

# CmpE 473

## Internet Programming

Pinar Yolum  
[pyolum@cmpe.boun.edu.tr](mailto:pyolum@cmpe.boun.edu.tr)

Department of  
Computer Engineering  
Boğaziçi University

### Course Information

- Topics
- Work Schedule
- Grading
- Resources
- Academic Integrity

# Chapter 1

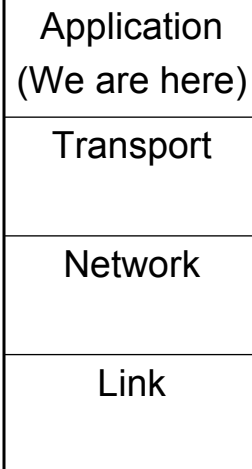
## Architectures

## Internet Architectures

- A set of nodes collaborate to carry out a job
  - A node wants to print, but doesn't have access to a printer
  - A node needs data that is available at a different node
- A common language for communication
  - Usually not a full-fledged language
  - Protocol that specifies what to do in specific situations

# TCP/IP Network Model

1. Link
  - Network hardware and device drivers
2. Network (IP)
  - Communication
  - Addressing
  - Routing
3. Transport (TCP, UDP)
  - Communication among programs
4. Application (HTTP, FTP, DNS)
  - End-user applications



Spring 2005— Pinar Yolum

5

# Transport

- Transmission Control Protocol (TCP)
  - Connection-based (like making a phone call)
  - Reliable (Failures are reported)
  - Order preserving (Why important?)
  - HTTP, FTP
- User Datagram Protocol (UDP)
  - Connectionless (like posting a letter)
  - No guarantee on order or delivery
  - PING, ECHO

Spring 2005— Pinar Yolum

6

## Ports

- Delivering incoming data to the right application
  - One connection
  - Multiple applications
- Application picks a (unique) port number
- Identify the computer (IP address) + identify the application (port number)
- Port numbers
  - Range from 0 to 65,535
  - 0 - 1023 are reserved ports
- Ports: HTTP 80, FTP 21, ...
- Example

Spring 2005—Pinar Yolum

7

## Protocol

- Set of rules that will be followed by the participants
  - Events that take place
  - The initiators
  - The timing of events
  - The data formats
- Does not specify how the protocol should be implemented
- Example\*: Hypertext Transfer Protocol (HTTP)

Spring 2005—Pinar Yolum

8

## Protocol Properties (1)

- Unambiguous:
  - The protocol state should state clearly what should be done in a situation
  - No misunderstandings
- Complete:
  - The protocol should cover all possible requests
  - Garbled data?
  - Illegal request?

## Protocol Properties (2)

- Extendable:
  - The protocol should allow new requests and responses to be added
  - Versioning of protocols
  - World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) work to standardize versions of protocols
- Accessible:
  - Different clients and different servers may be designed and implemented by different programmers
  - Should still be able to speak the same language

## Protocol Types

- Synchronous
  - Client sends a request and blocks
  - Server responds
  - Example: HTTP, SMTP
- Asynchronous
  - Clients and server send information at the same time
  - Example\*: TELNET
- Deferred Synchronous
  - Continue operation until a certain point and then wait
  - Example: CORBA

## Client/Server Architecture (1)

- Client and server asymmetric in capabilities
- Client\*:
  - Represents a user
  - Program that request tasks from servers
- Often users interact with a client through a GUI
- Client translate the user requests to protocol tokens
- Clients initiate the interaction with a server

## Client/Server Architecture (2)

- Server: Program that waits for incoming communication requests from a client
- Usually has some resources that the client does not have
  - Bandwidth, access to printer
- Takes the request
  - Process data
  - Perform a task
  - Return results client
- Examples: Mail servers, Print servers, Web servers

## Client/Server Architecture (3)

- Common protocols
  - Mail servers: SMTP
  - Print servers: LPP, IPP
  - Web servers: HTTP
- On request, the server performs a service and sometimes returns a success/fail response
- The client pulls the information

## Server Tasks

- Application-dependent
- Authentication: Verify the identity of the client
  - Check if the Username/Password supplied by the client matches the one already stored
- Authorization: Check that the client has rights to perform what it wants to perform
  - Check mail of another user?
  - Delete a file?
- Concurrency: Allow multiple clients to use the server at the same time

## Two-Tier Architecture

- Example\*: registration.boun.edu.tr
  - What does the client have? Database, logic?
  - Who uses the client?
  - What does the server have? Database, logic?
  - Two tiers?

## Three-Tier Architecture

- Three separate layers
- Presentation tier
  - Runs on client
  - Provides user interface
  - Invokes business logic
- Business logic tier
  - Runs on server
  - Has application logic, business rules, etc.
- Data tier
  - Runs on a database server
  - Stores and provides access to data
- Advantages?

Spring 2005— Pinar Yolum

17

## Invocation vs. Message-Oriented

- Invocation
  - Assumes knowledge of the other party
  - Gives access to others' resources
- Message-Oriented
  - Recipient deals with the message
  - Recipient can change its functioning
  - Increased abstraction

Spring 2005— Pinar Yolum

18

## Message-Oriented Middleware

- Intermediary message system
- Post and read messages from queues (point to point).
- Publish and subscribe (topic-based)
- Reliability guarantees of delivery or failure notification to sender.
- Some messages correspond to event notifications.
- Examples: IBM MQ Services, Java Message Service (JMS)

## Point-to-point

- Each node has an individual incoming and outgoing queue
- Nodes are uniquely identified
- Send a message: Write to outgoing queue
- MOM: Transfers the message to the recipients incoming queue
- Receive a message:
  - Automatically notified
  - Read from incoming queue
- Synchronous or asynchronous?

## Publish-and-subscribe

- Nodes subscribe to topics or events that they are interested in
- Nodes can publish about topics
- MOM: Notifies the subscribers of the new information
- Useful for group communication\*

## Peer-to-Peer Architectures

- Richer models of interaction (compared to client-server)
  - Symmetric client-server
  - Asynchrony
  - Federation of equals
- Control on all peers (not just the client)
- Peers enter and leave; the system stays
- Difficult to shut down
  - No bottleneck

## Symmetric client-server

- *Symmetric client-server*: (callbacks) each party can be the client of the other.
- Example:
  - File sharing
  - Each peer hosts some files (e.g., mp3s)
  - No central server, no central index
  - Query peers to find the file you need
  - Different protocols: Gnutella, Freenet

## Gnutella

- Each peer runs on a user's computer
- The user enters search words for a song
- The client contacts other peers to see if they have the MP3's the user is looking for
- The query propagates in the system
- Until
  - Matching MP3s are found (the peer contacts the user's node directly)
  - Time-to-live (TTL) Reached (won't propagate after a predefined number of hops)

## Asynchrony

- *Asynchrony\**: while the request-response paradigm corresponds to pull, asynchronous communication corresponds to push.
  - Intelligence on the server (pushing) side.
  - What can you push? When do you push?
  - Spamming?

## Federation of equals

- *Federation of equals\**: when the participants can enact the protocols they like.
  - Equal share and control over interactions
  - Choose and start a protocol based on needs