



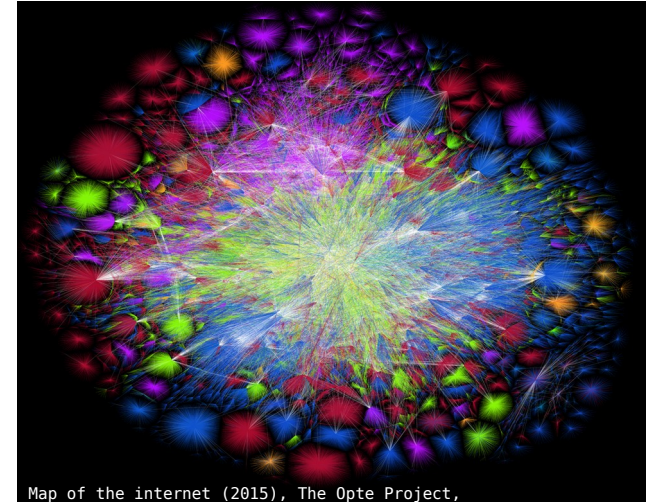
Computer Systems (Computer Networks)

David Marchant

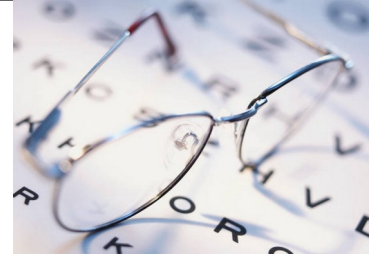
Based on slides compiled by Marcos Vaz Salles, with
adaptions by Vivek Shah and Michael Kirkedel Thomsen

Why study Computer Networks?

- How can we build networked applications?
- What are the protocols that power the Internet?
- How can we secure data transmission?



What should we learn in this portion of the course?



- Describe the design of application-layer protocols such as HTTP and DNS.
- Implement networked applications making use of sockets.
- Explain the mechanisms used by transport-layer protocols to achieve multiplexing, reliability, flow control, and congestion control.
- Describe network setups involving subnets, NAT, and LAN segments as well as related interconnection hardware such as routers, switches, and hubs.
- Explain the use of cryptography and operational measures to secure network protocols and applications.
- Build for performance by using all the hardware we have, both locally and remotely.
- Utilise events as basis for applications.
- De-mystify cloud computing and its impacts on data center networking

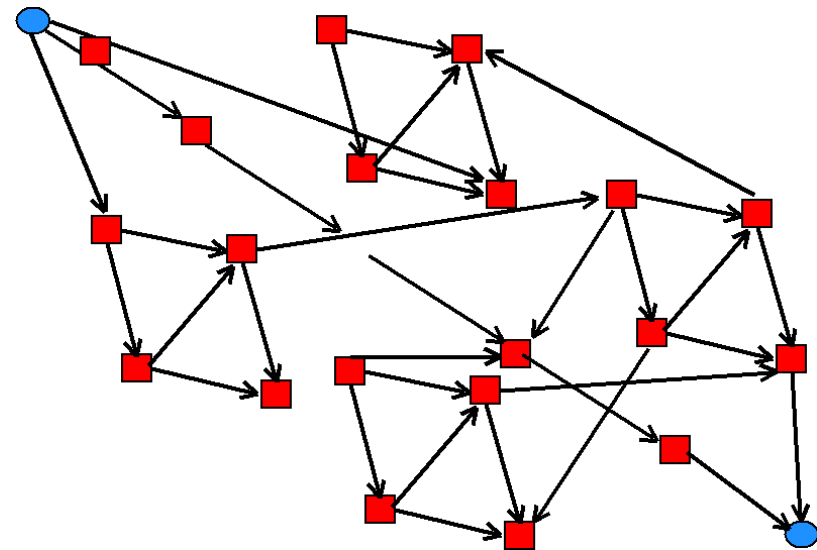
Computer Networks: What will we study?

- How can we build networked applications?
 - Application-level protocols, e.g., HTTP and content delivery
 - Programming with sockets
 - Resolving names with DNS



Computer Networks: What will we study?

- What are the protocols that power the Internet?
 - UDP: Basic transport
 - TCP: Reliable and ordered transport with flow and congestion control
 - IP: Addressing, forwarding, routing
 - Ethernet et al: Physical transmission



Largest portion of
the course

Computer Networks: What will we study?

- How can we secure data transmission?
 - Cryptography
 - Authentication
 - HTTPS, IPSec: Securing protocols



Computer Networks: What will we study?

- Not only theoretical knowledge, but also skills
 - Programming with sockets
 - Implementing protocols
 - Using network tools
 - Building distributed applications



References & Course Materials

- Book
 - Computer Networking, 8th ed., James F. Kurose and Keith W. Ross, Pearson, ISBN 13: 978-0-135-92873-8
- Ex-book
 - Computer Systems: A Programmers Perspective, 3rd ed., Randal Bryant and David O'Hallaron, ISBN 13: 978-0-134-09266-9
- Other references
 - Vast majority listed in the course schedule
 - Will keep updating them as we go



Teaching and Assignments

- Lectures
 - **Informal discussion oriented.**
 - Jeg kan ikke tale dansk :(
 - **“The only stupid question is the one not asked”**
 - **Do not wait for course evaluation for hot fixes**
 - Slides are not a substitute for the book
 - Office: HCØ, building B, room 772-01-0-S06
- Exercises
 - Practical and Theoretical, both are import
- Assignments
 - **Assignments A3-A4**
 - Exercises on **concepts** as well
 - Programming of **distributed P2P file sharer**
 - **No plagiarism - Cite appropriately**

Acknowledgements

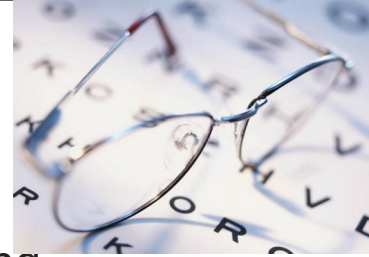
- Many of the slides in this course are based on or reproduce material kindly made available by Michael Freedman (Princeton), James Kurose & Keith Ross (RPI & NYU-Poly, textbook), Jerome Saltzer & M. Frans Kaashoek & Robert Morris (MIT), Randal E. Bryant & David R. Halloran (Computer Systems, textbook)
- Marcos Vaz Salles (Associate Professor, DIKU) for creating and compiling these slide sets



Questions so far?



What should we learn today?



- Identify the key concepts in networking of protocols, layering, resource allocation, and naming.
- Describe what a protocol is and the main issues in protocol design.
- Explain the goal of layering in networked systems and the multiple layers in the Internet protocol suite.
- Explain how circuit switching and packet switching address the resource allocation problem in networks.
- Explain Time-Division Multiplexing (TDM) and Frequency-Division Multiplexing (FDM) and predict transmission time in circuit switched networks.
- Explain the advantages and challenges of packet switching and store-and-forward, and identify the components of transfer time, including processing, queueing, transmission, and propagation.
- Predict transfer time in such store-and-forward networks.
- Explain the notion of throughput in store-and-forward networks and predict throughput in specific scenarios.



Networking is Relevant



Information wants to be free because it has become so cheap to distribute, copy, and recombine... It wants to be expensive because it can be immeasurably valuable to the recipient. (1985)

