

# Computer Networks

## – Introduction



---

### Outline:

- General course information.
- What are computer networks.
- Network architecture (Internet and OSI).
- Socket programming.



# General Course Information

---

- Course Web page
  - <http://www.cs.rochester.edu/~bukys/csc257-fall2008>
- Course email address
  - [cs257@cs.rochester.edu](mailto:cs257@cs.rochester.edu)
- Text and reference books
  - Kurose and Ross, "Computer Networking: A Top-Down Approach Featuring the Internet", 4th edition, 2007.
  - Peterson and Davie, "Computer Networks: A Systems Approach", 4th edition, 2007.
  - Network programming in C: Stevens, "Unix Network Programming, Volume I", 2nd edition, 1998.
  - Learn to use UNIX manual pages



# What we will learn?

---

- Principles

- basic engineering principles, design algorithms behind building blocks of computer networks
- why you'd select one technique over another

- Practices

- how things are done in practice:
  - in the case of Internet: the Web, p2p overlay, etc.
  - historical examples of protocols and implementations that were tried but are not so common any more



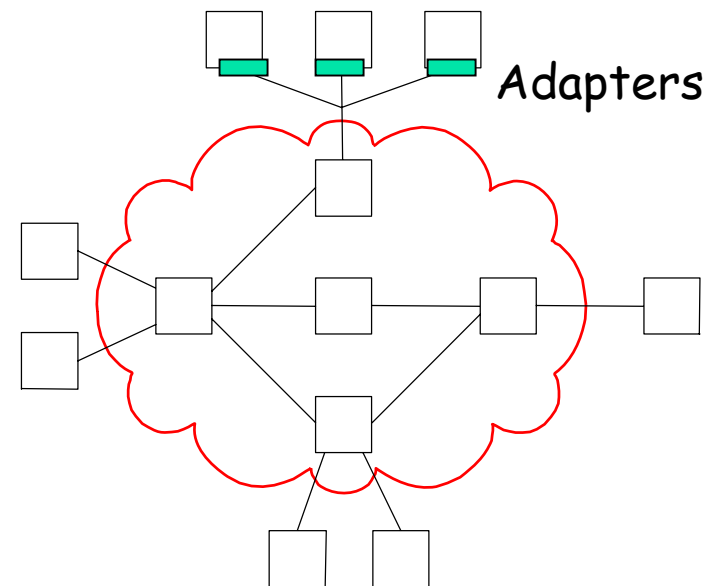
# Course Outline

---

- Traditional materials about computer networks
  - Data links: Ethernet etc.
  - Network: IP addressing and routing.
  - Transport: TCP/UDP.
  - Application: HTTP, FTP, SMTP, and DNS.
- Advanced topics in computer networks
  - Multimedia networking (quality of service), computer security, wireless networks, overlay networks.
- Case studies of emerging network systems/technologies
  - HTTP load balancing.
  - Network caching.
  - Content distribution (Akamai).
  - Peer-to-peer systems (Gnutella/BitTorrent).

# Basic Building Blocks for Computer Networks

- Nodes:
  - PC, server, special-purpose hardware, sensors ...
  - end hosts, switches
- Links:
  - twisted pair, coaxial cable, optical fiber, phone line, wireless radio channels ...
- Network: two or most hosts connected by links/switches





# Addressing and Routing

---

- Address: node identifier in a network.
  - often a unique byte string.
- Routing: finding a route (path) to the destination node based on its address.
- Types of traffic/addressing
  - unicast: to a single destination node;
  - broadcast: to all nodes on the network;
  - multicast: to some subset of nodes on the network;
  - anycast: to any one in some subset of nodes.



