

## Introduction to Computer Networks

Chen Yu  
Indiana University

---

---

---

---

---

---

---

---

## Basic Building Blocks for Computer Networks

- Nodes
  - PC, server, special-purpose hardware, sensors
  - 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: something that identifies a node often a unique byte string.
- Routing: find 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 subnet of nodes on the network.

---

---

---

---

---

---

---

---

## 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
  - Actions taken on message transmission, receipt
- Protocol define interface, not implementations
  - Allow two sides of a communication task to be independently designed / implemented.

---

---

---

---

---

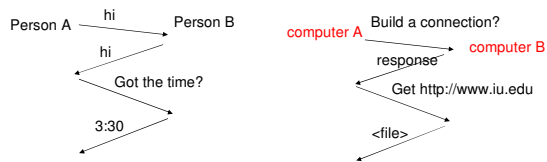
---

---

---

## An example

- A human protocol
- A computer protocol



---

---

---

---

---

---

---

---

## Network Layers

- Separation of network functions /features allow independent design and implementation.
- Layering is a vertical separation of network functions / features.
  - Hidden the complexity: each layer interface hides complexity in this layer and layers below.

---

---

---

---

---

---

---

---

## Internet Architecture

Bottom-up:

- **Physical:** electromagnetic signals on the wire.
- **Link:** data transfer between neighboring network elements. E.g. Ethernet encoding, framing, error correction
- **Network:** host-to-host connectivity. E.g. IP routing and addressing
- **Transport:** host-to-host data transport reliable data transport, congestion control, flow control. TCP/UDP
- **Application:** anything you want to do on computer networks. HTTP, FTP, SMTP




---

---

---

---

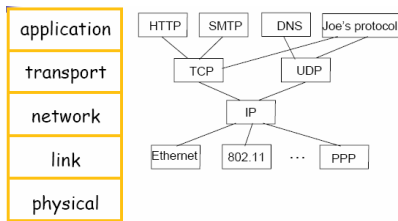
---

---

---

---

## Protocols for the Internet Architecture



- Difference between layers, protocols, protocol implementations.

---

---

---

---

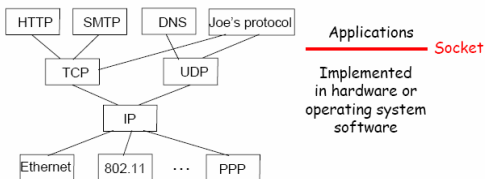
---

---

---

---

## Socket Programming



- Socket: the interface between application processes and underlying networking systems.

---

---

---

---

---

---

---

---

## Socket Programming

- Socket: an OS interface into which application processes can communicate with remote application process.
- Two types of transport services via socket APIs:
  - 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
- Usually, port numbers ranging from 0 to 1023 are called well-known port numbers and are restricted.

---

---

---

---

---

---

---

---

## TCP socket

- 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 sockets: client TCP establishes connection to server TCP.
- When contacted by client:
  - server TCP creates a new connection socket for communicating with the client, which allows server to talk with multiple clients.

---

---

---

---

---

---

---

---

## Berkeley Sockets

- Operations:  
fd = socket(): create the socket  
bind(fd,port): binds socket to local port.  
connect(fd,port): establish a connection to a remote port  
send(), recv(), write(), read(): operations for sending and receiving data  
close(fd)

---

---

---

---

---

---

---

---

## Association

- The 5-tuple that completely specifies the two processes that make up a connection.  
{protocol, local-addr, local-process, foreign-addr, foreign-process}

---

---

---

---

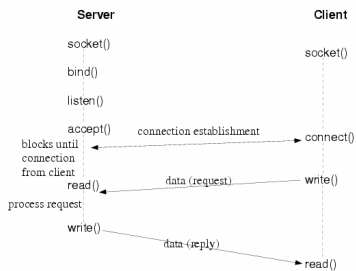
---

---

---

---

## Server-Client Model



---

---

---

---

---

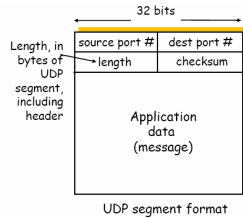
---

---

---

## UDP Socket

- UDP: connectionless  
no handshaking  
sender explicitly  
attaches IP address  
and port of destination  
to each datagram.
- Datagram-oriented