You Tube

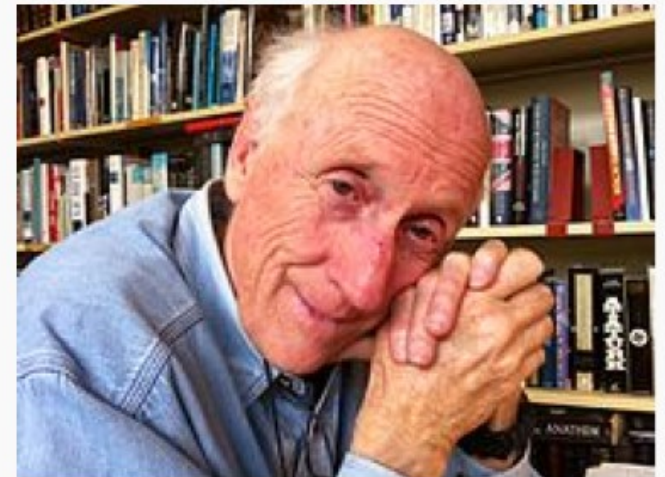
Google news

WIKIPEDIA

facebook

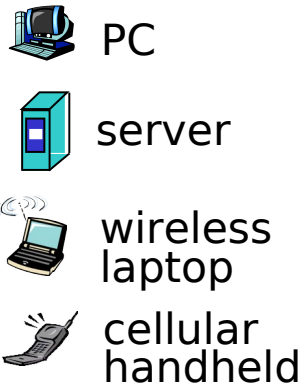
twitter

Stewart Brand

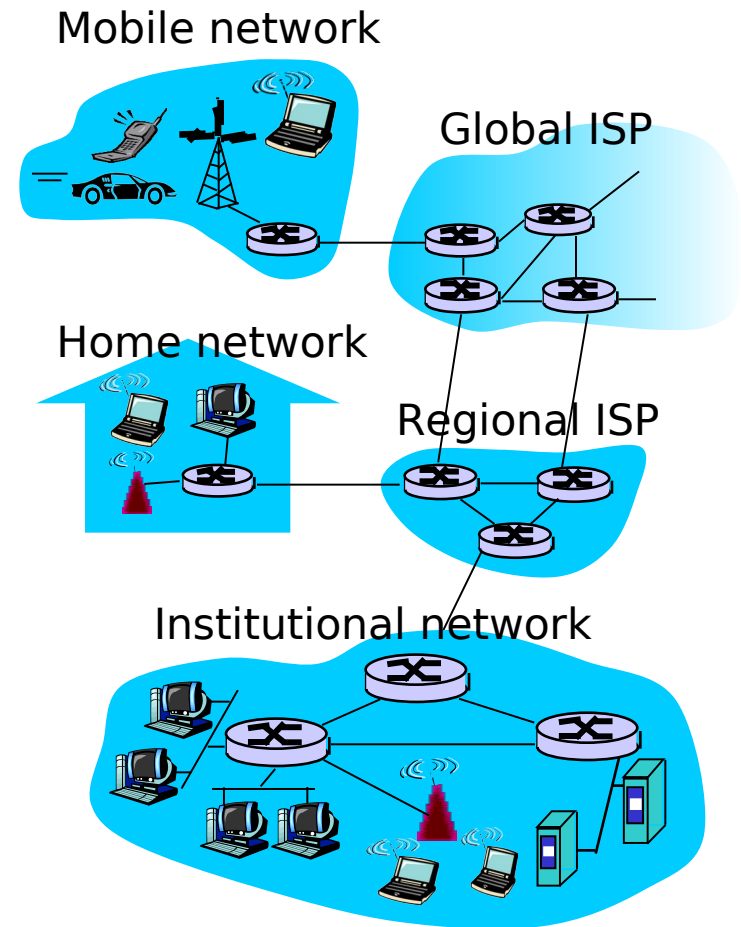


Source: Freedman

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



- A network of networks
- Millions of connected computing devices: *hosts = end systems*
 - running *network apps*
- *communication links*
 - fiber, copper, radio, satellite
 - transmission rate = *bandwidth*
- *routers*: forward packets (chunks of data)



Key Concepts in Networking

- **Protocols**
 - Speaking the same language
 - Syntax and semantics
- **Layering**
 - Standing on the shoulders of giants
 - A key to managing complexity
- **Resource allocation**
 - Dividing scarce resources among competing parties
 - Memory, link bandwidth, wireless spectrum, paths
- **Naming**
 - What to call computers, services, protocols, ...



- Protocols

- Speaking the same language
- Syntax and semantics

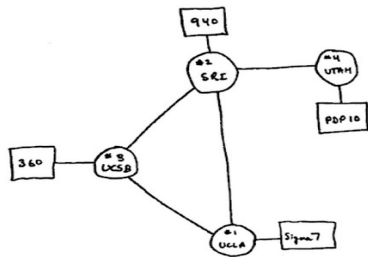


Diagram illustrating a network structure, likely a communication or data network, showing various nodes and connections. The nodes are labeled with names and codes, and the connections are represented by lines.

Legend:

- IMP (Impulse Modulation)
- PLURIBUS IMP (Pluribus Impulse Modulation)
- TIP (Time Interleaved Packet)
- SATELLITE CIRCUIT

Key Nodes and Connections:

- POP-10** (PLI) connects to **MOFFETT** (DEC 1000, DEC 8000).
- POP-11** connects to **STAN** (ILLINOIS) and **WPAFB** (POP-10).
- POP-12** connects to **DATA** (COMPUTER, POP-10) and **POP-13** (DEC-2050).
- POP-14** connects to **MITF** (POP-10) and **MIT 64** (POP-10).
- POP-15** connects to **MIT 64** (POP-10) and **MIT 44** (POP-10).
- POP-16** connects to **MIT 44** (POP-10) and **MIT 30** (POP-10).
- POP-17** connects to **MIT 30** (POP-10) and **MIT 20** (POP-10).
- POP-18** connects to **MIT 20** (POP-10) and **MIT 10** (POP-10).
- POP-19** connects to **MIT 10** (POP-10) and **MIT 0** (POP-10).
- POP-20** connects to **MIT 0** (POP-10) and **MIT 10** (POP-10).
- POP-21** connects to **MIT 10** (POP-10) and **MIT 20** (POP-10).
- POP-22** connects to **MIT 20** (POP-10) and **MIT 30** (POP-10).
- POP-23** connects to **MIT 30** (POP-10) and **MIT 40** (POP-10).
- POP-24** connects to **MIT 40** (POP-10) and **MIT 50** (POP-10).
- POP-25** connects to **MIT 50** (POP-10) and **MIT 60** (POP-10).
- POP-26** connects to **MIT 60** (POP-10) and **MIT 70** (POP-10).
- POP-27** connects to **MIT 70** (POP-10) and **MIT 80** (POP-10).
- POP-28** connects to **MIT 80** (POP-10) and **MIT 90** (POP-10).
- POP-29** connects to **MIT 90** (POP-10) and **MIT 100** (POP-10).
- POP-30** connects to **MIT 100** (POP-10) and **MIT 110** (POP-10).
- POP-31** connects to **MIT 110** (POP-10) and **MIT 120** (POP-10).
- POP-32** connects to **MIT 120** (POP-10) and **MIT 130** (POP-10).
- POP-33** connects to **MIT 130** (POP-10) and **MIT 140** (POP-10).
- POP-34** connects to **MIT 140** (POP-10) and **MIT 150** (POP-10).
- POP-35** connects to **MIT 150** (POP-10) and **MIT 160** (POP-10).
- POP-36** connects to **MIT 160** (POP-10) and **MIT 170** (POP-10).
- POP-37** connects to **MIT 170** (POP-10) and **MIT 180** (POP-10).
- POP-38** connects to **MIT 180** (POP-10) and **MIT 190** (POP-10).
- POP-39** connects to **MIT 190** (POP-10) and **MIT 200** (POP-10).
- POP-40** connects to **MIT 200** (POP-10) and **MIT 210** (POP-10).
- POP-41** connects to **MIT 210** (POP-10) and **MIT 220** (POP-10).
- POP-42** connects to **MIT 220** (POP-10) and **MIT 230** (POP-10).
- POP-43** connects to **MIT 230** (POP-10) and **MIT 240** (POP-10).
- POP-44** connects to **MIT 240** (POP-10) and **MIT 250** (POP-10).
- POP-45** connects to **MIT 250** (POP-10) and **MIT 260** (POP-10).
- POP-46** connects to **MIT 260** (POP-10) and **MIT 270** (POP-10).
- POP-47** connects to **MIT 270** (POP-10) and **MIT 280** (POP-10).
- POP-48** connects to **MIT 280** (POP-10) and **MIT 290** (POP-10).
- POP-49** connects to **MIT 290** (POP-10) and **MIT 300** (POP-10).
- POP-50** connects to **MIT 300** (POP-10) and **MIT 310** (POP-10).
- POP-51** connects to **MIT 310** (POP-10) and **MIT 320** (POP-10).
- POP-52** connects to **MIT 320** (POP-10) and **MIT 330** (POP-10).
- POP-53** connects to **MIT 330** (POP-10) and **MIT 340** (POP-10).
- POP-54** connects to **MIT 340** (POP-10) and **MIT 350** (POP-10).
- POP-55** connects to **MIT 350** (POP-10) and **MIT 360** (POP-10).
- POP-56** connects to **MIT 360** (POP-10) and **MIT 370** (POP-10).
- POP-57** connects to **MIT 370** (POP-10) and **MIT 380** (POP-10).
- POP-58** connects to **MIT 380** (POP-10) and **MIT 390** (POP-10).
- POP-59** connects to **MIT 390** (POP-10) and **MIT 400** (POP-10).
- POP-60** connects to **MIT 400** (POP-10) and **MIT 410** (POP-10).
- POP-61** connects to **MIT 410** (POP-10) and **MIT 420** (POP-10).
- POP-62** connects to **MIT 420** (POP-10) and **MIT 430** (POP-10).
- POP-63** connects to **MIT 430** (POP-10) and **MIT 440** (POP-10).
- POP-64** connects to **MIT 440** (POP-10) and **MIT 450** (POP-10).
- POP-65** connects to **MIT 450** (POP-10) and **MIT 460** (POP-10).
- POP-66** connects to **MIT 460** (POP-10) and **MIT 470** (POP-10).
- POP-67** connects to **MIT 470** (POP-10) and **MIT 480** (POP-10).
- POP-68** connects to **MIT 480** (POP-10) and **MIT 490** (POP-10).
- POP-69** connects to **MIT 490** (POP-10) and **MIT 500** (POP-10).
- POP-70** connects to **MIT 500** (POP-10) and **MIT 510** (POP-10).
- POP-71** connects to **MIT 510** (POP-10) and **MIT 520** (POP-10).
- POP-72** connects to **MIT 520** (POP-10) and **MIT 530** (POP-10).
- POP-73** connects to **MIT 530** (POP-10) and **MIT 540** (POP-10).
- POP-74** connects to **MIT 540** (POP-10) and **MIT 550** (POP-10).
- POP-75** connects to **MIT 550** (POP-10) and **MIT 560** (POP-10).
- POP-76** connects to **MIT 560** (POP-10) and **MIT 570** (POP-10).
- POP-77** connects to **MIT 570** (POP-10) and **MIT 580** (POP-10).
- POP-78** connects to **MIT 580** (POP-10) and **MIT 590** (POP-10).
- POP-79** connects to **MIT 590** (POP-10) and **MIT 600** (POP-10).
- POP-80** connects to **MIT 600** (POP-10) and **MIT 610** (POP-10).
- POP-81** connects to **MIT 610** (POP-10) and **MIT 620** (POP-10).
- POP-82** connects to **MIT 620** (POP-10) and **MIT 630** (POP-10).
- POP-83** connects to **MIT 630** (POP-10) and **MIT 640** (POP-10).
- POP-84** connects to **MIT 640** (POP-10) and **MIT 650** (POP-10).
- POP-85** connects to **MIT 650** (POP-10) and **MIT 660** (POP-10).
- POP-86** connects to **MIT 660** (POP-10) and **MIT 670** (POP-10).
- POP-87** connects to **MIT 670** (POP-10) and **MIT 680</**