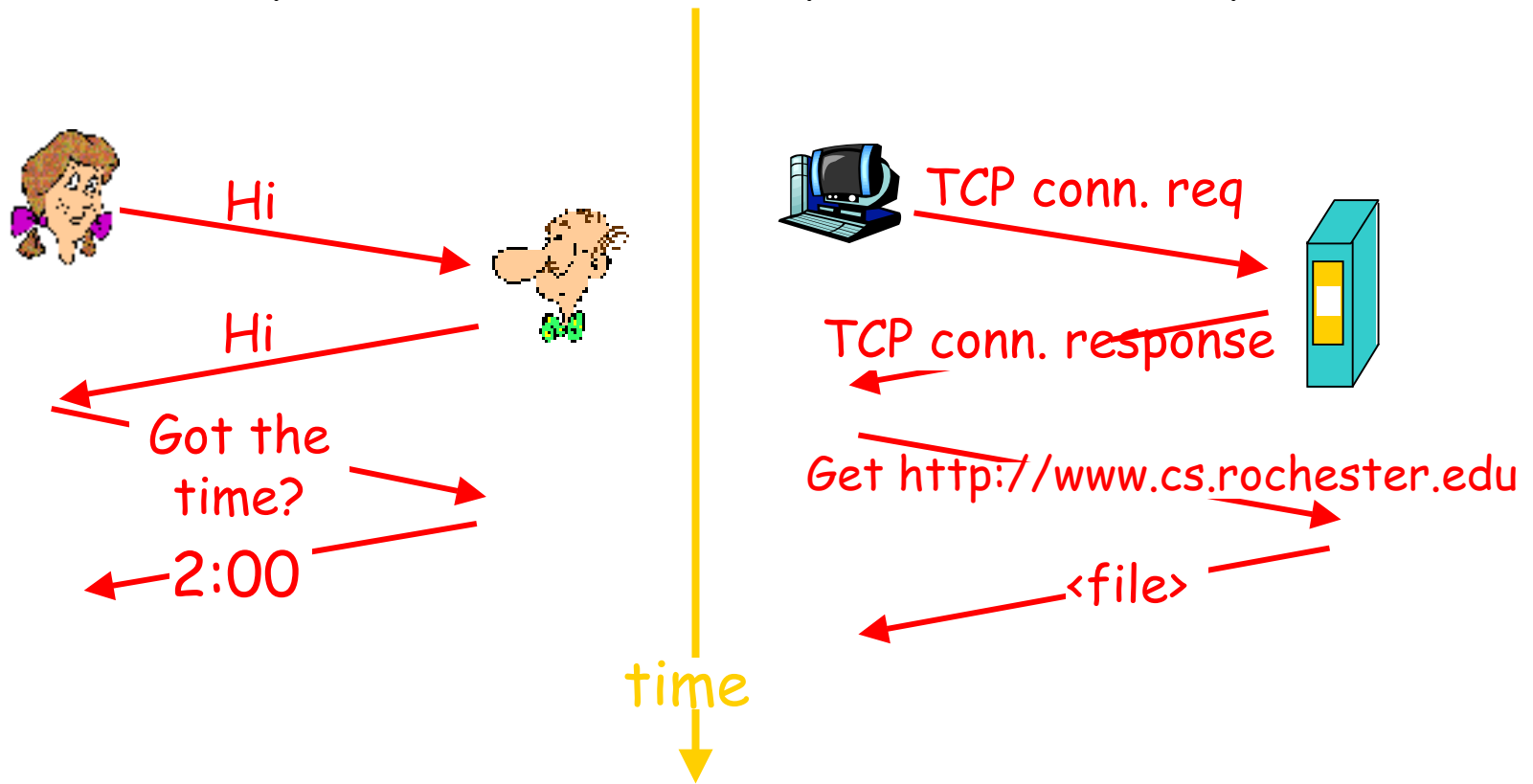
# Network Protocol

---

- A network **protocol** describes how communication entities (sender/receiver) should interact to accomplish a networking task.
  - format, order of messages sent and received among network entities
  - actions taken on message transmission, receipt
- Protocols define interface, not implementations
  - allow two sides of a communication task to be independently designed/implemented.