---

---

---

---

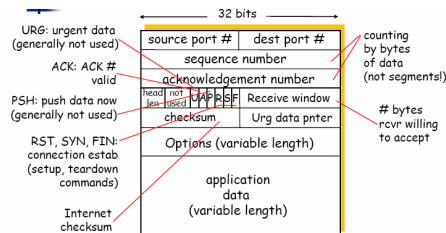
---

---

---

---

## TCP Segment Structure




---

---

---

---

---

---

---

---

## Protocols vs. Implementation

- They specify:
  - Syntax and semantics of messages exchanged.
  - Rules for when and how processes send & respond to messages.
- They don't specify the implementation:
  - Programming languages, data structures
- The goal:
  - components implemented independently can inter-operate with each other as long as they follow the protocol specification.

---

---

---

---

---

---

---

---

## Network Applications and Application-layer Protocols

- Network applications:
  - running in end systems (hosts)
  - distributed, communicating using network.
  - use communication services provided by lower layer protocols (TCP, UDP).
- Application-Layer Protocols define interface between application entities.

---

---

---

---

---

---

---

---

## Network Applications

- Some commonplace applications
  - Web and HTTP
  - FTP, telnet and ssh
  - E-mail: SMTP, POP3, IMAP
  - DNS
  - Distributed file sharing

---

---

---

---

---

---

---

---

## HTTP and web

- HTTP: hypertext transfer protocol
- Web: the application using HTTP protocol
- Client/Server model
  - Client: browser that requests, receives, and displays web objects
  - Server: web server sends objects in response to requests

---

---

---

---

---

---

---

---

## HTTP and Web

- HTTP specifies
  - Types of messages exchanged, e.g. request & response messages
  - Syntax of message types: what fields in messages & how fields are delineated.
  - Semantics of the fields, ie. Meaning of information
  - Rules for when and how processes send & respond to messages.
- Web client and server can inter-operate as long as they follow HTTP
  - Web client (browser): IE, Firefox
  - Web server: Apache, MS IIS

---

---

---

---

---

---

---

---

## Electronic Mail

- Two types of entities:
  - Mail servers
    - mailbox contains incoming messages for users
    - message queue of outgoing (to be sent) mail messages
  - User agents
    - compose, edit, read mail messages.
    - pine, outlook...

---

---

---

---

---

---

---

---

## Electronic Mail

- Two types of protocols:
  - Mail transfer protocol
    - from sender agent to the receiver's mail server (SMTP simple mail transfer protocol)
  - Mail access protocol
    - the receiver pulls mails from server to agent.
    - E.g. POP3, IMAP, HTTP

---

---

---

---

---

---

---

---

## Example

- You use your PC to compose a message with Bob's address.
- Your PC sends the message to her mail server through SMTP; the message is placed in a message queue.
- Your mail server sends message to Bob's mail server through SMTP.
- Bob's mail server places the message in Bob's mailbox
- Bob invokes his user agent to read the message.

---

---

---

---

---

---

---

---

## SMTP interaction between mail servers

```
• S: 220 www.example.com ESMTP Postfix
• C: HELO mydomain.com
• S: 250 Hello mydomain.com
• C: MAIL FROM:<sender@mydomain.com>
• S: 250 Ok
• C: RCPT TO:<friend@example.com>
• S: 250 Ok
• C: DATA
• S: 354 End data with <CR><LF>.<CR><LF>
• C: Subject: test message
• C: From: sender@mydomain.com
• C: To: friend@example.com
• C:
• C: Hello,
• C: This is a test.
• C: Goodbye.
• C:
• S: 250 Ok: queued as 12345
• C: QUIT
• S: 221 Bye
```

---

---

---

---

---

---

---

---

## Example

- telnet servername 25
- See 220 reply from the server
- Enter hello, mail from, rcpt to, data

This allows you

- Send email without using a normal email client
- ...

