Copyright notice

• Copyright 2010 by University of Illinois
• All rights reserved. Permission to reproduce this and all ECE/CS 438 course materials in whole or part for not-for-profit educational purposes is hereby granted. This document may not be reproduced for commercial purposes without the express written consent of the author.
• Includes content by Robin Kravets, Steve Lumetta, Bruce Hajek, Nitin Vaidya, Larry Peterson, Jennifer Rexford, Ion Stoica, Brighten Godfrey, and others
Course Information

• Instructors:
  – Prof. Matthew Caesar
    3118 SC
    217-244-0527
    caesar@cs.illinois.edu

• TAs:
  – Brian Cho, Virajith Jalaparti

• Class Webpage:
  – http://www.cs.illinois.edu/class/sp10/cs438

• Class News group:
  – cs.illinois.class.cs438
Prerequisites

• Operating Systems Concepts
  – CS 241 or equivalent

• C or C++ Programming
  – Preferably Unix

• Probability and Statistics
Grading Policy

- Homework 15%
  - 7 homework assignments
- Programming Projects 35%
  - 3 Programming projects
- Mid-term Exam 20%
  - March 17
- Final Exam 30%
  - May 12, 7 – 10 PM
Homework and Projects

- **Homework:**
  - Due Wednesdays at start of class.
  - General extension to Fridays start of class (hard deadline).
    - Solutions handed out in class on Fridays
  - No questions to Professor, TAs or on newsgroup after class on Wednesday.

- **Projects:**
  - Due Fridays at 9:00pm.
  - 2% off per hour late
  - MP1 is solo
  - MP2 and MP3 are 2 person teams
Academic Honesty

- Your work in this class **must** be your own.
- If students are found to have collaborated excessively or to have blatantly cheated (e.g., by copying or sharing answers during an examination or sharing code for the project), **all** involved will at a minimum receive grades of 0 for the first infraction.
  - We will run a similarity-checking system on code and binaries
- Further infractions will result in failure in the course and/or recommendation for dismissal from the university.
Graduate Students

• Graduate students MAY take an extra one hour project in conjunction with this class
  – Graduate students
    • Write a survey paper in a networking research area of your choice
    • Project proposal with list of 10+ academic references (no URL’s) due February 11th
    • Paper due last day of class
  – Undergraduates may not take this project course
    • However, if you are interested in networking research, please contact me
Course Objectives

- At the end of the semester, you should be able to
  - Identify the problems that arise in networked communication
  - Explain the advantages and disadvantages of existing solutions to these problems in the context of different networking regimes
  - Understand the implications of a given solution for performance in various networking regimes
  - Evaluate novel approaches to these problems
Programming Objectives

- At the end of the semester, you should be able to
  - Identify and describe the purpose of each component of the TCP/IP protocol suite
  - Develop solid client-server applications using TCP/IP
  - Understand the impact of trends in network hardware on network software issues
• Introduction to UNIX Network Programming
• Direct Link Networks
• Packet Switched Networks
• Internetworking
• End-to-End Protocols
• Congestion Control
• Performance Analysis and Queueing Theory

Some
• Mobile Ad hoc Networks
• P2P Networks
• Network Security
Why study networks?

• Internet has drastically changed the way we interact with computers
  – Business, collaborations, retail, news, communications, gaming, media... all increasingly conducted online

• Emergence of new distributed software to support these applications
  – Webmail, online retail, online auctions, wikis, online storage, search, utility computing, network management, messaging and email, wireless services, social networking
What do these two things have in common?

- First printing press
  - Key idea: movable type
- The Internet

Both lowered the cost of distributing information
The ARPANet

- 1962: J.C.R. Licklider appointed head of ARPA
  - Envisions shared network to connect computers at different sites
- 1968: RFQ sent to 140 potential bidders to build ARPANet
  - 12 submissions (most regard proposal as outlandish)
  - BBN Technologies selected as winner
- 2 September 1969: UCLA first node on ARPANet
- December 1969: 4 nodes connected by phone lines
ARPANet evolves into Internet

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SATNet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRNet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SATNet: Satellite network
PRNet: Radio Network

Web Hosting
Multiple ISPs
Internet2 Backbone
Internet Exchanges

Application Hosting
ASP: Application Service Provider
SaS: Software as a Service Provider (e-commerce toolkit, etc.)
ARPANet, 1971
ARPANet, 1980

ARPANET GEOGRAPHIC MAP, OCTOBER 1980

- SATELLITE CIRCUIT
  - IMP
  - TIP
  - PLURIBUS IMP
  - PLURIBUS TIP
  - C30

(NOTE: THIS MAP DOES NOT SHOW ARPA'S EXPERIMENTAL SATELLITE CONNECTIONS)
NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES
Transition to NSFNet

NSFNET T1 Network 1991

© Merit Network, Inc.
Internet, today
Networking: Actually Not Boring

• How hard can it be?