1998

All speak IPv4

“Internet Protocol version 4”

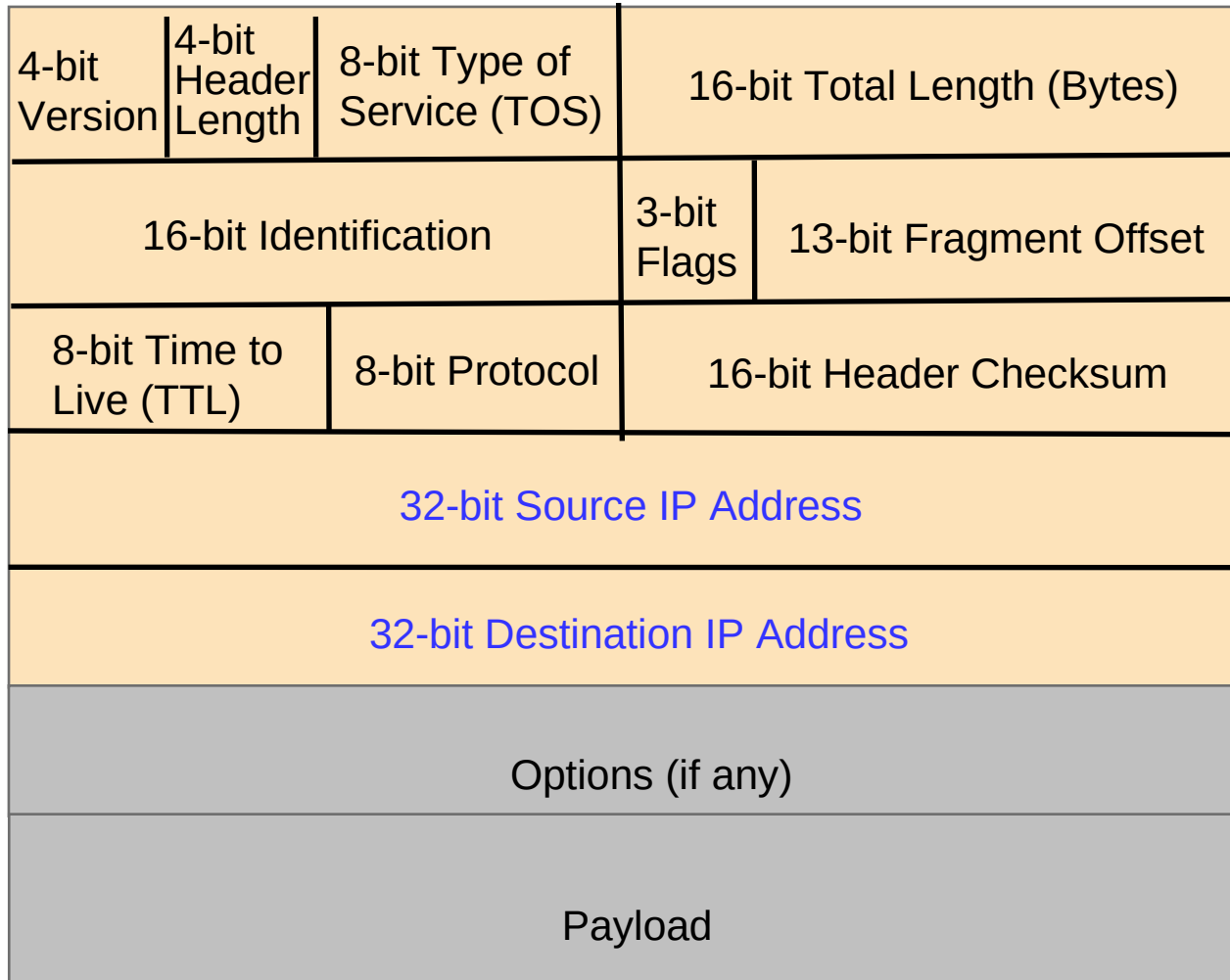


Protocol design is about tradeoffs

- **How should hosts and routers communicate?**
 - Standard protocol
 - Fast: Machine readable in hardware at line rates
- **Browsers, web servers, and proxies?**
 - Can be slower: software readable
 - Human readable
 - Extensible and forward-compatible
 - Not everybody might be familiar with extensions



IPv4 Packet



20-byte header

Example: HyperText Transfer Protocol

```
GET /courses/archive/spr09/cos461/ HTTP/1.1
Host: www.cs.princeton.edu
User-Agent: Mozilla/4.03
CRLF
```

Request

```
HTTP/1.1 200 OK
Date: Mon, 2 Feb 2009 13:09:03 GMT
Server: Netscape-Enterprise/3.5.1
Last-Modified: Mon, 21 Feb 2009 11:12:23 GMT
Content-Length: 42
CRLF
Site under construction
```

Response



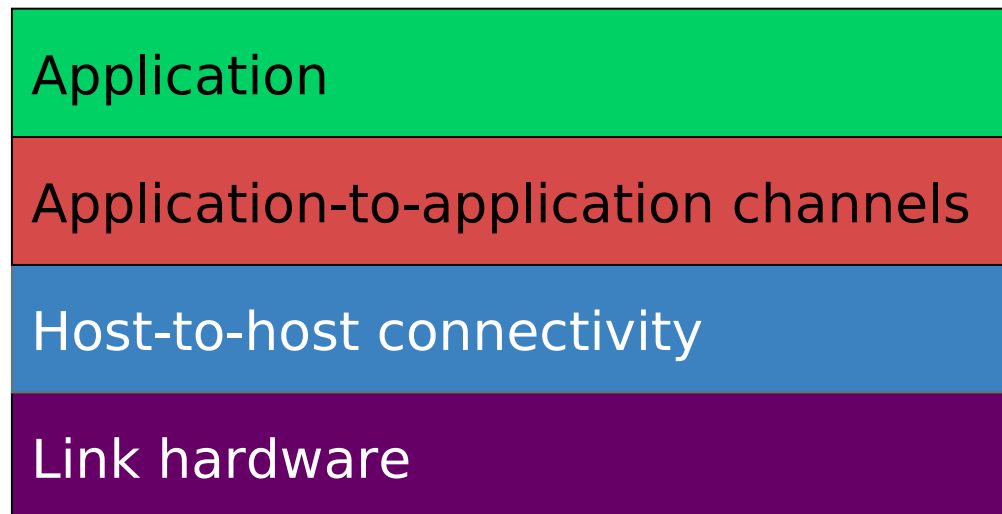
Key Concepts in Networking

- **Protocols**
 - Speaking the same language
 - Syntax and semantics
- **Layering**
 - Standing on the shoulders of giants
 - A key to managing complexity

