COS 461: Computer Networks

Spring 2009 (MW 1:30-2:50 in CS 105)

Mike Freedman

Teaching Assistants: Wyatt Lloyd and Jeff Terrace

http://www.cs.princeton.edu/courses/archive/spr09/cos461/

Goals for Today's Class

- COS 461 overview
  - Goals of the course
  - Structure of the course
  - Learning the material
  - Programming assignments
  - Course grading
  - Academic policies
- Key concepts in data networking
  - Protocols
  - Layering
  - Resource allocation
  - Naming

What You Learn in This Course

- Skill: network programming
  - Socket programming
  - Designing and implementing protocols
- Knowledge: how the Internet works
  - IP protocol suite
  - Internet architecture
  - Applications (Web, DNS, P2P, …)
- Insight: key concepts in networking
  - Protocols
  - Layering
  - Resource allocation
  - Naming

Structure of the Course (1st Half)

- Start at the top
  - Sockets: how applications view the Internet
  - Protocols: essential elements of a protocol
- Then study the “narrow waist” of IP
  - IP best-effort packet-delivery service
  - IP addressing and packet forwarding
- And how to build on top of the narrow waist
  - Transport protocols (TCP, UDP)
  - Domain Name System (DNS)
  - Glue (ARP, DHCP, ICMP)
  - End-system security and privacy (NAT, firewalls)
- Looking underneath IP
  - Link technologies (Ethernet)

Structure of the Course (2nd Half)

- And how to get the traffic from here to there
  - Internet routing architecture (the “inter” in Internet)
  - Intradomain and interdomain routing protocols
- Building applications
  - DNS and HTTP
  - Content-distribution networks
  - Peer-to-peer and distributed hash tables
  - P2P Applications: BitTorrent, etc.
  - Multimedia streaming
- Other approaching to building networks
  - Delay /disruption tolerant networks
  - Multicast, anycast, …

Learning the Material: People

- Lecture (Prof. Mike Freedman)
  - When: MW 1:30-2:50 in Computer Science 105
  - Slides available online at course Web site
  - Office hours (room 208) by appointment
  - Email: mfreed+cos461 at cs.princeton.edu

- Teaching Assistants
  - Wyatt Lloyd
    - Office hours: Mon 3-4pm, Tue 4-5pm (CS 003)
    - E-mail: wlloyd+cos461 at cs.princeton.edu
  - Jeff Terrace
    - Office hours: Wed 3-4pm, Thu 1-2pm (CS 001B)
    - E-mail: jterrace+cos461 at cs.princeton.edu

- Main email: cos461-staff at lists.cs.princeton.edu

Learning the Material: Mailing List

- Mailing list
  - If you're enrolled, you should be on it
  - E-mail: cos461 at lists.cs.princeton.edu
  - Sign up: https://lists.cs.princeton.edu/mailman/listinfo/cos461
- Read often
  - Good place to ask questions
  - But do not post your code ☺
- Reply, too
  - Good place to *answer* questions (participation!)

Learning the Material: Books

- Required textbook
  - *Computer Networks: A Systems Approach (4th edition)*, by Peterson and Davie [Okay to use the 3rd edition]
  - Mostly covers the material in the first half of the class
- Books on reserve
  - Networking text books
    - *Computer Networking: A Top-Down Approach Featuring the Internet (3rd edition),* by Kurose and Ross
    - *Computer Networks (4th edition)*, by Tanenbaum
  - Network programming references
    - *TCP/IP Illustrated, Volume 1: The Protocols*, by Stevens
    - *Unix Network Programming, Volume 1: The Sockets Networking API (3rd Edition)*, by Stevens, Fenner, & Rudolf
- Online resources
  - E.g. on socket programming

Grading and Schedule

- Assignments (10% each)
  - 95% 3 hours, 75% two day late, 50% two days late
  - One free late day during semester
- Two exams (40% total)
  - Midterm exam in week six (20%)
  - Final exam during exam period (20%)
- Class participation (10%)
  - In lecture
  - On the listserv

Programming Assignments

- Five assignments (with target due dates)
  - Introduction to sockets (Sun Feb 15)
  - HTTP proxy (Sun Mar 8)
  - IP router (Sun Apr 12)
  - Firewall (Sun Apr 26)
  - Reliable transport (Tue May 12) – no late accepted

- Last three: Stanford Virtual Network System
  - Constructs virtual network topologies that integrate directly into physical networks
  - Traffic forwarded to your program, running in user space
  - http://yuba.stanford.edu/vns/

Facilities for Programming

- Computer cluster in Friend Center 010
  - Friend Center 010 computers
  - Machines: labpc-XX.cs.princeton.edu for XX of 01 to 30
  - Why: common environment, and access to lab TAs

- Accessing your account
  - If you are enrolled, you have a labpc account
  - Using your OIT login and password

- Logging in to the machines remotely
  - SSH to portal.cs.princeton.edu, and then to FC 010
  - Need a CS account, if you don't have one already
  - https://csguide.cs.princeton.edu/requests/account
  - Request a "class account"

Facilities for Programming

- Other option: your own PC (*not* recommended)
  - Running GNU tools on Linux, or
  - Running GNU tools on Windows, or
  - Running a standard C development environment
  - Development environment not supported by TAs
- Suggestion: test access this week
  - Logging in to the FC 010 cluster
  - Writing and running "Hello World" in C
- Submitting assignments
  - Using Blackboard
  - More details coming soon!

Graduate Students: Two Choices

- Pick one of two options
  - Do the five programming assignments
  - Or, first two assignments, plus research project

- Research projects
  - Networking-related research problem
  - Must have a *systems* programming component
  - Write-up of project due on Dean’s Date
  - Talk to me about a project before spring break

Policies: Write Your Own Code

Programming in an individual creative process much like composition. You must reach your own understanding of the problem and discover a path to its solution. During this time, discussions with friends are encouraged. However, when the time comes to write code that solves the problem, such discussions are no longer appropriate - the program must be your own work.

If you have a question about how to use some feature of C, UNIX, etc., you can certainly ask your friends or the TA, but **do not, under any circumstances, copy another person's program.** Letting someone copy your program or using someone else's code in any form is a **violation of academic regulations**. "Using someone else's code" includes using solutions or partial solutions to assignments provided by commercial web sites, instructors, preceptors, teaching assistants, friends, or students from any previous offering of this course or any other course.

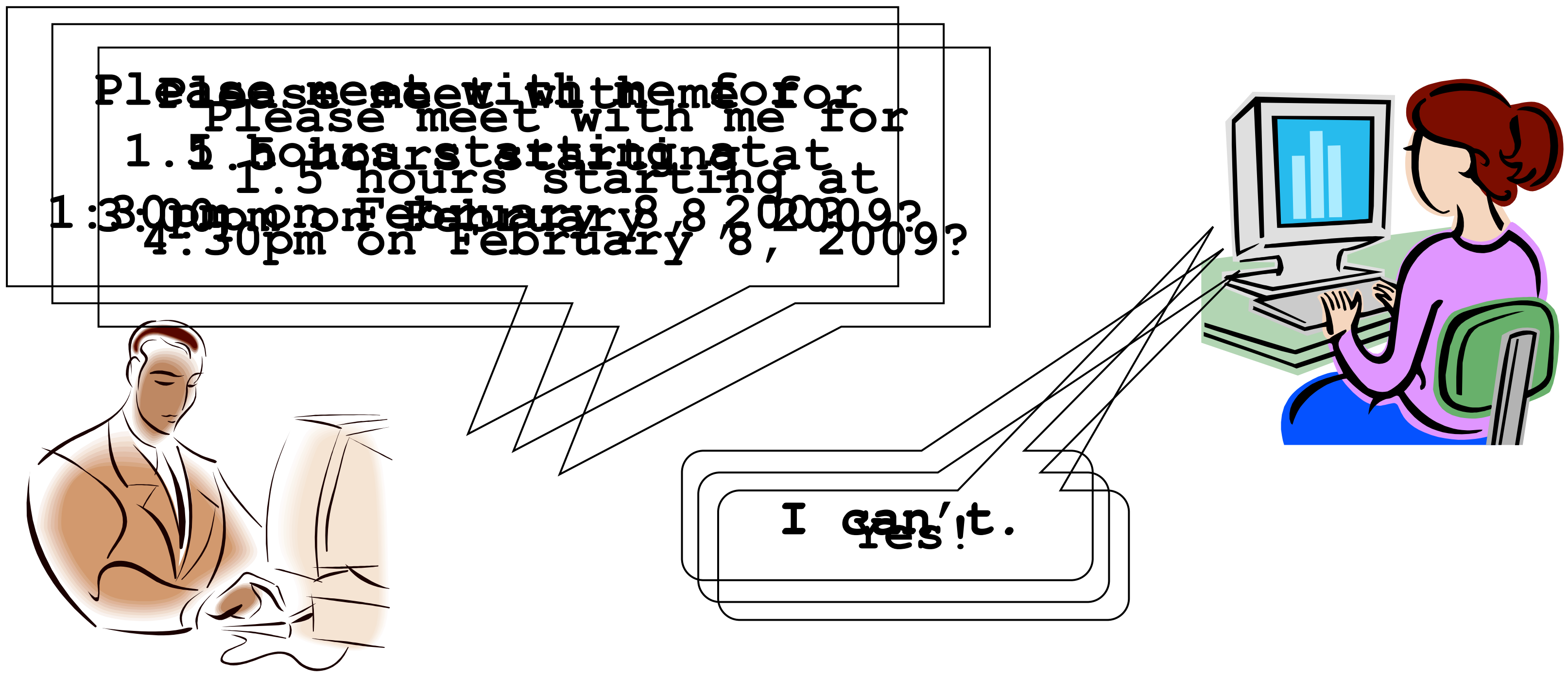
Okay, so let’s get started… with a crash course in data networking

