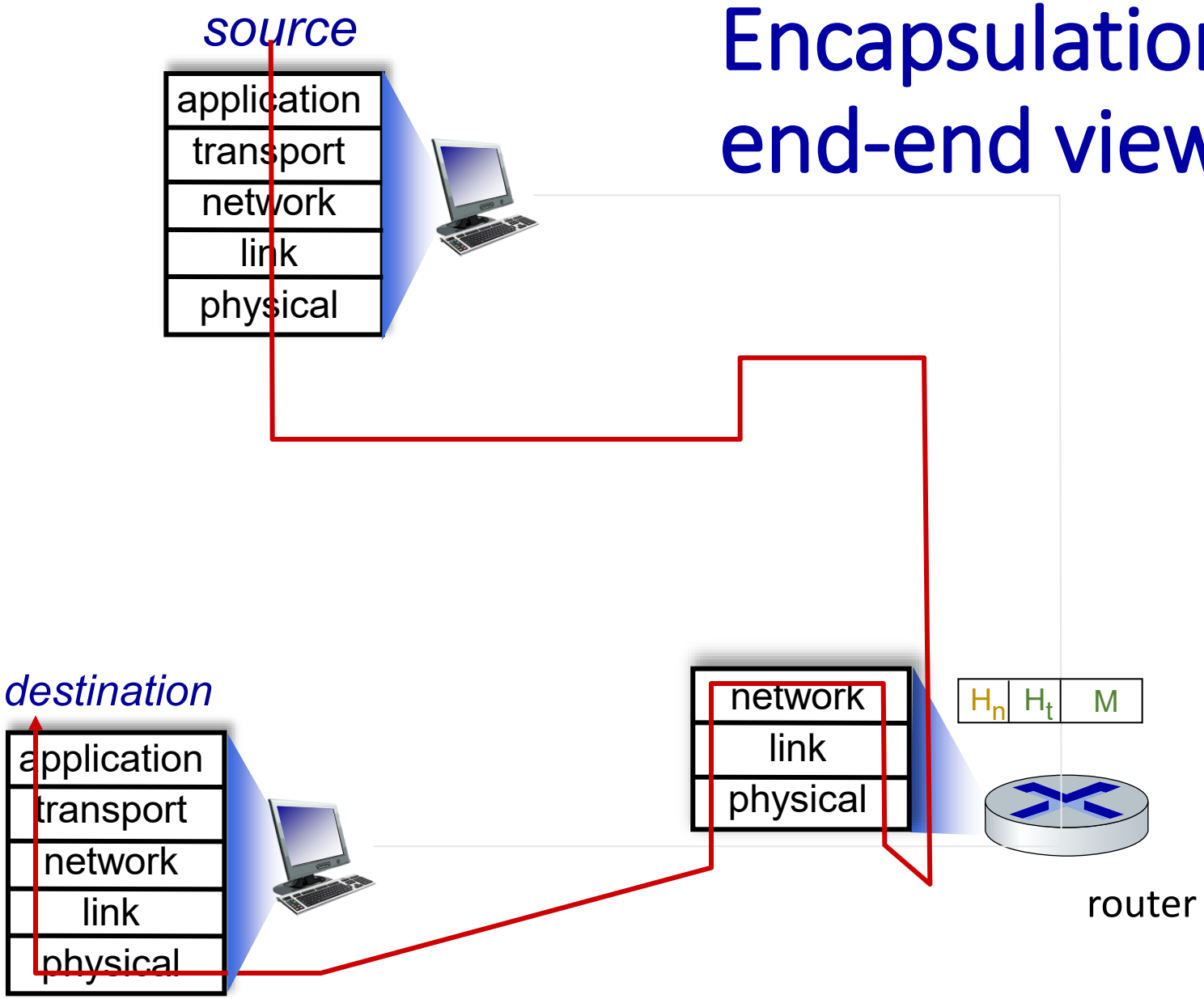
Encapsulation: an end-end view



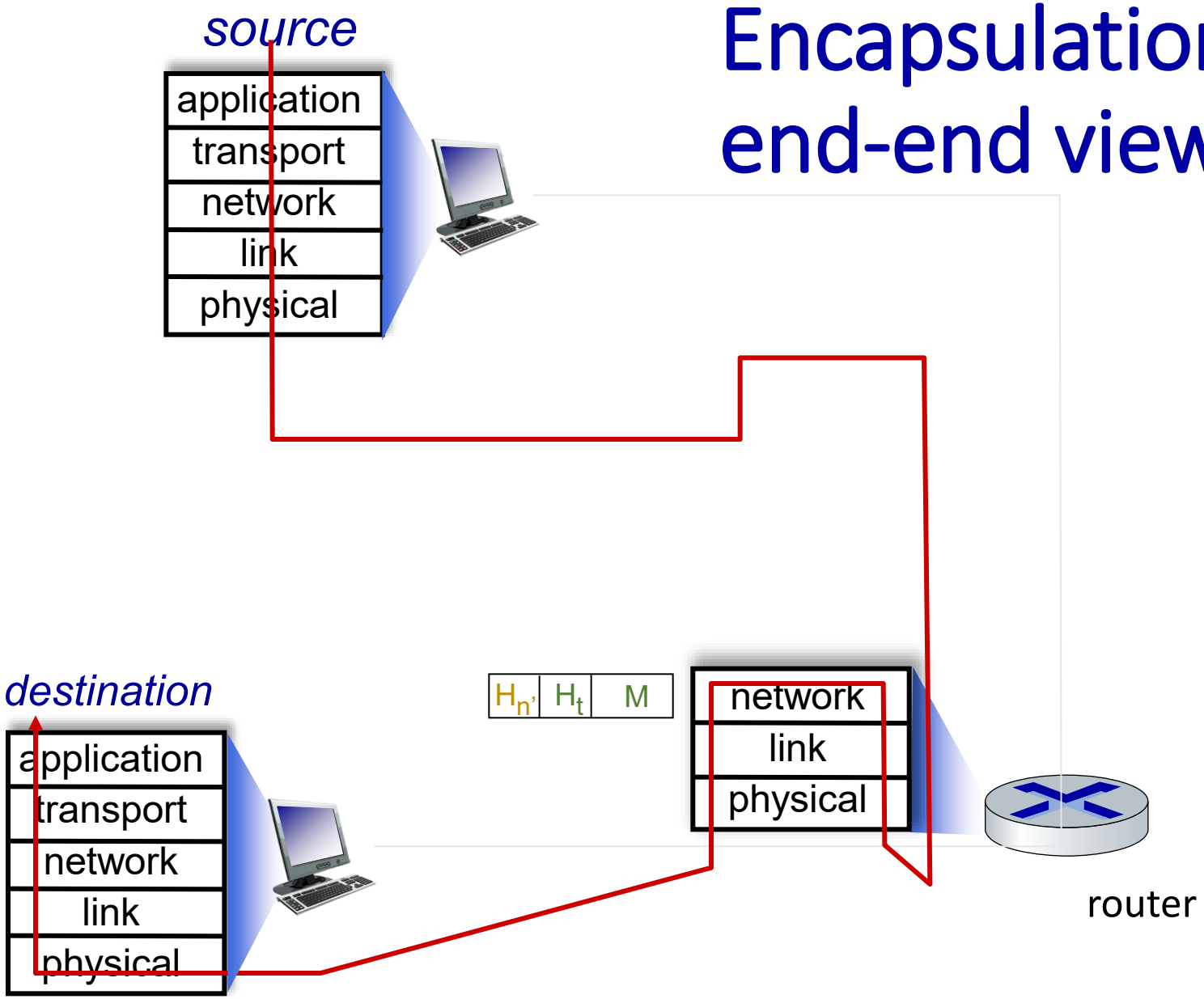
Encapsulation: an end-end view



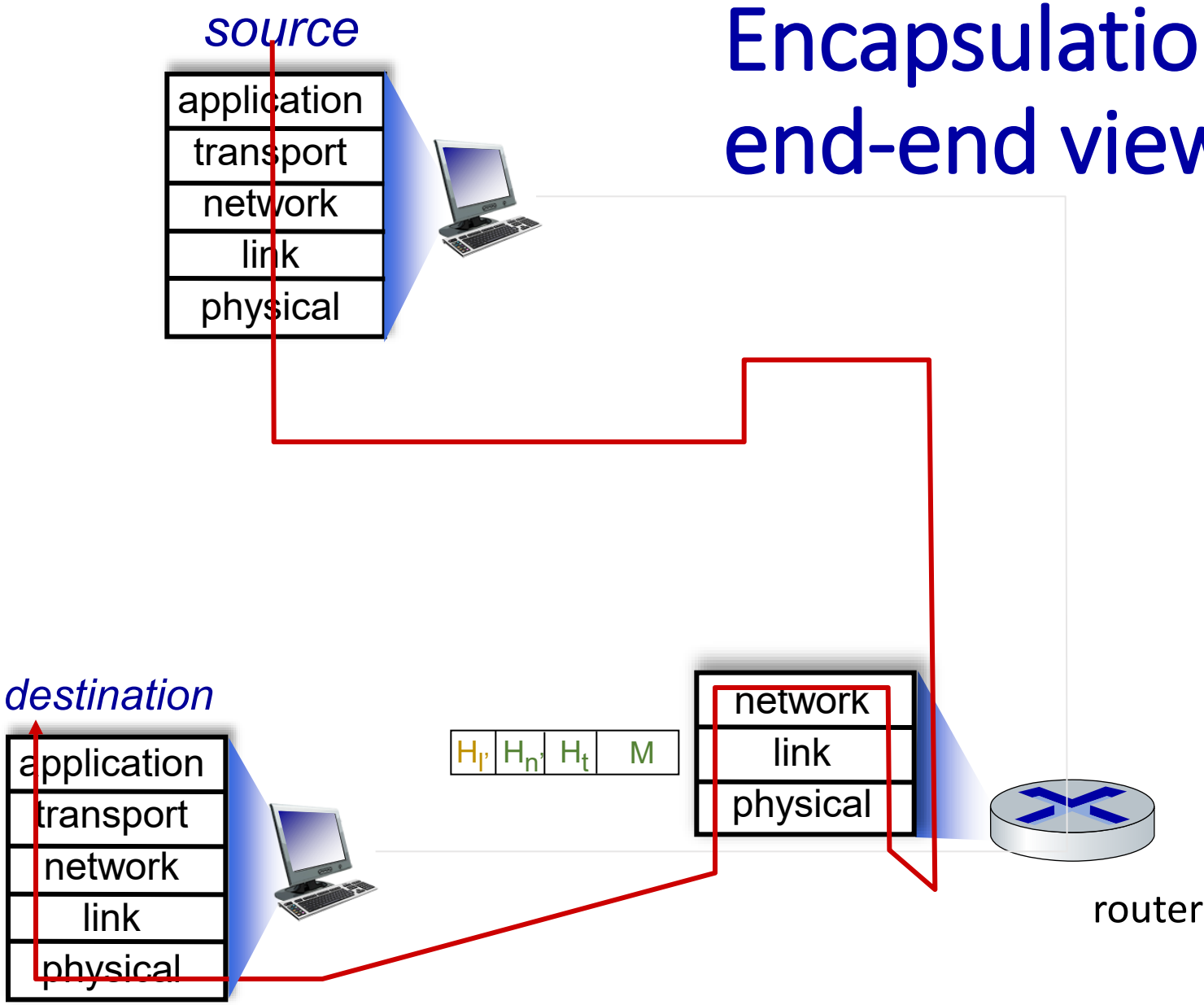
Encapsulation: an end-end view



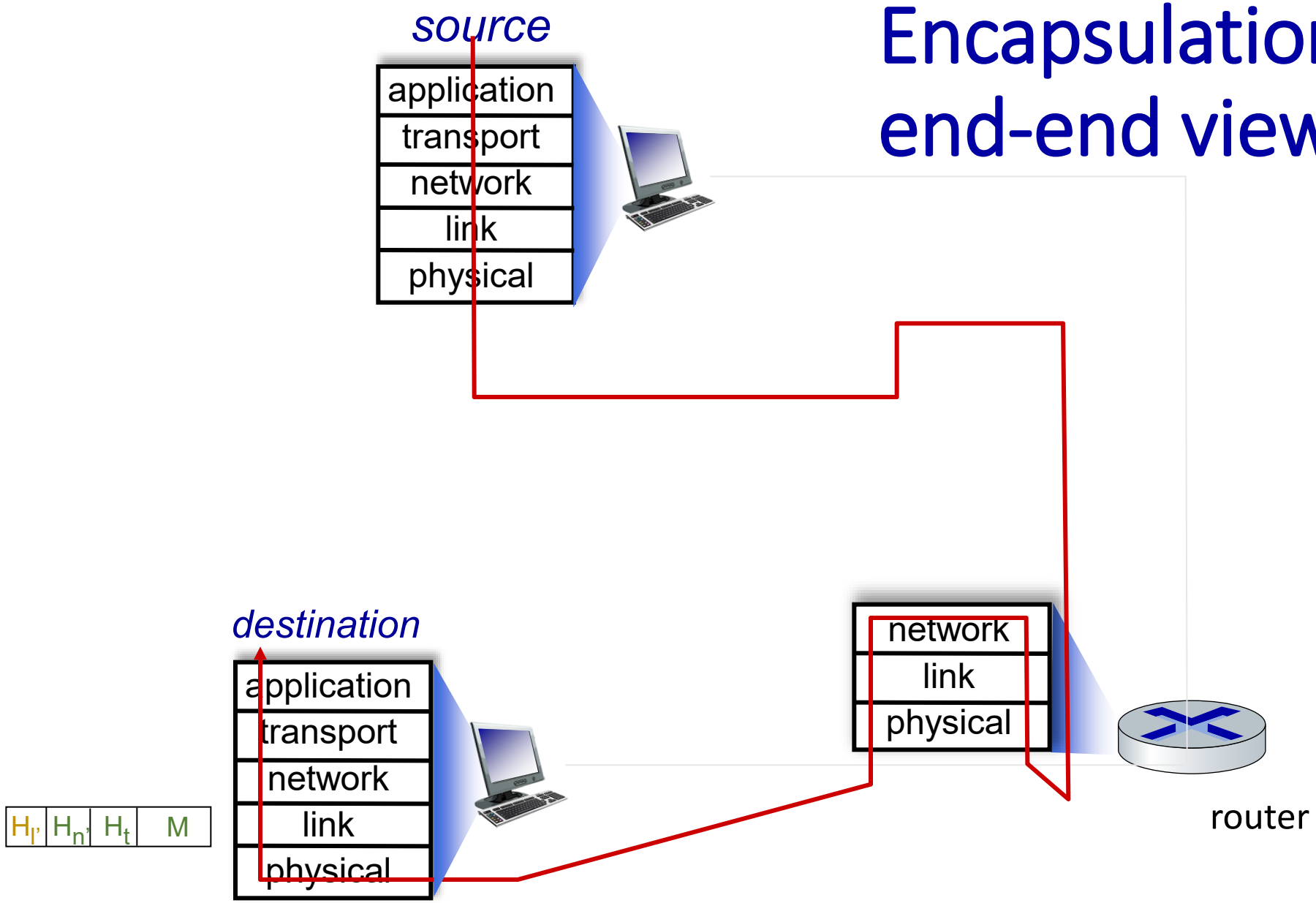
Encapsulation: an end-end view



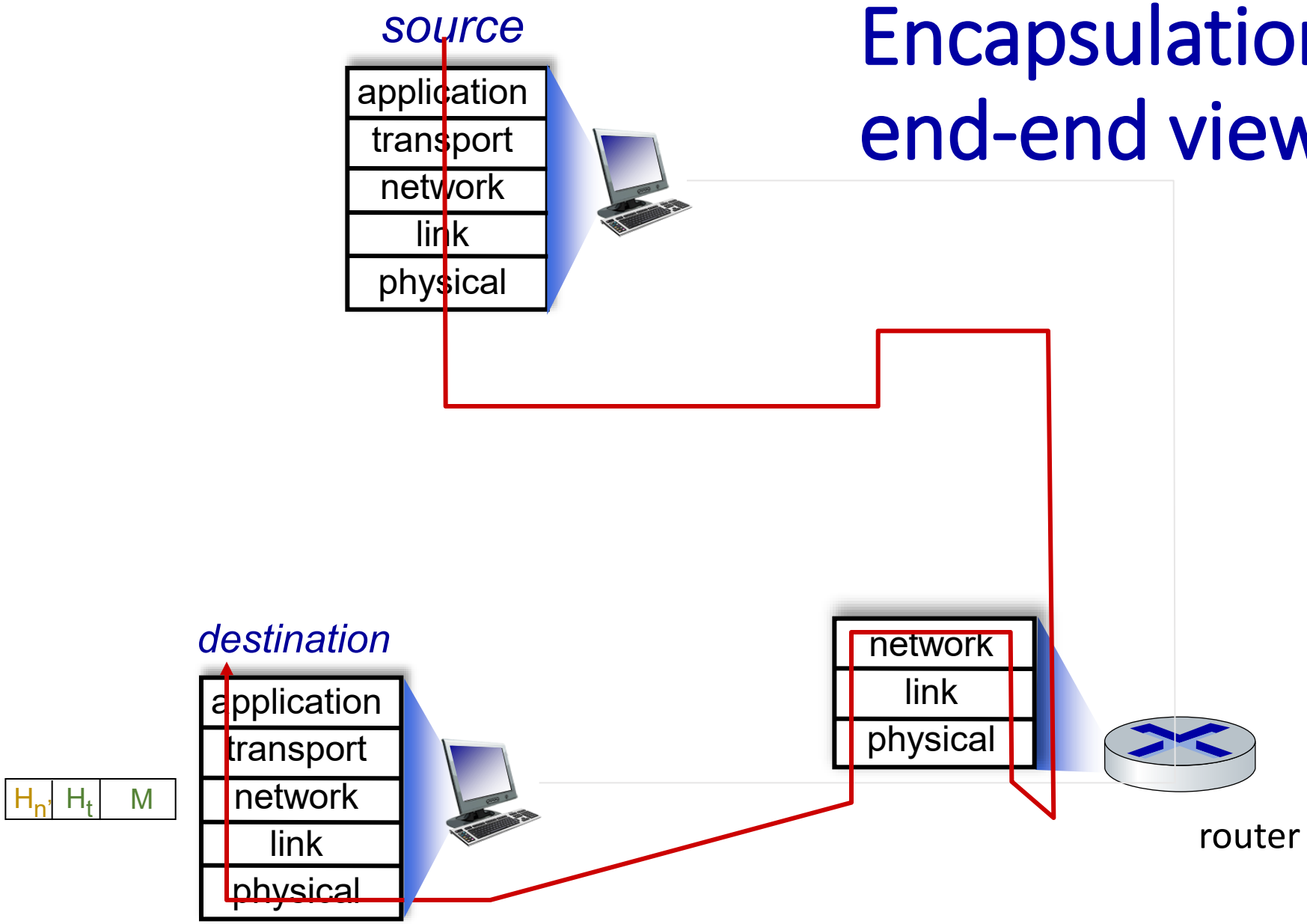
Encapsulation: an end-end view



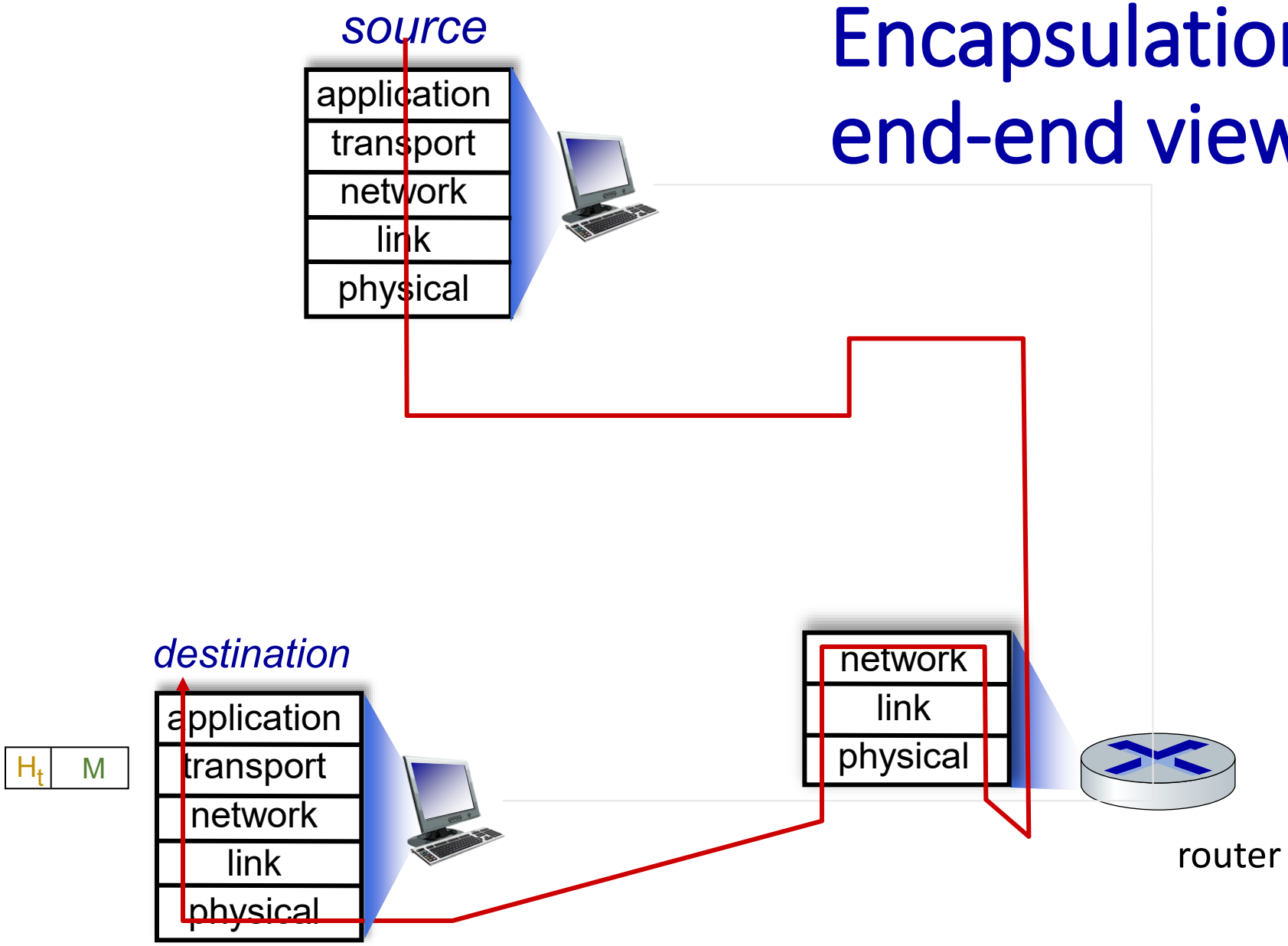
Encapsulation: an end-end view



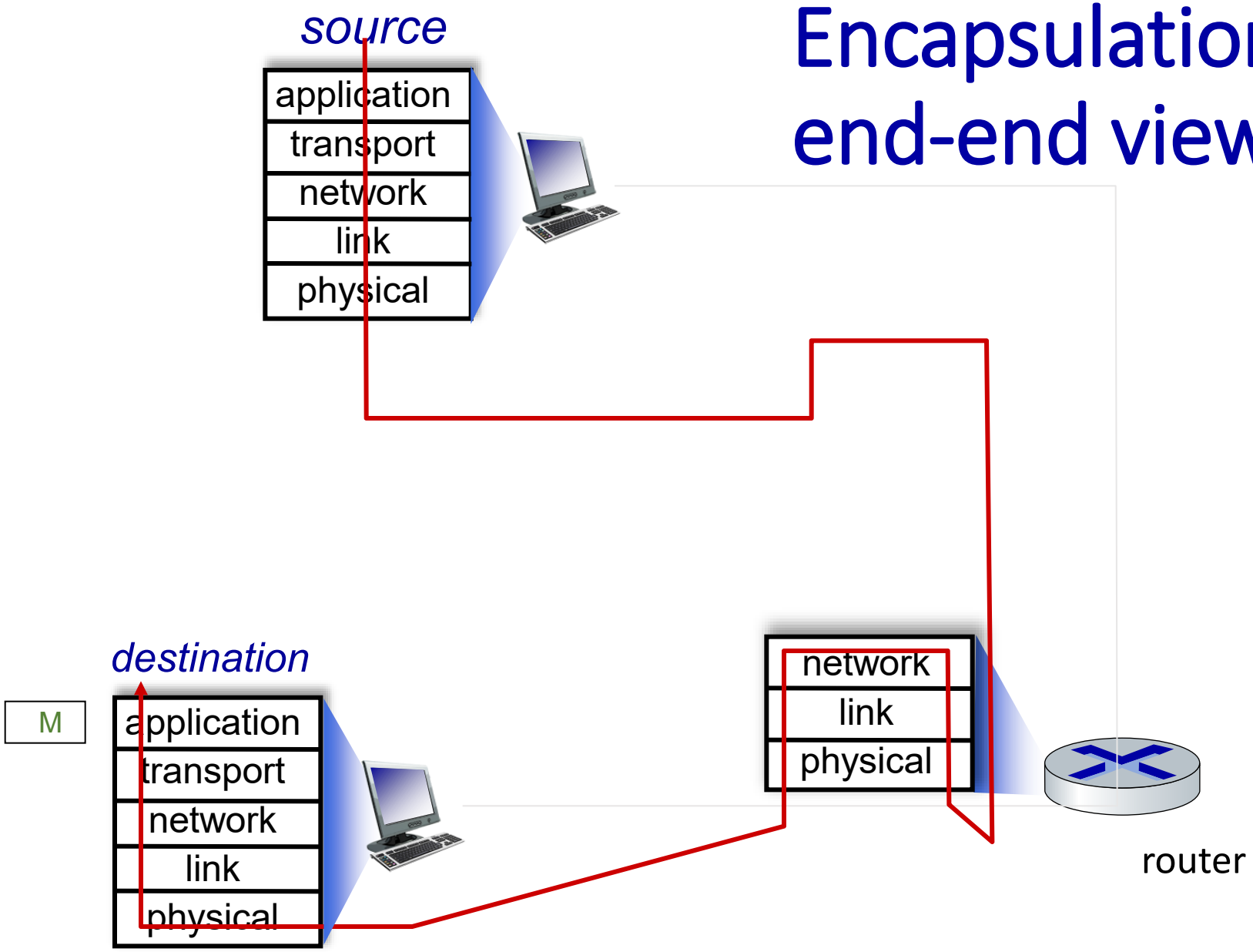
Encapsulation: an end-end view



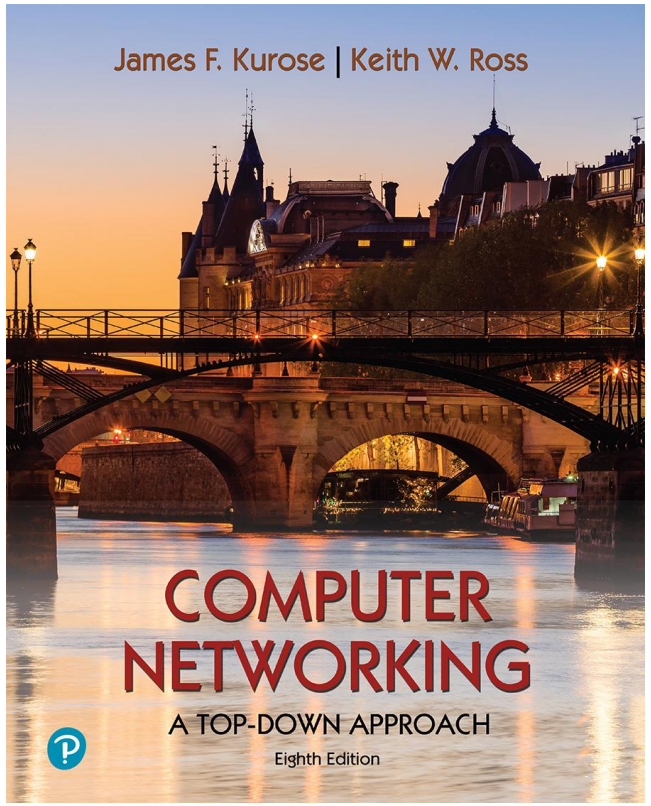
Encapsulation: an end-end view



Encapsulation: an end-end view



Chapter 2
Application Layer



