Διαχείριση Δικτύων Βασισμένων στο Λογισμικό 2025 (DIT306)

Δρ. Ειρήνη Λιώτου

eliotou@hua.gr

27/2/2025

Διάρθρωση μαθήματος

- **<u>Διαλέξεις:</u>**
  - Θεωρία
  - Εργαστήριο:
    - Παράδοση απαντήσεων εργαστηρίου απευθείας στο τέλος του μαθήματος ή εντός 1 εβδομάδας
    - Συνήθως ζητείται screenshots απαντήσεων από Wireshark/Mininet
    - Ερωτήσεις στα Αγγλικά → θα τις μεταφράζουμε προφορικά
    - Σύνολο 10 εργαστήρια – 1 αναφορά προαιρετική (επιλογής σας)
    - Αν παραδοθούν και οι 10, λαμβάνονται υπόψιν οι 9 με το μεγαλύτερο βαθμό

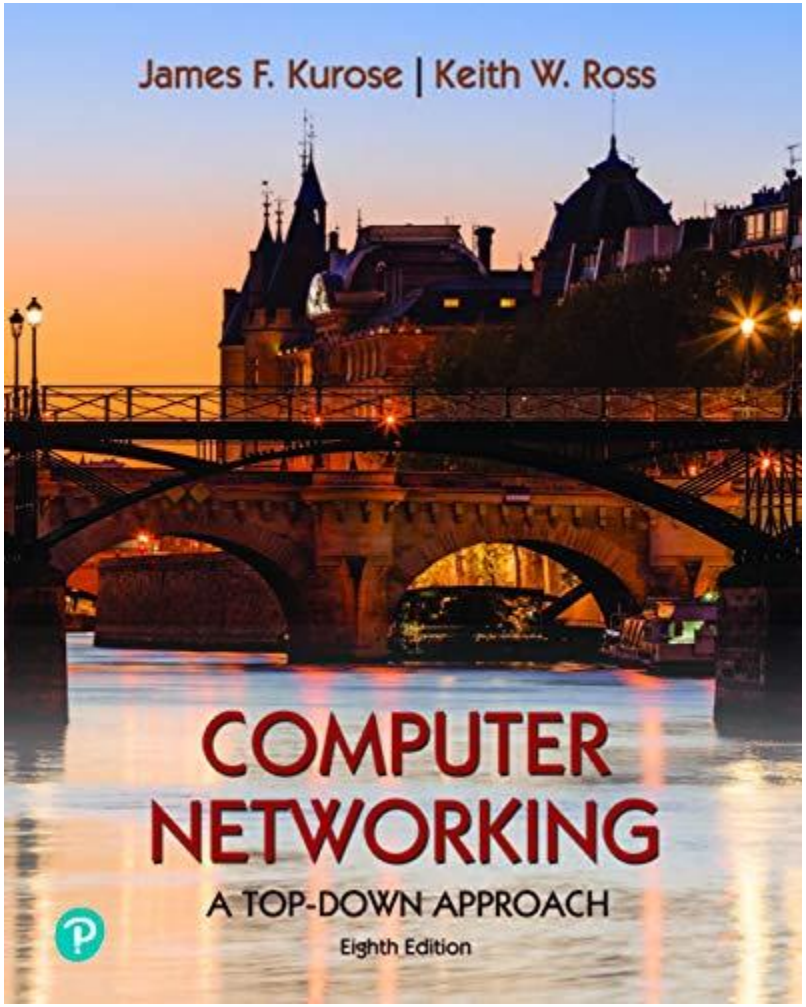
Διάρθρωση μαθήματος

- **Βαθμός**
  - Δεν υπάρχει τελική εξέταση
  - Παράδοση απαντήσεων εργαστηρίου (70%)
  - Εργασία παρουσίασης συγκεκριμένων θεμάτων στο state of the art του μαθήματος (30%) → Word & PowerPoint presentation

Βιβλίο μαθήματος

Κυρίως σύγγραμμα:

- KUROSE & ROSS, «Δικτύωση Υπολογιστών: Προσέγγιση από Πάνω προς τα Κάτω», 7η/8η έκδοση, Εκδόσεις Γκιούρδας https://gaia.cs.umass.edu/kurose_ross/index.php



Σχεδιάγραμμα μαθήματος - ΘΕΩΡΙΑ

Διάλεξη Α/Α	ΠΕΡΙΕΧΟΜΕΝΟ ΔΙΑΛΕΞΗΣ	
1	Εισαγωγή στο διαδίκτυο, πρότυπο αναφοράς OSI, αρχιτεκτονική διαδικτύου, βασικές μετρικές απόδοσης δικτύου	27/2
2	End-to-end αίτημα ιστοσελίδας και βασικά πρωτόκολλα	6/3
3	Πρωτόκολλο διαχείρισης δικτύου (SNMP), πρωτόκολλο ελέγχου μηνυμάτων διαδικτύου (ICMP)	13/3
4	Επίπεδο Δικτύου basics	20/3
5	Εισαγωγή στα δίκτυα βασισμένα στο λογισμικό (Software-Defined Networking - SDN)	27/3
6	Install & run Mininet	3/4
7	Mininet Walkthrough & MiniEdit	10/4
8	POX SDN Controller	8/5
9	SDN applications without controller	15/5
10	Ryu Controller MiniNAM Traffic engineering	22/5
11	Παρουσίαση εργασιών φοιτητών 1/3 (στα πλαίσια της εξέτασης του μαθήματος)	29/5
12	Παρουσίαση εργασιών φοιτητών 2/3 (στα πλαίσια της εξέτασης του μαθήματος)	5/6
13	Παρουσίαση εργασιών φοιτητών 3/3 (στα πλαίσια της εξέτασης του μαθήματος)	12/6

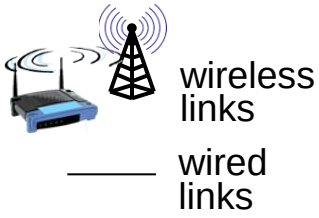
Σχεδιάγραμμα μαθήματος – ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ

Διάλεξη Α/Α	ΠΕΡΙΕΧΟΜΕΝΟ ΕΡΓΑΣΤΗΡΙΟΥ	
1	Wireshark Lab: Εισαγωγή και γνωριμία	27/2
2	Wireshark Lab: HTTP	6/3
3	Wireshark Lab: DNS, ICMP, Ethernet/ARP	13/3
4	Wireshark Lab: IP, DHCP	20/3
5	Knowledge Checks + NAT	27/3
6	Mininet lab: Εγκατάσταση, εισαγωγή και γνωριμία	3/4
7	Mininet lab: Δημιουργία τοπολογιών δικτύου και performance tests MiniEdit lab: Εισαγωγή και γνωριμία	10/4
8	SDN lab: POX SDN controller: hub, switch, firewall	8/5
9	SDN lab: Open VSwitch	15/5
10	SDN lab: Ryu Controller, MiniNAM & Traffic tests	22/5
11	Παρουσίαση εργασιών φοιτητών 1/3 (στα πλαίσια της εξέτασης του μαθήματος)	29/5
12	Παρουσίαση εργασιών φοιτητών 2/3 (στα πλαίσια της εξέτασης του μαθήματος)	5/6
13	Παρουσίαση εργασιών φοιτητών 3/3 (στα πλαίσια της εξέτασης του μαθήματος)	12/6

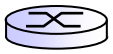
Chapter 1
Introduction

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

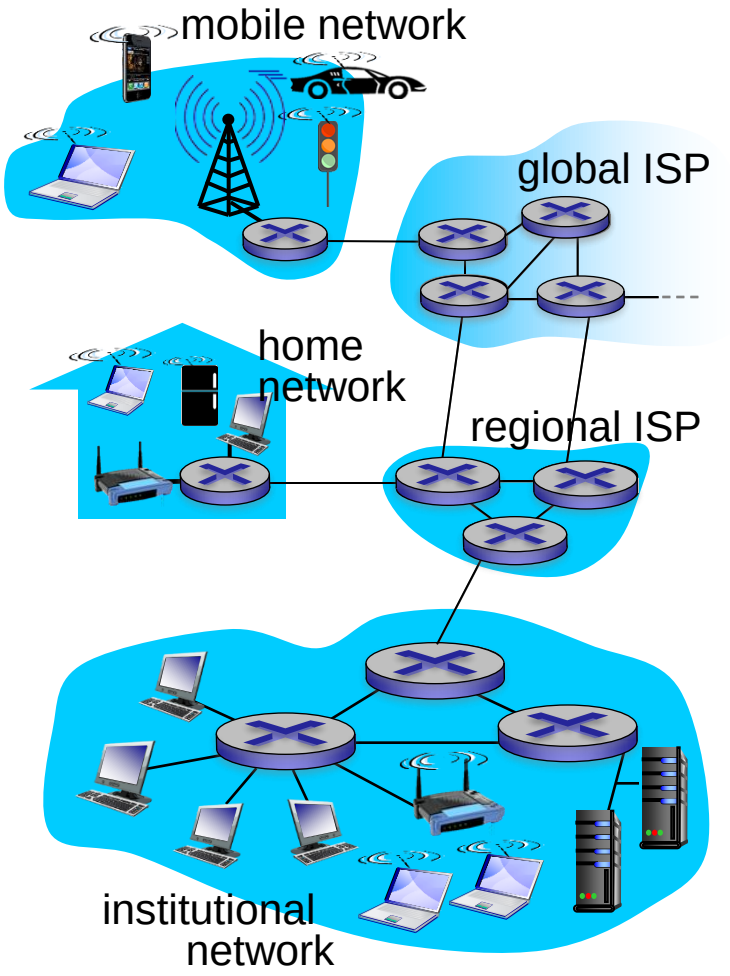
- billions of connected computing devices:
  - *hosts* = *end systems*
  - running *network apps*



