CISC 3120

C18: Networking and Network I/O

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

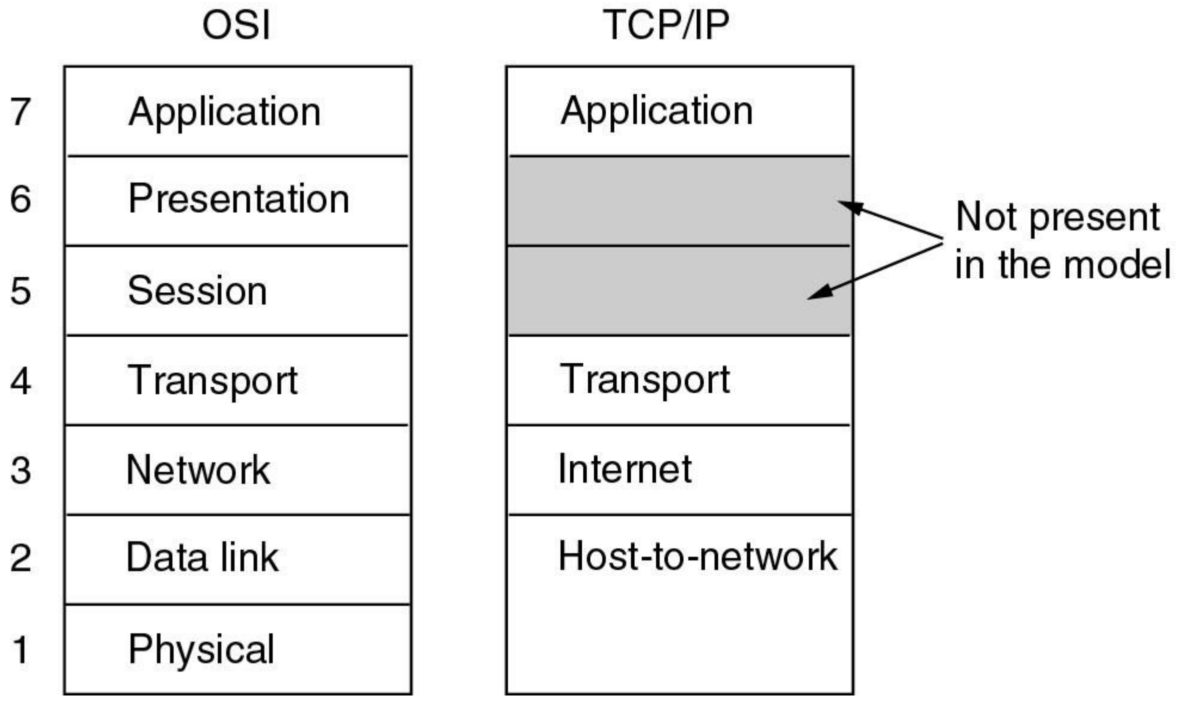
- Networking fundamentals
- Network interfaces
- Sockets and network I/O
- Multi-threading
- Client/server and peer-to-peer architectures
- Object serialization

Sample Programs

- All sample programs are in the “network” folder of the “sampleprograms” repository