A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top-Down Approach

8th edition
Jim Kurose, Keith Ross
Pearson, 2020

DNS: Domain Name System

people: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

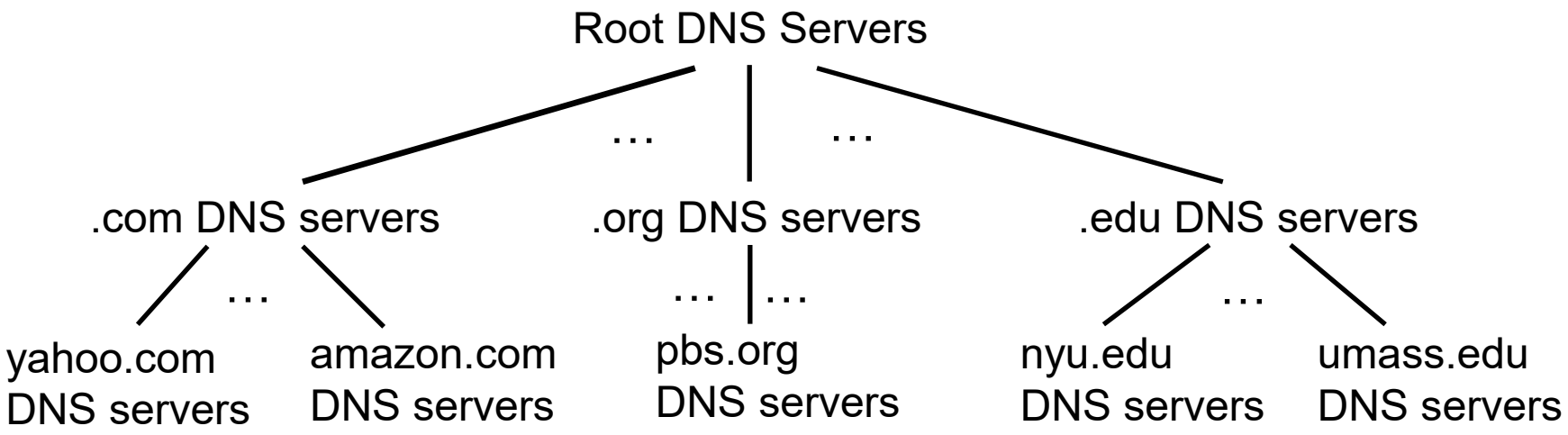
- IP address (32 bit) - used for addressing datagrams
- "name", e.g., cs.umass.edu - used by humans

Q: how to map between IP address and name, and vice versa?

Domain Name System (DNS):

- *application-layer protocol:* hosts and DNS servers communicate to *resolve* names (address/name translation)
  - *note:* core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"
- implemented as a hierarchy of many *DNS servers* (or *name servers*)