# An Example

A human protocol and a computer network protocol:

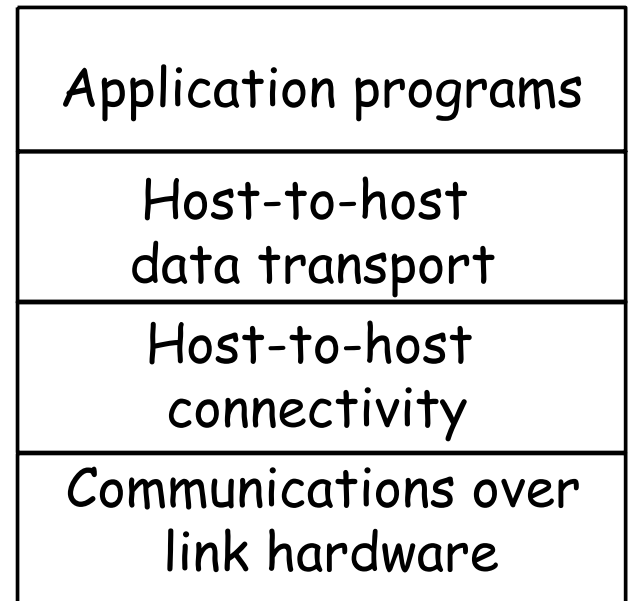




# Network Layers

---

- Separation of network functions/features allow independent design and implementation  $\Rightarrow$  thus **distributing the complexity**.
- Layering is a vertical separation of network functions/features.
  - **Hiding the complexity**: each layer interface hides complexity in this layer and layers below.
- Disadvantage of layering?
  - lack of full information sharing





# Protocol functions and features

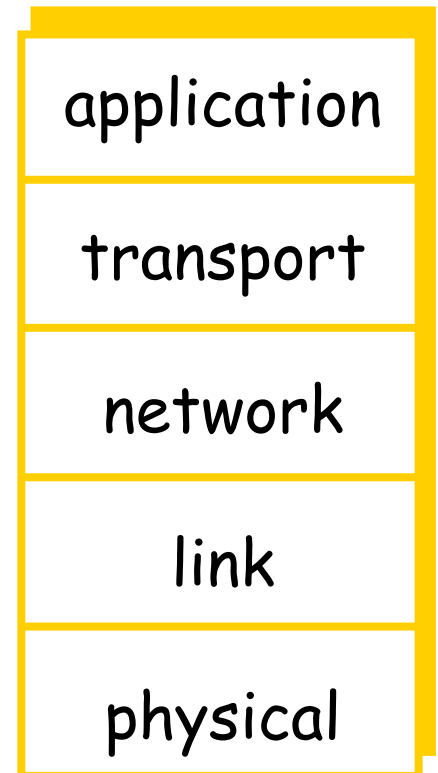
---

- Protocols may offer some useful features or properties:
  - Transport of large streams of data, or discrete messages
  - Reliability
  - Low latency, low jitter (variance of latency)
  - Efficient use of bandwidth
  - Fairness
- Applications may have differing requirements
  - Two-way voice benefits from low jitter and low latency, can tolerate occasional loss of data
  - File transfer tends to need some integrity and reliability guarantees