Layered Architecture

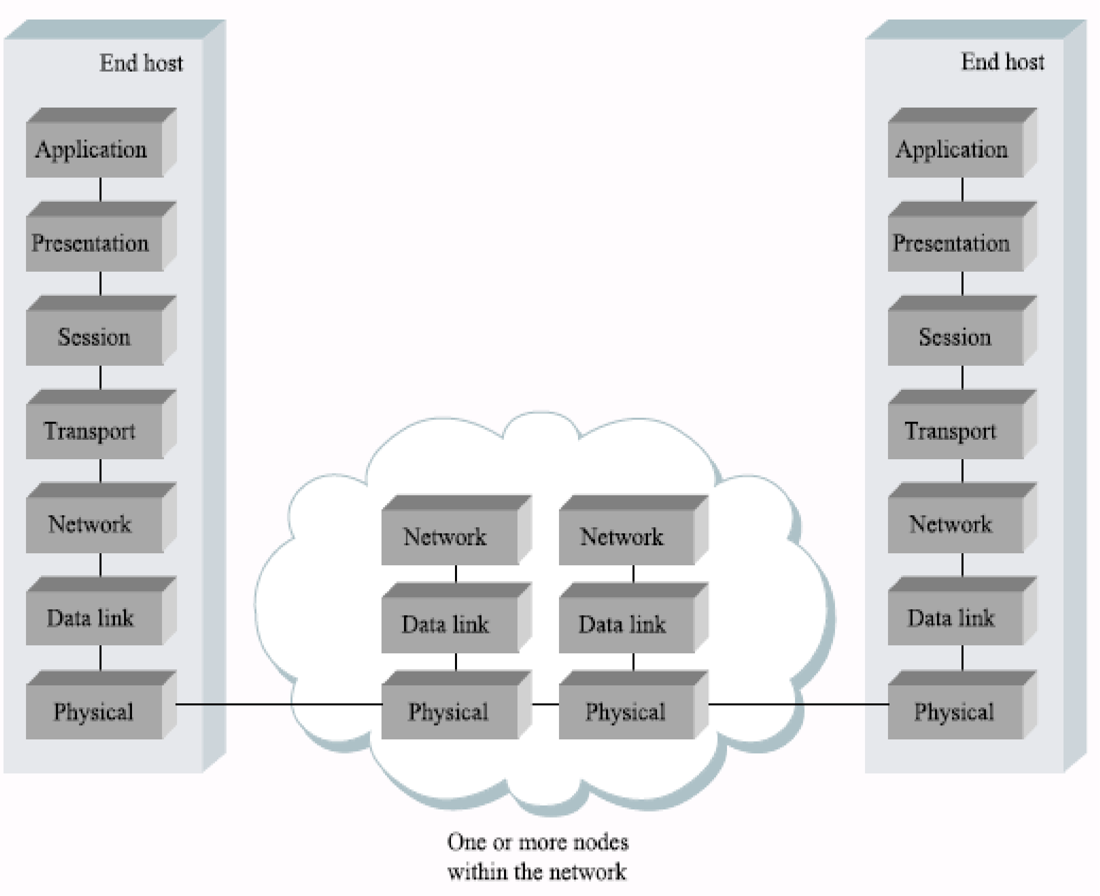
- OSI model and TCP/IP



	OSI	TCP/IP
7	Application	Application
6	Presentation	
5	Session	
4	Transport	Transport
3	Network	Internet
2	Data link	Host-to-network
1	Physical	