DNS: a distributed, hierarchical system



Root

Top Level Domain

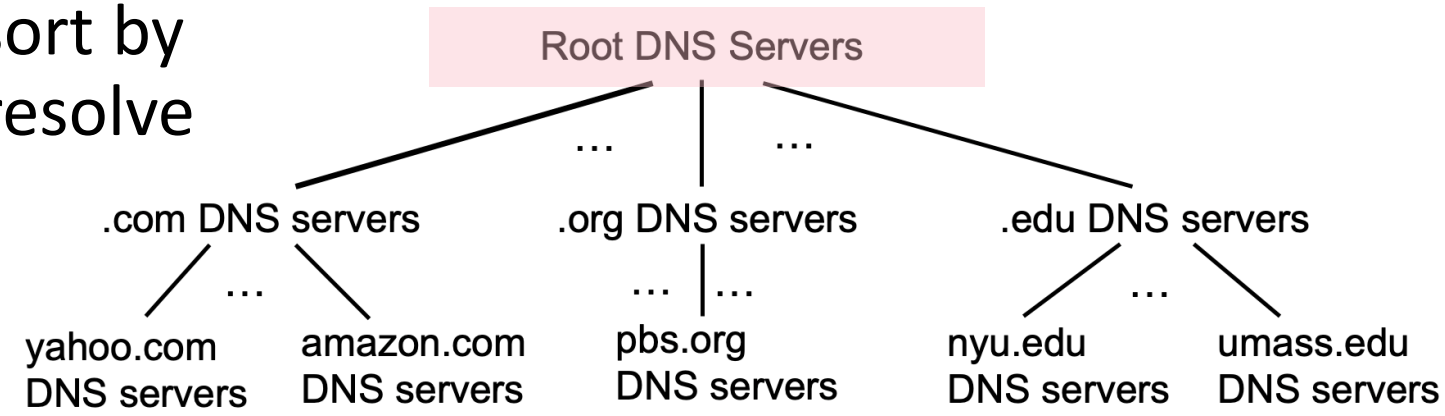
Authoritative

Client wants IP address for www.amazon.com; 1st approximation:

- client contacts root DNS server to find .com DNS server
- client contacts .com DNS server to get amazon.com DNS server
- client contacts amazon.com DNS server to get IP address for www.amazon.com

DNS: root servers

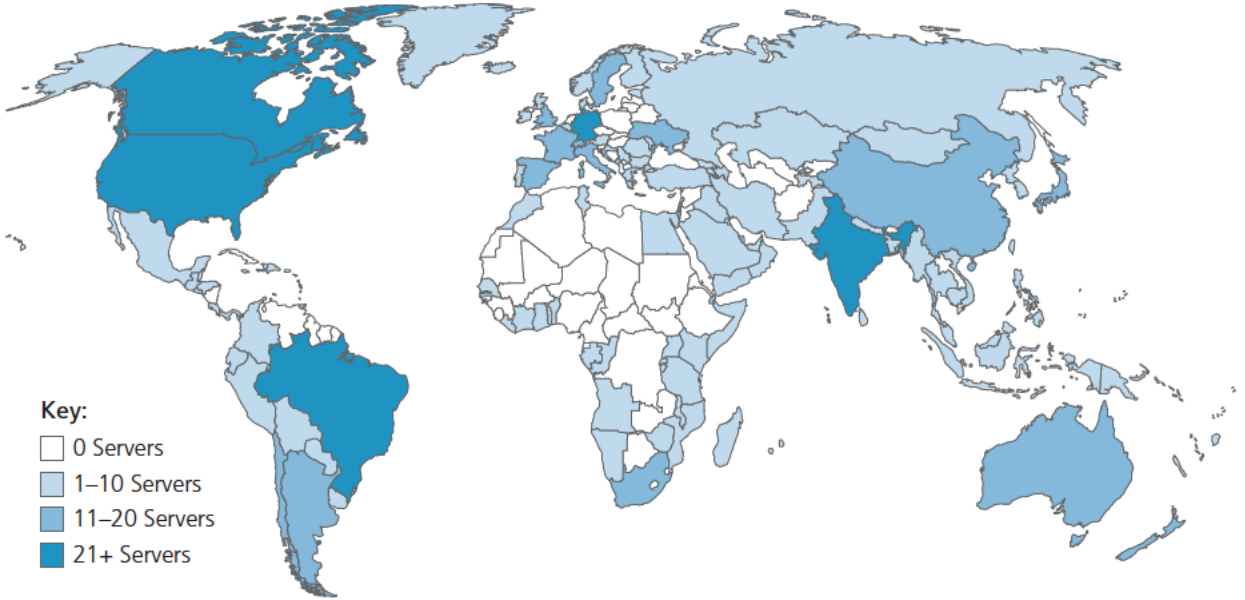
- official, contact-of-last-resort by DNS servers that can not resolve name



DNS: root servers

- official, contact-of-last-resort by DNS servers that can not resolve name
- *incredibly important* Internet function
  - Internet couldn't function without it!