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

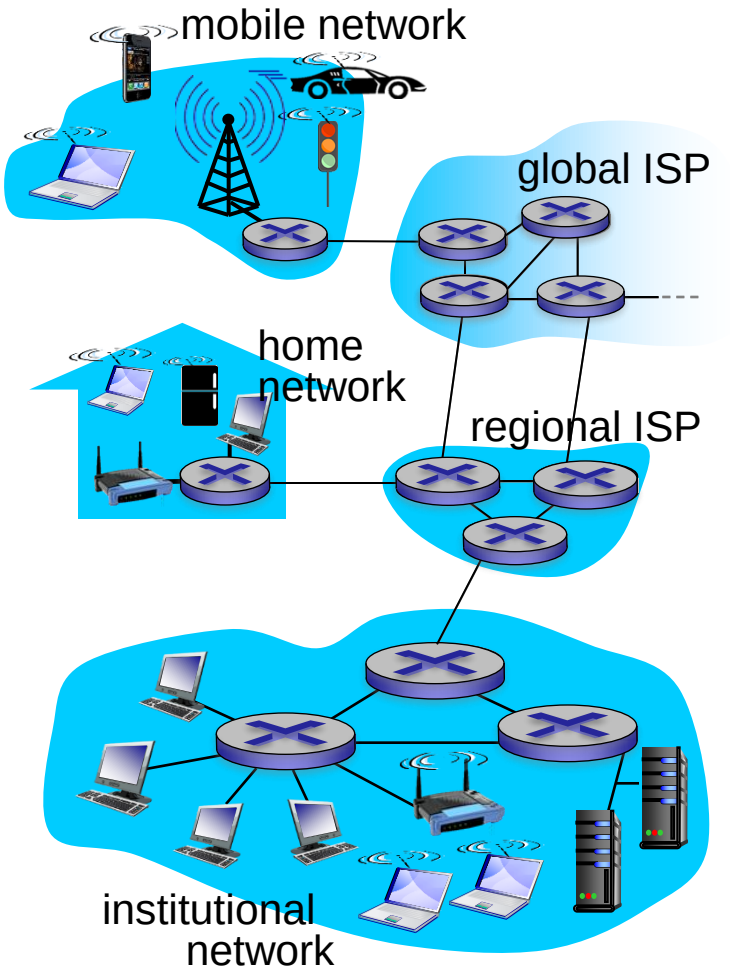


- *packet switches:* forward packets (chunks of data)
  - *routers* and *switches*



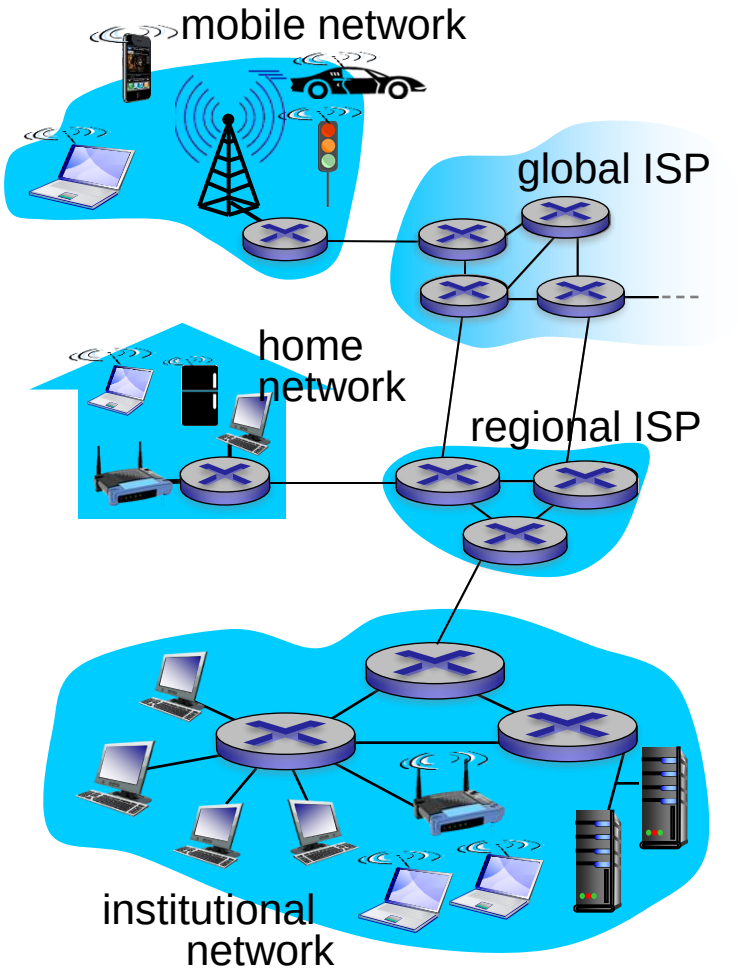
What's the Internet: "nuts and bolts" view

- *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



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?

human protocols:

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

… specific messages sent

… specific actions taken when messages received, or other events

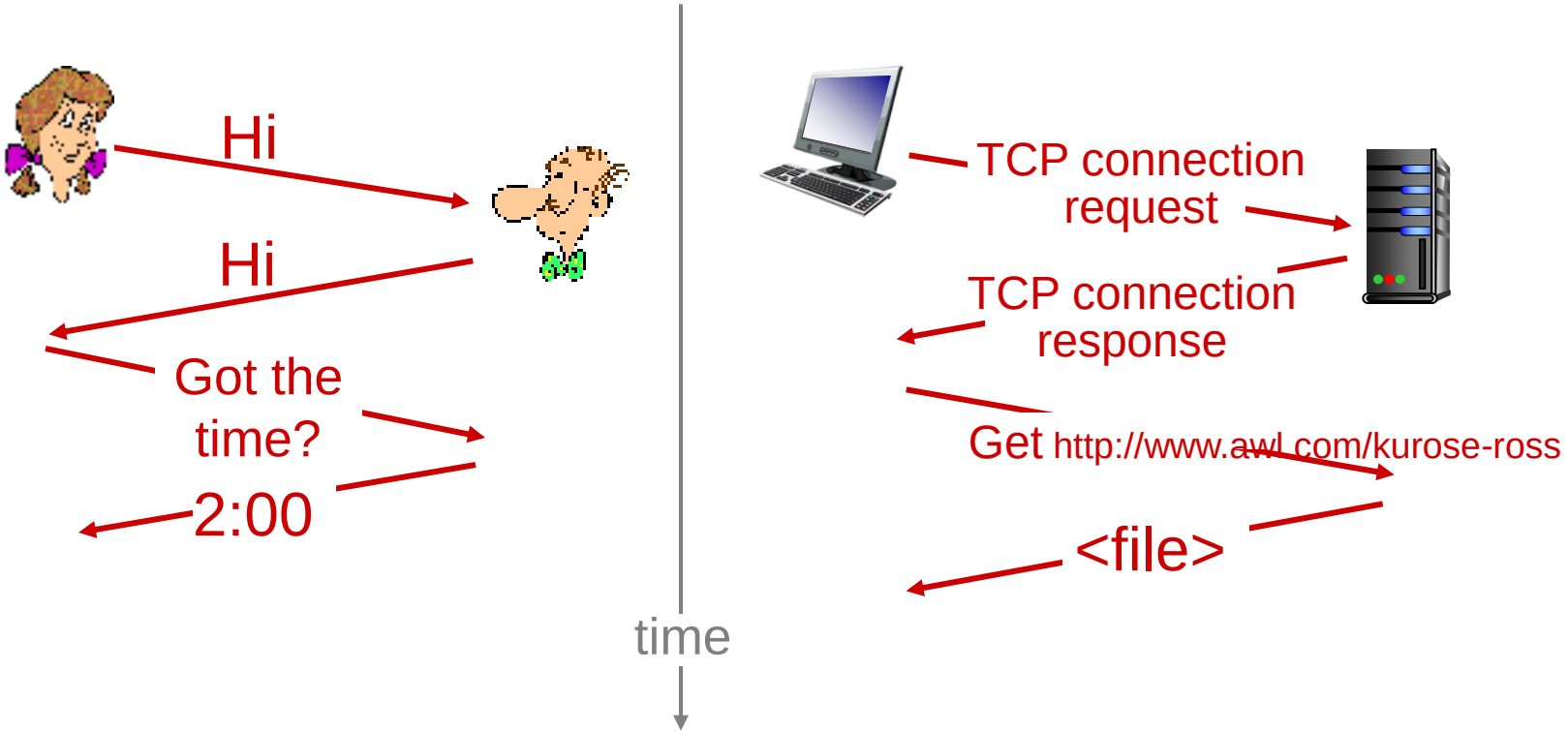
network protocols:

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

protocols define 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:

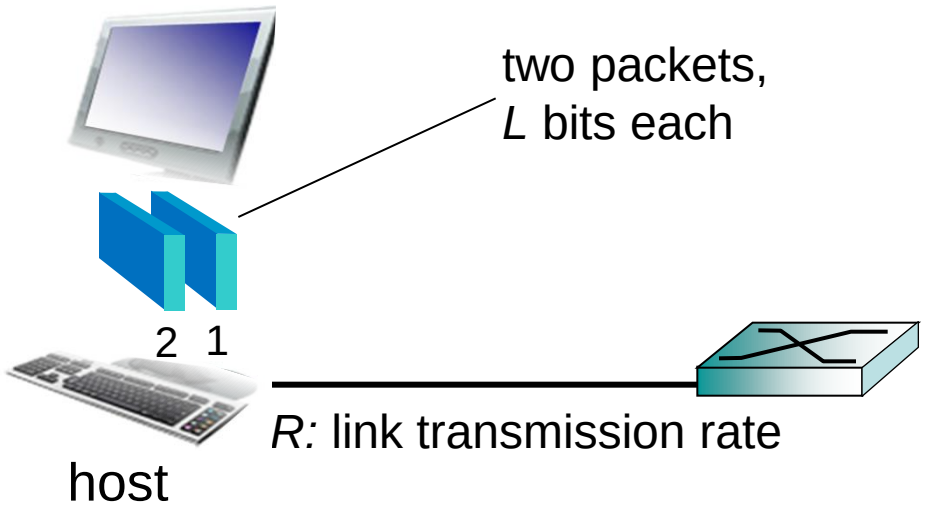


Q: other human protocols?

Host: sends *packets* of data

host sending function:

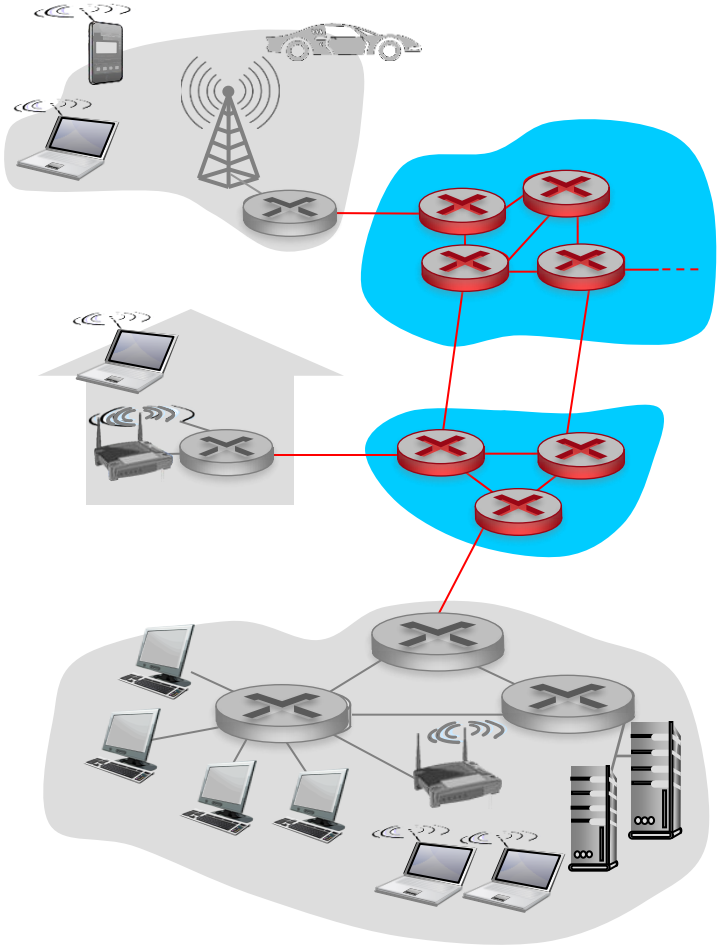
- takes application message
- breaks into smaller chunks, known as *packets*, of length *L* bits
- transmits packet into access network at *transmission rate R*
  - link transmission rate, aka link *capacity, aka link bandwidth*



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

The network core

- mesh of interconnected routers
- packet-switching: hosts break application-layer messages into *packets*
  - forward packets from one router to the next, across links on path from source to destination
  - each packet transmitted at full link capacity