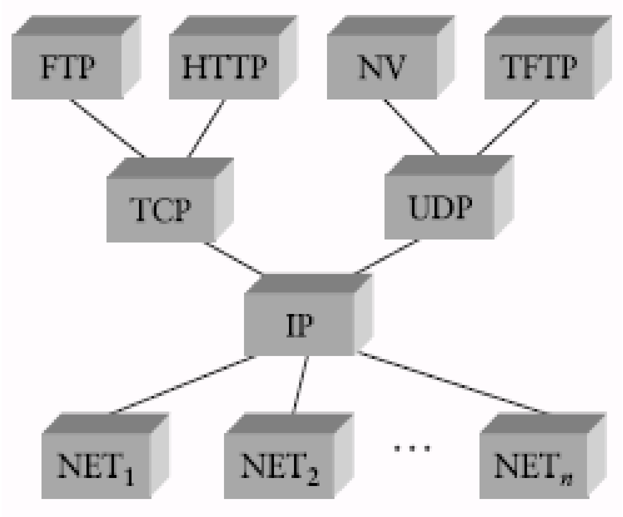
Not present in the model

Two Hosts and a Router

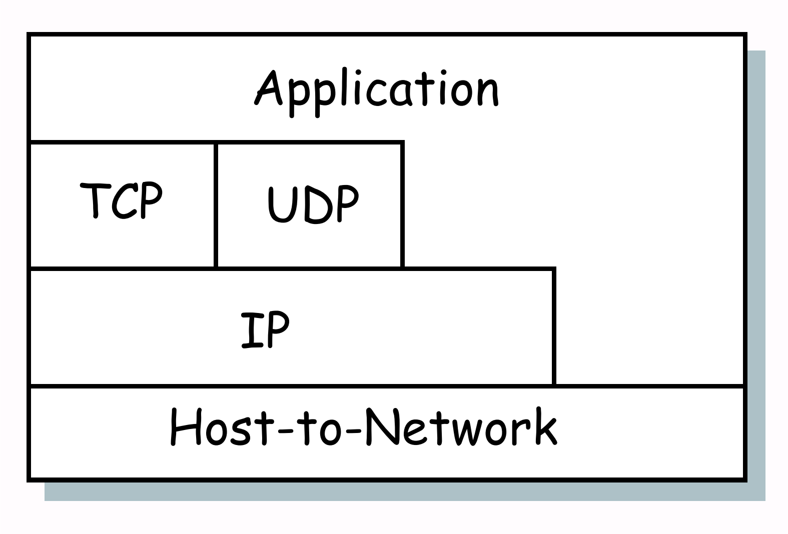


The Internet Architecture (TCP/IP)

- Layering is not strict, hourglass design, representative implementation



Internet protocol graph.



Internet architecture.

Network Protocol

- A distributed algorithm and associated data structures for data communication over a network
- Each layer may have many protocols

Host and Network Interface

- A host may have multiple network interface
- A network interface typically implements physical layer and link layer functionality (or the host-to-network layer)

IP

- The Internet Protocol
- Communication protocol for hosts
- Transmit and receive IP packets
- To identify a host, use IP address or host name

IP Address

- Currently deployed Internet Protocols
  - IP version 4 (IPv4)
  - IP version 6 (IPv6)
  - The very first field in an IP packet indicates the version of IP protocol
  - Globally unique except local networks & private networks
  - Hierarchical (network number + host number)

IPv4 Address

- 32 bit integer
  - Divided into two parts
    - Network number and host number (using prefix or network mask)
- Human-readable form
  - IPv4 numbers-and-dots notation, each number corresponds to a byte in the address
  - Example: 146.245.201.50
- Facing exhaustion of address space, moving to IPv6

IPv4 Private Networks

- Private networks
  - Not routable in a public network
  - 24-bit block 10.0.0.0–10.255.255.255
  - 20-bit block 172.16.0.0–172.31.255.255
  - 16-bit block 192.168.0.0–192.168.255.255

IPv4 Link Local and Loopback Address

- Link local address
  - Not routable
  - For configuration purpose
  - 169.254.0.0/16 (16 bit block: 169.254.0.0 – 169.254.255.255)
- Loopback address
  - Only stay within the host
  - 127.0.0.0/8 (24 bit block: 127.0.0.0 – 127.255.255.255)

Broadcast, Multicast, and Unicast

- The addresses are divided into broadcast, multicast, and unicast address
  - Broadcast address: all 1's in the network number for the network
  - IPv4 Multicast: 224.0.0.0/4 (224.0.0.0 – 239.255.255.255)

A Few IPv4 Address Types

Address Type	Binary Prefix	IPv4 CIDR Notation
Private Network	1100 0000 1010 1000	192.168.0.0/16
	1010 1100 0001	172.16.0.0/12
	1010 0000	10.0.0.0/8
Loopback	0111 1111	127.0.0.0/8
Link-local Unicast	1111 1110 10	169.254.0.0/16
Documentation (TEST-NET-1)	1100 0000 0000 0000 0000 0010	192.0.2.0/24
Documentation (TEST-NET-2)	1100 0110 0011 0011 0110 0100	198.51.100.0/24
Documentation (TEST-NET-3)	1100 1011 0000 0000 0111 0001	203.0.113.0/24
Multicast	1110	224.0.0.0/4
Global Unicast	Everything else (with exceptions)	