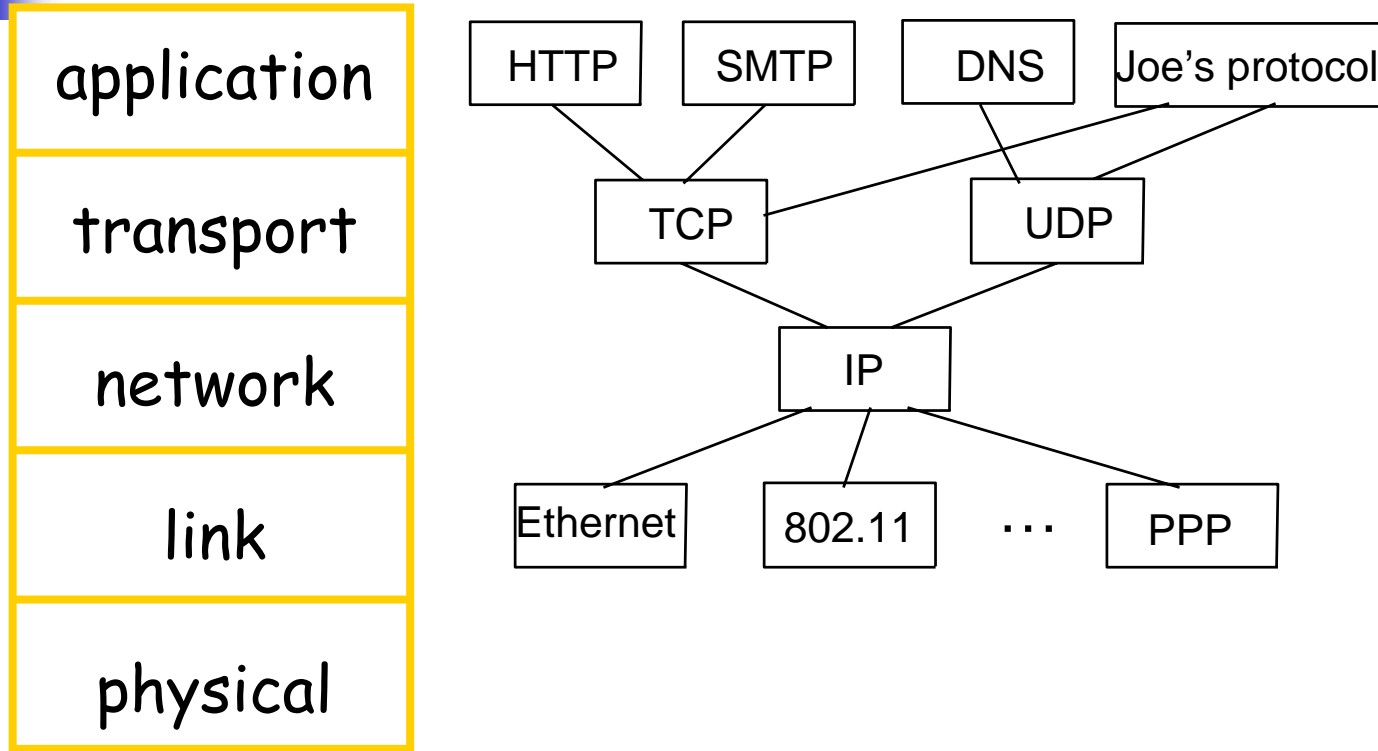
# Internet Architecture

## Bottom-up:

- **physical:** electromagnetic signals "on the wire"
- **link:** data transfer between neighboring network elements
  - encoding, framing, error correction, access control for shared links
- **network:** host-to-host connectivity
  - routing, addressing
- **transport:** host-to-host data transport
  - reliable data transport, congestion control, flow control
- **application:** anything you want to do on computer networks