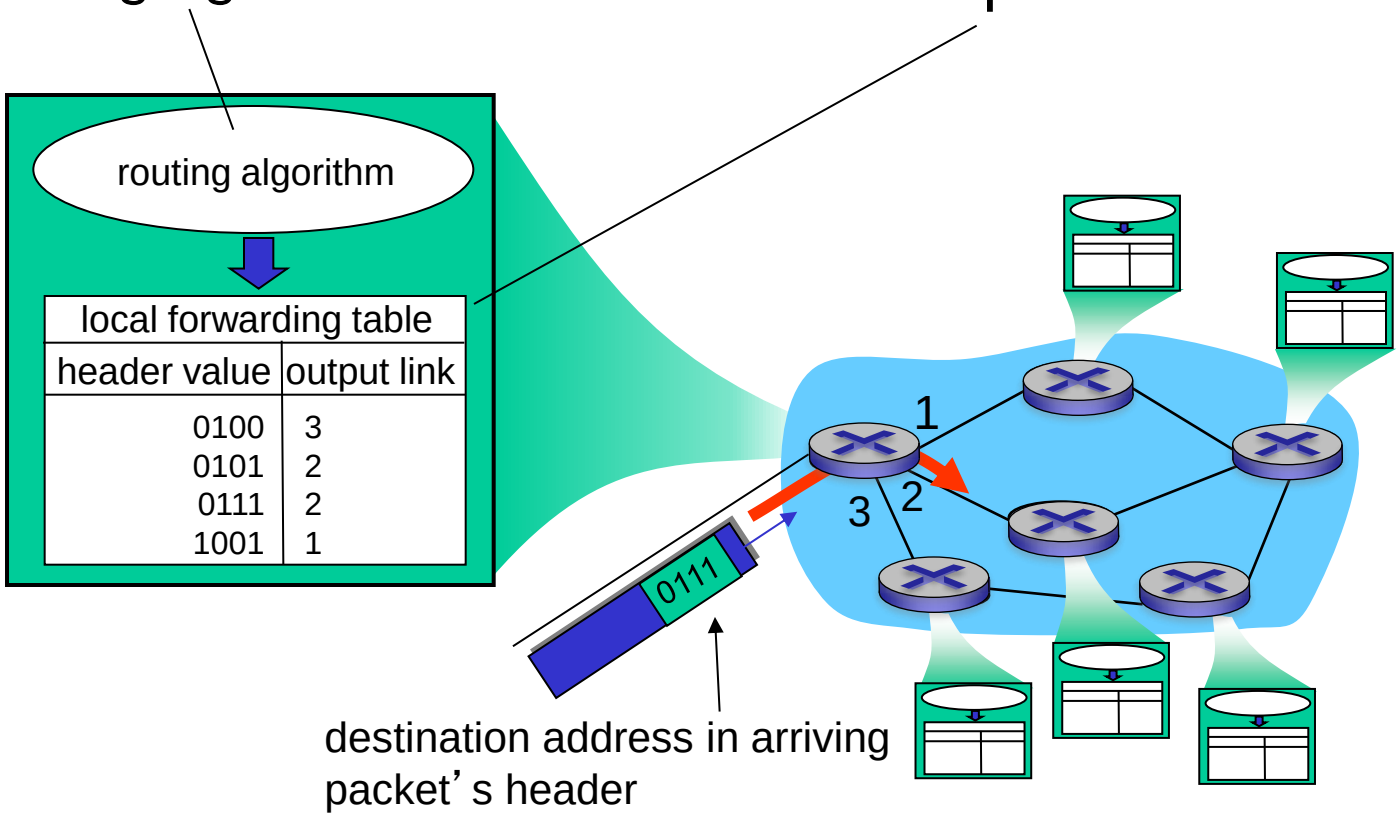


Two key network-core functions

routing: determines source-destination route taken by packets

- *routing algorithms*

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



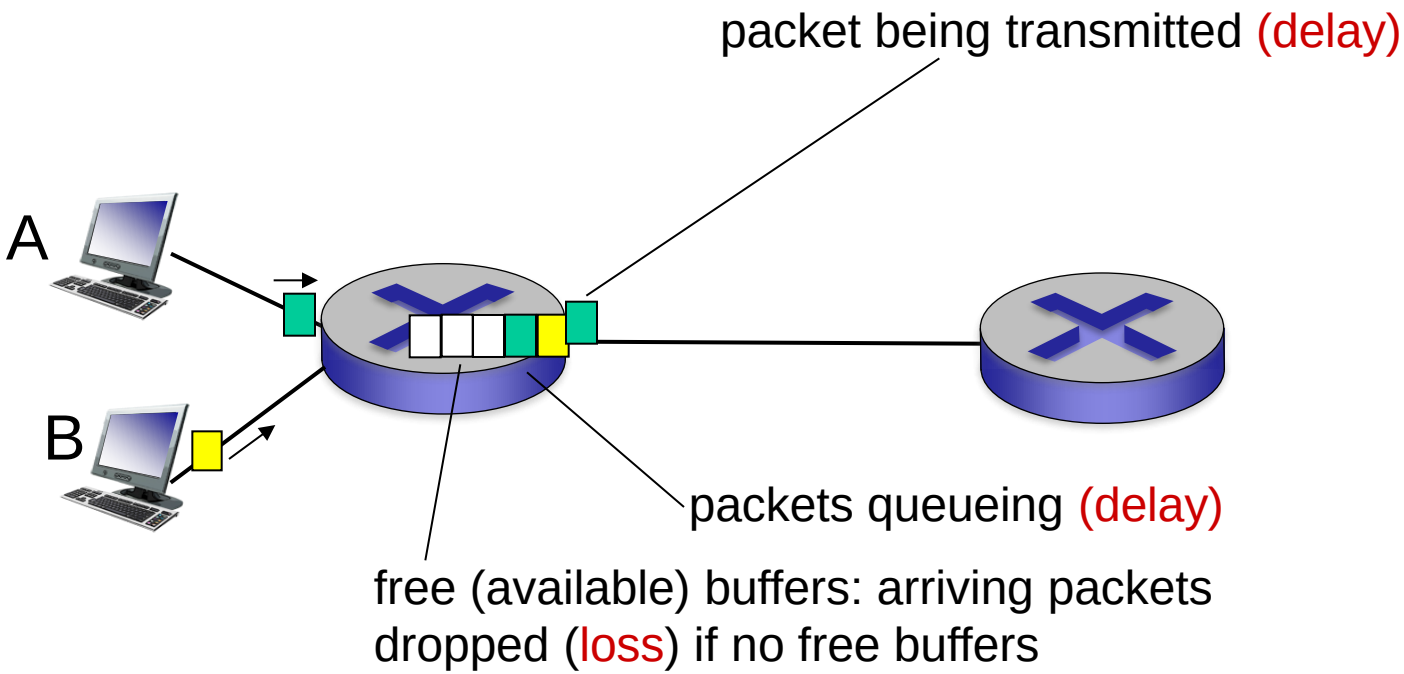

routing
45 h
3,053 miles
48 h
3,309 miles
United States
Victoria
Seattle
WASHINGTON
Portland
OREGON
IDAHO
MONTANA
NORTH DAKOTA
MINNESOTA
SOUTH DAKOTA
WYOMING
Minneapolis
WISCONSIN
MICHIGAN
Toronto
Quebec City
Montreal
Ottawa
MAINE
VERMONT
NEW HAMPSHIRE
Northampton
CT
RI
Detroit
Chicago
IOWA
NEVADA
UTAH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


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

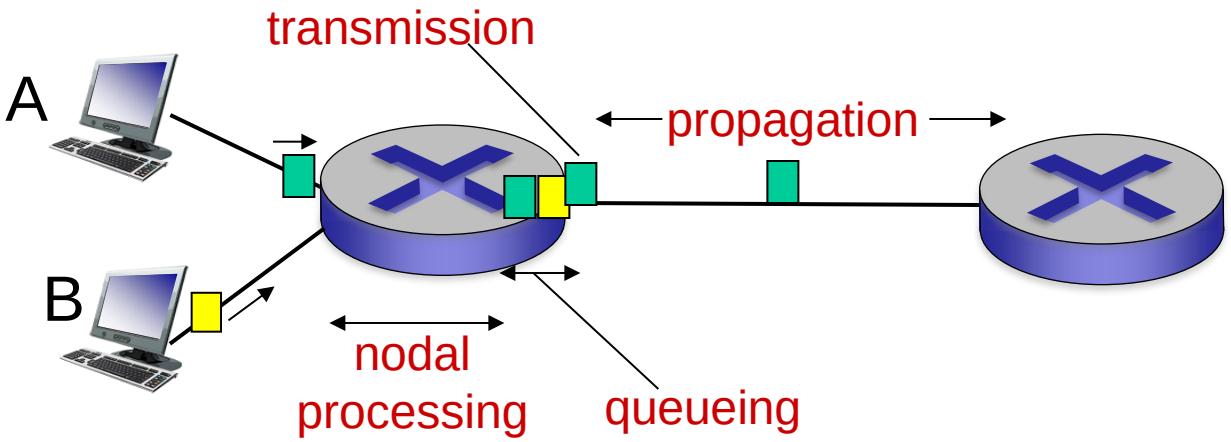
How do loss and delay occur?

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



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

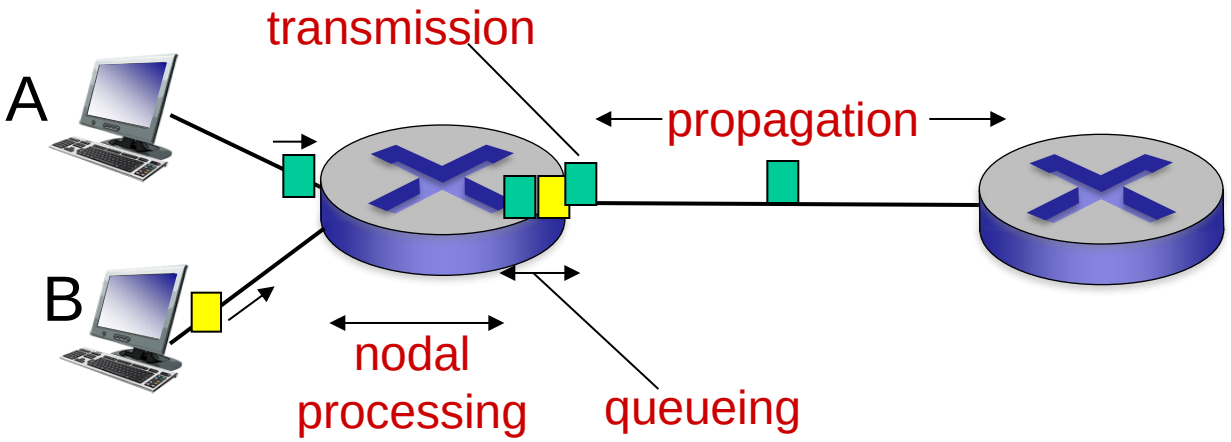
d_{proc}: nodal processing

- 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)*
- $d_{trans} = L/R$

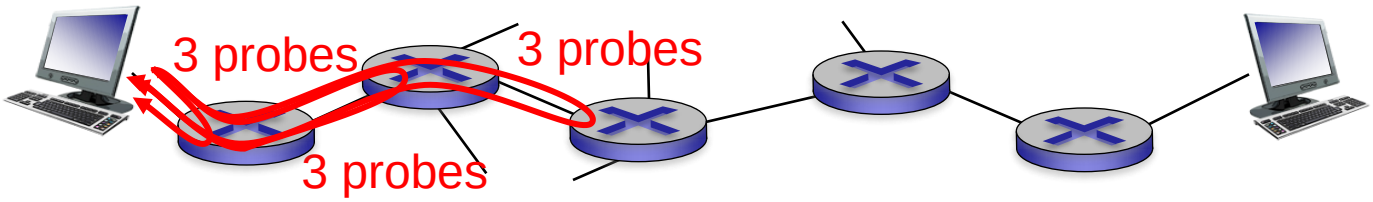
d_{trans} and d_{prop} *very* different

d_{prop}: propagation delay:

- d: length of physical link
- s: propagation speed (~3×10^8 m/sec)
- $d_{prop} = d/s$

"Real" Internet delays and routes

- what do "real" Internet delay & loss look like?
- `traceroute` program: provides delay measurement from source to router along end-end Internet path towards destination. For all *i:*
  - sends three packets that will reach router *i* on path towards destination
  - router *i* will return packets to sender
  - sender times interval between transmission and reply