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 scare resources among competing parties
  - Memory, link bandwidth, wireless spectrum, paths, …
  - Distributed vs. centralized algorithms
- Naming
  - What to call computers, services, protocols, …

Protocols: Calendar Service

- Making an appointment with your advisor


Please meet with me for 1.5 hours starting at 1:30pm on February 8, 2009?
Please meet with me for 1.5 hours starting at 3:00pm on February 8, 2009?
Please meet with me for 1.5 hours starting at 4:30pm on February 8, 2009?
I can't.
Yes!

- Specifying the messages that go back and forth
  - – And an understanding of what each party is doing

Okay, So This is Getting Tedious

- You: When are you free to meet for 1.5 hours during the next two weeks?
- Advisor: 10:30am on Feb 8 and 1:15pm on Feb 9.
- You: Book me for 1.5 hours at 10:30am on Feb 8.
- Advisor: Yes.

Well, Not Quite Enough

- Student #1: When can you meet for 1.5 hours during the next two weeks?
- Advisor: 10:30am on Feb 8 and 1:15pm on Feb 9.
- Student #2: When can you meet for 1.5 hours during the next two weeks?
- Advisor: 10:30am on Feb 8 and 1:15pm on Feb 9.
- Student #1: Book me for 1.5 hours at 10:30am on Feb 8.
- Advisor: Yes.
- Student #2: Book me for 1.5 hours at 10:30am on Feb 8.
- Advisor: Uh… well… I can no longer can meet then. I'm free at 1:15pm on Feb 9.
- Book me for 1.5 hours at 1:15pm on Feb 9.
- Advisor: Yes.
- Advisor: Wait…am I talking to Student 1 or 2?

Specifying the Details

- How to identify yourself?
  - Name? Social security number?
- How to represent dates and time?
  - Time, day, month, year? In what time zone?
  - Number of seconds since Jan 1, 1970?
- What granularities of times to use?
  - Any possible start time and meeting duration?
  - Multiples of five minutes?
- How to represent the messages?
  - Strings? Record with name, start time, and duration?
- What do you do if you don’t get a response?
  - Ask again? Reply again?

Example: HyperText Transfer Protocol

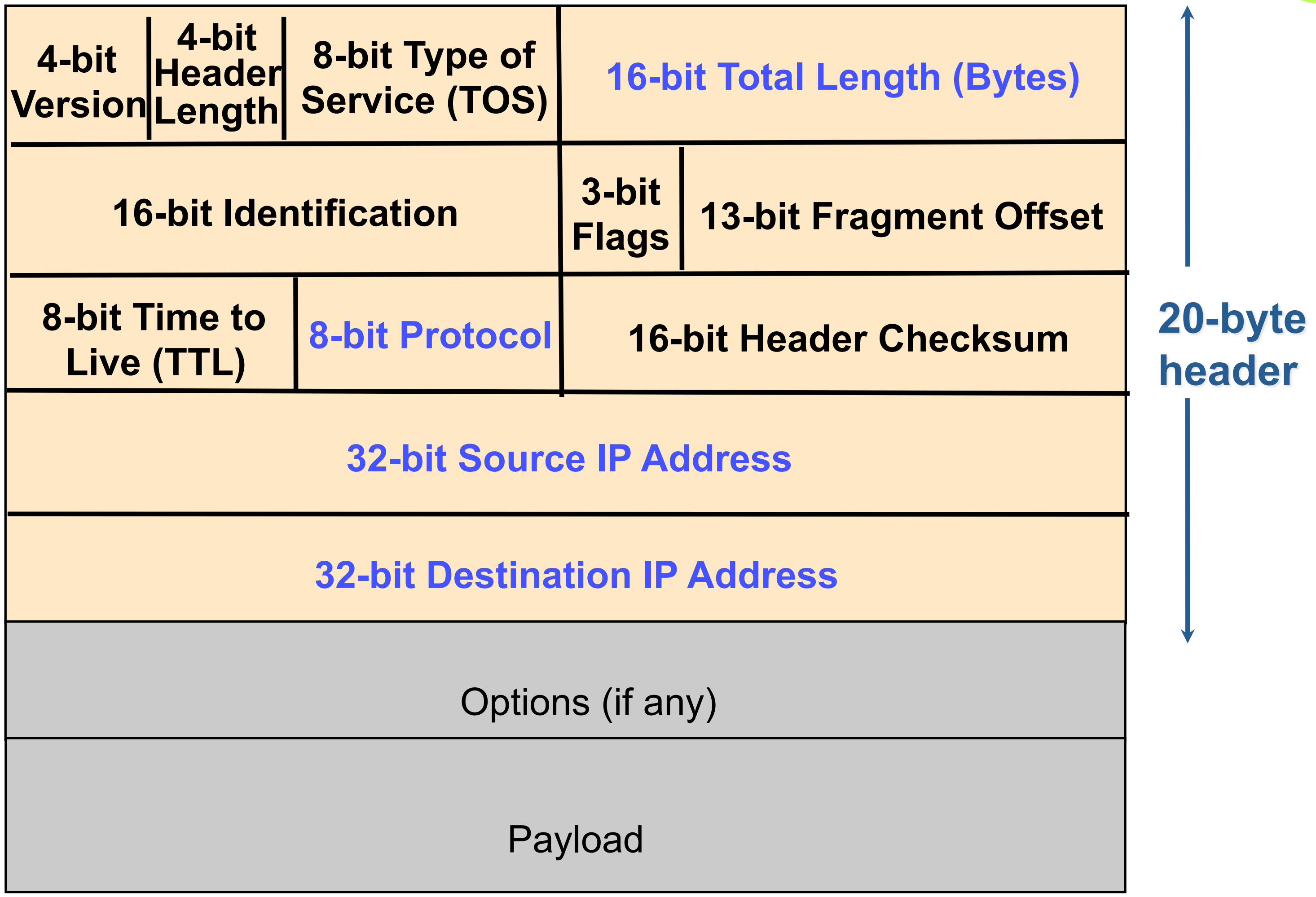
```
GET /courses/archive/spr09/cos461/ HTTP/1.1
Host: www.cs.princeton.edu
User-Agent: Mozilla/4.03
CRLF
```