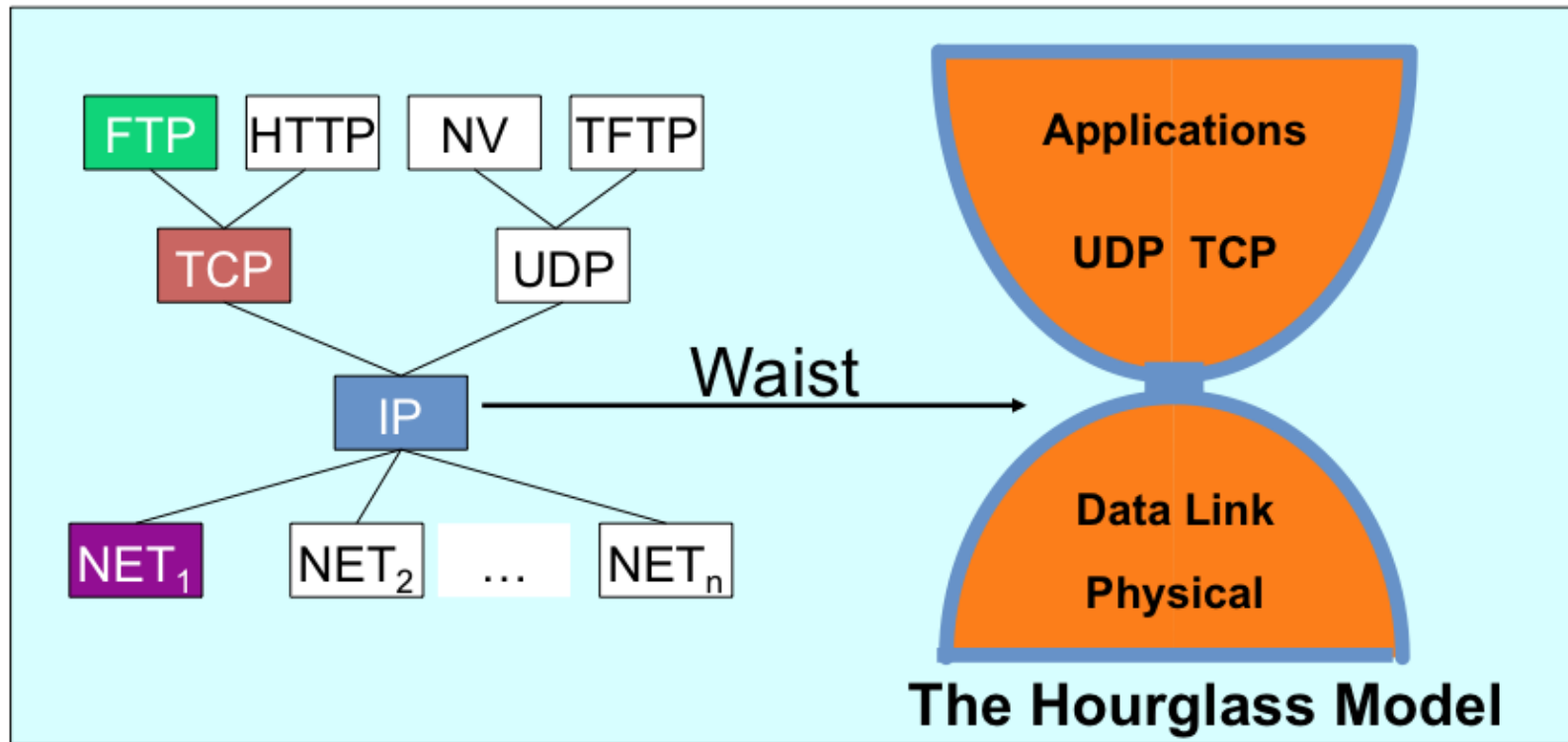


Layering = Functional Abstraction

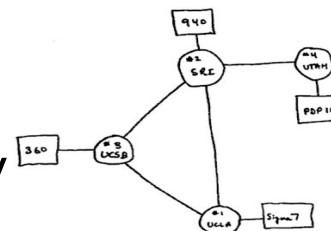
- Sub-divide the problem
 - Each layer relies on services from layer below
 - Each layer exports services to layer above
- Interface between layers defines interaction
 - Hides implementation details
 - Layers can change without disturbing other layers



The Internet Protocol Suite

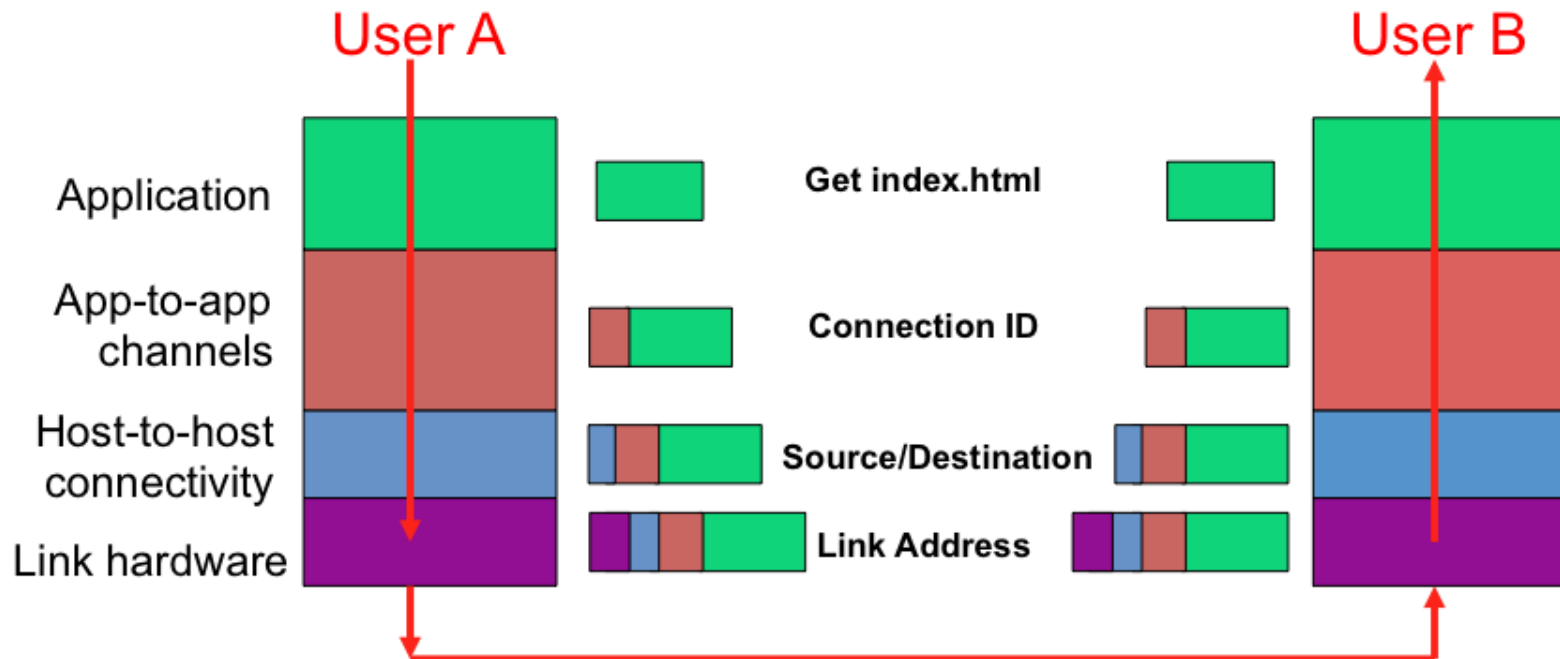
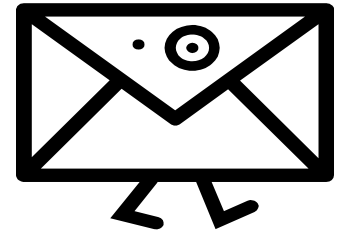


The waist facilitates interoperability

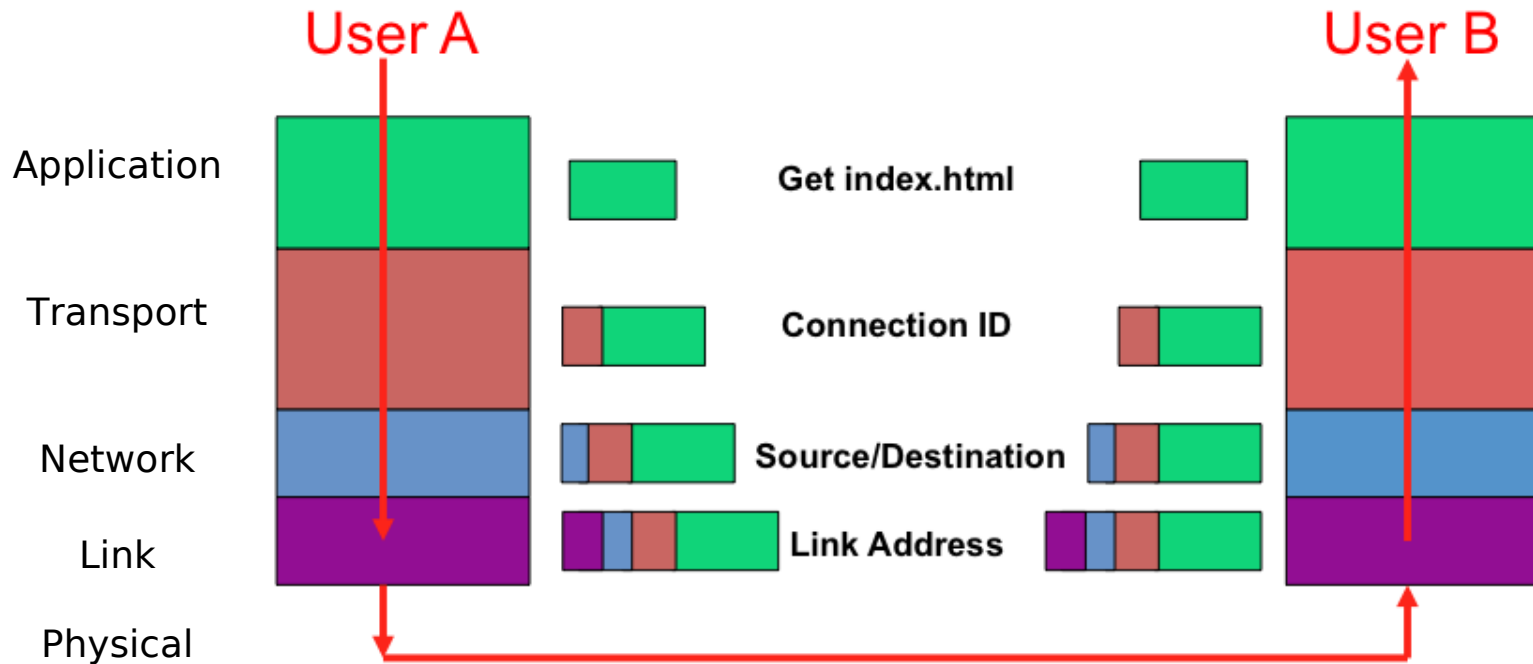
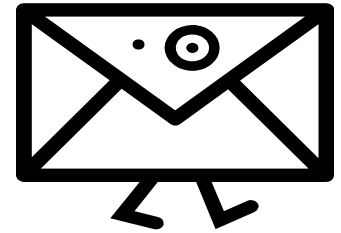


