Network Programming TDC 561

Lecture # 1

Dr. Ehab S. Al-Shaer

School of Computer Science & Telecommunication
DePaul University
Chicago, IL

1

Network Programming

- Goals of this Course:
  Studying, evaluating and developing alternative client-server network architectures
- Perquisites
- Text Books
- Assignments and the course Project
- Rules and Policies
- What do you expect?
- Applications of this course

2

Applications of Network Programming

- Network Tools
  - traditional: telnet, ftp, rsh, rlogin, SMTP,
- Internet Tools
  - gopher, HTTP, NTP, Chat rooms
- Collaborative Tools
  - Application Sharing , Desktop Conferencing, Distance Learning
- Distributed Object Computing
  - SUN RPC, CORBA, JAVA RMI
- Distributed Databases

3

IRI: Interactive Distance Learning

MASH: Internet-based Video Conferencing

Network Overview OSI Seven Layer Model

- Layering (service abstraction)
  - Problem decomposition
  - Modular design
  - Interoperability
- OSI Principles:
  - Different level of abstraction
  - Well-defined functions
  - Supporting open system
  - Well-defined interfaces (layers boundaries)
  - Different functions are in different layers

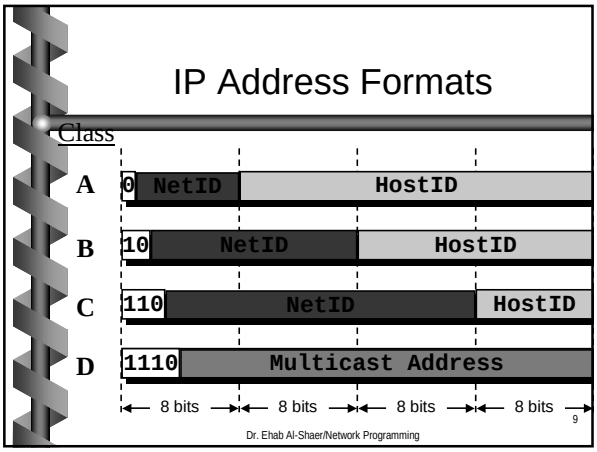
Network Overview
OSI Seven Layer Model

- OSI Seven Layers
  - Physical Layer
  - Data Link Layer
  - Network Layer
  - Transport Layer
  - Session Layer
  - Presentation Layer
  - Application Layer
- Internet (TCP/IP) Model
  - Application Layer (telnet, ftp)
  - Transport Layer (TCP)
  - Network Layer (IP)
  - Host-to-Network: device drivers of various MAC protocols (Ethernet, token-ring)

TCP/IP Overview
(Transmission Control Protocol/Internet Protocol)

- Internet Protocol (IPv4 and IPv6)
  - IP is the network layer: a packet delivery service (host-to-host).
  - *Connectionless*: each datagram is independent of all others.
  - *Unreliable:* there is no guaranteed delivery
  - Fragmentation / Reassembly (based on MTU).
  - Routing.
  - Error detection (link).

IP Address Formats

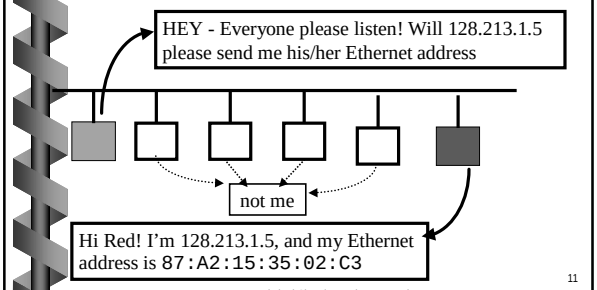


TCP/IP Overview (cont.)

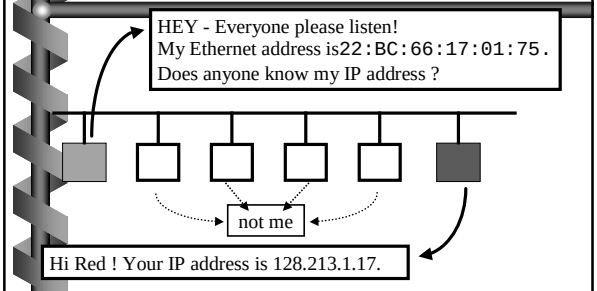
- IP Addresses
  - 32 bit, Logical, unique
  - eg. condor.cs.depaul.edu is 140.192.1.6
- IP Packet Format:

VERS	HL	TOS	Fragment Length
Datagram ID		FLAG	Fragment Offset
TTL	Protocol		Header Checksum
Source Address			
Destination Address			
Options (if any)			
Data			

ARP
(Address Resolution Protocols)