Request

Response

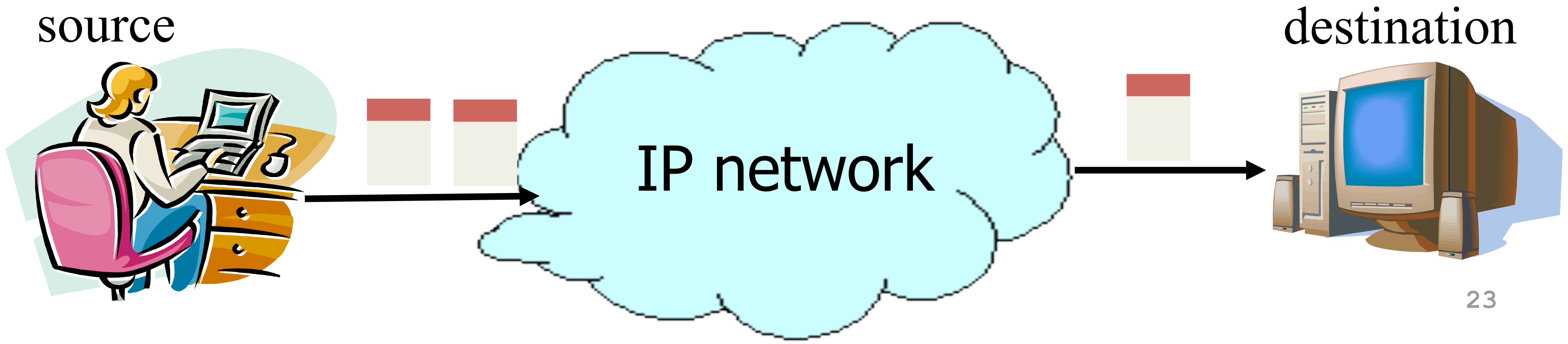
```
HTTP/1.1 200 OK
Date: Mon, 2 Feb 2009 13:09:03 GMT
Server: Netscape-Enterprise/3.5.1
Last-Modified: Mon, 42 Feb  2009 11:12:23 GMT
Content-Length: 21
CRLF
Site under construction
```

Example: IP Packet



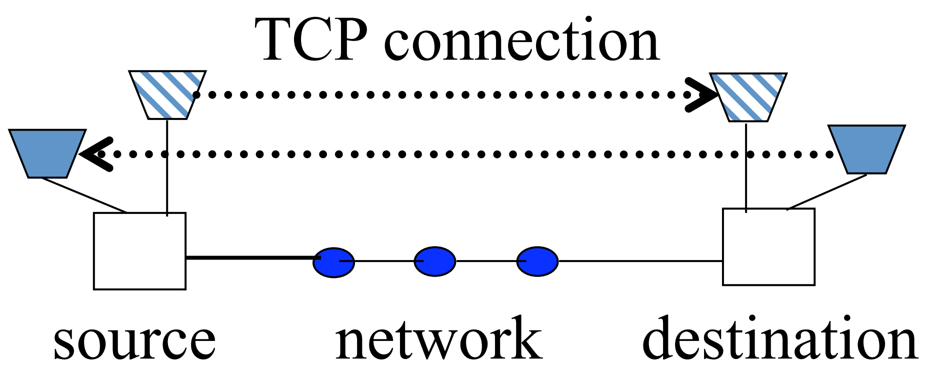
IP: Best-Effort Packet Delivery

- Packet switching
  - Send data in packets
  - Header with source & destination address
- Best-effort delivery
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



Example: Transmission Control Protocol

- Communication service (socket)
  - Ordered, reliable byte stream
  - Simultaneous transmission in both directions
- Key mechanisms at end hosts
  - Retransmit lost and corrupted packets
  - Discard duplicate packets and put packets in order
  - Flow control to avoid overloading the receiver buffer
  - Congestion control to adapt sending rate to network load

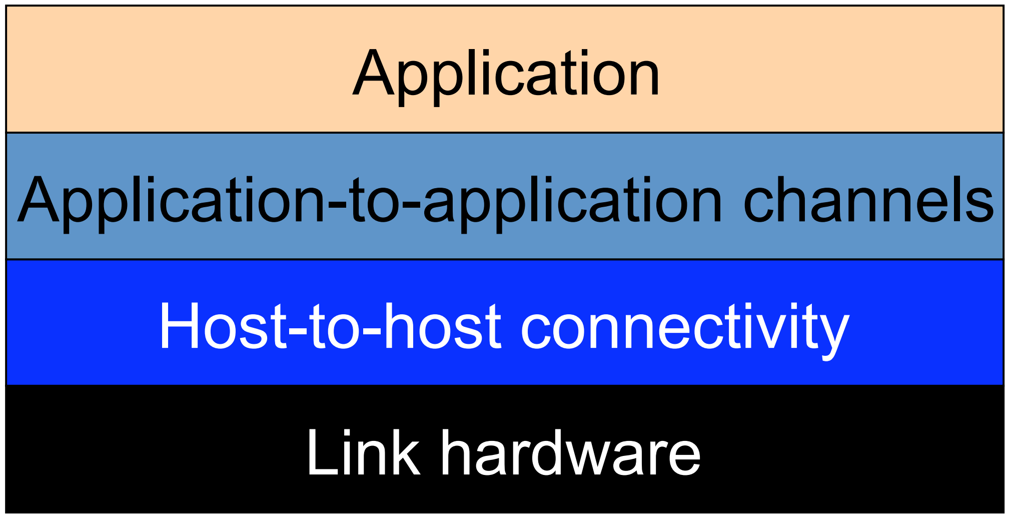


Protocol Standardization

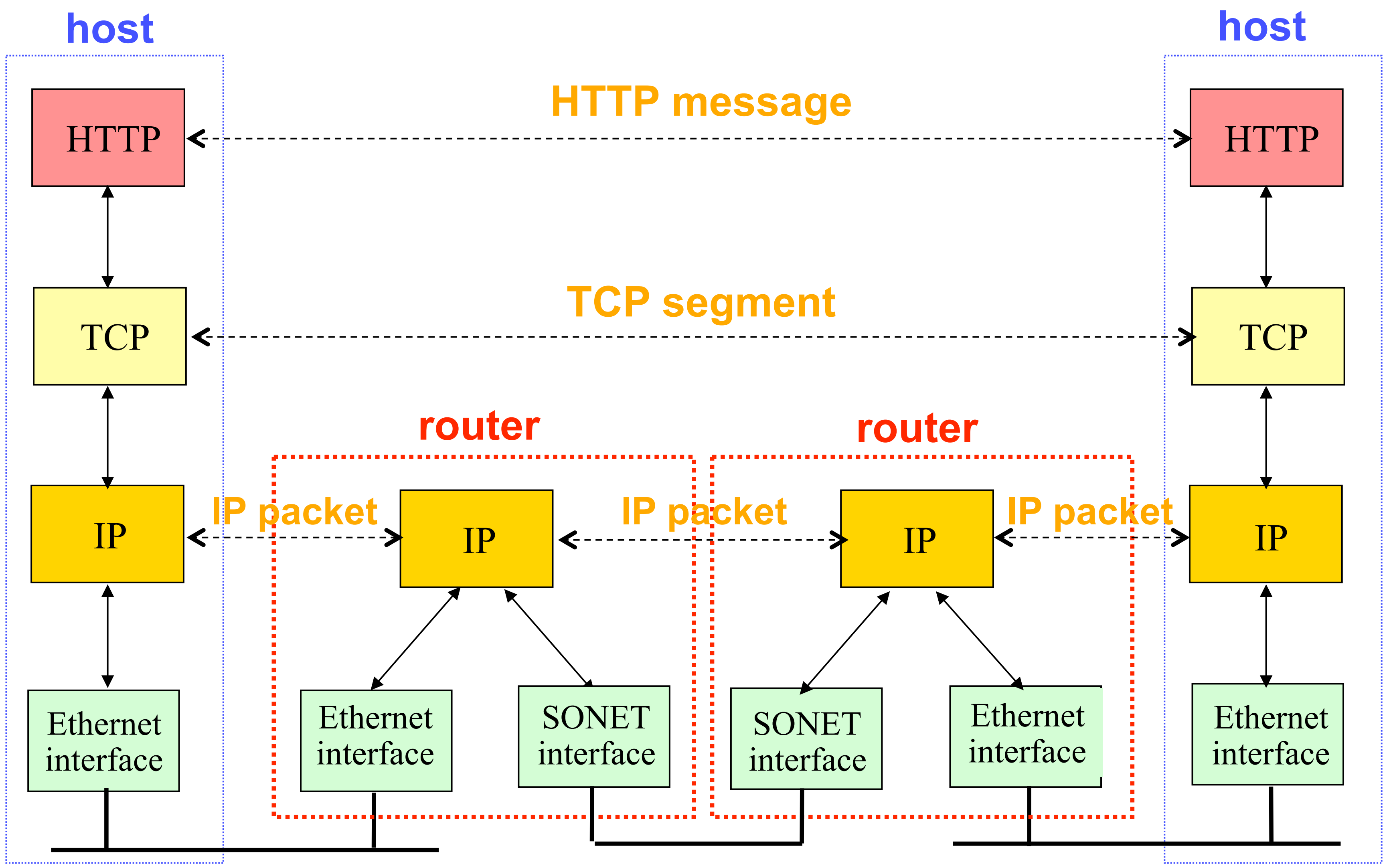
- Communicating hosts speaking the same protocol
  - Standardization to enable multiple implementations
  - Or, the same folks have to write all the software
- Standardization: Internet Engineering Task Force
  - Based on working groups that focus on specific issues
  - Produces "Request For Comments" (RFCs)
    - Promoted to standards via rough consensus and running code
    - E.g., RFC 1945 on "HyperText Transfer Protocol – HTTP/1.0"
  - IETF Web site is http://www.ietf.org
- De facto standards: same folks writing the code
  - P2P file sharing, BitTorrent, Skype, Flash videos (RTMP)
  - <your protocol here>…