---

---

---

---

---

---

---

---

## Mail Access Protocols

- SMTP: delivery/storage to the receiver's server
- Mail access protocol: retrieve from server
  - POP: Post Office Protocol
  - IMAP: Internet Mail Access Protocol
  - HTTP: Hotmail, yahoo!mail

---

---

---

---

---

---

---

---

## DNS: Domain Name System

- People: multiple identifiers
  - SSN – unique, for tax reporting
  - Name – Human friendly, easy to remember
- Internet hosts:
  - IP address –used for addressing, routing on the Internet
  - "name": e.g. heart.psyc.indiana.edu – human friendly.
- Question: Map between IP addresses and names?  
DNS query: find the IP address for a given name  
core function for Internet applications  
"ssh avidd-b.indiana.edu" vs. "ssh 192.76.168.10"  
<http://www.cnn.com> vs <http://64.236.24.20>

---

---

---

---

---

---

---

---

## DNS: decentralized

- Distributed database – implemented with collaboration of many name servers distributed all over the network.  
No server has all name-to-IP address mappings.
- Why not centralize DNS?
- What if massively replicating it?

---

---

---

---

---

---

---

---

## Query

- Recursive Query:  
puts the burden of name resolution on the contacted name server.
- Iterative Query:  
contacted server replies with name of server to contact  
“I don't know this name, but ask this server”

---

---

---

---

---

---

---

---

## DNS Caching

- Once a name is learned by the name server, it caches mapping so the next query for the same name can be answered directly.  
This can happen at any step of the name lookup.
- Cache entries timeout (disappear) after some time (e.g. two days).  
Timeout is necessary because the mapping can change.

---

---

---

---

---

---

---

---

## DNS Scalability & Reliability

- Scalability  
Poor scalability at few root name servers
- Reliability  
Problems at a root server can cause big troubles.  
too many replicated root name servers would make it difficult to synchronize them.

---

---

---

---

---

---

---

---

## Distributed File sharing

- Napster: central index
- Gnutella (query flooding)  
non-hierarchical, equal status

---

---

---

---

---

---

---

---

## Peer-to-Peer Networks

- Fundamental advantage of p2p networks  
better scalability – no performance  
bottleneck.  
better robustness – more tolerant to  
random failures, intentional attacks.
- Challenge: peer coordination without  
complete global knowledge.

---

---

---

---

---

---

---

---

## Peer

- A pure peer-to-peer network does not  
have the notion of clients or servers, but  
only equal *peer* node that simultaneously  
function as both "clients" and "servers" to  
the other nodes on the network.

---

---

---

---

---

---

---

---

## Keyword search

- Users input a few keywords, the system returns a list of documents matching the keywords. E.g. google
- Google maintains a central search index:  
a search index contains a list of all searchable words, each of which contains a list of documents relevant to the word.  
interaction document lists for multiple-word queries.  
Scalability?  
Robustness?

---

---

---

---

---

---

---

---

## Peer-to-Peer Keyword Search

- Splitting the central search index into smaller pieces and distributing them in peer-to-peer fashion.
- Challenge: make them work collaboratively to achieve similar speed and quality of search.

---

---

---

---

---

---

---

---

## Two solutions

- Splitting based on keywords  
split the index database to many pieces based on keywords and distribute them to many nodes in the network.
- Split based on documents

---

---

---

---

---

---

---

---

## Putting them together

- Split based on keywords  
weakness: transferring large data segments for multiple keyword queries.
- Split based on documents  
weakness: too many sites to visit for each query.

---

---

---

---

---

---

---

---