Source: Freedman

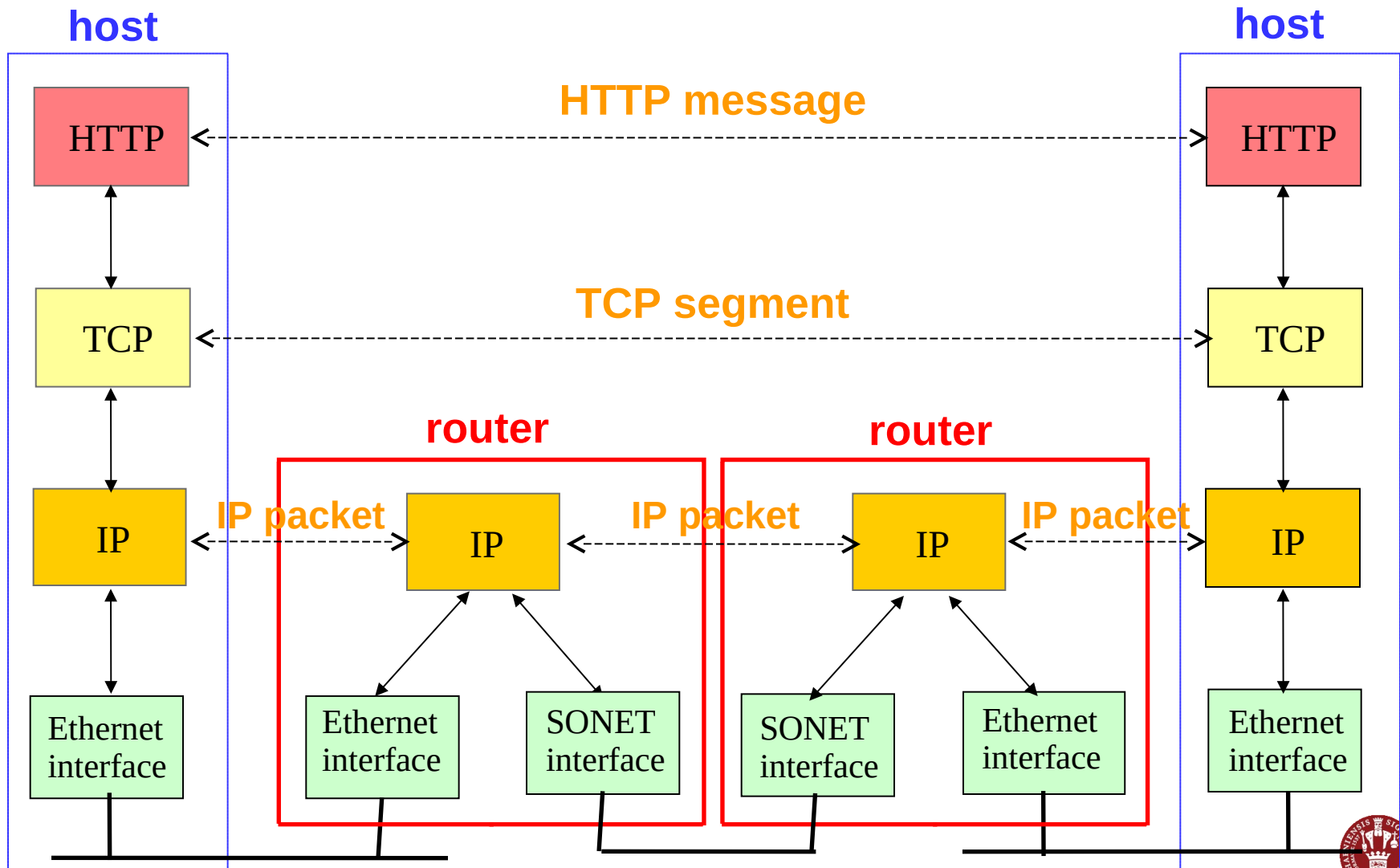
Layer Encapsulation in HTTP



Layer Encapsulation in HTTP



IP Suite: End Hosts vs. Routers



Key Concepts in Networking

- **Protocols**
 - Speaking the same language
 - Syntax and semantics
- **Layering**
 - Standing on the shoulders of giants
 - A key to managing complexity
- **Resource allocation**
 - Dividing scarce resources among competing parties
 - Memory, link bandwidth, wireless spectrum, paths

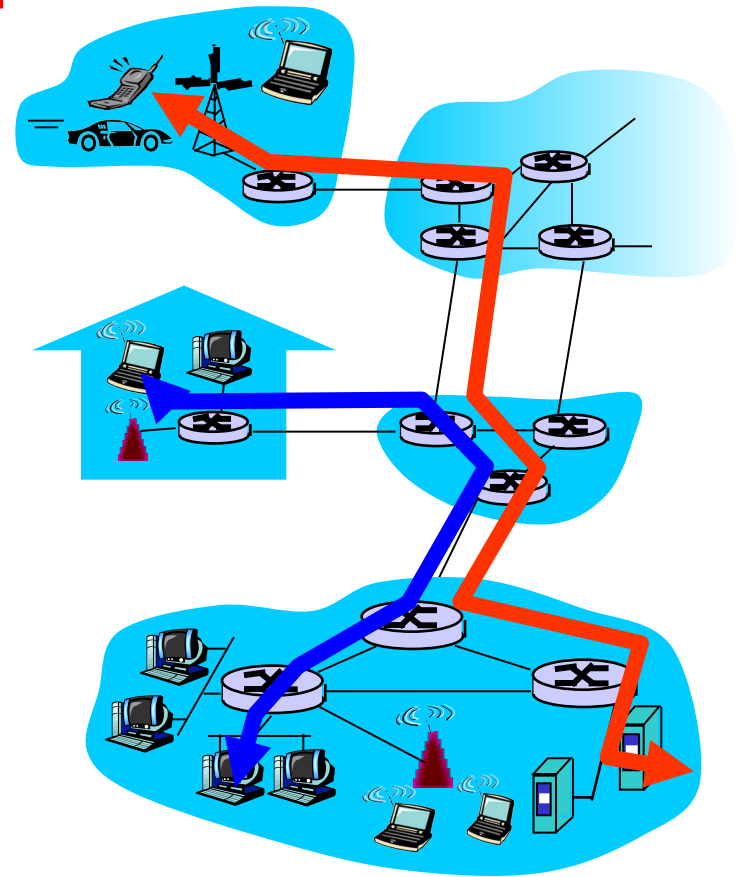
First Example: circuit vs. packet switching



Network Core: Circuit Switching

end-end resources reserved
for “call”

- link bandwidth, switch capacity
- dedicated resources: no sharing
- circuit-like (guaranteed) performance
- call setup required

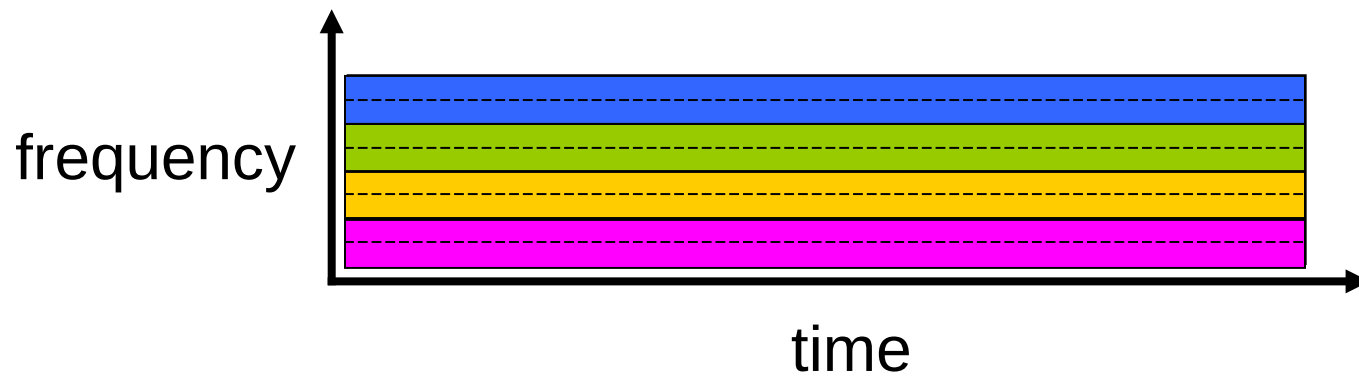


Circuit Switching: FDM and TDM

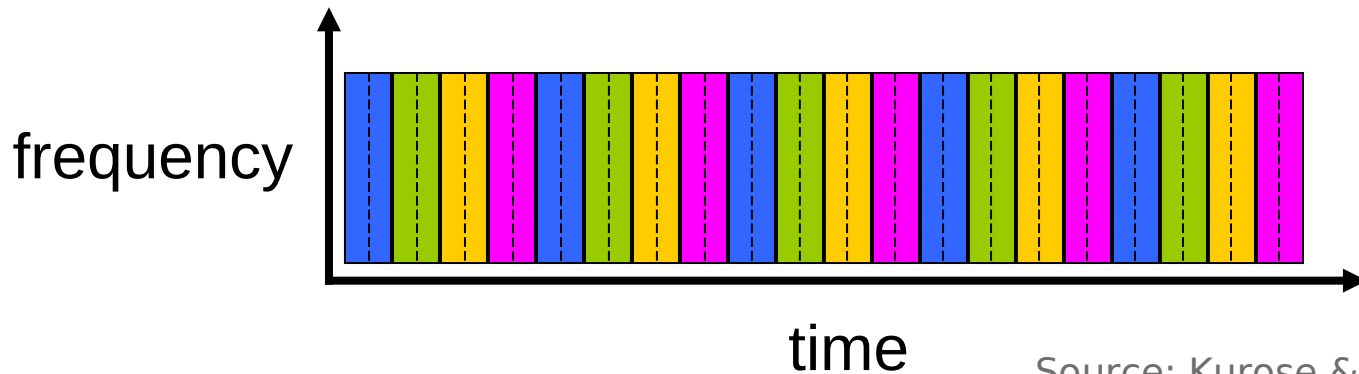
FDM

Example:

4 users



TDM



Numerical example

- How long does it take to send a file of 640,000 bits from host A to host B over a circuit-switched network?
 - all link speeds: 1536 Mbps
 - each link uses TDM with 24 slots/sec
 - 500 msec to establish end-to-end circuit
 - Note: 1 Mbps = 10^6 bps
- Let's work it out!
- Possible answers
 - (a) 500 msec
 - (b) 500.4 msec
 - (c) 510 msec
 - (d) 1 sec



Numerical example

- How long does it take to send a file of 640,000 bits from host A to host B over a circuit-switched network?