Layering: A Modular Approach

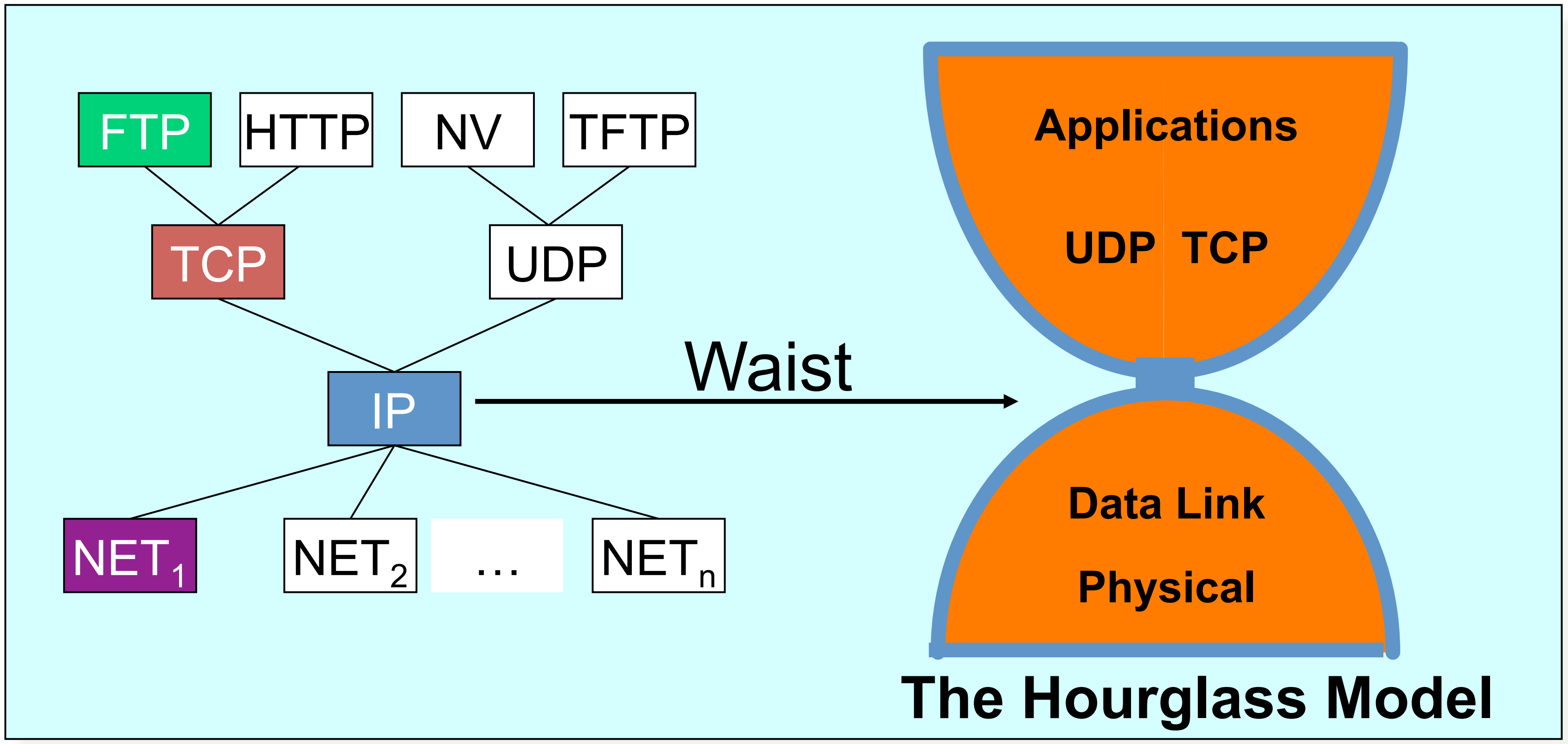
- Sub-divide the problem
  - Each layer relies on services from layer below
  - Each layer exports services to layer above
- Interface between layers defines interaction
  - Hides implementation details
  - Layers can change without disturbing other layers


Application
Application-to-application channels
Host-to-host connectivity
Link hardware