13 root "servers" worldwide; each "server" replicated many times (~20 servers in Italy)

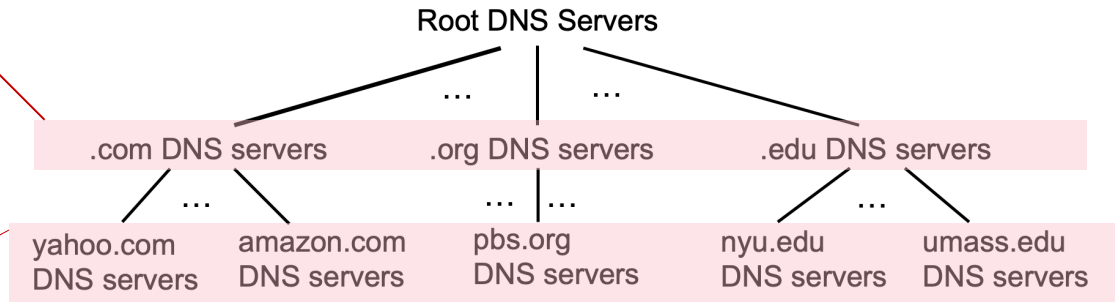


Check root name server replicas at https://root-servers.org/

Top-Level Domain, and authoritative servers

Top-Level Domain (TLD) servers:

- responsible for .com, .org, .net, .edu, .aero, .jobs, .museums, and all top-level country domains, e.g.: .cn, .uk, .fr, .ca, .jp



authoritative DNS servers:

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

Local DNS name servers

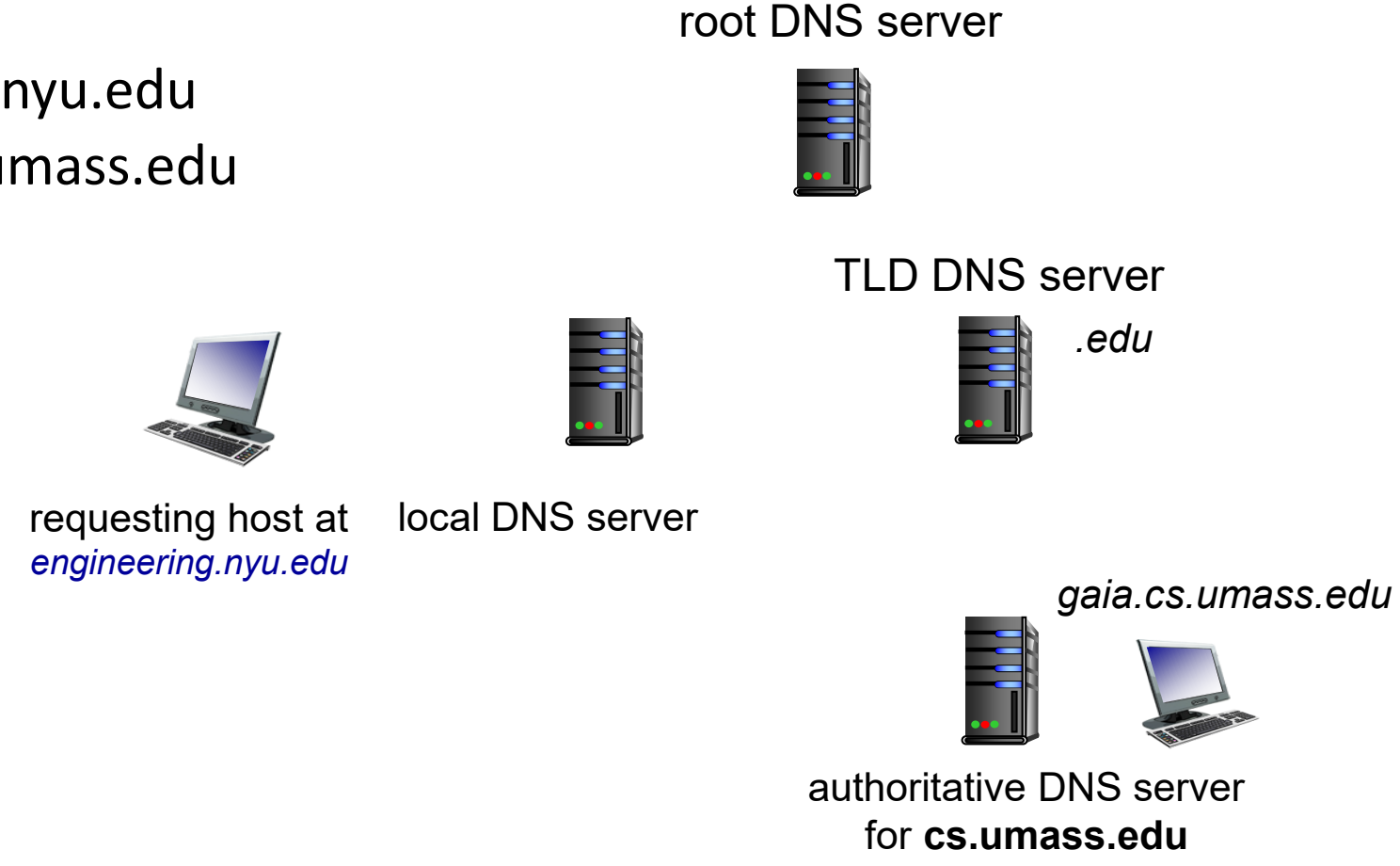
- when host makes DNS resolution request (i.e., query), it is sent to its *local* DNS server
  - local DNS server returns reply, answering:
    - from its **local cache** of recent hostname-to-IP-address translation pairs
    - forwarding request into DNS hierarchy for resolution
  - each ISP has local DNS server; to find yours:
    - MacOS: `% scutil --dns`
    - Windows: `> ipconfig /all`
  - also external local DNS servers can be used (e.g. Google's local DNS server)

DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"

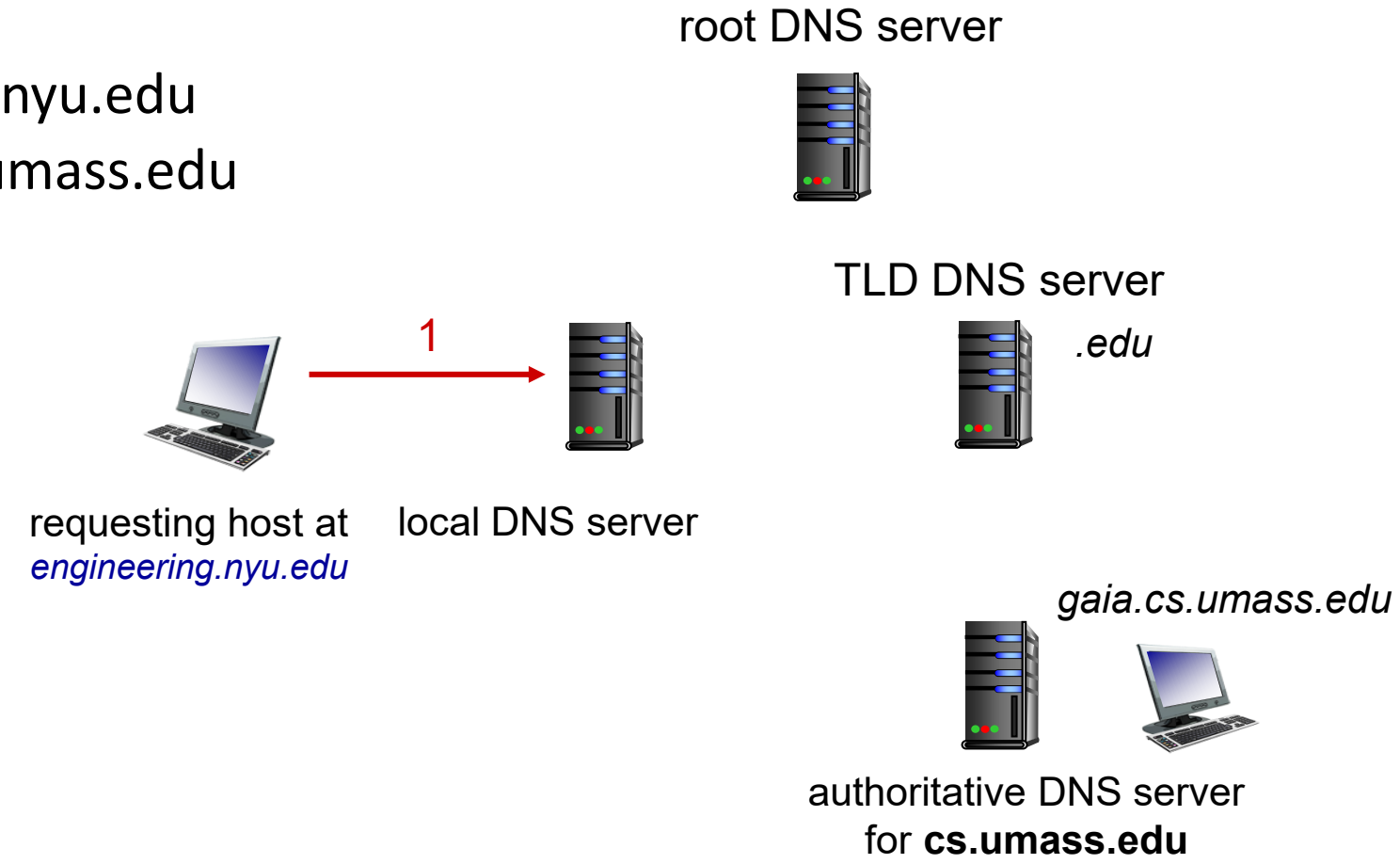


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"

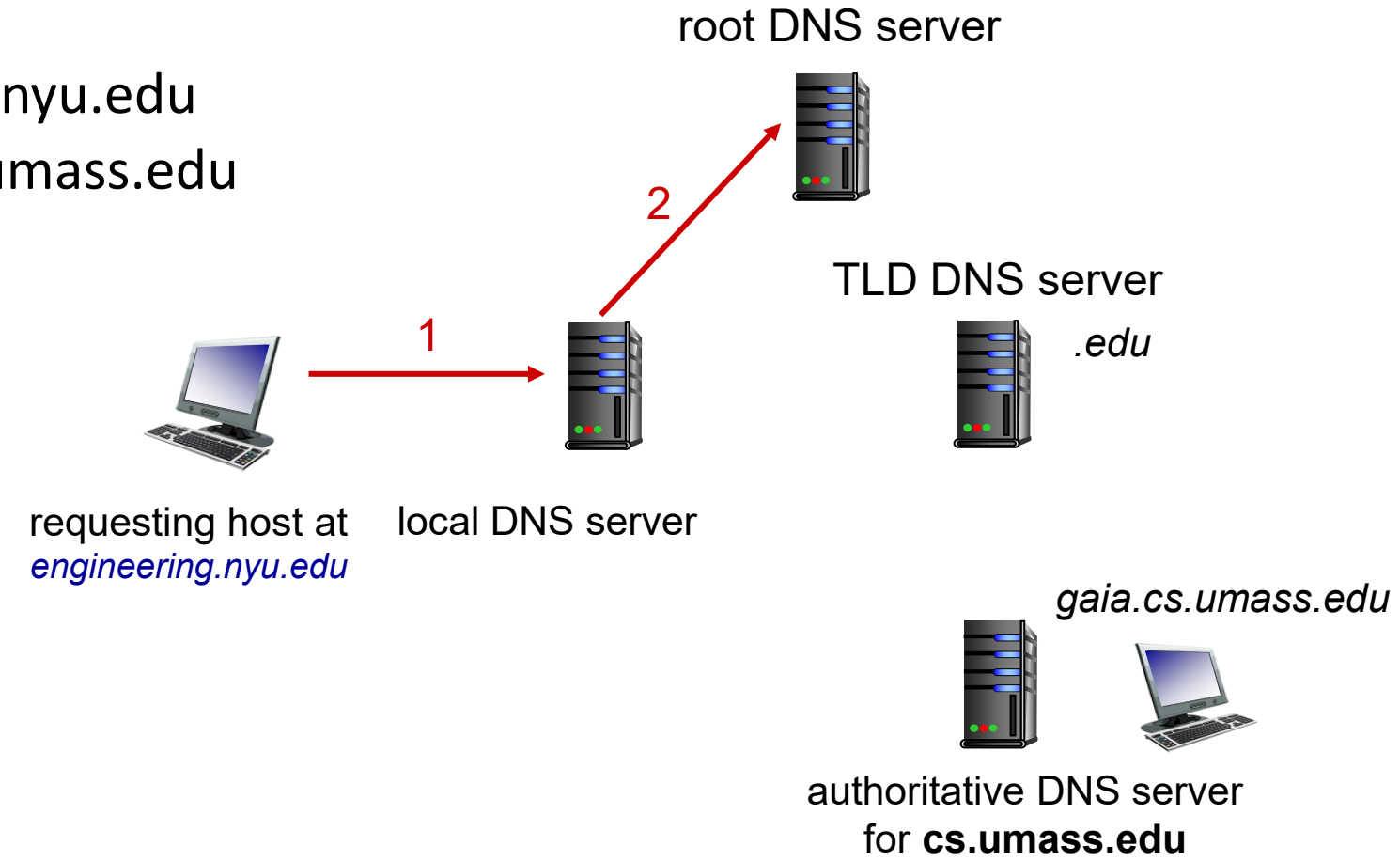


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"