- all link speeds: 1536 Mbps
- each link uses TDM with 24 slots/sec
- 500 msec to establish end-to-end circuit
- Note: 1 Mbps = 10^6 bps

- Let's work it out!
- Possible answers

(a) 500 msec

(b) 500.4 msec

(c) 510 msec (Assuming only 1 link)

(d) 1 sec

$$d = N(L/R)$$



Network Core: Packet Switching

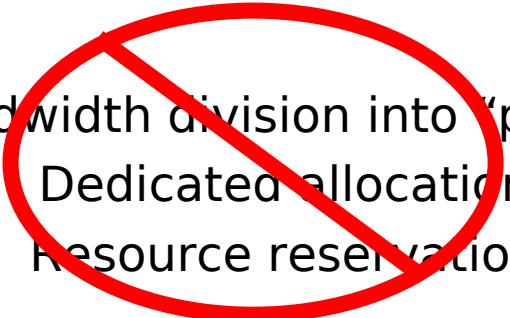
each end-end data stream divided into packets

- user A, B packets share network resources
- each packet uses full link bandwidth
- resources used as needed

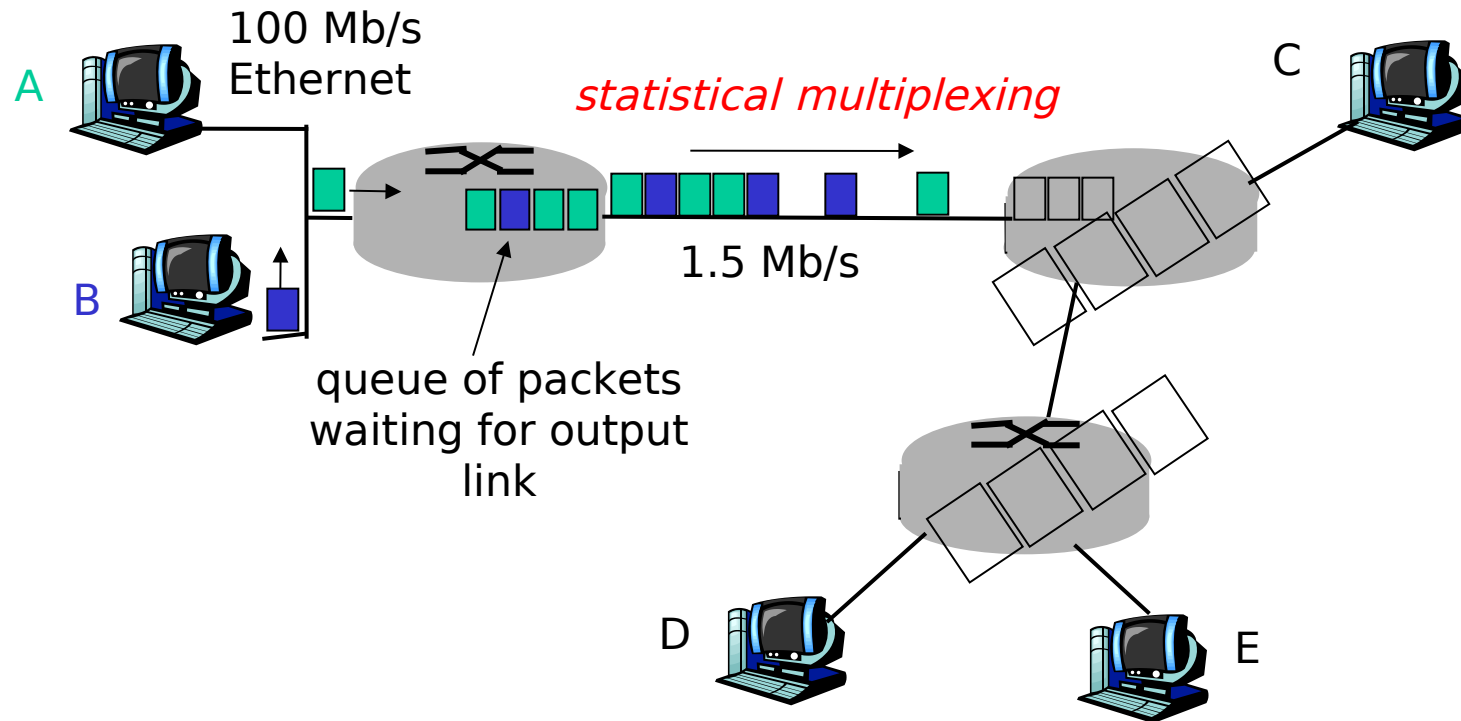
resource contention:

- aggregate resource demand can exceed amount available
- congestion: packets queue, wait for link use
- store and forward: packets move one hop at a time
 - node receives complete packet before forwarding

Bandwidth division into “pieces”
Dedicated allocation
Resource reservation



Packet Switching: Statistical Multiplexing



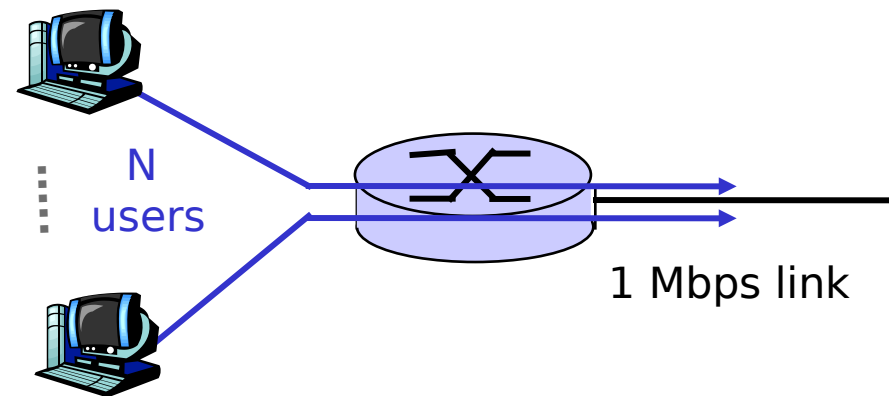
- sequence of A & B packets has no fixed timing pattern
 - bandwidth shared on demand: **statistical multiplexing**.
- TDM: each host gets same slot in revolving TDM frame.

Packet switching versus circuit switching

Packet switching allows more users to use network!

Example:

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



•circuit-switching:

- 10 users, utilization?

•packet switching:

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

Q: How did we get value 0.0004?

A: $P(N=10) =$

$$C(35,10) * P(A)^{10} * (1-P(A))^{(35-10)}$$

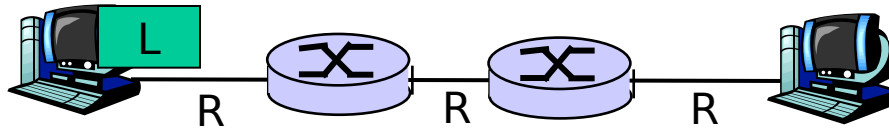
$P(N>10) \rightarrow$ sum the above for $N=10..35$

Q: what happens if > 35 users ?