IP Suite: End Hosts vs. Routers

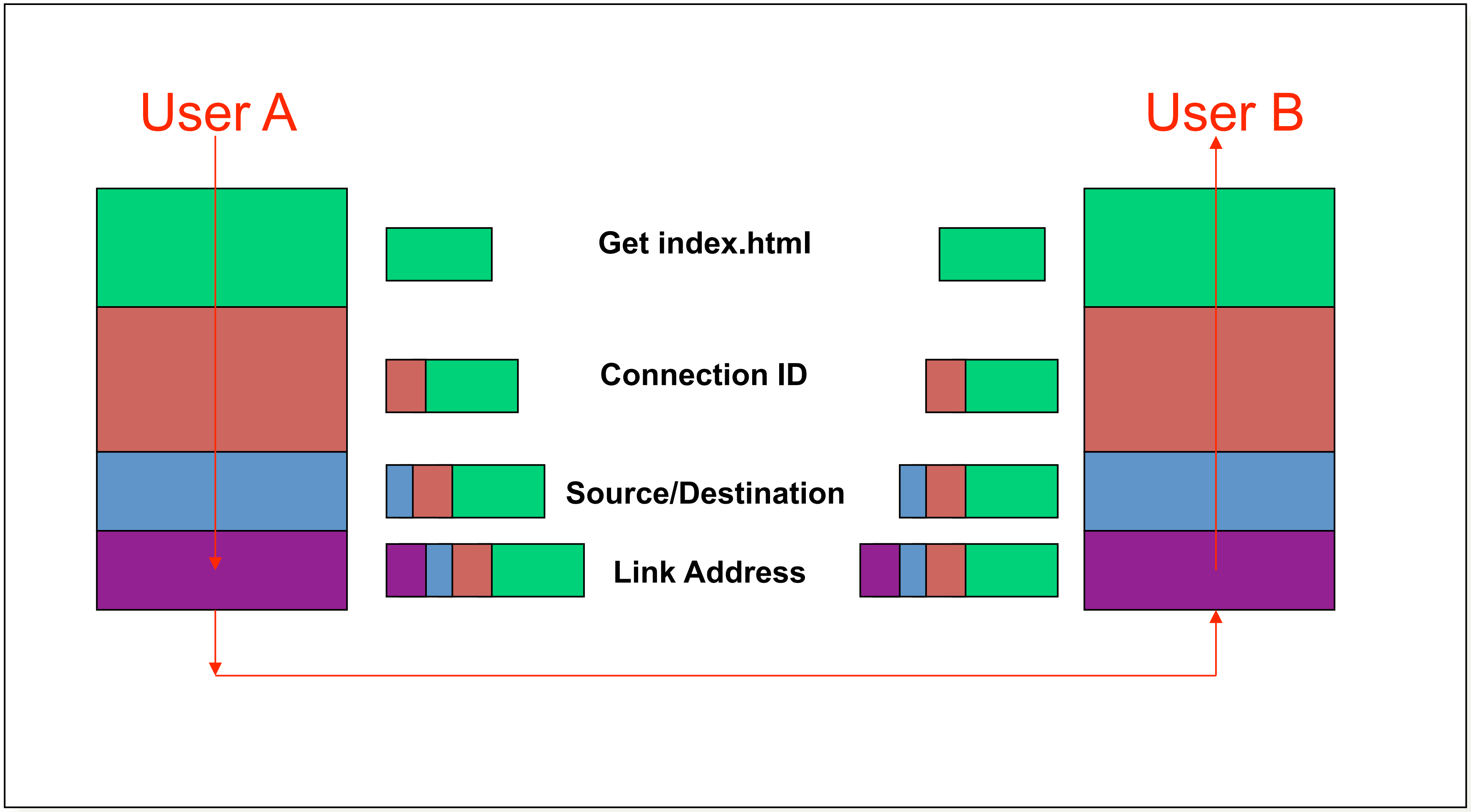


The Internet Protocol Suite



The waist facilitates interoperability

Layer Encapsulation

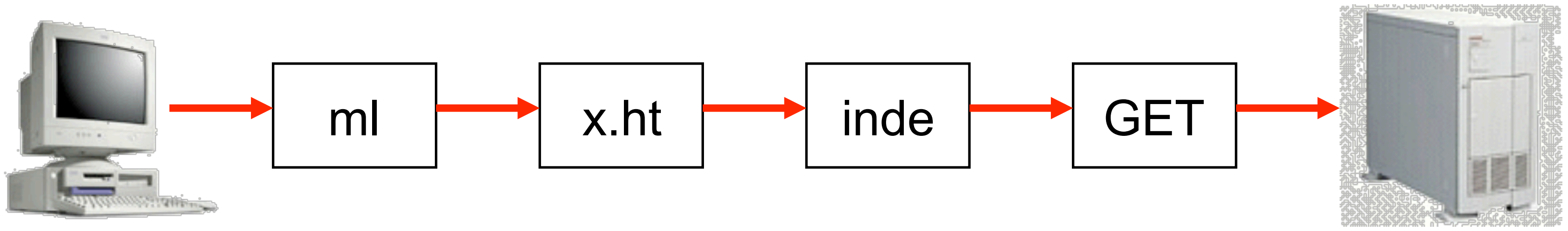


What if the Data Doesn't Fit?

Problem: Packet size

- On Ethernet, max IP packet is 1500 bytes
- Typical Web page is 10 kbytes

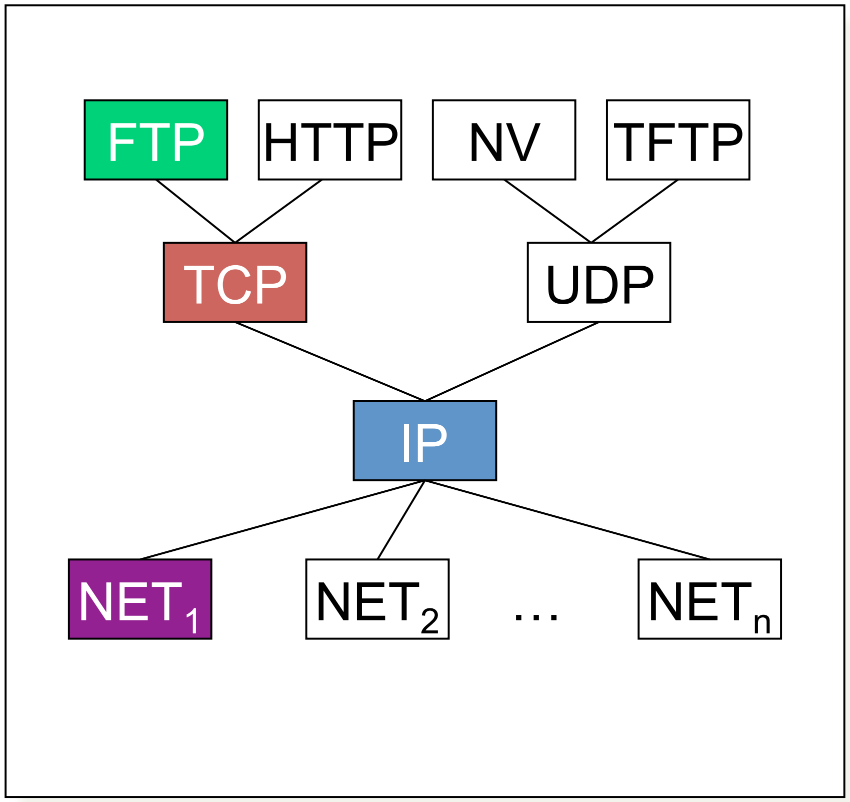
Solution: Split the data across multiple packets