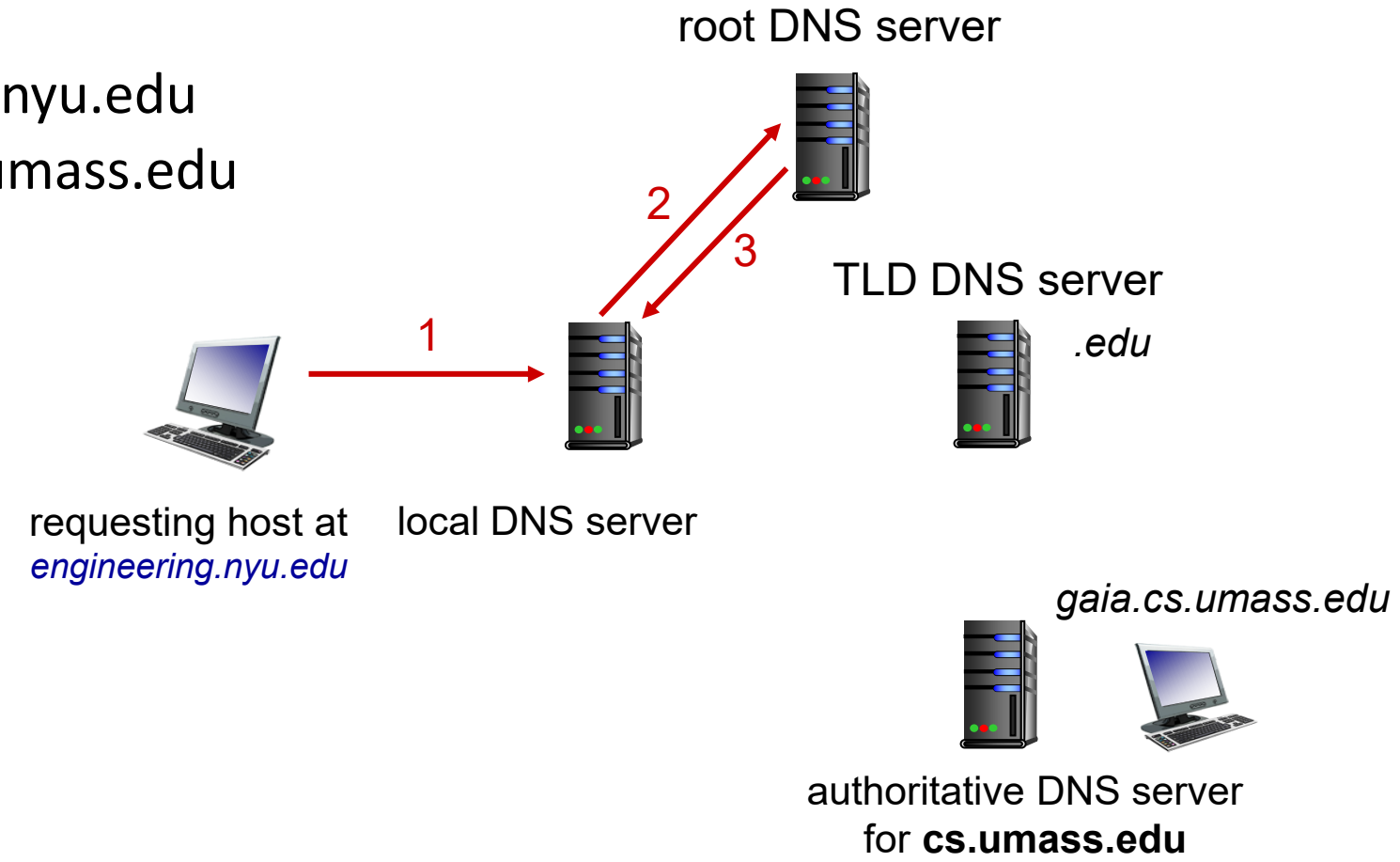


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"

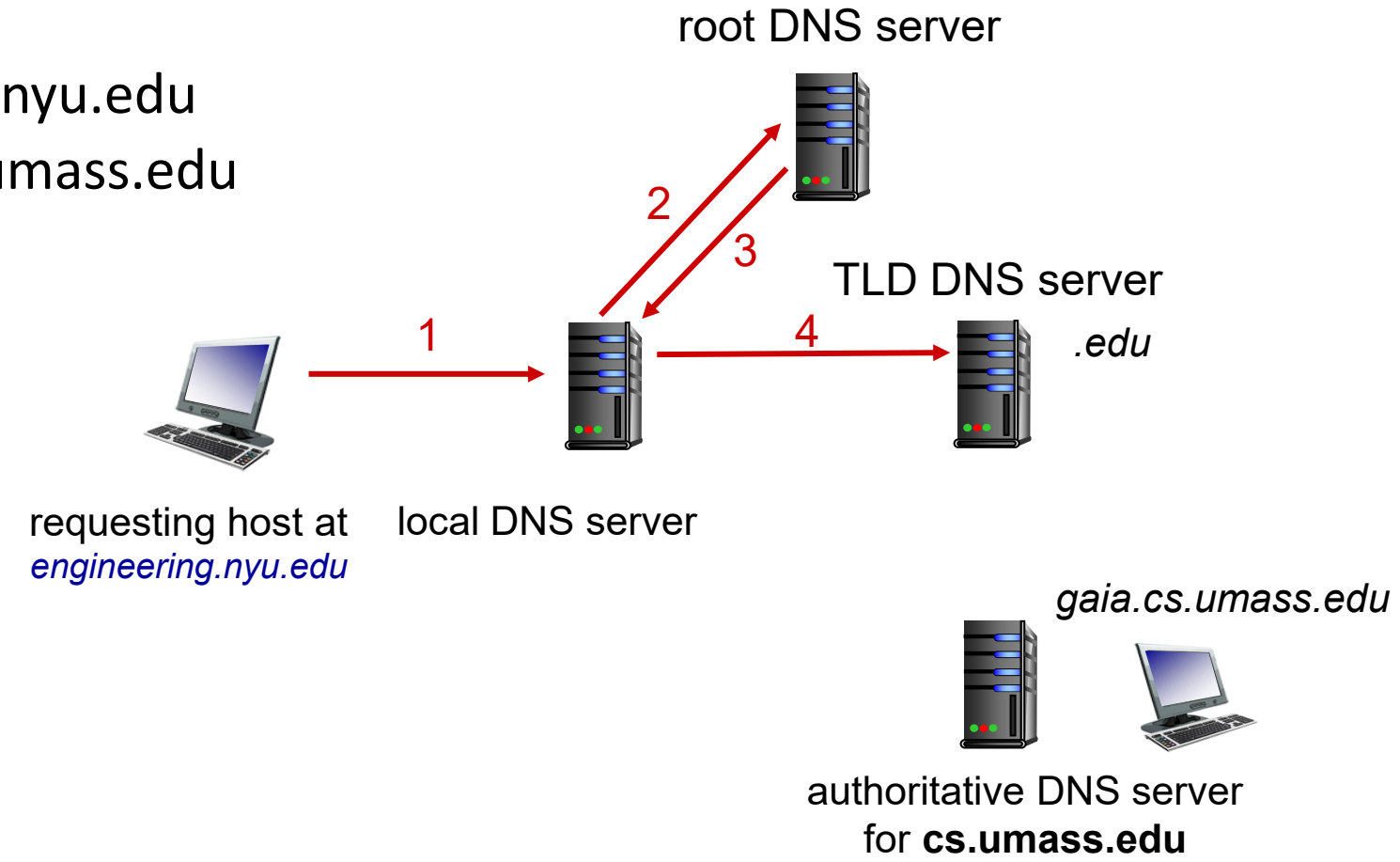


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"

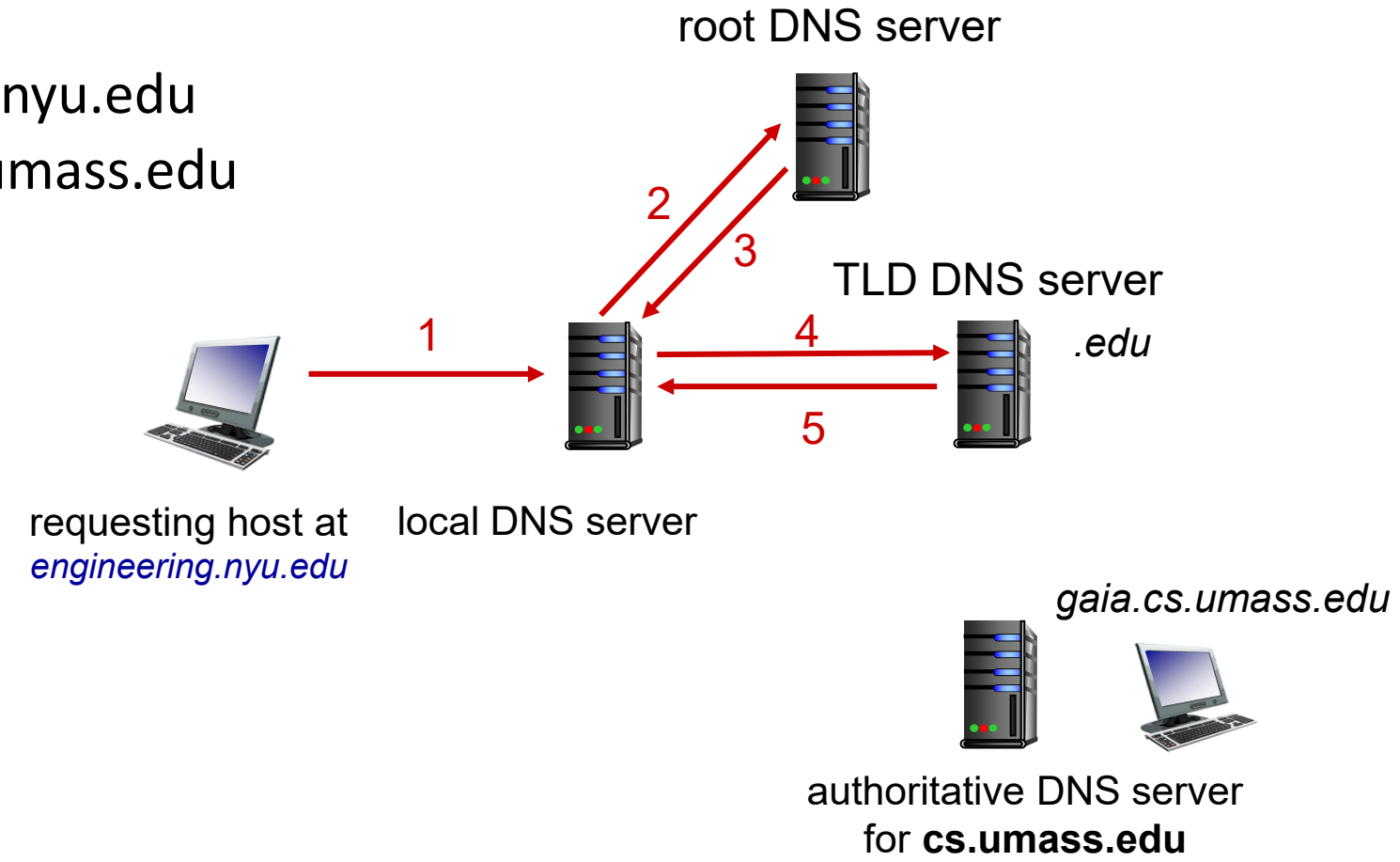


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"