• You just string a wire (or other signaling path) between two computers...
• ... first one pushes bits down the link...
• ... and the second one gets them up... right?
• Where does it get tricky?
• What are the challenges?
Why Networking is Challenging

• Fundamental challenge: the speed of light

• Question: how long does it take light to travel from UIUC to Mountain View, CA (Google Headquarters)?

• Answer:
  – Distance UIUC --> Mountain View is 2,935 km
  – Traveling 300,000 km/s: 9.78ms
Fundamental Challenge: Speed of Light

- Question: how long does it take an Internet “packet” to travel from UIUC to Mountain View?
- Answer:
  - For sure >= 9.78ms
  - But also depends on:
    - The route the packet takes (could be circuitous!)
    - The propagation speed of the links the packet traverses
      - E.g. in optical fiber light propagates only at 2/3 C
    - The transmission rate (bandwidth) of the links (bits/sec)
      - And also the size of the packet
    - Number of hops traversed (“store and forward” delay)
    - The “competition” for bandwidth the packet encounters (congestion). It may have to wait in router queues.
  - In practice this boils down to >=40ms
    - With variance (can be hard to predict!)
Fundamental Challenge: Speed of light

- Question: how many cycles does your PC execute before it can possibly get a reply to a message it sent to a Mountain View web server?
- Answer:
  - Round trip takes >= 80ms
  - PC runs at (say) 3 GHz
  - $3,000,000,000$ cycles/sec * $0.08$ sec = $240,000,000$ cycles
- Thus,
  - Communication feedback is always dated
  - Communication fundamentally asynchronous
Fundamental Challenge: Speed of Light

- Question: what about machines directly connected (via a local area network or LAN)?
- Answer:

  ```
  % ping www.cs.uiuc.edu
  PING dcs-www.cs.uiuc.edu (128.174.252.83) 56(84) bytes of data.
  64 bytes from 128.174.252.83: icmp_seq=1 ttl=63 time=0.263 ms
  64 bytes from 128.174.252.83: icmp_seq=2 ttl=63 time=0.595 ms
  64 bytes from 128.174.252.83: icmp_seq=3 ttl=63 time=0.588 ms
  64 bytes from 128.174.252.83: icmp_seq=4 ttl=63 time=0.554 ms
  ...
  ```

- 500us = 1,500,000 cycles
  - Still a looooonng time...
  - ... and asynchronous...
• Fundamental challenge: **components fail**
  – Network communication involves a chain of **interfaces, links, routers, and switches**...
Examples of Network Components

**Links**
- Fibers
- Coaxial Cable

**Interfaces**
- Ethernet card
- Wireless card

**Switches/routers**
- Large router
- Telephone switch
Why networking is challenging (cont’d)

• Fundamental challenge: components fail
  – Network communication involves a chain of interfaces, links, routers, and switches...
  – All of which must function correctly

• Question: suppose a communication involves 50 components which work correctly (independently) 99% of the time. What’s the likelihood the communication fails at a given point in time?
  – Answer: success requires that they all function, so failure probability = \(1 - 0.99^{50} = 39.5\%\)

• So we have a lot of components, which tend to fail...
  – ... and we may not find out for a loooong time
Why networking is challenging (cont’d)

• Challenge: enormous dynamic range
  – Round trip times (latency) vary from 10 us’s to sec’s \(10^5\)
  – Data rates (bandwidth) vary from kbps to 10 Gbps \(10^7\)
  – Queuing delays inside the network vary from 0 to sec’s
  – Packet loss varies from 0 to 90+%  
  – End system (host) capabilities vary from cell phones to supercomputing clusters
  – Application needs vary enormously: size of transfers, bidirectionality, need for reliability, tolerance of jitter

• Related challenge: very often, there is no such thing as “typical”. Beware of your “mental models”!
  – Must think in terms of design ranges, not points
  – Mechanisms need to be adaptive
Why networking is challenging (cont’d)

• Challenge: different parties must work together
  – Multiple parties with different agendas must agree how to divide the task between them

• Working together requires:
  – Protocols (defining who does what)
    • These generally need to be standardized
  – Agreements regarding how different types of activity are treated (policy)

• Different parties very well might try to “game” the network’s mechanisms to their advantage
Why networking is challenging