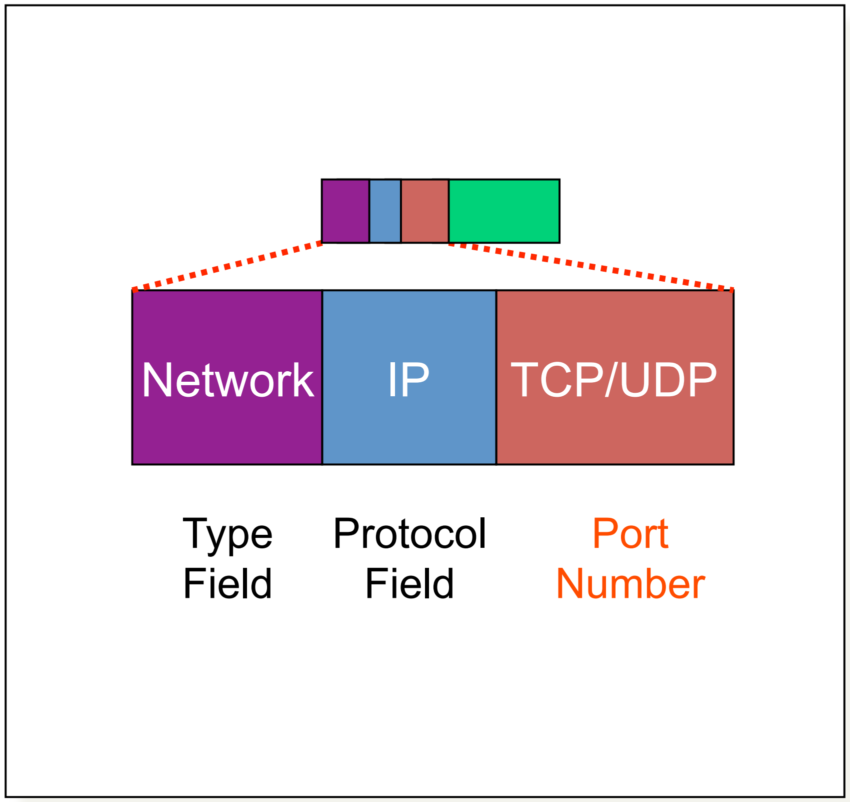


GET index.html

Protocol Demultiplexing

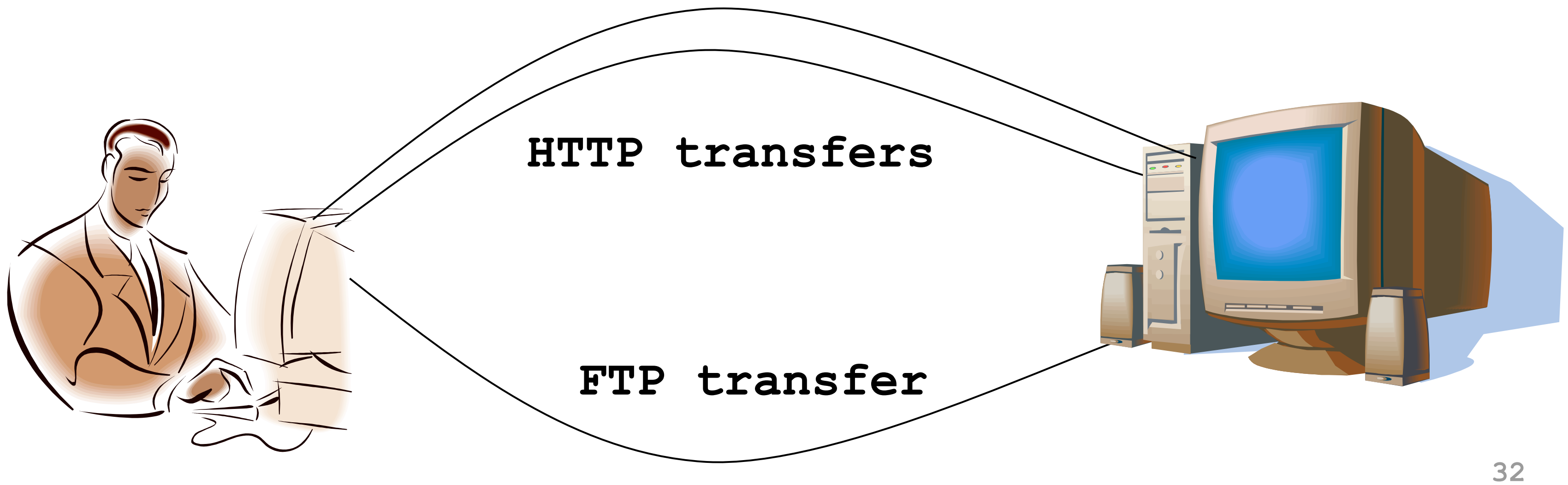
- Multiple choices at each layer





Demultiplexing: Port Numbers

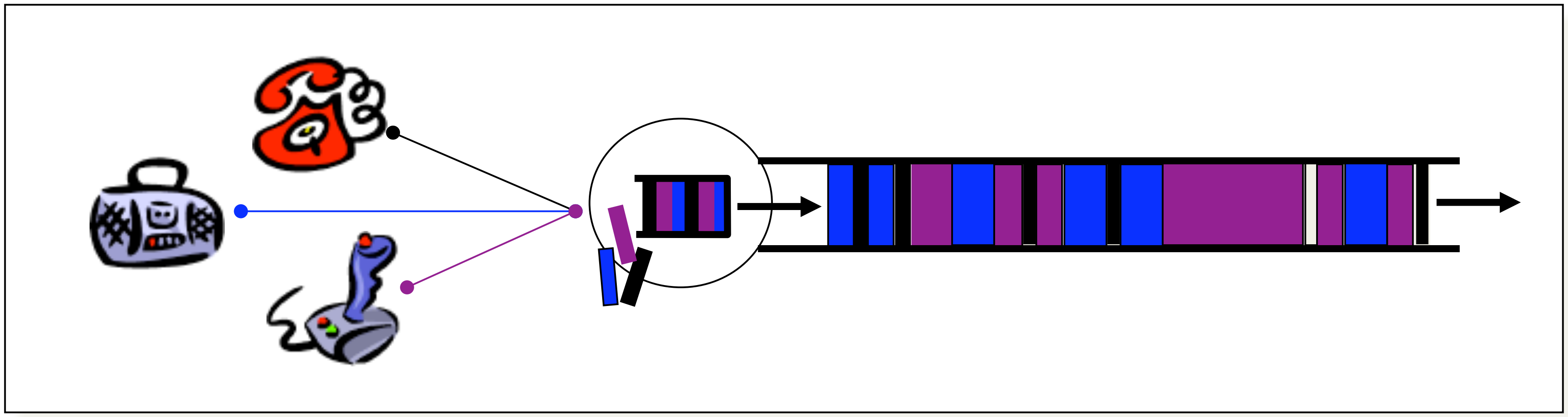
- Differentiate between multiple transfers
  - Knowing source and destination host is not enough
  - Need an id for *each transfer* between the hosts
- Use well-known ports to discover a particular service
  - E.g., HTTP server running on port 80
  - E.g., FTP server running on port 21
  - But how differentiate if server always port 80?



Is Layering Harmful?

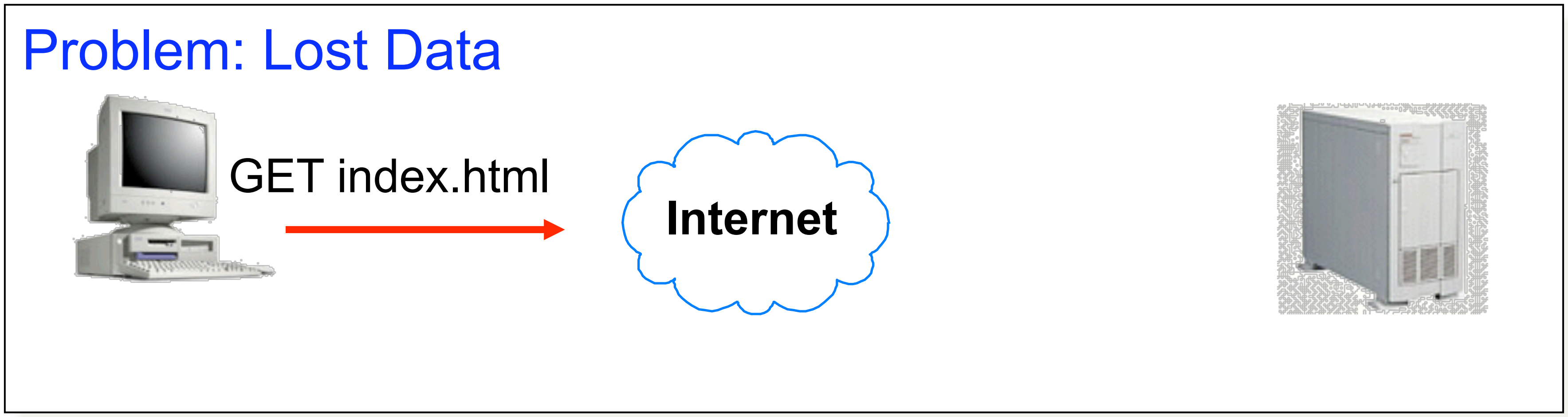
- Layer N may duplicate lower level functionality
  - E.g., error recovery to retransmit lost data
- Layers may need same information
  - E.g., timestamps, maximum transmission unit size
- Strict adherence to layering may hurt performance
  - E.g., hiding details about what is really going on
- Some layers are not always cleanly separated
  - Inter-layer dependencies for performance reasons
  - Some dependencies in standards (header checksums)
- Headers start to get really big
  - Sometimes more header bytes than actual content

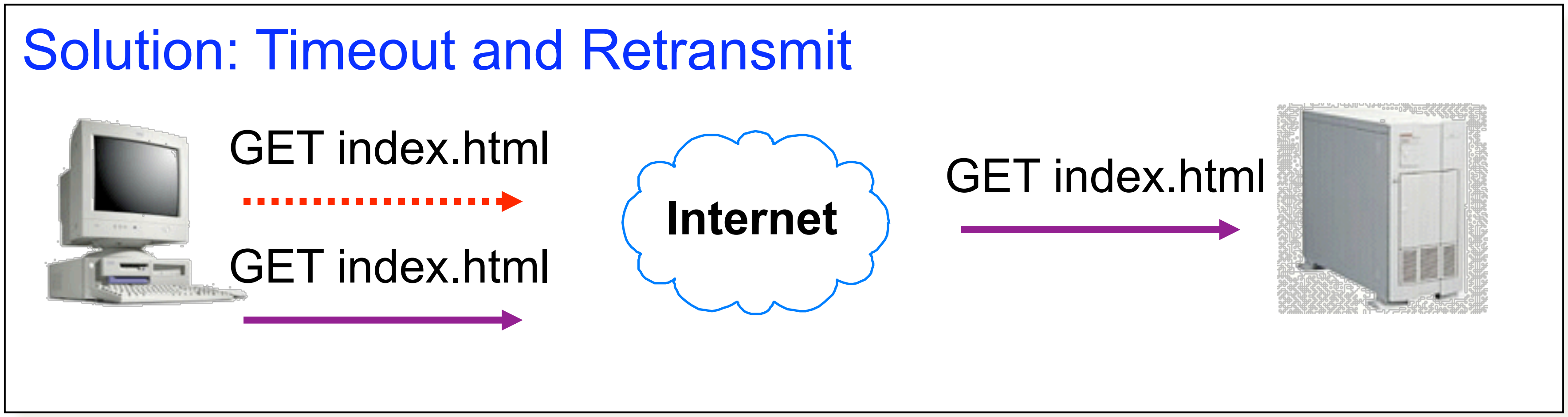
Resource Allocation: Queues



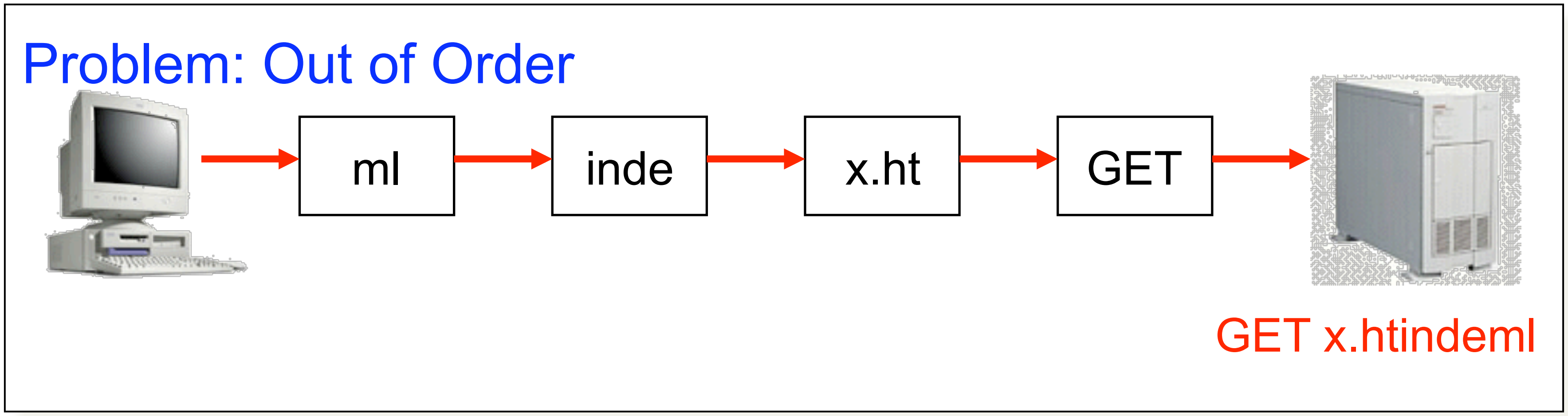
- Sharing access to limited resources
  - E.g., a link with fixed service rate
- Simplest case: first-in-first out queue
  - Serve packets in the order they arrive
  - When busy, store arriving packets in a buffer
  - Drop packets when the queue is full
- Anybody hear of “Network Neutrality”?