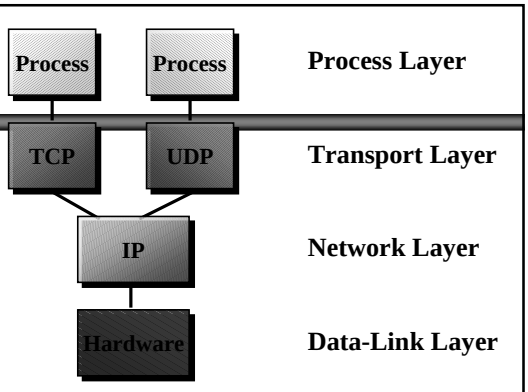


RARP
(Reverse Address Resolution Protocols)



Transport Layer of TCP/IP

- IP is the network protocols, means TCP is the transport protocol, right?
  - Not exactly, TCP is part of the transport protocol, and the other part is UDP (User Datagram Protocol).
- Ports
  - Abstract destination point
  - 16 bit positive integer for UDP and TCP
  - Reserved/ Well-known ports: 1-1023
  - Registered ports: 1024-49151
  - Dynamic ports: 49151-65535



Transport Layer of TCP/IP

- TCP supports:
  - Connection-oriented (establish, terminate, notify)
  - Reliable (ordered, no loose, and no duplicates)
  - Buffering (configurable, though)
  - Byte stream
  - Full-duplex
- UDP supports:
  - Connectionless
  - Unreliable
  - Message stream (Datagram)
  - Full-duplex

TCP Segment Format

Source Port			Destination Port
Sequence Number			
Request Number			
offset	Reser.	Control	Window
Checksum			Urgent Pointer
Options (if any)			
Data			

TCP/IP Overview

- UDP Datagram Format

Source Port	Destination Port
Length	Checksum
Data	

- Hmmmmm, Which one to use: TCP or UDP?

Client-Server Architecture Design

- Terminology
  - A client initiates communication
  - A server waits for incoming requests
  - Integrated model (e.g. server as client)
  - Does TCP/IP care?
- Client-Server Paradigms
  - Message passing (Synch and Asynch)
  - Remote Procedure Call (Syncg and Asynch)

Client-Server Architecture Design (cont.)

- Client-Server Communication Models
  - Point-to-point Vs. Multi-point/Group
    - A client to multiple servers
    - A server to multiple clients
  - Broadcast Vs. Multicast
  - Connection-oriented Vs. Connectionless

Client-Server Architecture Design (cont.)

- Servers Design Issues
  - System Ports Vs. User Ports
  - Statefull Vs. Stateless Servers
  - Privileges and Complexity: authentication, authorization, data sharing, privacy, protection
  - Robustness and Reliability
  - Client or Message Independent
  - Client or Message Independent (dispatching)
  - Concurrency
    - Multi-processes
    - Multi-threaded
    - I/O Multiplexing
    - Asynchronous I/O

Client-Server Architecture Design (cont.)

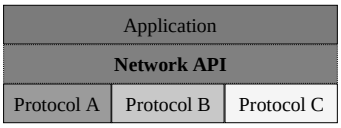
- Client Design Issues
  - Server Independent (dispatching)
  - Stateless Protocol/Messages (e.g. slide tool)
  - Rich and Flexible Parameters (e.g. telnet)

Sockets Programming

- Network API
- Socket Structures
- Socket Functions

Network Application Programming Interface (API)

- The services provided by the operating system that provide the interface between application and protocol software.



Network API

- Generic Programming Interface.
- Support for message oriented and connection oriented communication.
- Uses the existing I/O services
- Operating System independence.
- Support multiple communication protocol suites (families): IPv4, IPv6, XNS, UNIX.
- Provide special services for Client and Server?

TCP/IP

- There are a variety of APIs for use with TCP/IP:
  - Sockets
  - TLI
  - Winsock
  - MacTCP

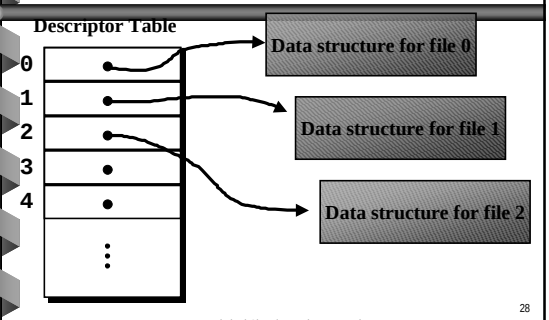
Functions needed

- Specify local and remote communication endpoints
- Initiate a connection
- Wait for incoming connection
- Send and receive data
- Terminate a connection gracefully
- Error handling

Berkeley Sockets

- Generic:
  - support for multiple protocol families.
  - address representation independence
- Uses existing I/O programming interface as much as possible.

Unix Descriptor Table



Socket

- A socket is a process-level abstract representation of a communication endpoint.
- Sockets work with Unix I/O services just like files, pipes & FIFOs.
- Sockets (obviously) have special needs:
  - establishing a connection
  - specifying communication endpoint addresses