• Challenge: incessant rapid growth
  – Utility of the network scales with its size
    • Fuels exponential growth (for more than 2 decades!)
  – Data centers contain 100k+ of hosts, Internet contains 600M+ hosts, 2.6M routers
    • Microsoft’s data center in Chicago: 500k servers
• Adds another dimension of dynamic range...
  – and quite a number of ad hoc artifacts...
Why networking is challenging (cont’d)

• Challenge: there are Bad Guys out there
• As the network population grows in size, so does the number of
  – Vandals
  – Crazies
• What really matters, though: as network population grows, it becomes more and more attractive to
  – Crooks
  – (and also spies and militaries)
Why Crooks Matter for Networking

• They (and other attackers) seek ways to misuse the network towards their gain
  – Carefully crafted “bogus” traffic to manipulate the network’s operation
  – Torrents of traffic to overwhelm a service (denial-of-service) for purposes of extortion/competition
  – Passively recording network traffic in transit (sniffing)
  – Exploit flaws in clients and servers using the network to trick into executing the attacker’s code (compromise)

• They all do this energetically because there is significant $$$ to be made
Why networking is challenging (cont’d)

• Challenge: you cannot reboot the Internet!
  – Everyone depends on the Internet
    • Businesses
    • Hospitals
    • Education institutions
    • Financial sector
    • ...
  – Cannot stop, fix, or restart it
  – ... akin to changing the engine while you are flying the plane!
Summary so far...

- Networking is about design in the presence of challenges/contraints:
  - Not akin to e.g., programming languages/compilers
    - Which have well-developed theories to draw upon
  - Much more akin to operating systems
    - Abstractions
    - Tradeoffs
    - Design principles / “taste”
Roadmap for rest of lecture

• Today, let’s study a few key questions that networks need to solve:

1. How can many hosts communicate?

2. How can we identify hosts?

3. How can we make protocols easy to design/deploy?
Today, let’s study a few key questions that networks need to solve:

1. How can many hosts communicate?

2. How can we identify hosts?

3. How can we make protocols easy to design/deploy?
How can two hosts communicate?

- Encode information on modulated “Carrier signal”
  - Phase, frequency, and amplitude modulation, and combinations thereof
  - Ethernet: self-clocking Manchester coding ensures one transition per clock
  - Technologies: copper, optical, wireless
How can many hosts communicate?

- Naïve approach: full mesh
- Problem:
  - Obviously doesn’t scale to the 570,937,778 hosts in the Internet (estimated, Aug 2008)
How can many hosts communicate?

- Multiplex traffic with routers
- Goals: make network robust to failures, maintain spare capacity, reduce operational costs
Connectivity

• Building Block
  – Links: coax cable, optical fiber, ...
  – Nodes: workstations, routers, ...

• Links:
  – Point-to-point
  – Multiple access
Indirect Connectivity

- Switched Networks
- Internetworks

- Recursive definition of a network
  - Two or more nodes connected by a physical link
  - Two or more networks connected by one or more nodes
Effects of Indirect Connectivity

- Nodes receive data on one link and forward it onto the next ⇒ switching network
  - Circuit Switching
    - Telephone
    - Stream-based (dedicated circuit)
    - Links reserved for use by communication channel
    - Send/receive bit stream at constant rate
  - Packet Switching
    - Internet
    - Message-based (store-and-forward)
    - Links used dynamically
    - Admission policies and other traffic determine bandwidth
Today, let’s study a few key questions that networks need to solve:

1. How can many hosts communicate?

2. How can we identify hosts?

3. How can we make protocols easy to design/deploy?
How can we identify hosts?

- Hosts assigned topology-dependent addresses
- Routers advertise address blocks ("prefixes")
- Routers compute "shortest" paths to prefixes
- Map IP addresses to names with DNS
Addressing

- **Addressing**
  - Unique byte-string used to indicate which node is the target of communication

- **Types of Addresses**
  - Unicast: node-specific
  - Broadcast: all nodes on the network
  - Multicast: subset of nodes on the network

- **Routing**
  - The process of determining how to forward messages toward the destination node based on its address
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”
Caching based on a time-to-live (TTL) assigned by the DNS server responsible for the host name to reduce latency in DNS translation.
Today, let’s study a few key questions that networks need to solve:

1. How can many hosts communicate?
2. How can we identify hosts?
3. How can we make protocols easy to design/deploy?
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, 2006? I can’t.

Please meet with me for 1.5 hours starting at 3:00pm on February 8, 2006? Please meet with me for 1.5 hours starting at 4:30pm on February 8, 2006?