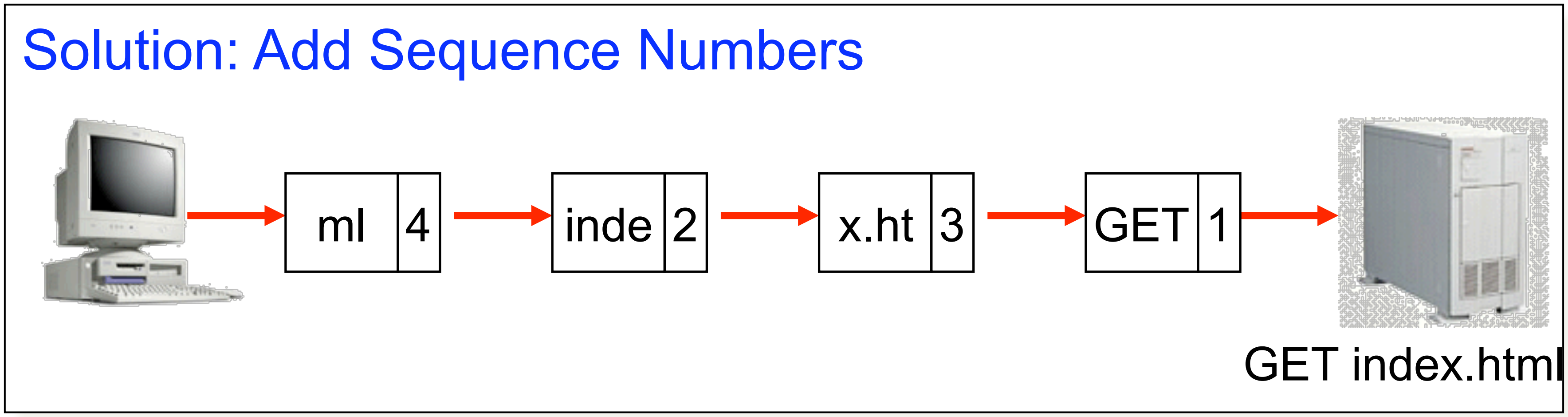
What if the Data gets Dropped?



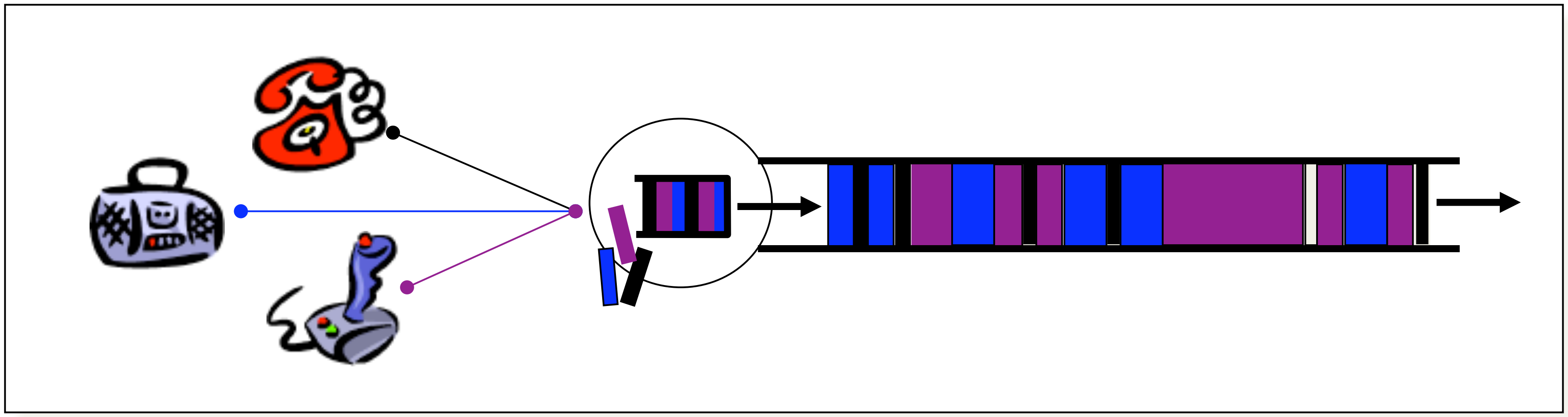


What if the Data is Out of Order?





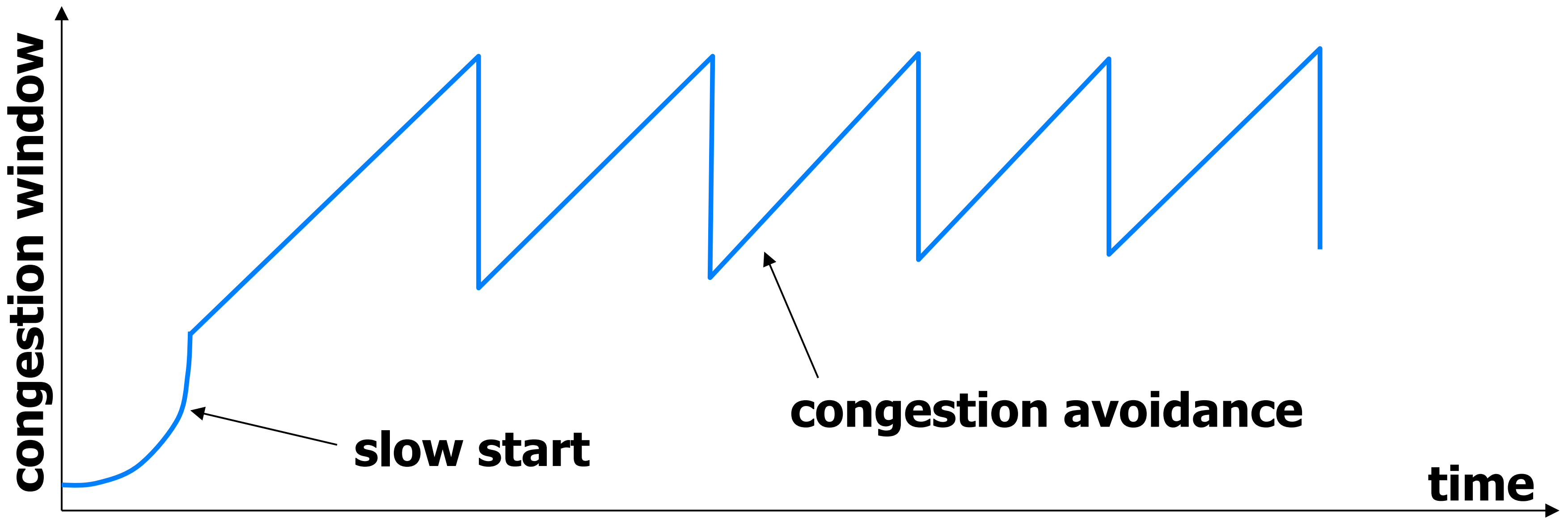
Resource Allocation: Congestion Control



- What if too many folks are sending data?
  - Senders agree to slow down their sending rates
  - … in response to their packets getting dropped
- The essence of TCP congestion control
  - Key to preventing congestion collapse of the Internet