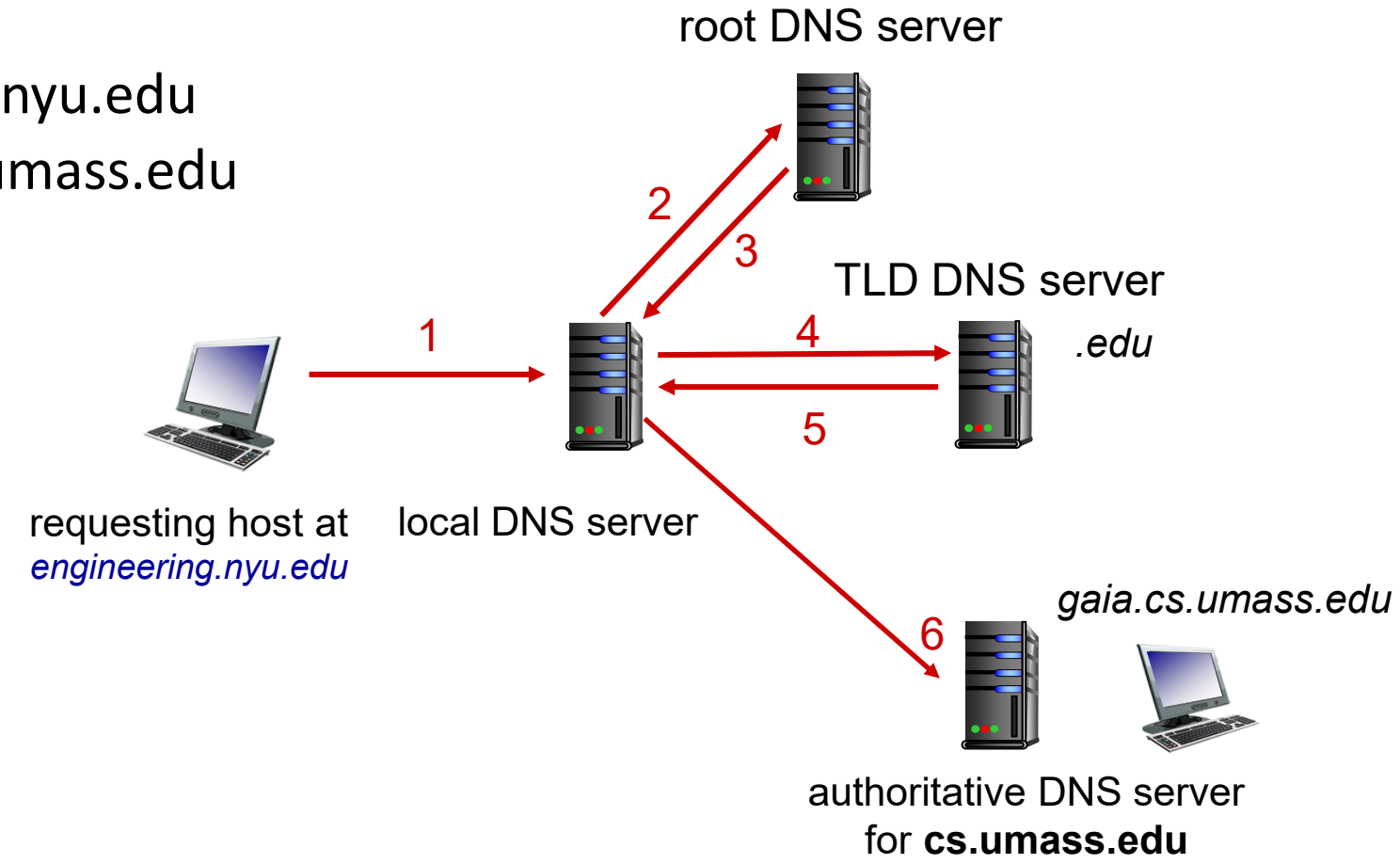


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"

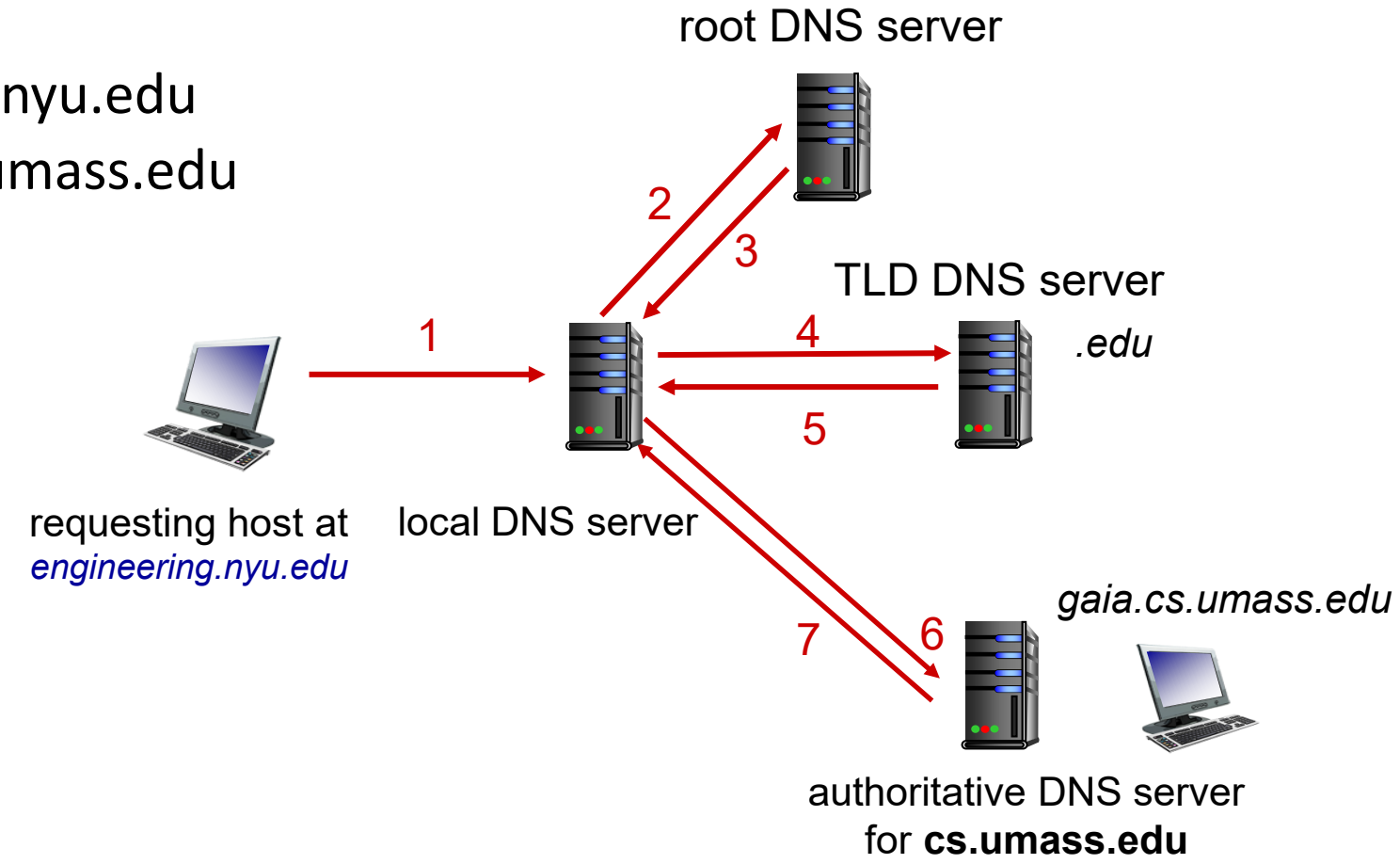


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"

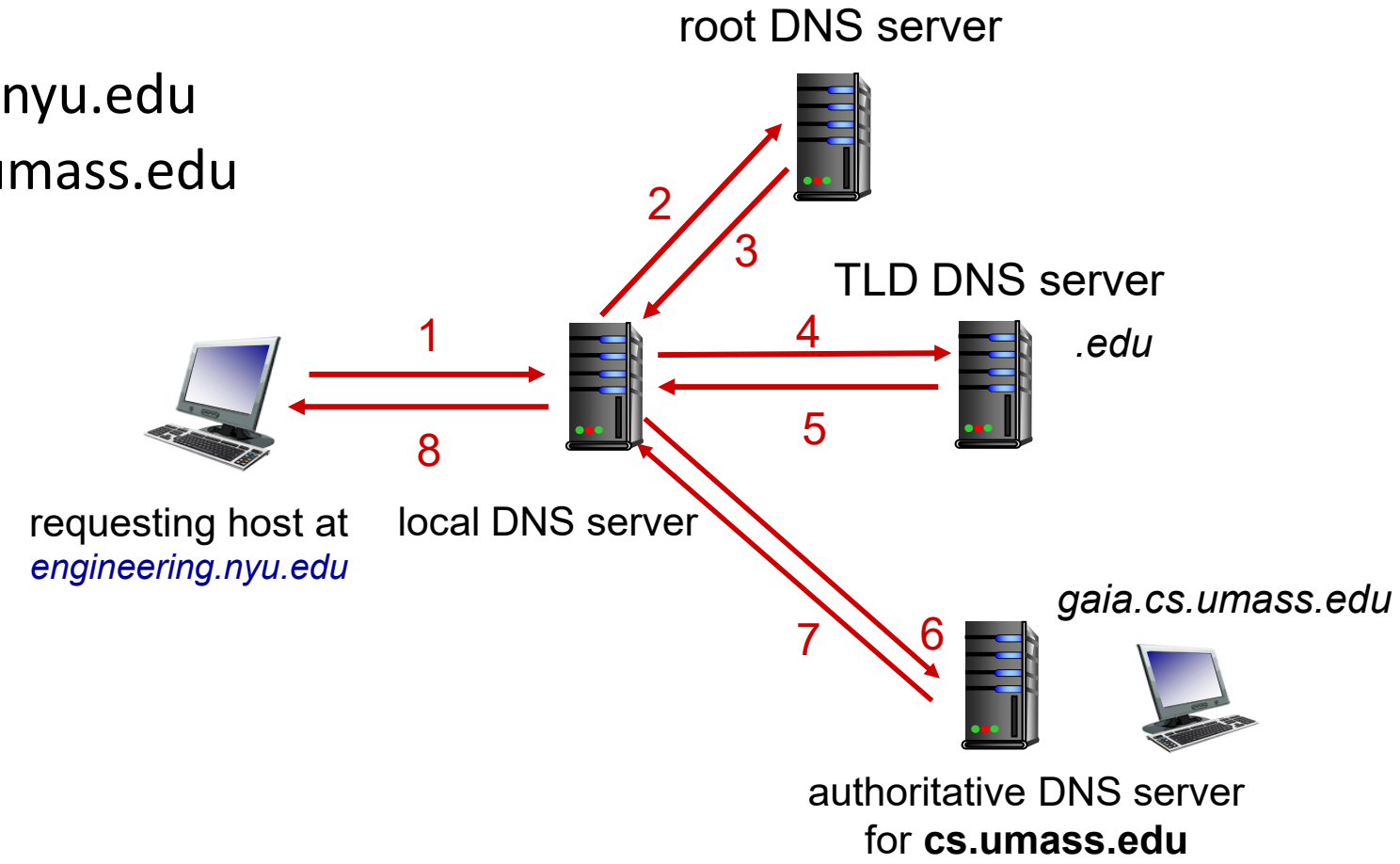


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies to local DNS server with IP address of server to contact
- "I don't know this name, but ask this server"



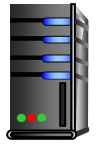
DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy

requesting host at *engineering.nyu.edu*



local DNS server

root DNS server

TLD DNS server
.edu

gaia.cs.umass.edu

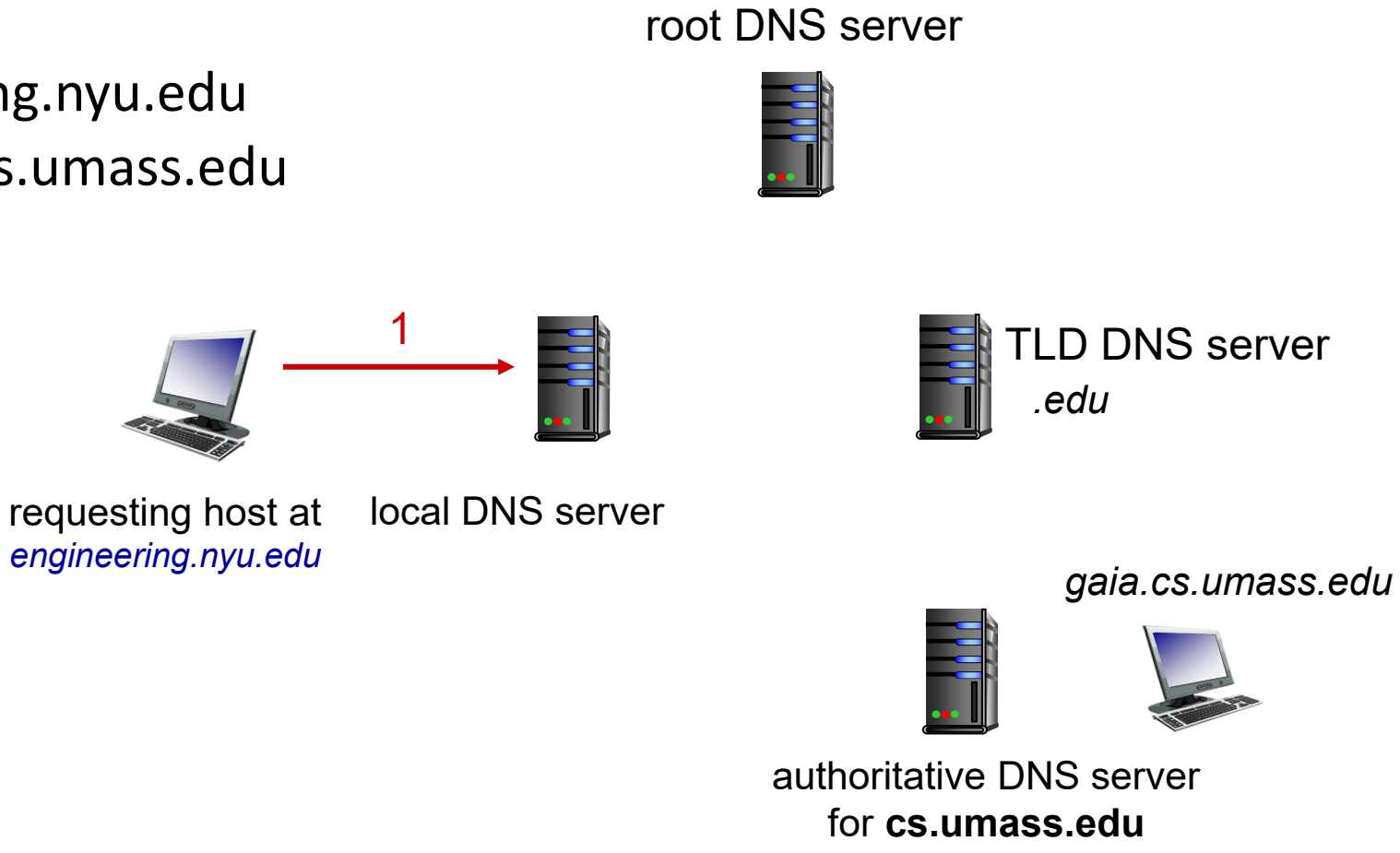
authoritative DNS server
for **cs.umass.edu**

DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy

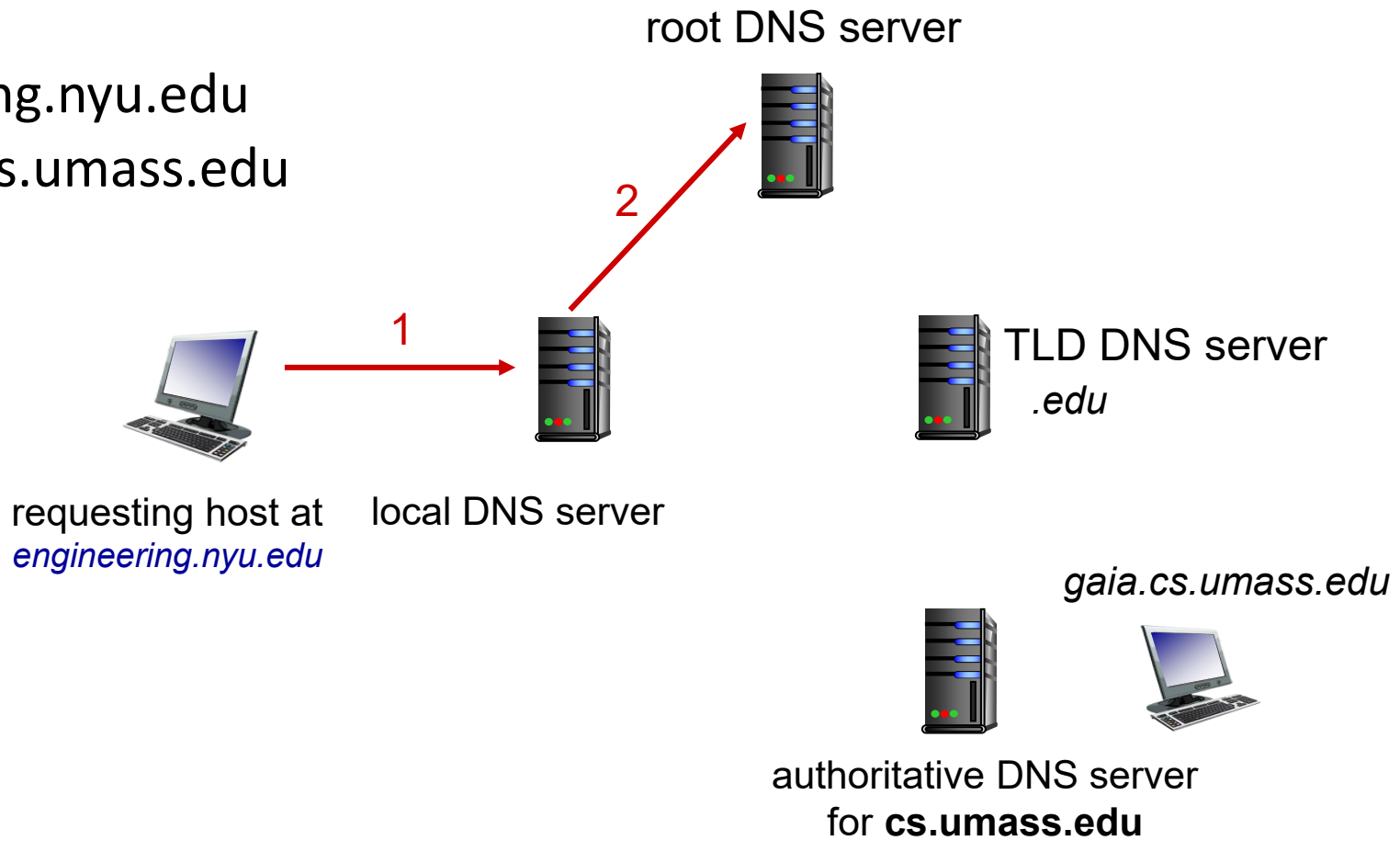


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy

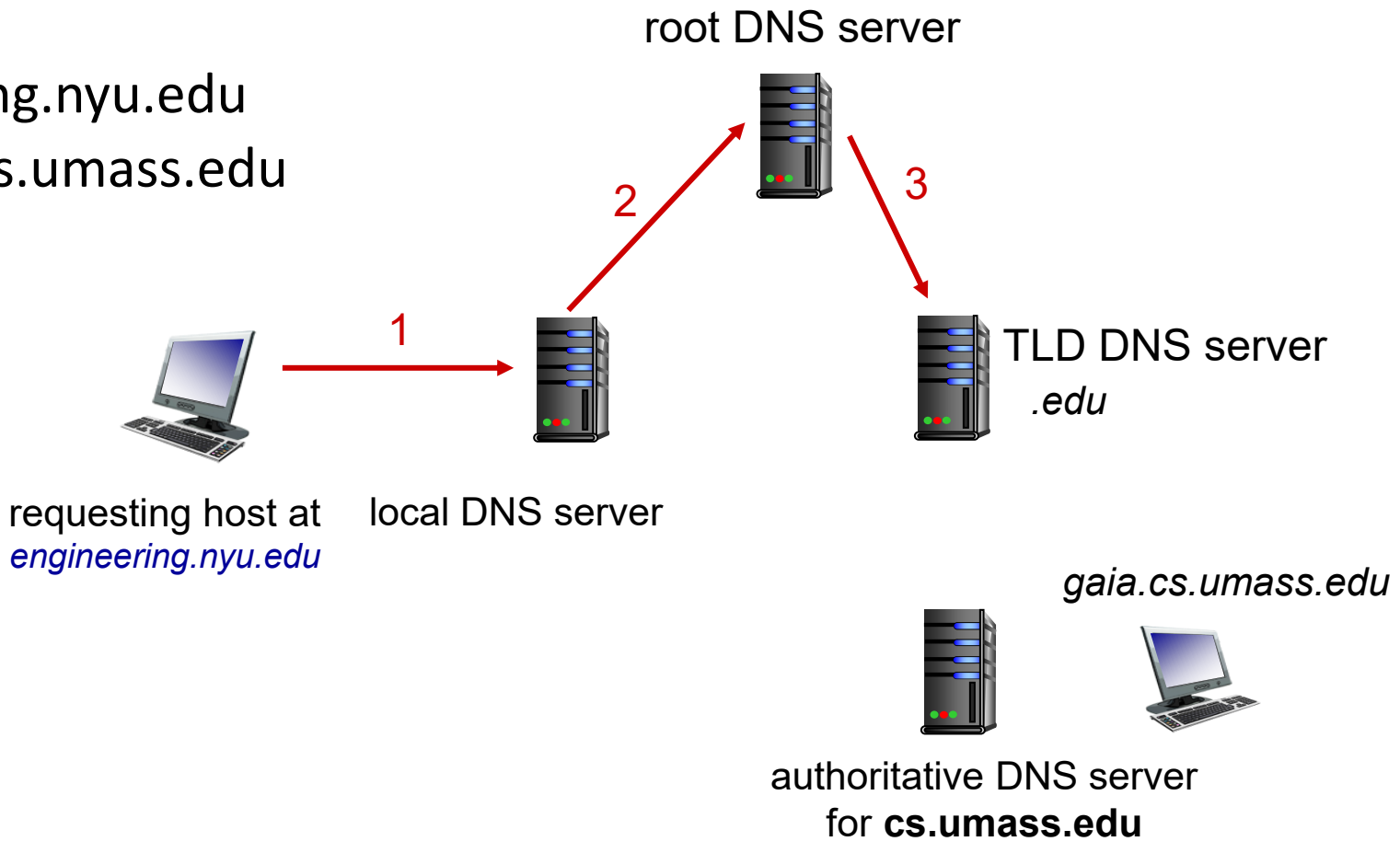


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy

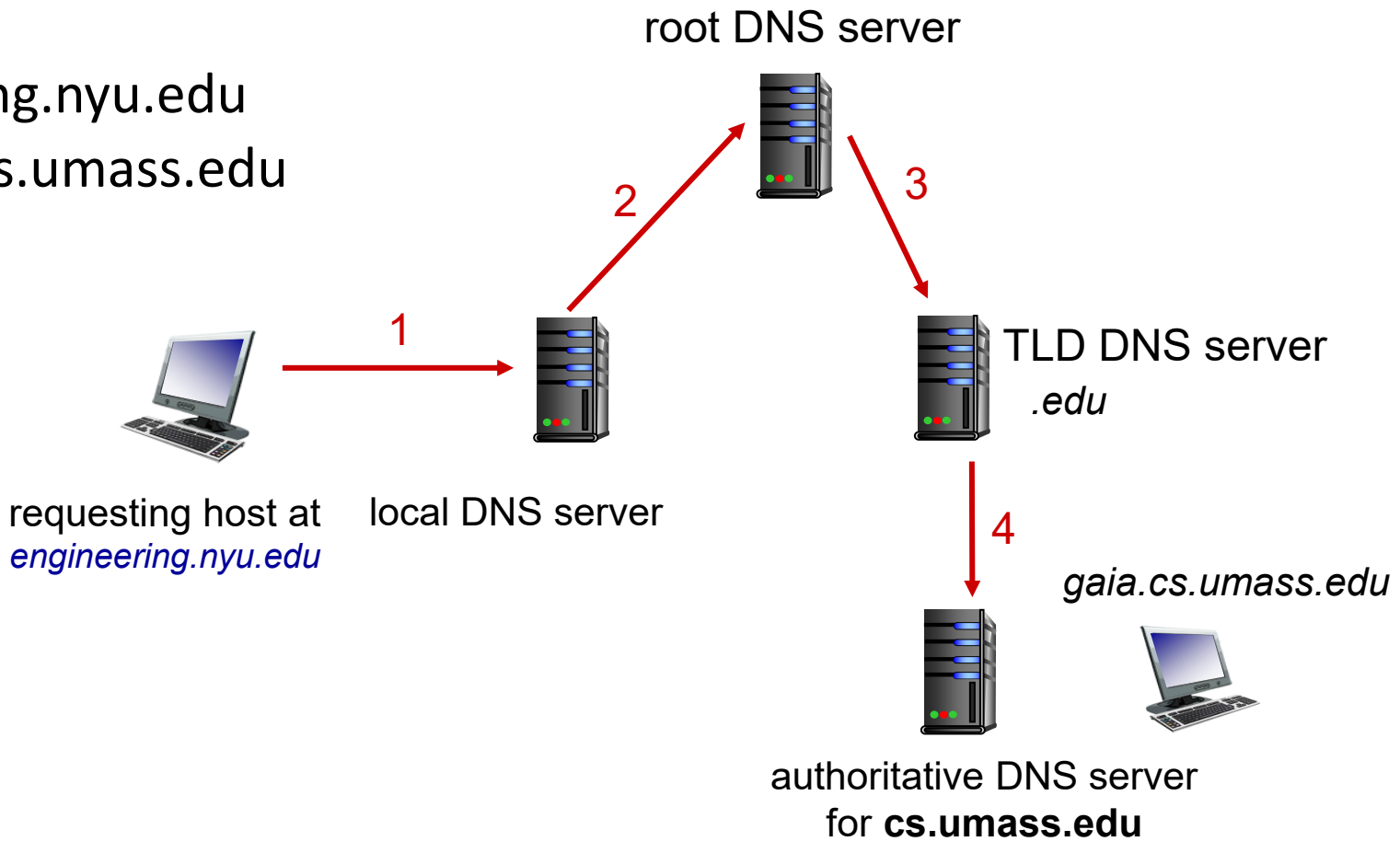


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy

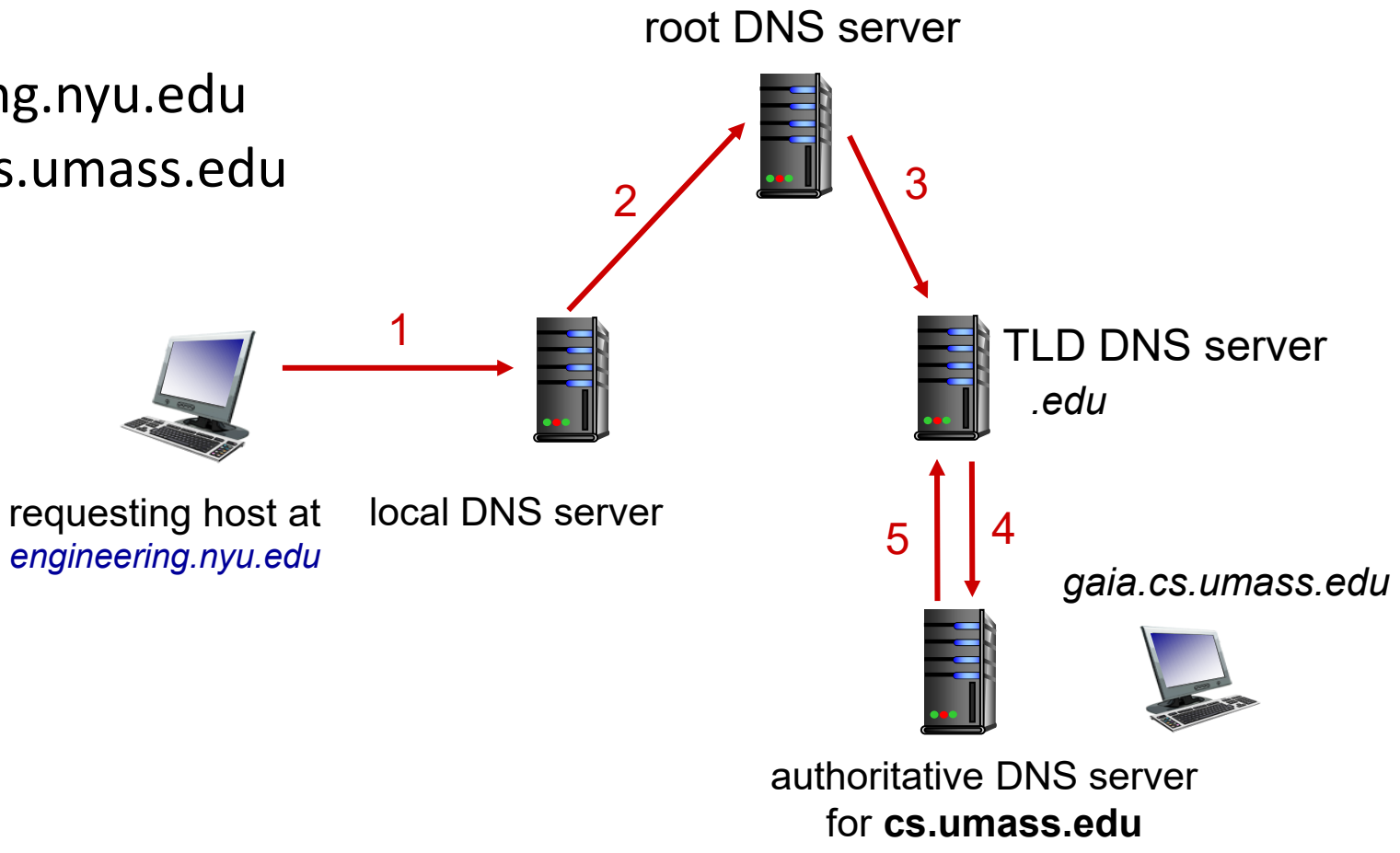