I can’t. 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.
• Student #2: Book me for 1.5 hours at 1:15pm on Feb 9.
  • Advisor: Yes.
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

Request

GET /courses/archive/spring08/cos461/ HTTP/1.1
Host: www.cs.princeton.edu
User-Agent: Mozilla/4.03
CRLF

Response

HTTP/1.1 200 OK
Date: Mon, 4 Feb 2008 13:09:03 GMT
Server: Netscape-Enterprise/3.5.1
Last-Modified: Mon, 4 Feb 2008 11:12:23 GMT
Content-Length: 21
CRLF

Site under construction
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

```
<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application-to-application channels</td>
</tr>
<tr>
<td>Host-to-host connectivity</td>
</tr>
<tr>
<td>Link hardware</td>
</tr>
</tbody>
</table>
```
The waist facilitates interoperability
Layer Encapsulation

User A

Get index.html

Connection ID

Source/Destination

Link Address

User B
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

- Specify a particular service running on a host
  - E.g., HTTP server running on port 80
  - E.g., FTP server running on port 21
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
• 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

![Diagram showing slow start and congestion avoidance phases of congestion control.](image-url)
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
Cost-Effective Sharing of Resources

- Physical links and switches must be shared among many users

- Common multiplexing strategies
  - (Synchronous) time-division multiplexing (TDM)
  - Frequency-division multiplexing (FDM)
Statistical Multiplexing

• Statistical Multiplexing (SM)
  – On-demand time-division multiplexing
  – Scheduled on a per-packet basis
  – Packets from different sources are interleaved
  – Uses upper bounds to limit transmission
  • Queue size determines capacity per source
Statistical Multiplexing in a Switch

- Packets buffered in switch until forwarded
- Selection of next packet depends on policy
  - How do we make these decisions in a fair manner? Round Robin? FIFO?
  - How should the switch handle congestion?
Channels

- **Channel**
  - The abstraction for application-level communication
- **Idea**
  - Turn host-to-host connectivity into process-to-process communication
Performance

• Latency/delay
  – Time from A to B
  – Example: 30 msec (milliseconds)
  – Many applications depend on round-trip time (RTT)
  – Components
    • Propagation delay over links
    • Transmission time
    • Queueing delays
    • Software processing overheads
Bandwidth vs. Latency

- Relative importance of bandwidth and latency
  - Depends on application
- Large file transfers
  - Bandwidth is critical
- Small messages (HTTP, NFS, etc.)
  - Latency is critical
- Variance in latency (jitter)
  - Can also affect some applications (e.g., audio/video conferencing)
Delay x Bandwidth Product

- Amount of data in “pipe”
  - channel = pipe
  - delay = length
  - bandwidth = area of a cross section
  - bandwidth x delay product = volume
Delay x Bandwidth Product

• Pipe
  – Half of data that must be buffered before sender responds to slowdown request
Delay x Bandwidth Product

- Bandwidth x delay product
  - How many bits the sender must transmit before the first bit arrives at the receiver if the sender keeps the pipe full
  - Takes another one-way latency to receive a response from the receiver
Delay x Bandwidth Product

- **Example: Transcontinental Channel**
  - BW = 45 Mbps
  - delay = 50ms
  - bandwidth x delay product
    $= (50 \times 10^{-3} \text{ sec}) \times (45 \times 10^6 \text{ bits/sec})$
    $= 2.25 \times 10^6 \text{ bits}$
Bandwidth vs. Latency

- Relative importance
  - 1-byte: Latency bound
    - 1ms vs 100ms latency dominates 1Mbps vs 100Mbps BW
  - 25MB: Bandwidth bound
    - 1Mbps vs 100Mbps BW dominates 1ms vs 100ms latency
Bandwidth vs. Latency

- Infinite bandwidth
  - RTT dominates
    - Throughput = TransferSize / TransferTime
    - TransferTime = RTT + 1/Bandwidth x TransferSize

- It's all relative
  - 1-MB file on a 1-Gbps link looks like a 1-KB packet on a 1-Mbps link
Performance Notes

- **Speed of Light**
  - $3.0 \times 10^8$ meters/second in a vacuum
  - $2.3 \times 10^8$ meters/second in a cable
  - $2.0 \times 10^8$ meters/second in a fiber

- **Comments**
  - No queueing delays in a direct link
  - Bandwidth is not relevant if size = 1bit
  - Software overhead can dominate when distance is small

- **Key Point**
  - Latency dominates small transmissions
  - Bandwidth dominates large
Supplementary slides
Network Architecture

• Challenge
  – Fill the gap between hardware capabilities and application expectations, and to do so while delivering “good” performance