Transmission Control Protocol

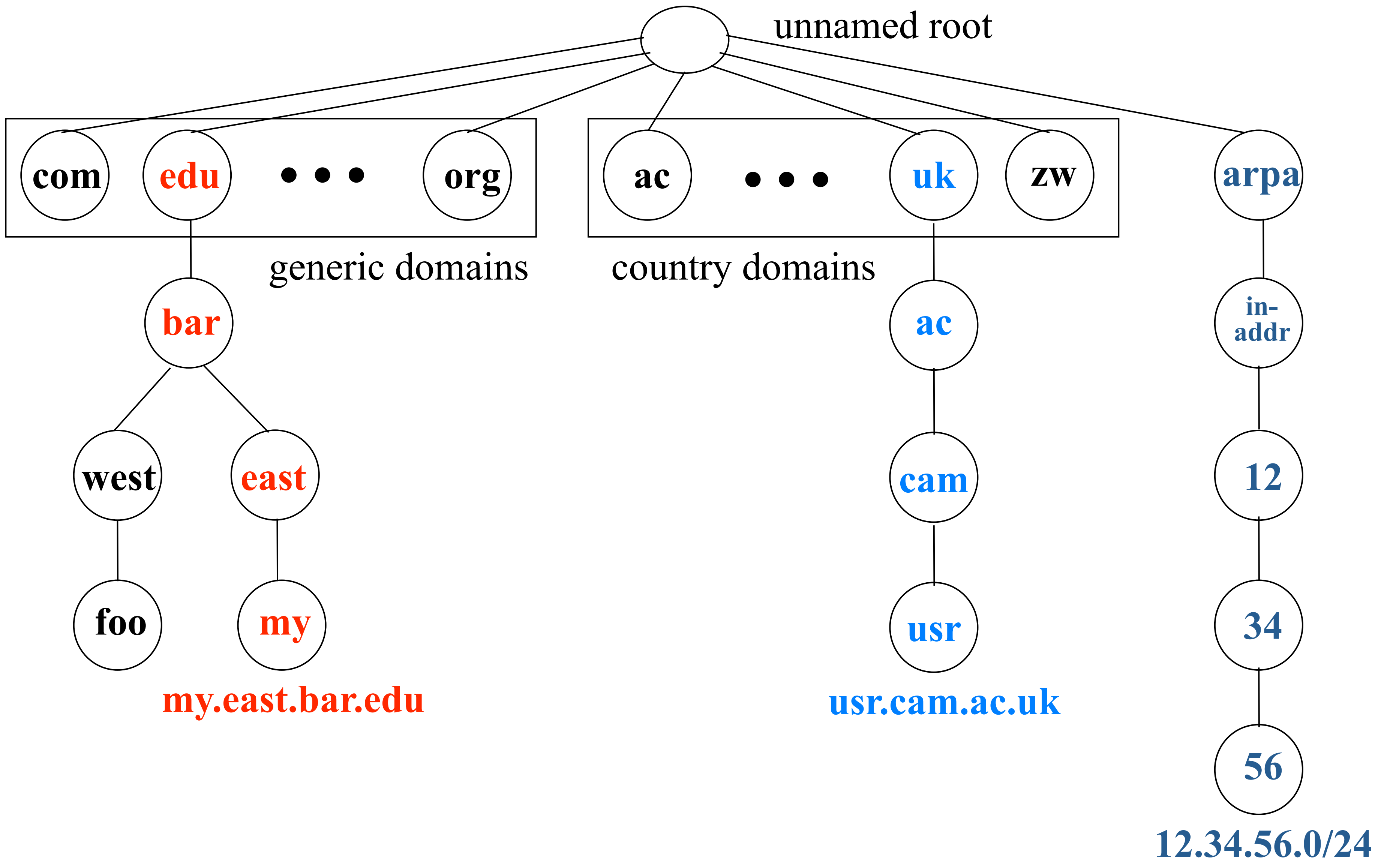
- Flow control: window-based
  - Sender limits number of outstanding bytes (window size)
  - *Receiver window* ensures data does not overflow receiver
- Congestion control: adapting to packet losses
  - *Congestion window* tries to avoid overloading the network (increase with successful delivery, decrease with loss)
  - TCP connection starts with small initial congestion window



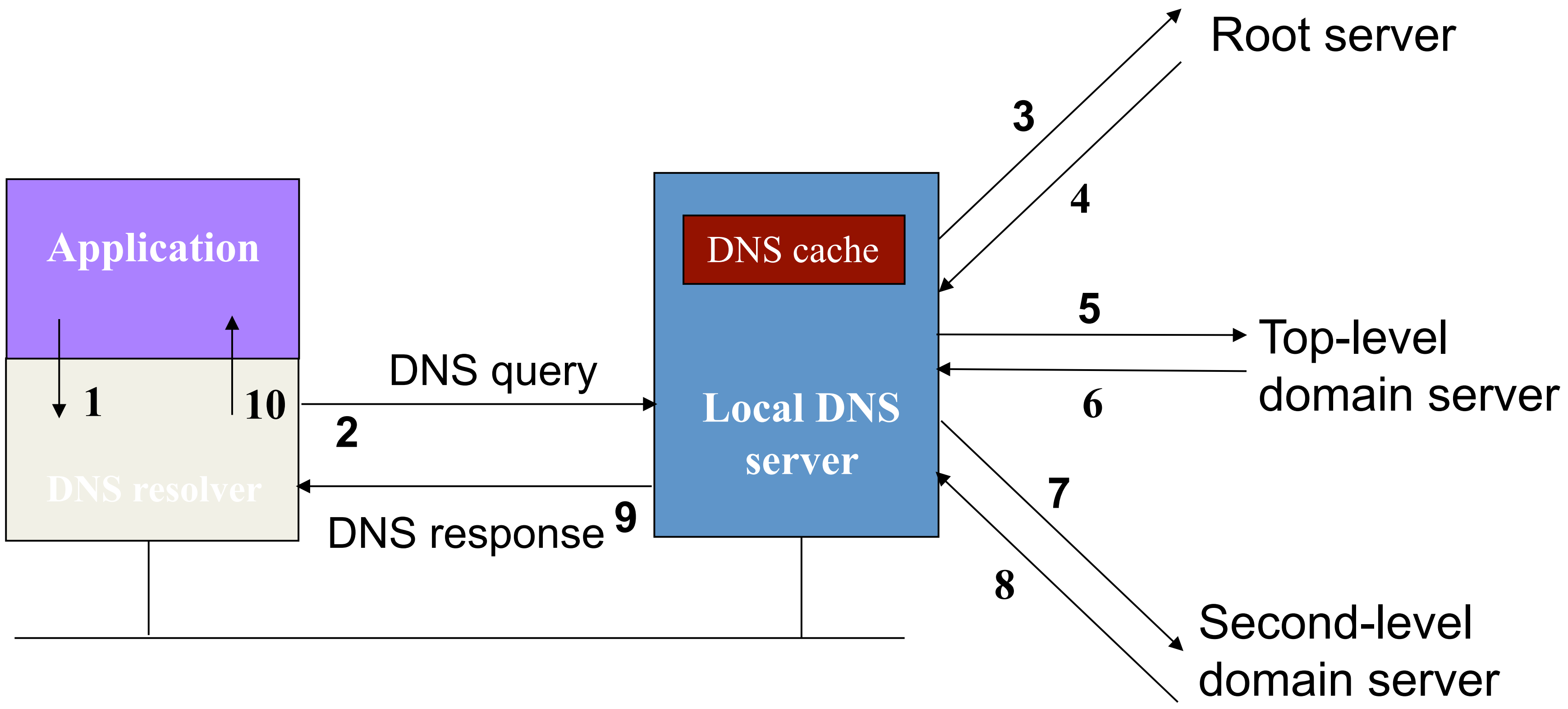
Naming: Domain Name System (DNS)

- Properties of DNS
  - Hierarchical name space divided into zones
  - Translation of names to/from IP addresses
  - Distributed over a collection of DNS servers
- Client application
  - Extract server name (e.g., from the URL)
  - Invoke system call to trigger DNS resolver code
    - E.g., *gethostbyname()* on www.cs.princeton.edu
- Server application
  - Extract client IP address from socket
  - Optionally invoke system call to translate into name
    - E.g., *gethostbyaddr()* on “12.34.158.5”

Domain Name System



DNS Resolver and Local DNS Server



- Caching based on time-to-live (TTL) specified by *authoritative* DNS server, i.e., one responsible for domain name
  - Reduces latency in DNS translation
  - Reduces load on authoritative DNS servers

Conclusions

- Course objectives
  - Network programming, how the Internet works, and key concepts in networking
- Key concepts in networking
  - Protocols, layers, resource allocation, and naming
- Next lecture: network programming
  - Socket abstraction (important for assignment #1)
  - Read Chapter 1 of the Peterson/Davie book
  - Skim the online reference material on sockets
  - (Re)familiarize yourself with C programming