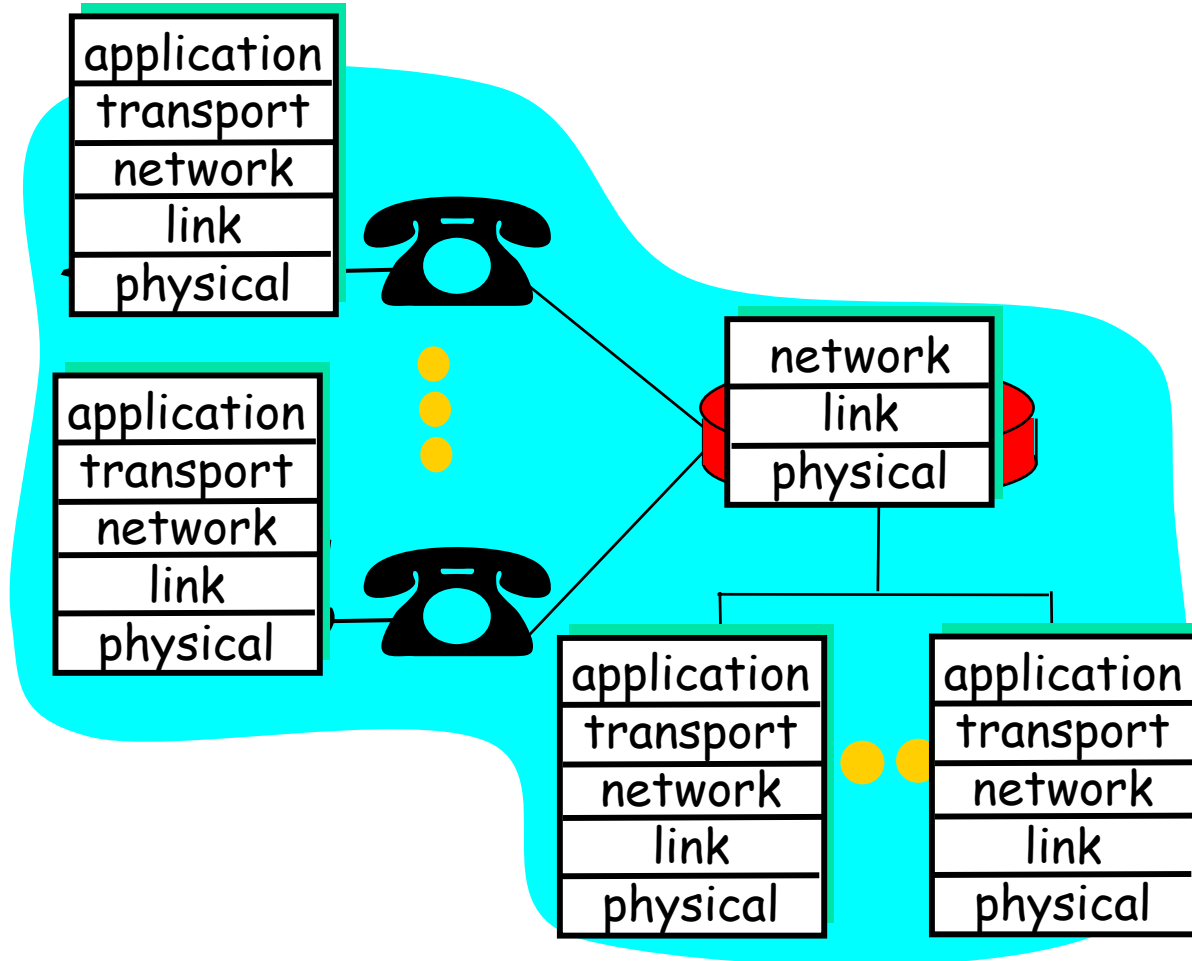


# Protocols for the Internet Architecture



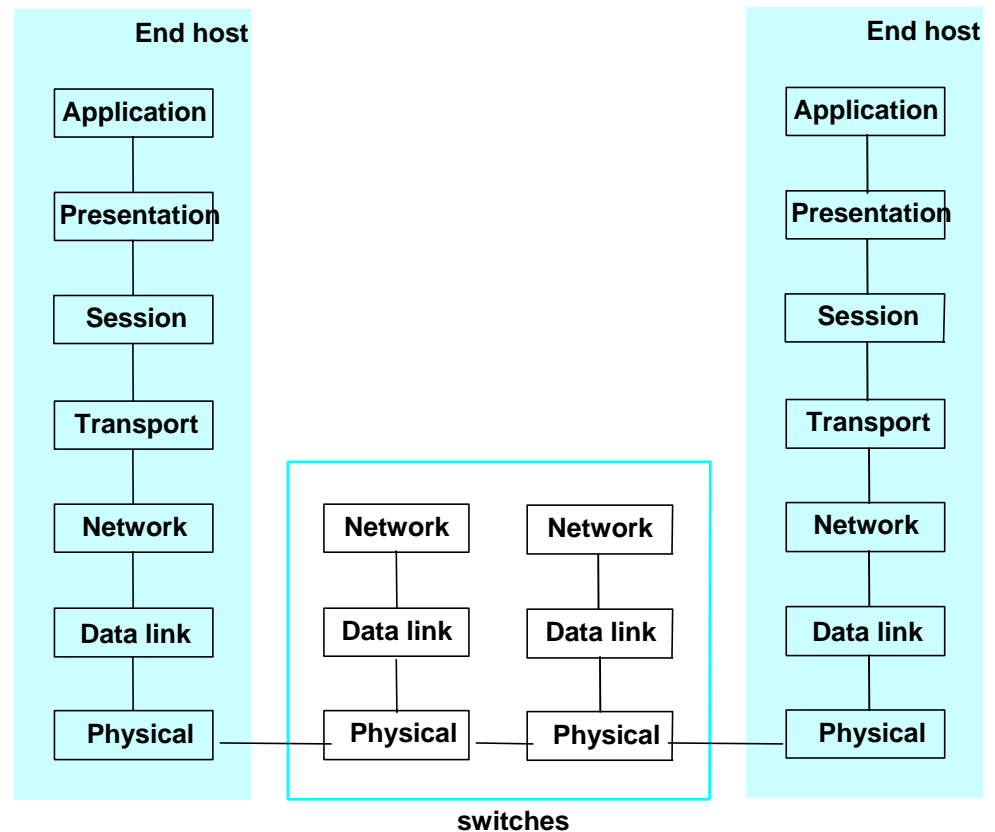
- Hourglass design.
- Difference between layers, protocols, protocol implementations.