IPv6 Address

- 128 bits/16 bytes in length
- IPv6 Notation: a human friendly text representation
- x:x:x:x:x:x:x:x where x is a 16-bit (or 2-byte) hexadecimal number, e.g.,
  - `47CD:1234:4422:ACO2:0022:0022:1234:A456`
- Contiguous 0s can be compressed, e.g.,
  - `47CD:0000:0000:0000:0000:0000:A456:0124`
  - can be written as
  - `47CD::A456:0124`

A Few IPv6 Address Types

Address Type	Binary Prefix	IPv6 Notation
Unspecified	00…0 (128 bits)	::/128
Loopback	00…1 (128 bits)	::1/128
Multicast	1111 1111	FF00::/8
Link-local Unicast	1111 1110 10	FE80::/10
Private Network	1111 110	FC00::/7
Documentation	0010 0000 0000 0001 0000 1101 1011 1000	2001:0DB8::/32
Global Unicast	Everything else (with exceptions)	

Look up Host IP Address

- Be aware that a host may have multiple IP addresses since an IP address is assigned to a network interface on a host
- Windows
  - ipconfig
- Mac OS X
  - ifconfig
- Linux
  - ip address or ifconfig

Host Name

- A host can also be identified by its name
- Domain Name Service (DNS)
    - A global name database
    - Example
        - www.brooklyn.cuny.edu
        - www.google.com
    - Communications are done using IP addresses
        - DNS provides translation

Look up a Host’s IP address

- Use nslookup, available on many operating systems (Windows, Mac OS X, Linux …)
- Example
  - nslookup www.google.com
  - nslookup www.brooklyn.cuny.edu

Work with Network Interface

- See LinkNetInterfaceExplorer in the network folder of the sampleprograms repository
- In Java, use java.net.NetworkInterface to deal with network interfaces on a host

Questions

- Network architecture and layered model
- Host, node, and network interface
- IP addresses
  - IPv4 and IPv6
- Practical operations
  - Look up hosts' IP addresses
  - Examine network interfaces

TCP and UDP

- Transport Control Protocol
- User Datagram protocol
- Communication protocol for <u>processes</u> (a process represents a running program)

UDP

- User Datagram Protocol
- Transmit independent datagram one at a time
- Communication is not reliable
  - No guarantee the order of datagrams
  - No guarantee the delivery of datagrams

TCP

- Connection-oriented reliable byte stream
  - Must establish connection
  - Guarantee delivery of data, otherwise, an error is reported
  - Create an abstraction data are transmitted or received one byte at a time
  - Maintain the order of the data

TCP and UDP Port Numbers

- UDP port numbers
  - 16 bit integer
  - Use them to differentiate different processes on a host
- TCP port numbers
  - 16 bit integer
  - Use them to differentiate different processes on a host

List TCP/UDP Port Statistics

- Use netstat , available on many operating systems (Windows, Mac OS X, Linux …)
- Windows
  - Examples
    - netstat -n -o -p TCP; netstat -f -o -p TCP; netstat -n -o -p UDP; and netstat -f -o -p TCP
- Linux
  - Examples
    - netstat –n –p -a -t; netstat –p -a -t; netstat –n –p -a -u; and netstat–p -a -u
- Mac OS X
  - Examples
    - netstat -n –a –p tcp; netstat –a –p tcp; netstat -n –a –p udp; and netstat –a –p udp;

Some Practical Considerations

- Are the port (TCP, UDP, both) available to to use in the program?
  - 1 – 1023 are privileged
  - Registered ports
    - Well-known ports (iana.org)
      - See /etc/services on Mac OS X, or, Linux or Unix
      - See C:\Windows\system32\drivers\etc on Windows
- Does the host-based or network-based firewall get in your way (at home, at the college, or at the coffee shop …)?