"Real" Internet delays, routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

3 delay measurements from gaia.cs.umass.edu to cs-gw.cs.umass.edu

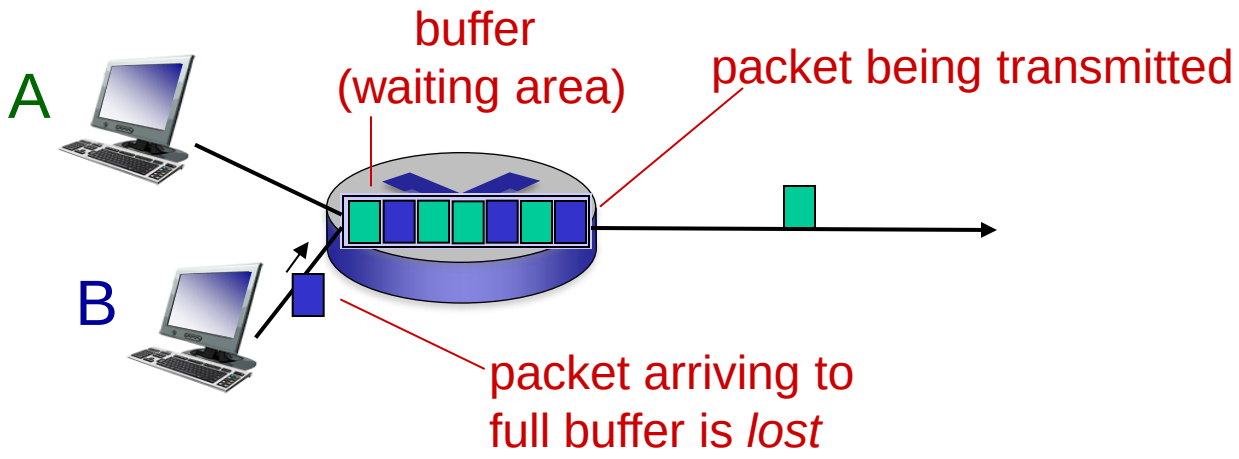
```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 ms 18 ms
6  abilene-vbns.abilene.ucaid.edu (198.32.11.9)  22 ms  18 ms  22 ms
7  nycm-wash.abilene.ucaid.edu (198.32.8.46)  22 ms  22 ms  22 ms
8  62.40.103.253 (62.40.103.253)  104 ms 109 ms 106 ms
9  de2-1.de1.de.geant.net (62.40.96.129)  109 ms 102 ms 104 ms
10  de.fr1.fr.geant.net (62.40.96.50)  113 ms 121 ms 114 ms
11  renater-gw.fr1.fr.geant.net (62.40.103.54)  112 ms  114 ms  112 ms
12  nio-n2.cssi.renater.fr (193.51.206.13)  111 ms  114 ms  116 ms
13  nice.cssi.renater.fr (195.220.98.102)  123 ms  125 ms  124 ms
14  r3t2-nice.cssi.renater.fr (195.220.98.110)  126 ms  126 ms  124 ms
15  eurecom-valbonne.r3t2.ft.net (193.48.50.54)  135 ms  128 ms  133 ms
16  194.214.211.25 (194.214.211.25)  126 ms  128 ms  126 ms
17  * * *
18  * * *
19  fantasia.eurecom.fr (193.55.113.142)  132 ms  128 ms  136 ms
```

trans-oceanic link

* means no response (e.g., router not replying)

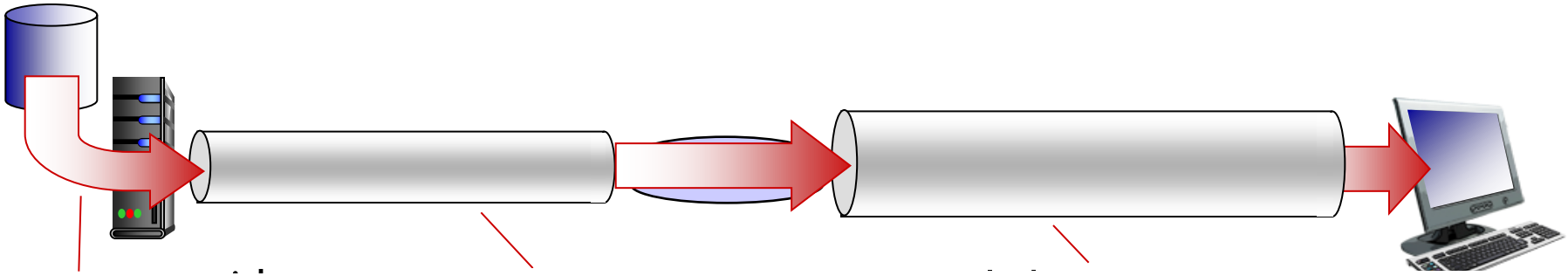
Packet loss

- 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



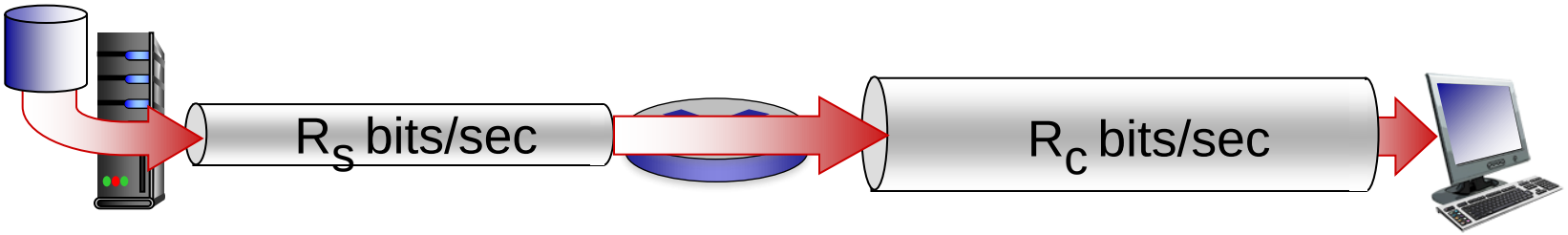
Throughput

- *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

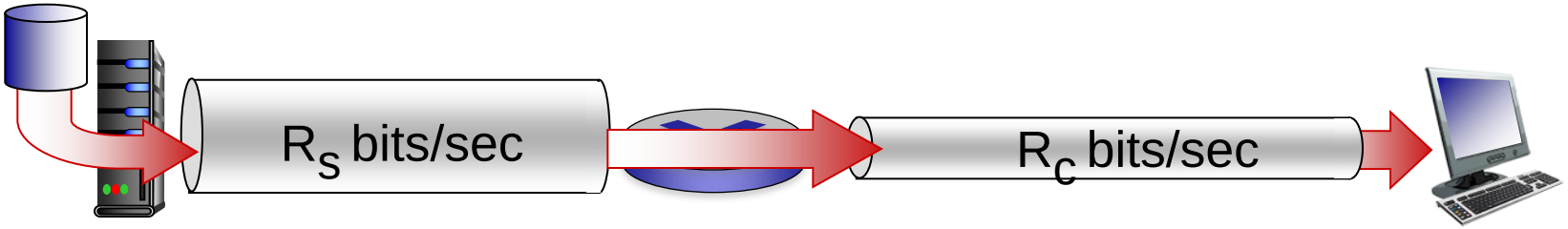


Throughput (more)