# Internet Layering on Hosts and Switches



# OSI Architecture

- **Session layer:** tie up different transport streams belonging to the same application
- **Presentation layer:** defining uniform format for data representation



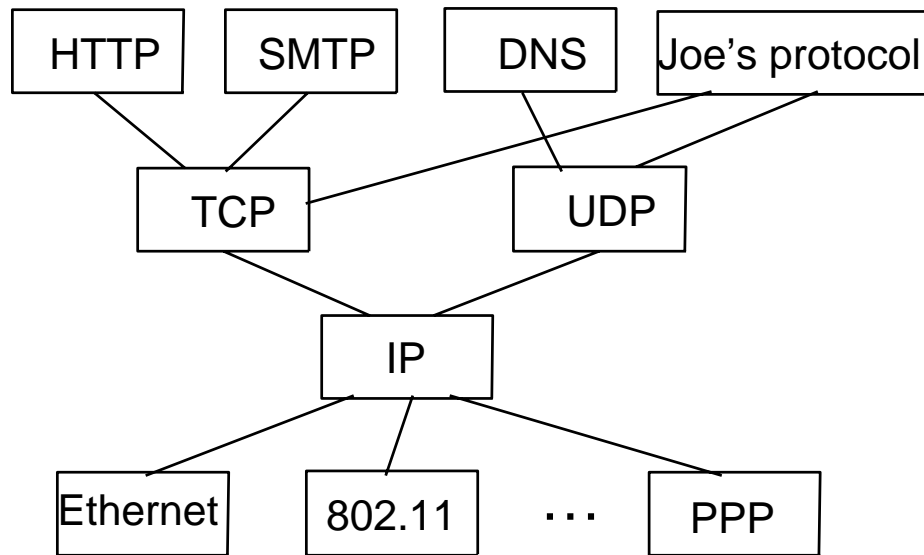


# Internet vs. OSI Architectures

---

- Internet (TCP/IP) architecture evolved from a Defense Department-supported project called ARPANET.
- OSI architecture was developed by an international body (arguably a more thought-out design).
- The Internet is dominant today largely because
  - A good TCP/IP implementation was bundled with a popular UNIX operating system (Berkeley UNIX).