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy

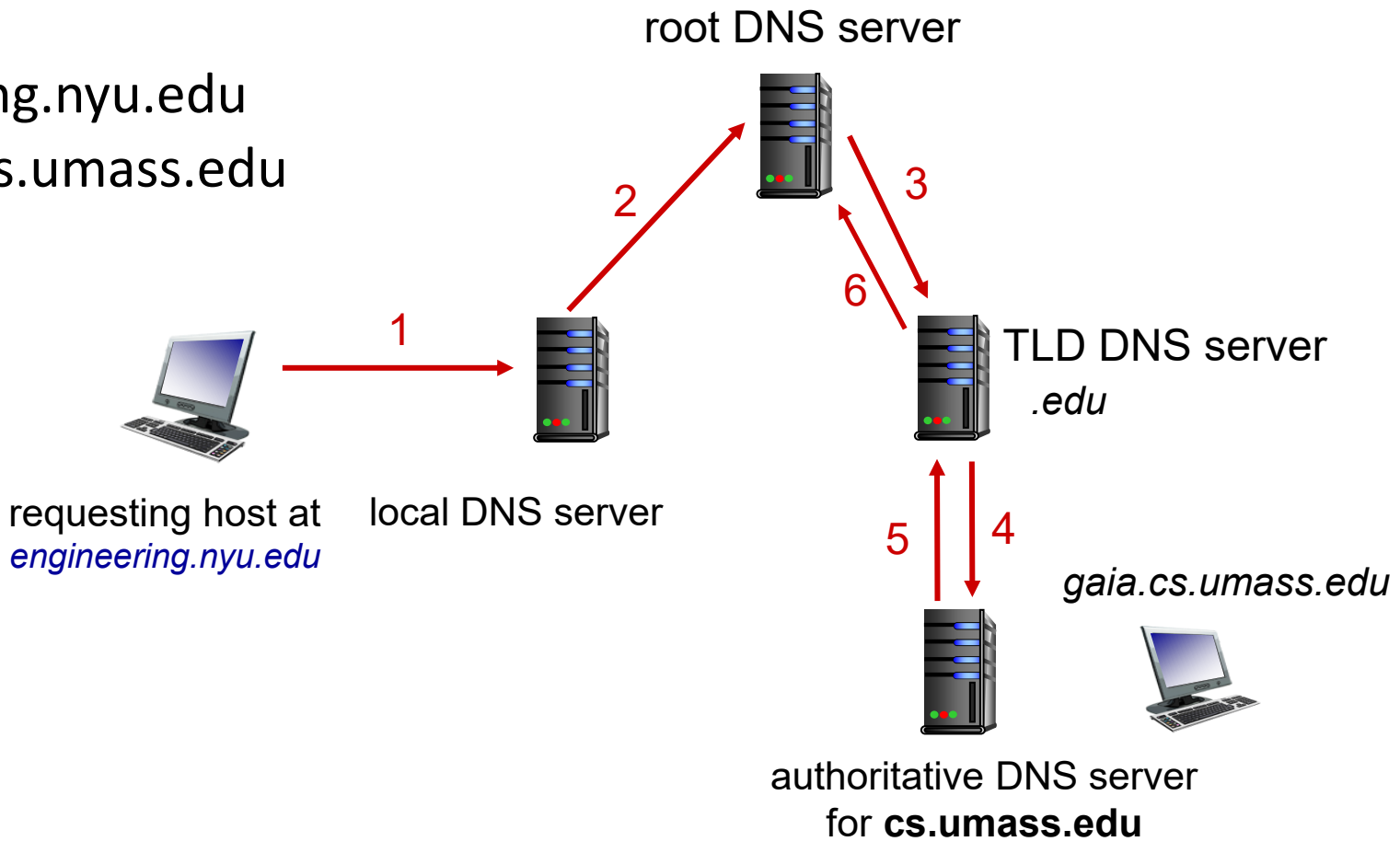


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy

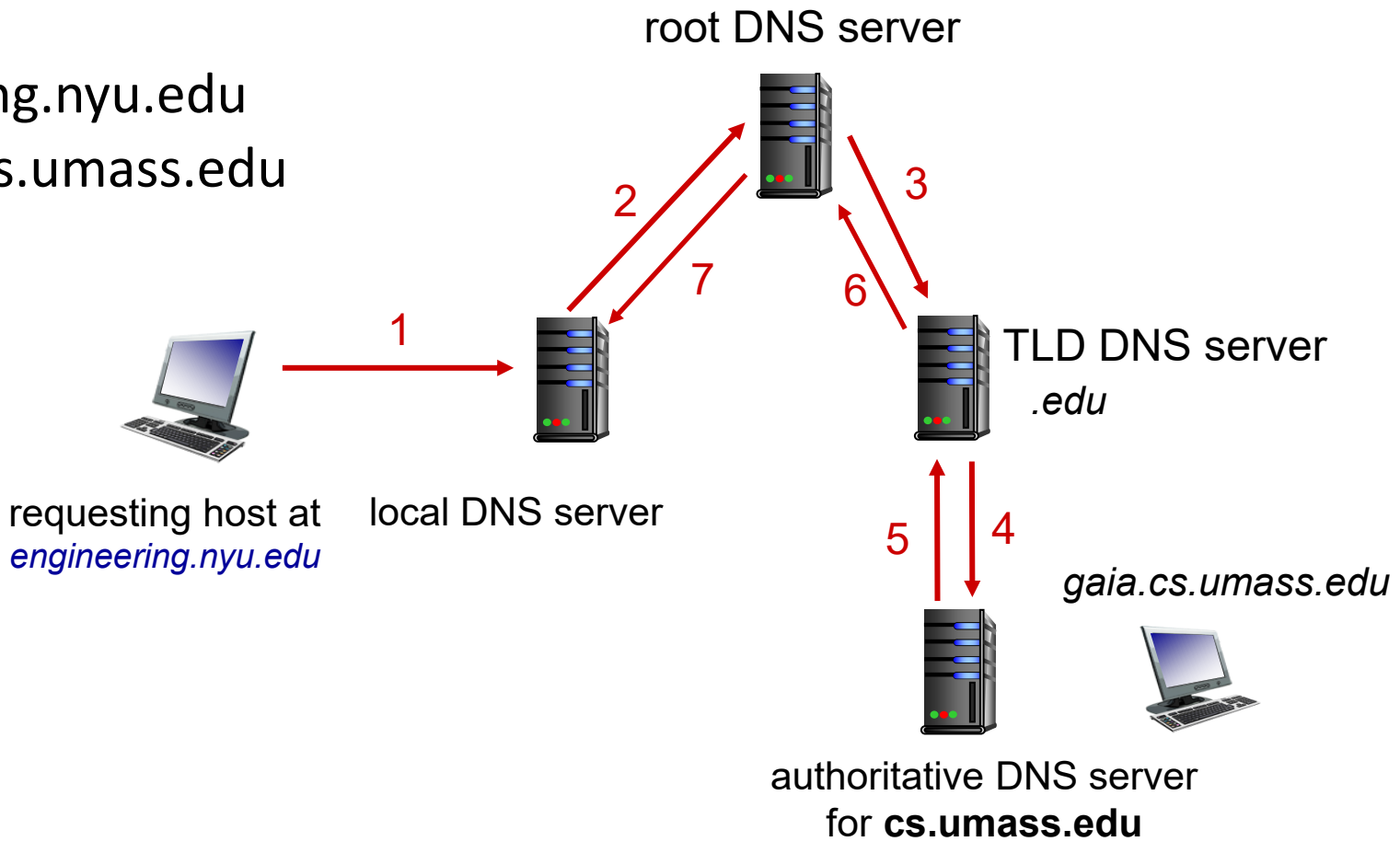


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy



DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

- puts burden of name resolution on contacted name server (less load to local DNS server)
- heavy load at upper levels of hierarchy