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



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

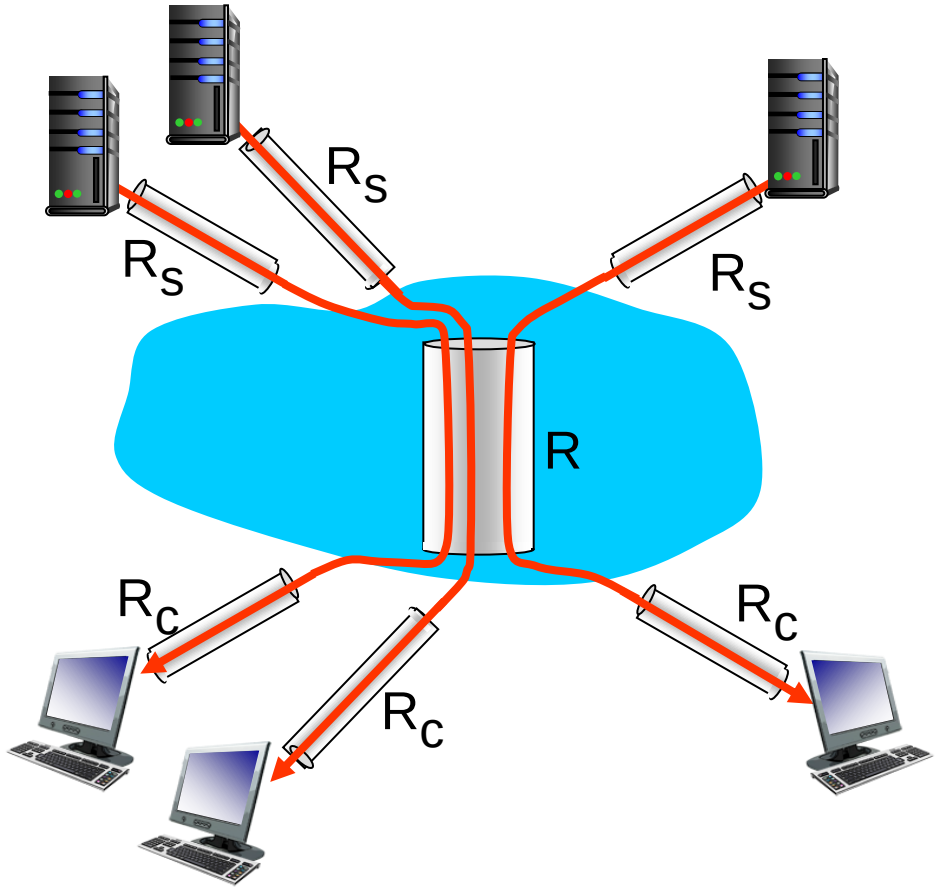


bottleneck link

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

Throughput: Internet scenario

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



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

Protocol "layers"

Networks are complex, with many "pieces":

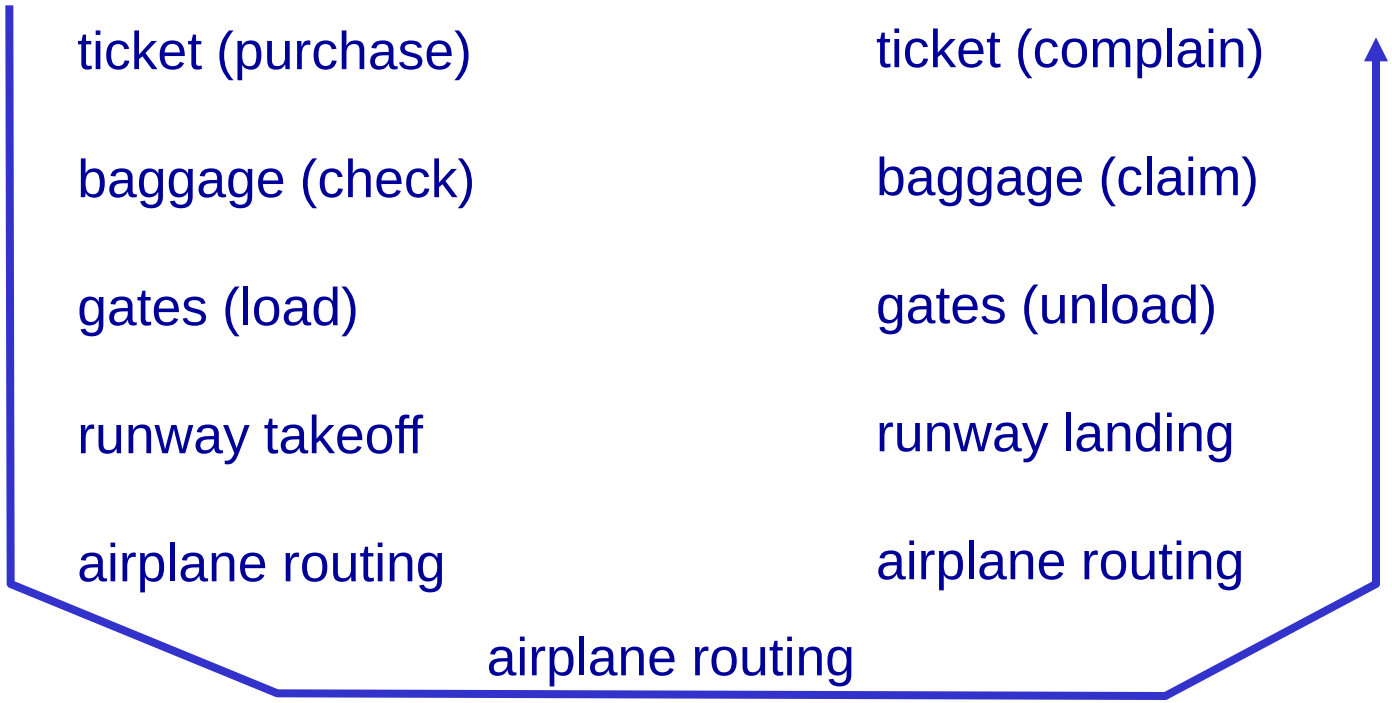
- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:

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

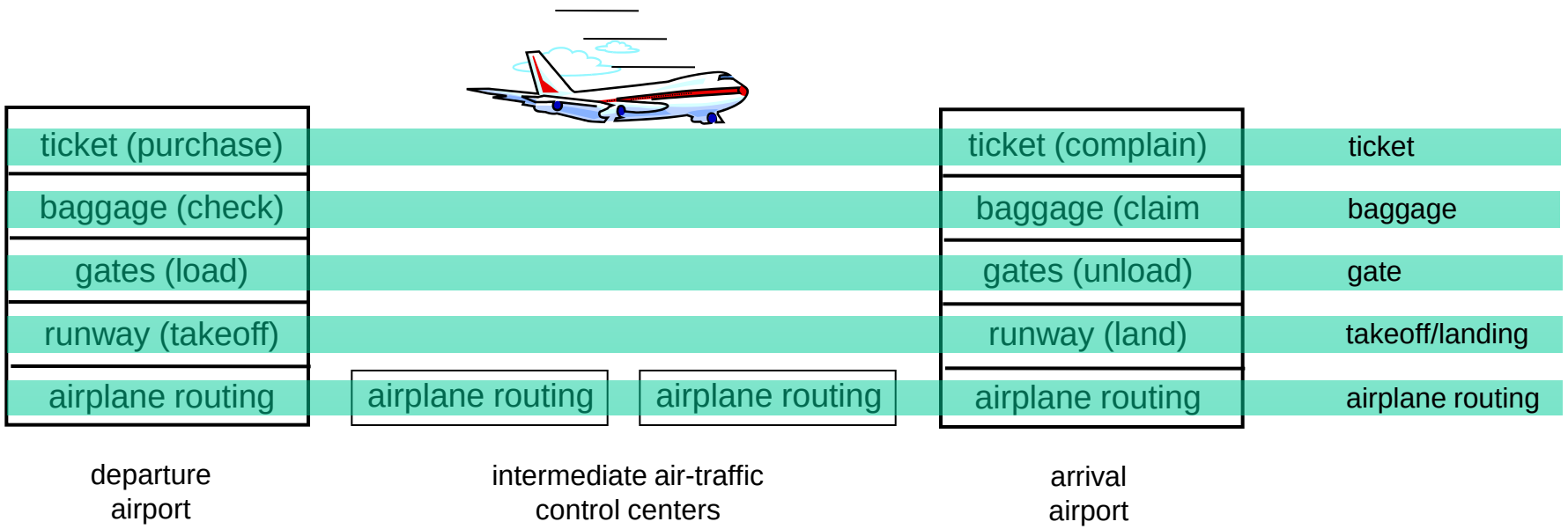
…. or at least our discussion of networks?

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

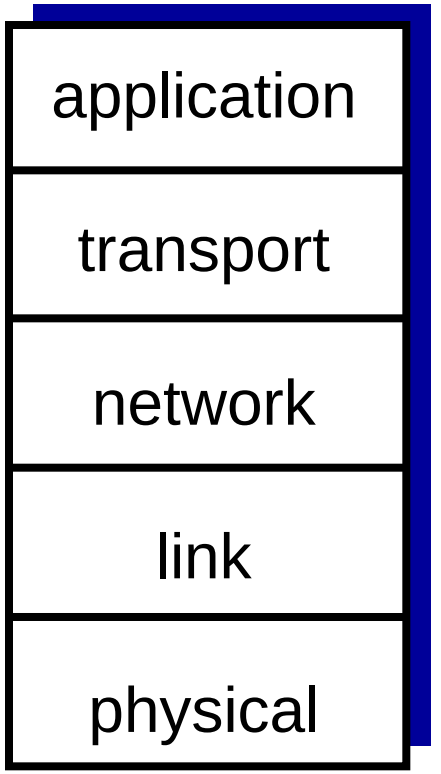
Why layering?

dealing with complex systems:

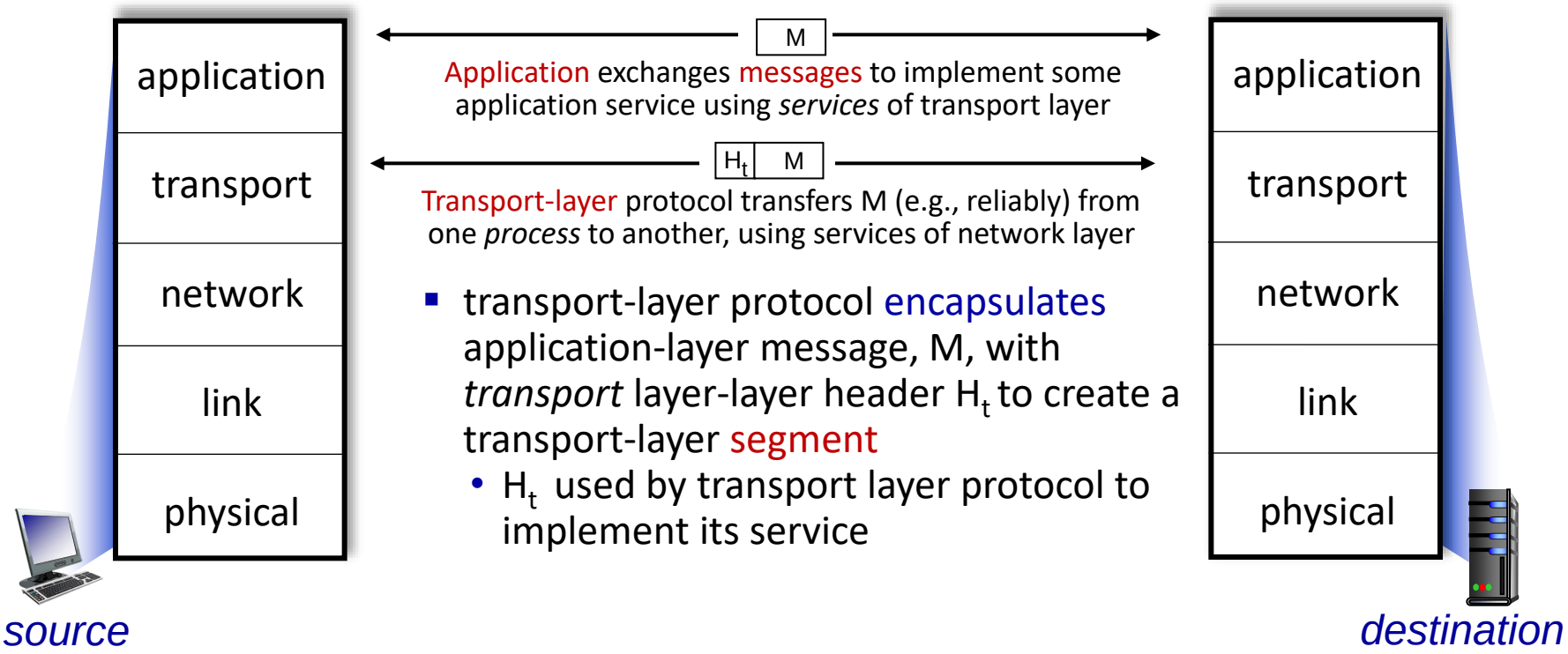
- explicit structure allows identification, relationship of complex system's pieces
  - layered *reference model* for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't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

Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP, 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, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"

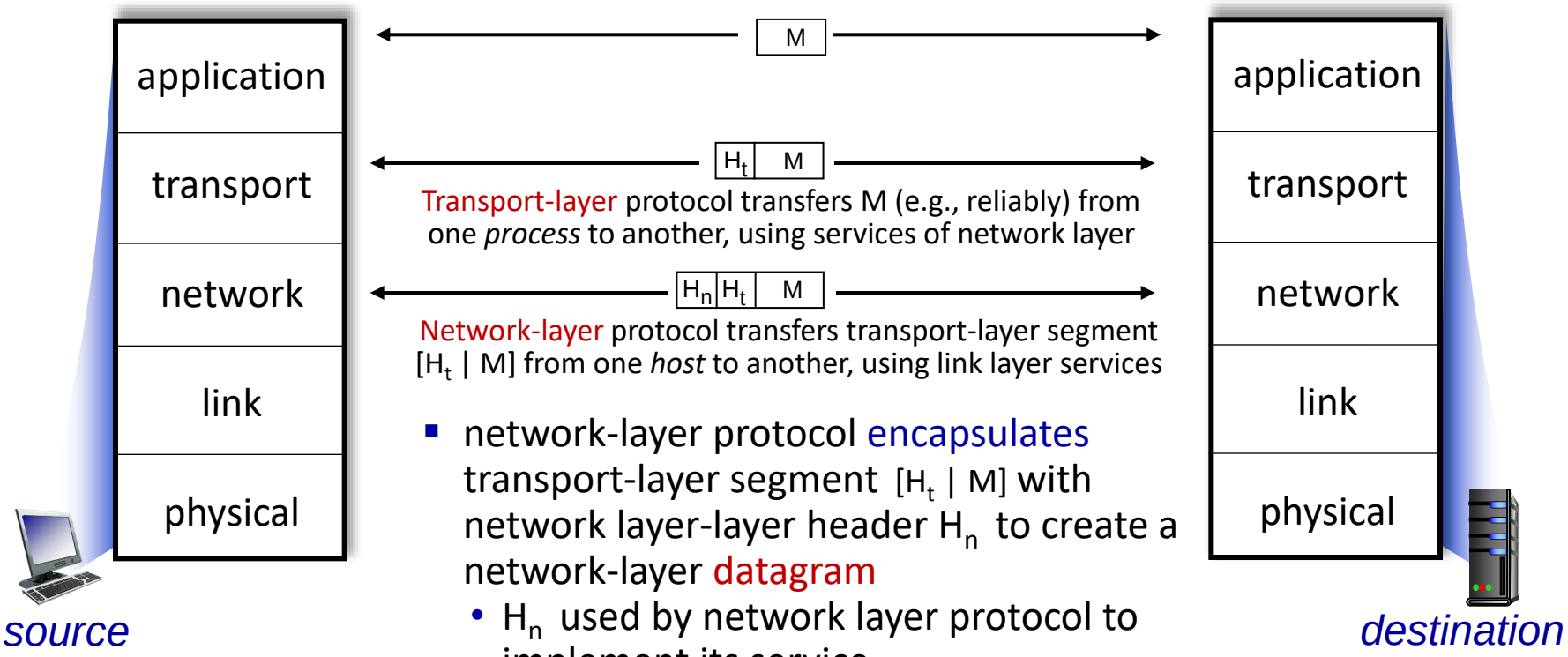


Services, Layering and Encapsulation



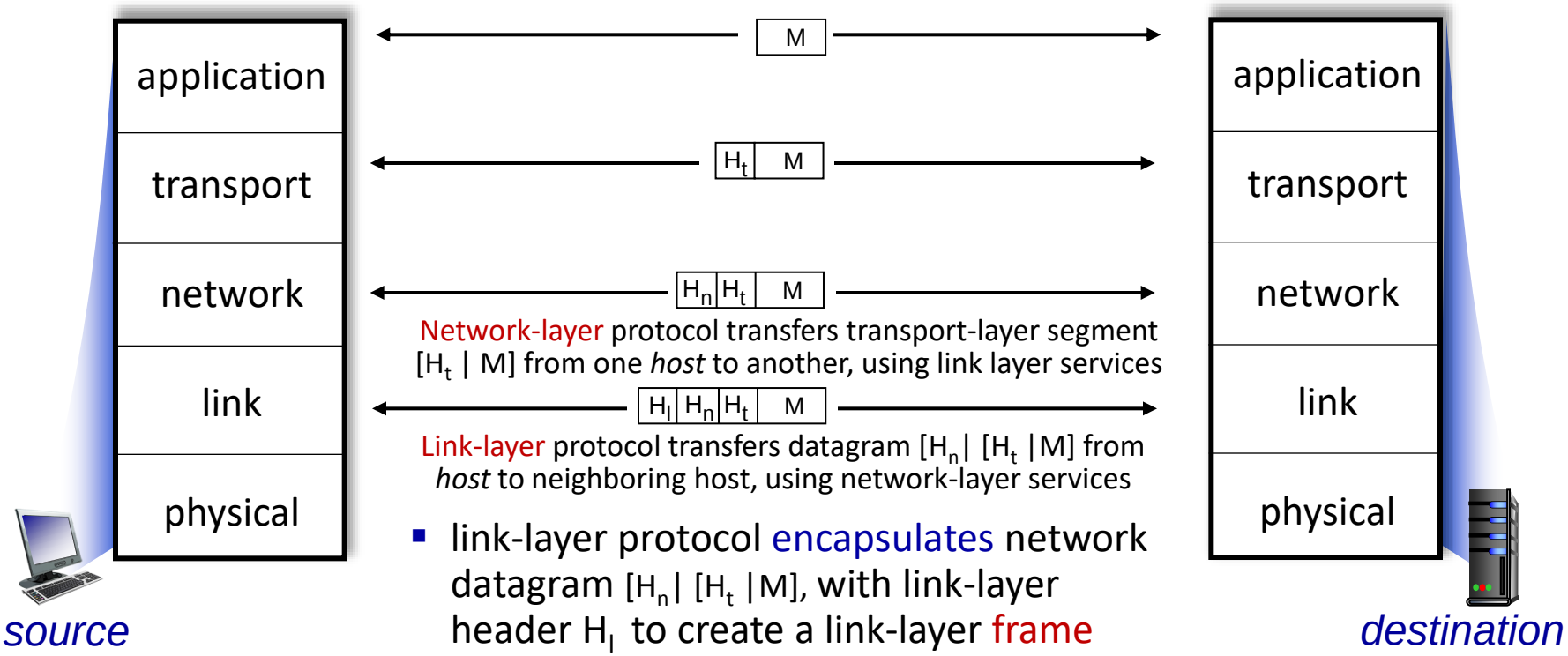
- transport-layer protocol encapsulates application-layer message, M, with *transport* layer-layer header H_t to create a transport-layer segment
  - H_t used by transport layer protocol to implement its service