# Socket Programming



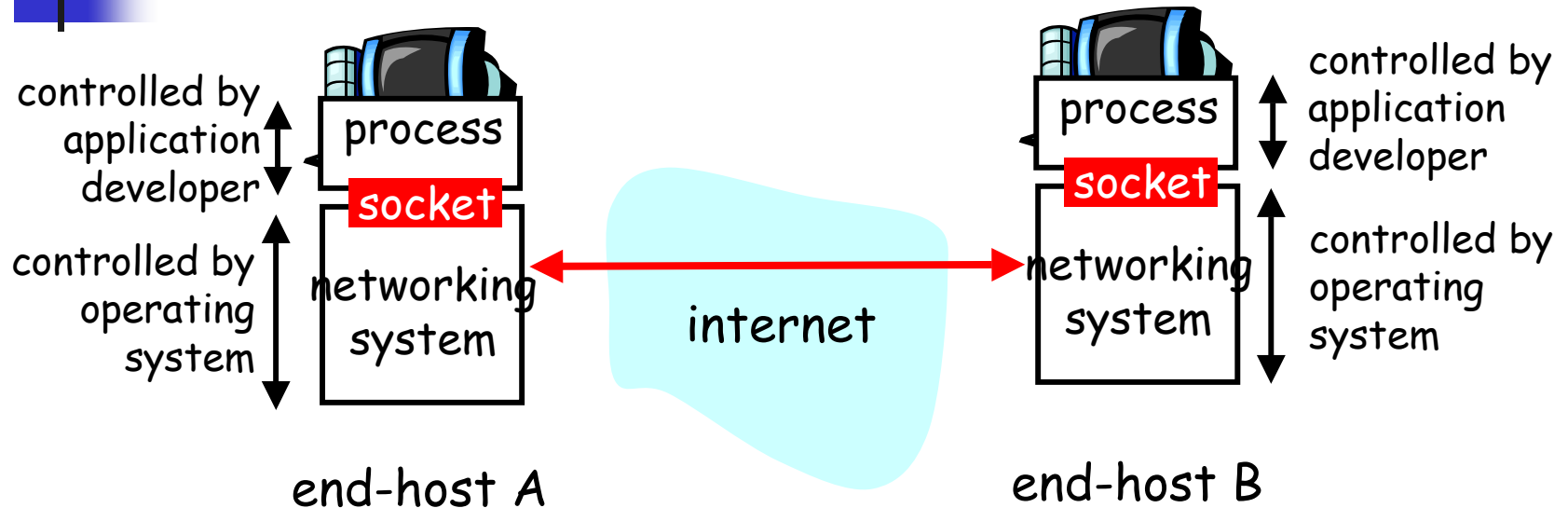
Applications

Socket

Implemented  
in hardware or  
operating system  
software

Socket: the interface between application processes and underlying networking systems

# Socket Programming (cont.)



**Socket**: an OS interface (a "door") into which application processes can communicate with remote application process.

Two types of transport services via socket API:

- connection-oriented byte stream (TCP)
- connectionless datagram (UDP)



# Port Numbers

---

- Multiple sockets might exist in each host.
- A **port number** identifies each such socket in each host.
- Each port number is a 16-bit number, ranging from 0 to 65535.
- Port numbers ranging from 0 to 1023 are called **well-known port numbers** and are restricted.



# Socket Programming with TCP

## At the server:

- server must have created socket (door) that welcomes client's contact
- server process must first be running

## Client contacts server by:

- creating client-local TCP socket
- specifying IP address, port number of the server socket
- When **client creates socket**: client TCP establishes connection to server TCP

- When contacted by client, **server TCP creates a new connection socket** for communicating with the client
  - allows server to talk with multiple clients

## application viewpoint

TCP provides connection-oriented byte stream between client and server



# Socket Programming with UDP

---

## UDP: connectionless

- no handshaking
- sender explicitly attaches IP address and port of destination to each datagram

## UDP: datagram-oriented

- clear boundaries between groups of bytes
- no ordering among datagrams

### application viewpoint

UDP provides transfer  
of groups of bytes  
("datagrams")  
between client and server



# Assignment #0

---

- Part I: little warmup on socket programming, will not be graded
- Part II: network measurement
  - measuring transmission delays for varying TCP/UDP traffic
  - use high resolution timer
  - must report stable results
- C/C++ versus Java



# Disclaimer

---

- Parts of the lecture slides contain original work of James Kurose, Larry Peterson, Keith Ross, Kai Shen, and Sandhya Dwarkadas, and Liudvikas Bukys, and remain copyrighted materials by original owner(s). The slides are intended for the sole purpose of instruction of computer networks at the University of Rochester.