• Hardware and expectations are moving targets

• How do network designers cope with complexity?
  – Layering
  – Protocols
  – Standards
Abstraction through Layering

- Abstract system into layers:
  - Decompose the problem of building a network into manageable components
    - Each layer provides some functionality
  - Modular design provides flexibility
    - Modify layer independently
    - Allows alternative abstractions

<table>
<thead>
<tr>
<th>Application programs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Message stream channel</td>
<td>Request/reply channel</td>
</tr>
<tr>
<td>Host-to-host connectivity</td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td></td>
</tr>
</tbody>
</table>
Example: Air Travel

- **Layers**
  - Each layer implements a service
  - Via its own internal-layer actions
  - Relying on services provided by layer below

```
ticket (purchase)   ticket (complain)
baggage (check)     baggage (claim)
gates (load)        gates (unload)
runway (takeoff)    runway (landing)
airplane routing    airplane routing
```
Air Travel: Services

- check-in-counter-to-baggage-claim delivery
- people transfer: loading gate to arrival gate
- bag transfer: belt at check-in counter to belt at baggage claim
- runway-to-runway delivery of plane
- airplane routing from source to destination
Layering Concepts

- **Encapsulation**
  - Higher layer protocols create messages and send them via the lower layer protocols
  - These messages are treated as data by the lower-level protocol
  - Higher-layer protocol adds its own control information in the form of headers or trailers

- **Multiplexing and Demultiplexing**
  - Use protocol keys in the header to determine correct upper-layer protocol
Encapsulation

![Diagram showing the encapsulation process with Application program, Request/Reply, Host-to-Host, RRP HDR, DATA, and HHP HDR.]
Multiplexing/Demultiplexing

- **Transport Layer**
  - Provide logical communication between application processes running on different hosts
- **Transport protocols run in end systems**
  - Send side:
    - Break application messages into segments
    - Pass to network layer
  - Receive side:
    - Reassemble segments into messages
    - Pass to application layer
- **Multiple available transport protocols**
  - Internet: TCP and UDP
OSI Architecture

- Open Systems Interconnect (OSI) Architecture
  - International Standards Organization (ISO)
  - International Telecommunications Union (ITU, formerly CCITT)
  - “X dot” series: X.25, X.400, X.500
  - Primarily a reference model
OSI Protocol Stack

- **Application:** Application specific protocols
- **Presentation:** Format of exchanged data
- **Session:** Name space for connection mgmt
- **Transport:** Process-to-process channel
- **Network:** Host-to-host packet delivery
- **Data Link:** Framing of data bits
- **Physical:** Transmission of raw bits
OSI Protocol Stack

Host
User-Level
OS
Kernel

Application
Presentation
Session
Transport
Network
Data Link
Physical

Router

Application
Presentation
Session
Transport
Network
Data Link
Physical

Host

© University of Illinois - Spring 2010
Transport vs. Network Layer

- **Network layer**
  - Logical communication between hosts

- **Transport layer**
  - Logical communication between processes
  - Relies on, enhances, network layer services
Internet Architecture

- Internet Architecture (TCP/IP)
  - Developed with ARPANET and NSFNET
  - Internet Engineering Task Force (IETF)
    - Culture: implement, then standardize
    - OSI culture: standardize, then implement
  - Made popular with release of Berkeley Software Distribution (BSD) Unix; i.e., free software
  - Standard suggestions debated publicly through “requests for comments” (RFC’s)
    - We reject kings, presidents, and voting. We believe in rough consensus and running code. – David Clark
Internet Architecture – Hourglass Design

FTP
HTTP
NV
TFTP
TCP
UDP
IP
Ethernet
FDDI
ATM
Modem
Internet Architecture

• Features:
  – No strict layering
  – Hourglass shape – IP is the focal point
Protocol Acronyms

- (T)FTP - (Trivial) File Transfer Protocol
- HTTP - HyperText Transport Protocol
- NV - Network Video
- SMTP - Simple Mail Transfer Protocol
- NTP - Network Time Protocol
- TCP - Transmission Control Protocol
- UDP - User Datagram Protocol
- IP - Internet Protocol
- FDDI - Fiber Distributed Data Interface
- ATM - Asynchronous Transfer Mode
Summary

• Goal
  – Understanding of computer network functionality, with experience building and using computer networks

• Steps
  – Identify what concepts we expect from a network
  – Define a layered architecture
  – Implement network protocols and application programs
Assignments

• Homework 1
  – Due Wednesday February 3\textsuperscript{rd} at the start of class

• Project 1
  – Due Friday February 5\textsuperscript{th} at 9:00pm