Source: Kurose & Ross (partial)

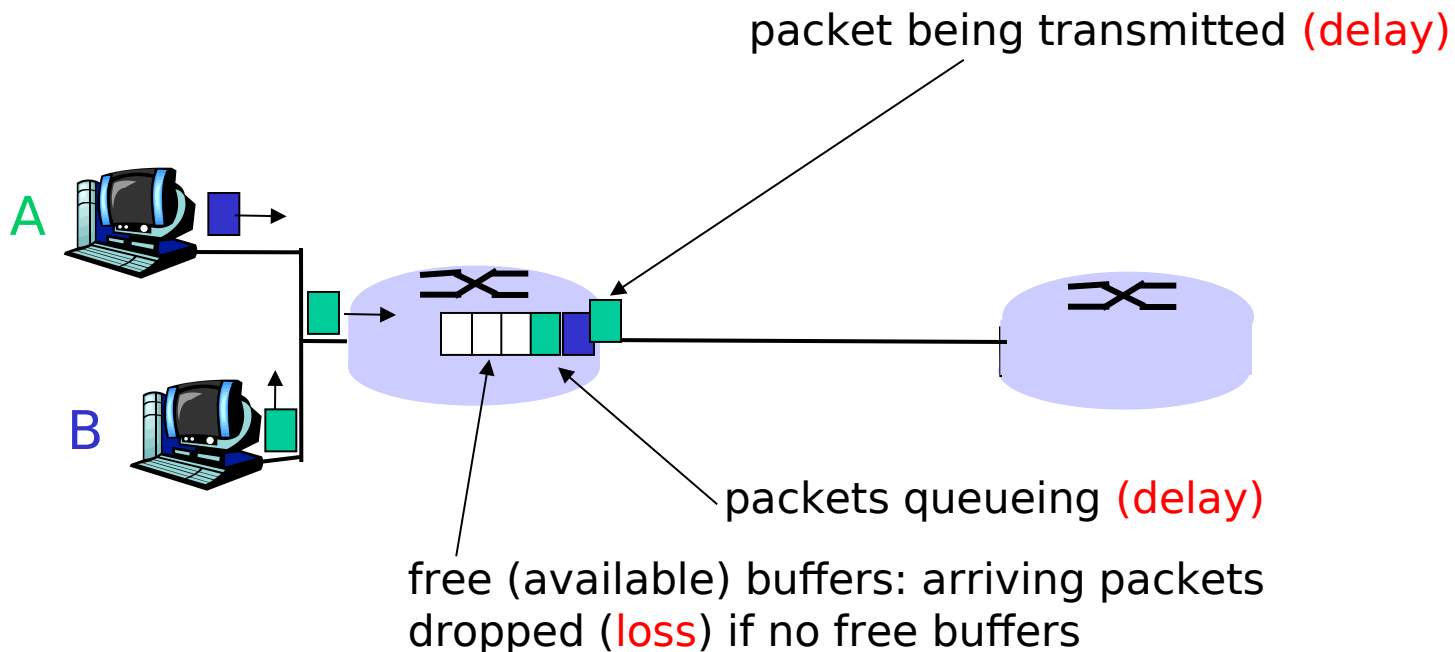
Packet-switching: store-and-forward



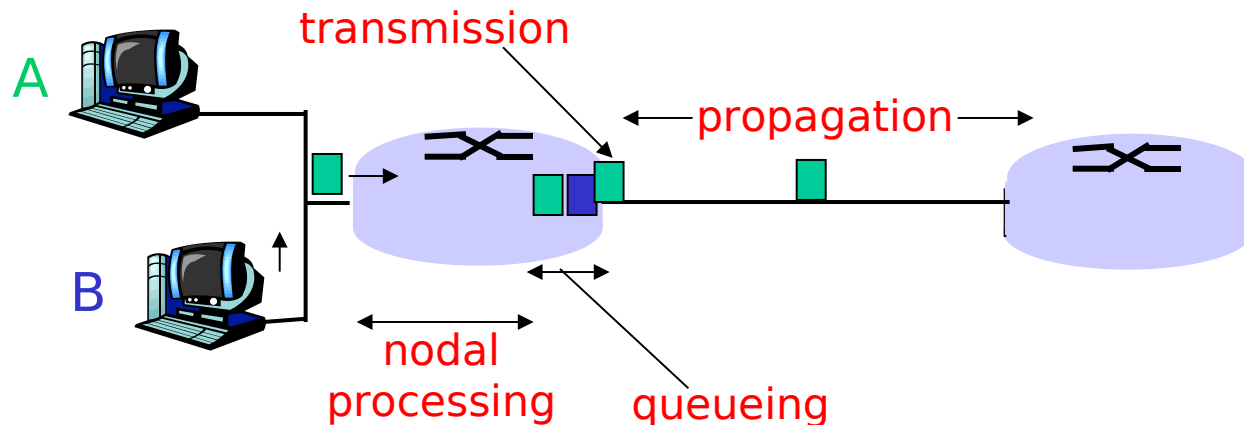
- takes L/R seconds to transmit (push out) packet of L bits on to link at R bps
- **store and forward:** entire packet must arrive at router before it can be transmitted on next link
- delay = $3L/R$ (assuming zero propagation delay) } more on delay shortly ...
- **Example:**
 - $L = 7.5$ Mbits
 - $R = 1.5$ Mbps
 - transmission delay =
 - (a) 5 sec
 - (b) 10 sec
 - (c) 15 sec
 - (d) 20 sec

How do loss and delay occur?

- packets queue in router buffers
- packet arrival rate to link exceeds output link capacity
- packets queue, wait for turn



Four sources of packet delay



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

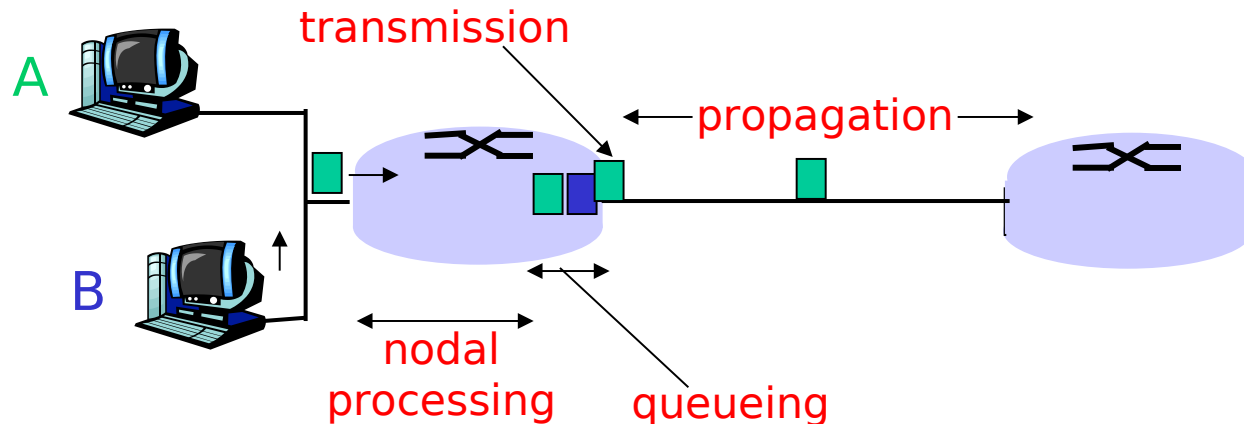
d_{proc} : nodal processing

- 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



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

d_{trans} : transmission delay

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

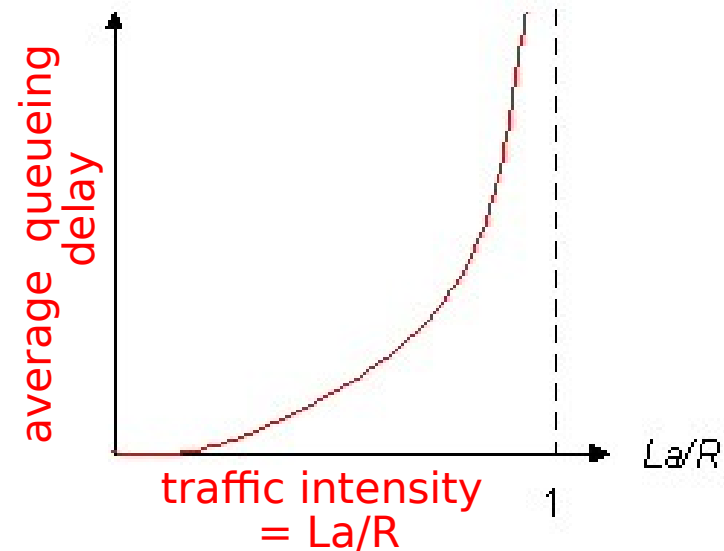
d_{prop} : propagation delay

- d: length of physical link
- s: propagation speed in medium ($\sim 2 \times 10^8$ m/sec)
- $d_{\text{prop}} = d/s$

d_{trans} and d_{prop}
very different

Queuing delay

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



- $La/R \sim 0$: avg. queuing delay small
- $La/R \rightarrow 1$: avg. queuing delay large
- $La/R > 1$: more “work” arriving than can be serviced, average delay infinite!