Services, Layering and Encapsulation



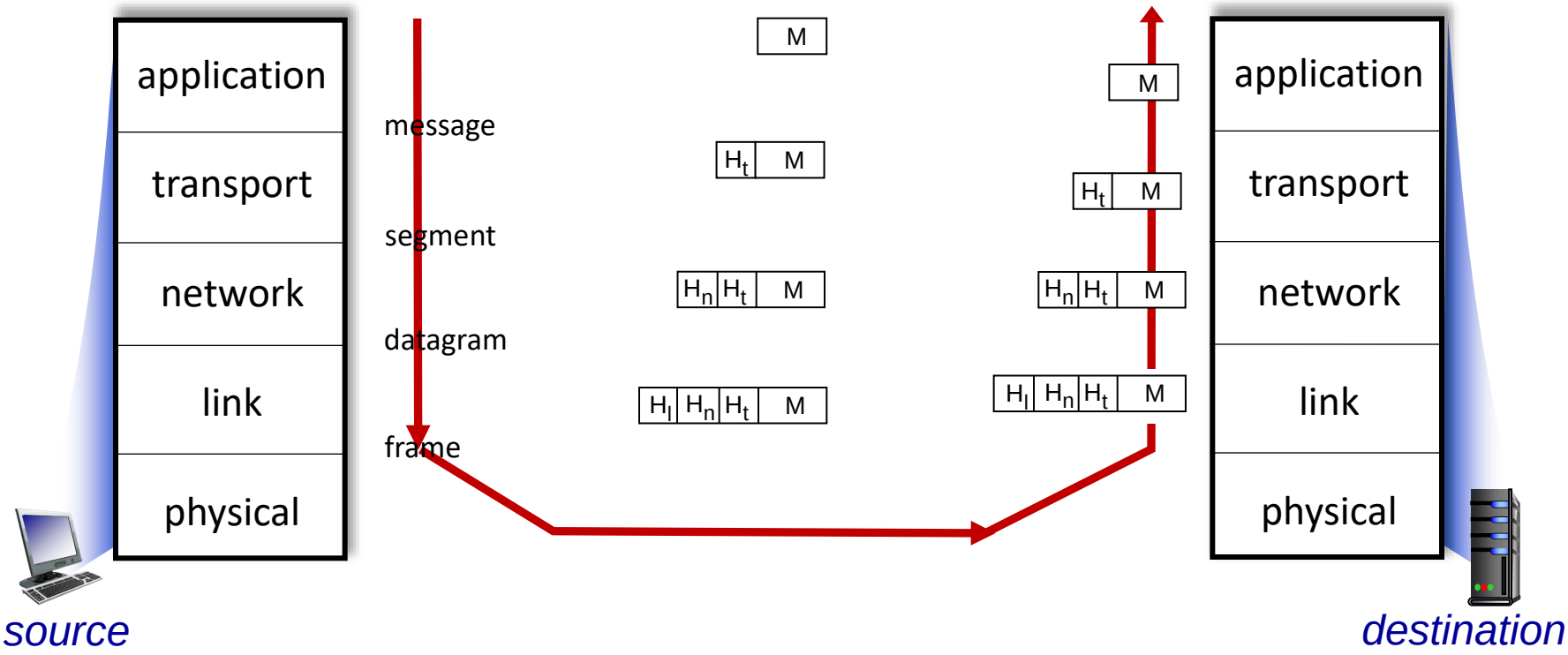
- network-layer protocol encapsulates transport-layer segment [H_t | M] with network layer-layer header H_n to create a network-layer datagram
  - H_n used by network layer protocol to implement its service

Services, Layering and Encapsulation

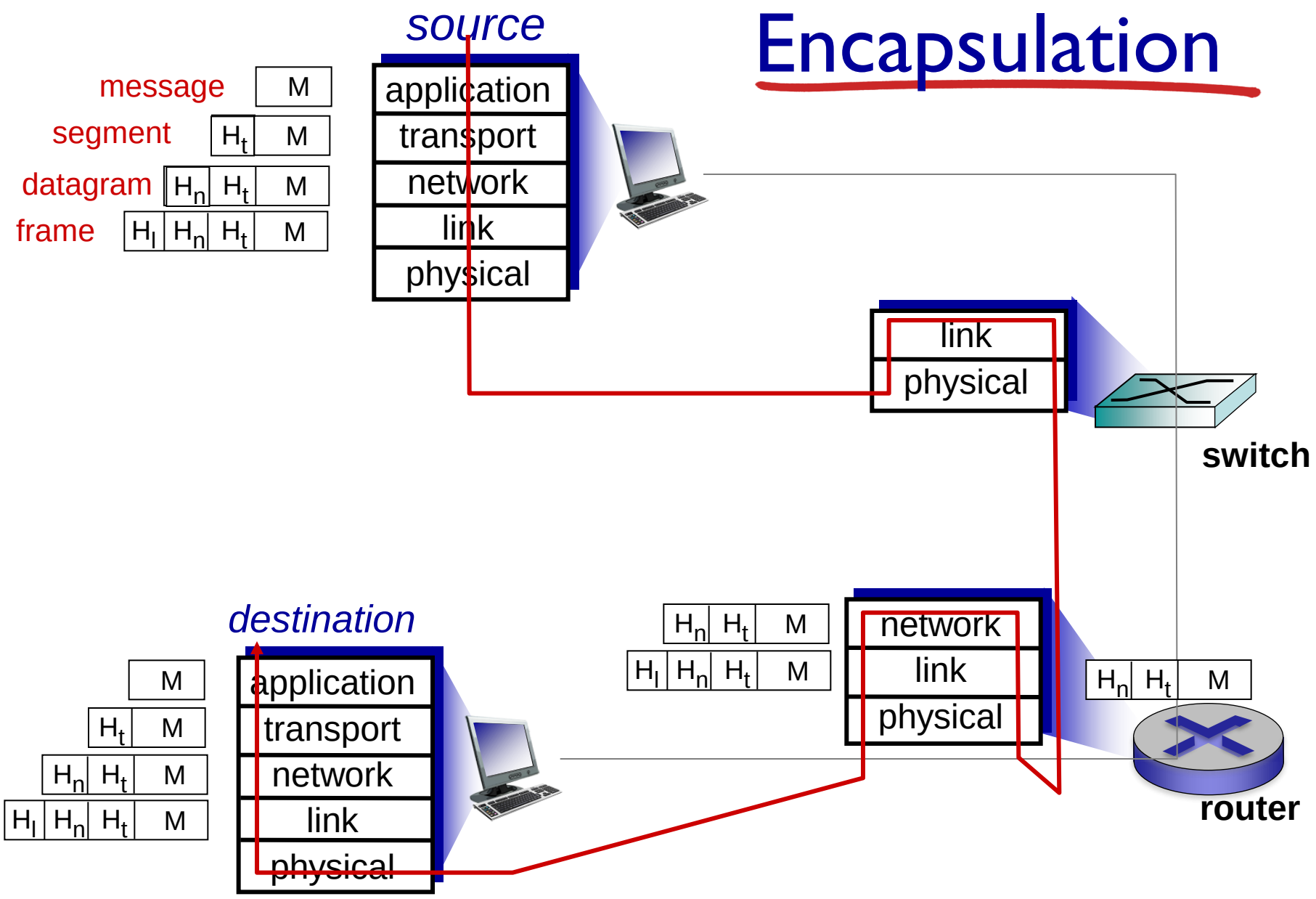


- link-layer protocol encapsulates network datagram [H_n| [H_t |M], with link-layer header H_l to create a link-layer frame

Services, Layering and Encapsulation



Encapsulation



THANK YOU!