Programming with TCP and UDP

- Most network applications uses TCP or UDP to communicate
- Program at the application layer for Java application
- Typical no need to concern with TCP or UDP
  - Use java.net package or other network related packages
  - TCP communications: The Socket, ServerSocket, URL, and URLConnection classes
  - UDP communications: The DatagramPacket, DatagramSocket, and MulticastSocket classes

Socket

- An end-point of a two-way communication link between two programs running on the network
  - A combination of IP address and port number
- Socket classes
  - Represent the connection between a client program and a server program.
  - Socket class: for the client side of the connection
  - ServerSocket class: for the server side of the connection
  - Low-level communication directly using TCP

Client-Server Application using Socket

- A server programs runs on a host and has a socket that is bound to a port number.
- The server just waits, listening to the socket for a client to make a connection request.
- The client attempts to connect to the server program by using the endpoint's address and port.
    - To identify itself to the server, the client binds to a local port number (usually assigned by the system) that it will use during this connection.
- When the server accepts the connection, the server does the following,
    - It gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client.
    - It creates a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.
- On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.
- The client and server can now communicate by writing to or reading from their sockets.

Example: FileDownloader

- See the FileDownloader in the sampleprograms repository
- Can you use try-with-resources to make it cleaner?

Use Datagram

- Datagram
  - Independent, self-contained message
  - Unreliable: there is no guarantee on arrival, arrival time, and order of arrival
- UDP communications: The DatagramPacket, DatagramSocket, and MulticastSocket classes
- Often use for broadcasting or multicasting

Example: PingPongMessenger

- See the UdpPingPongMessenger in the sampleprograms repository
- Server
  - Create a DatagramSocket bound to one or more addresses and a listening port, both of the server
  - Receive a packet
  - Prepare and Send a reply packet
- Client
  - Create a DatagramSocket, let JVM/OS determine the port number
  - Prepare and send a packet, filled with destination address and port number
  - Receive a packet

Datagram Multicasting

- One important use case of Datagram is to realize multicasting
  - Multicasting: one-to-many
- Where do you think multicast can be very helpful (or what application may be better off to use multicast than unicast?) How is it being helpful?

Example: MulticastDemo

- See the UdpMulticastDemo in the sampleprograms repository
- Server
  - Create multicast group using a multicast address
  - Send packets to the multicast group
- Clients
  - Join the multicast group
  - Receive packets from the server

Questions

- TCP and UDP
  - When to use them?
- Practical consideration
  - TCP and UDP in practice
- Sockets and network I/O

Let's look at another example

- TcpMessengerHalfDuplex

Communication Channel

- Simplex
- Half duplex
- Full duplex

How do we achieve full-duplex?

- Two programs (processes) at each side
  - One does receiving
  - One does transmitting
- Two threads at each side
  - One does receiving
  - One does transmitting

Process and Thread

- Processes and threads exist to support multitasking
- Process
  - *A program in execution and associated data structures (e.g., a process control block)*
  - *A process may have one or more threads of execution*

Process and Thread: Comparison

	Process	Thread
Address space	A process usually has its own address space (implication: two processes cannot access each other's variables)	Multiple threads of a process shares the same address space (implications: they can access the process's variables)
States and Controls	Processes usually have larger set of states and supporting data structure	Multiple threads of a process shares share the process states and other resources, in addition to memory
Interfacing	Inter-process communication (IPC)	Share the process memory
Context-switch	Generally slower than threads	Generally faster than processes when switching between different processes

Multithreading in Java

- Two approaches
  - Implementing the Runnable interface
  - Extending the Thread class
- Prefer to implementing the Runnable interface

Multithreading: Text-based App Example

- See TcpMessengerThreadedFullDuplex in the network directory of the sampleprograms repository
- One thread deals with InputStream of a Socket
- One thread deals with OutputStream of the Socket

Multithreading: JavaFX App Example

- See TcpMessenger in the network directory of the sampleprograms repository
- One thread deals with listening, accepting and InputStream of a Socket
- An EventHandler deals with OutputStream of the Socket

Network Applications

- Client-Server
- Peer-to-Peer
- Hybrid

Questions

- Channel
  - Simplex, half-duplex, full-duplex
- Application
  - Client/server, peer-to-peer, hybrid
- Process and threads
- Application examples

Questions

- Networking fundamentals
- Network interfaces
- Sockets and network I/O
- Multi-threading
- Client/server and peer-to-peer architectures

Assignments

- Practice
- Project 4