$La/R \sim 0$

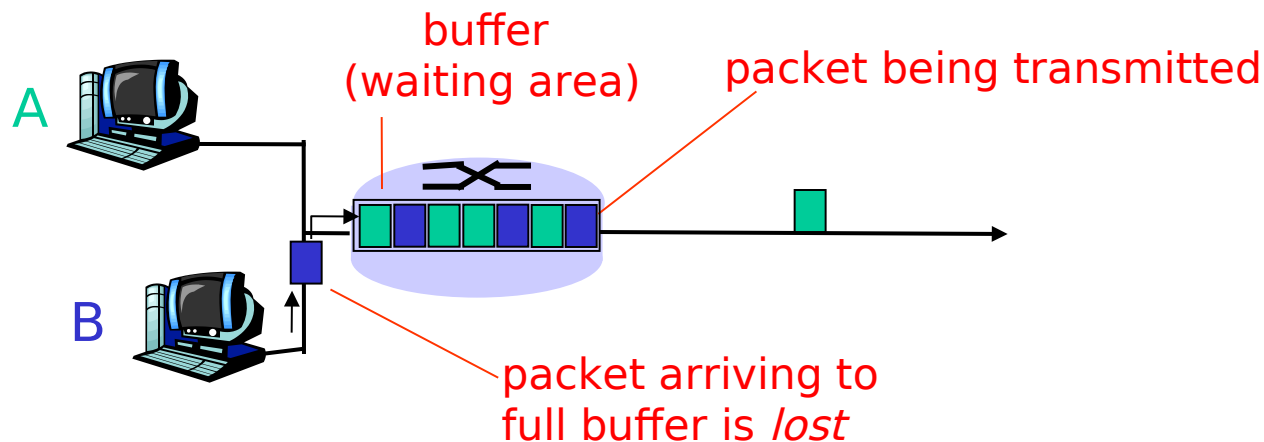


$La/R \rightarrow 1$

Source: Kurose & Ross

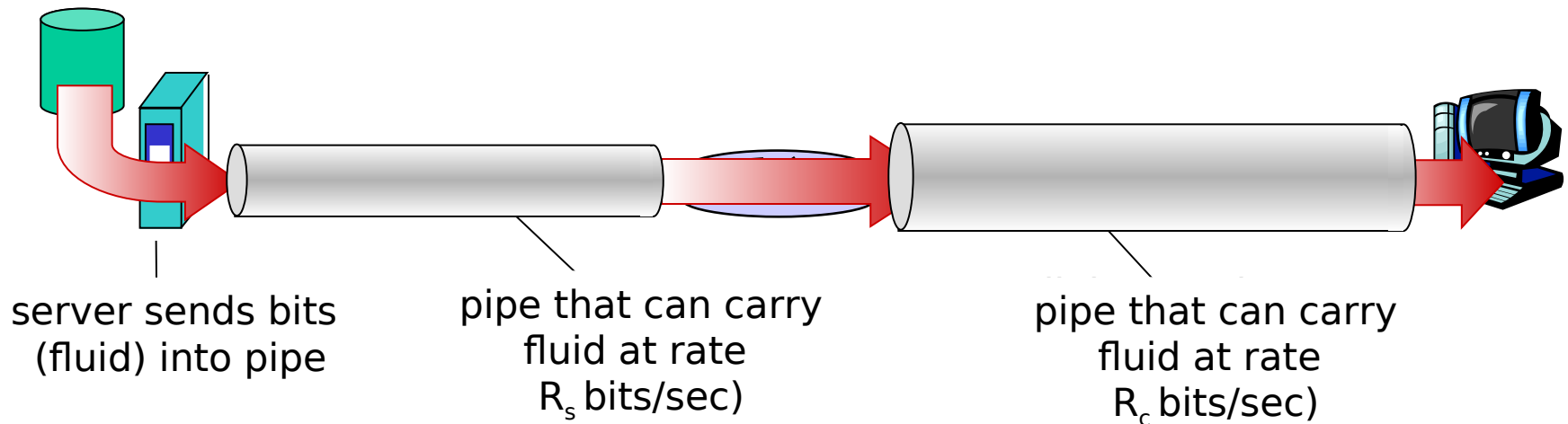
Packet Loss

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



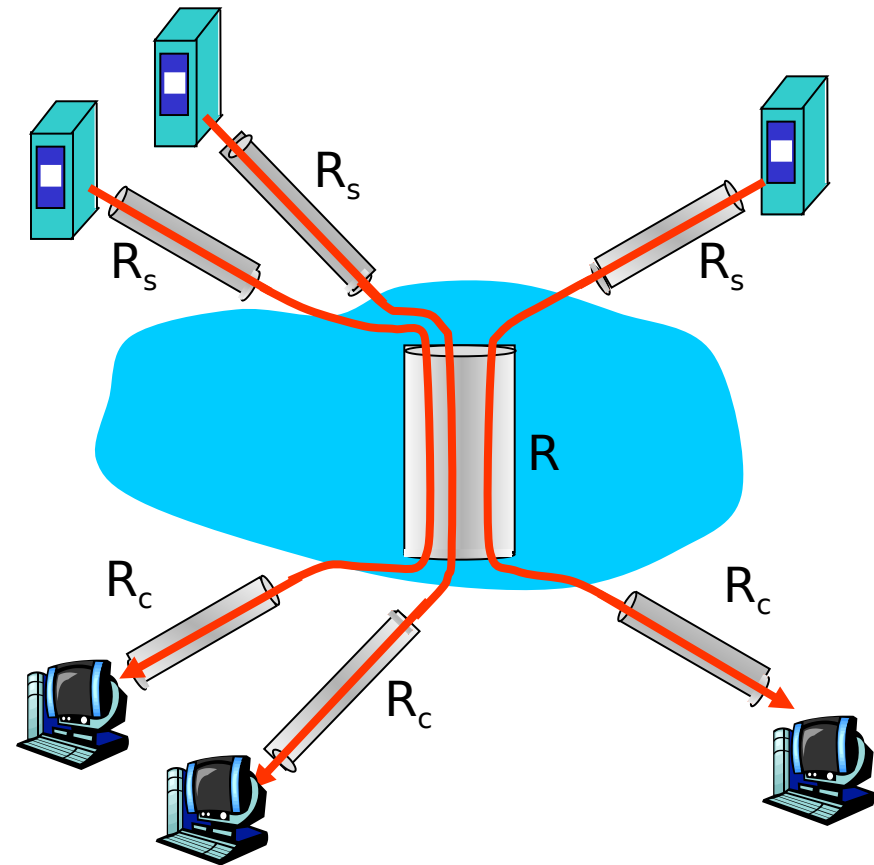
Throughput

- **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: 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

Source: Kurose & Ross

Summary

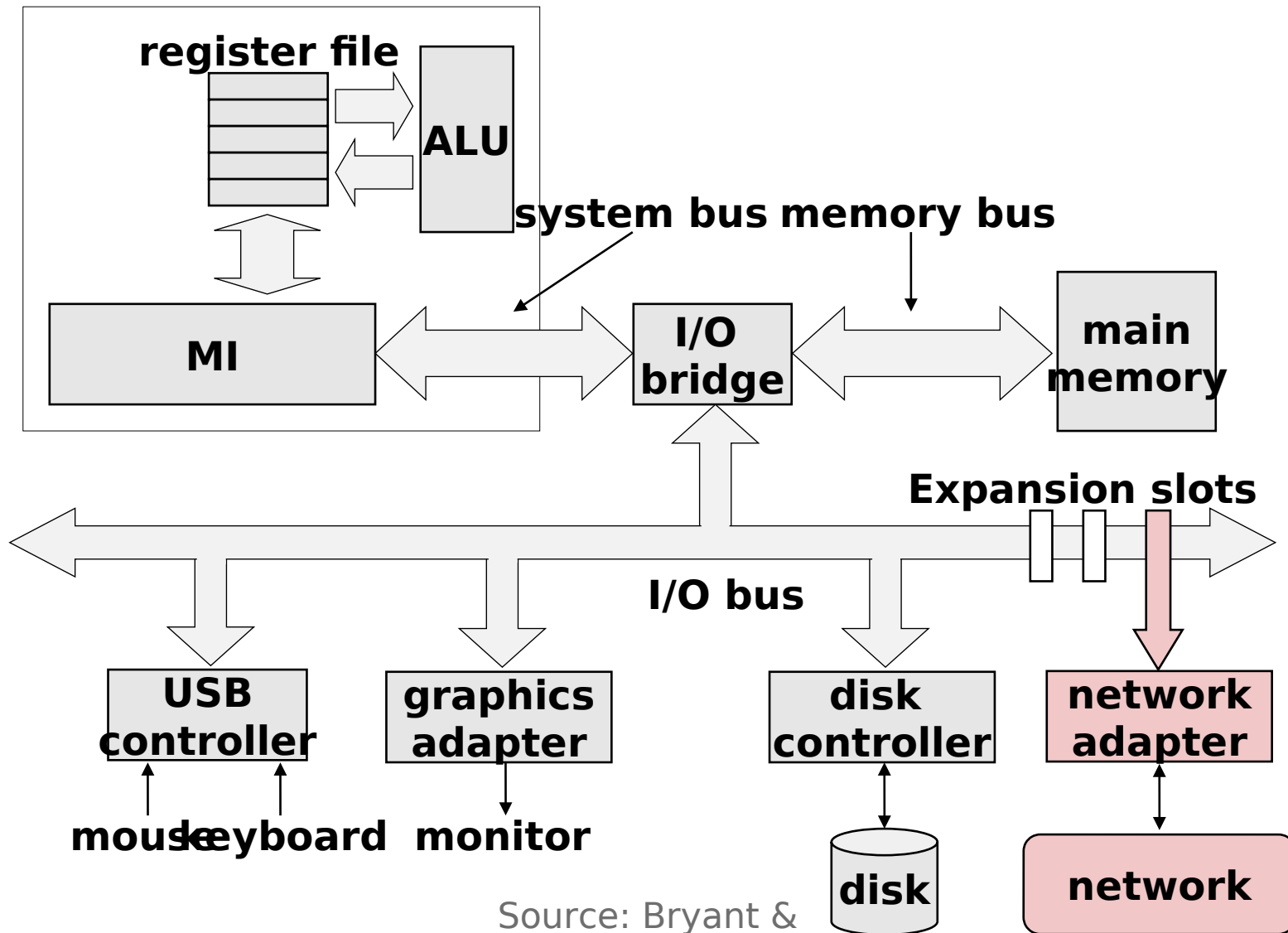
- Internet is a network of networks
 - Inter-operability, power to the edges
- Key concepts in networking
 - Protocols, layers, resource allocation, and naming
- Circuit switching
 - FDM
 - TDM
- Packet switching
 - Store-and-forward
 - Delay, loss, throughput



What's next ? What to program on the network ?



What's next ? How to program the network ?



Source: Bryant & Halloran

