Overview

- What is a computer network?
- What is the Internet?
- What are the popular network reference model?
  - OSI, TCP/IP
- What is the main responsibilities and issues for each layer?

TCP/IP Layering Architecture

- Data-Link: communication between machines on the same network.
- Network: communication between machines on possibly different networks.
- Transport: communication between processes (running on machines on possibly different networks).

Addresses at Layers

- Physical Layer: no address necessary
- Data Link Layer – address must specify the host
  - MAC address
- Network Layer – address must identify the network
  - IP address
- Transport Layer - address must identify the destination process.
  - Port #
Date Link Layer Functionality
- What is the main functionality of the data-link layer?
  - Provides reliable transfer of information between two adjacent nodes
- What is the service provided by the data-link layer?
  - Encoding: Convert bits to signals and recover bits from received signals
  - Framing: decide on a minimum unit for sending bits
  - Error detection and/or correction of frames
    - Parity, CRC
  - Flow control
    - ARQ, Sliding WINDOW

Framing
- A frame is a group of bits, typically in sequence
- Issues:
  - Frame creation
  - Frame delineation
- Use starting and ending characters (tags) to mark boundaries of frame
  - Problem: what if tag characters occur in the data or control portions of the frame
- Use preamble + packet length

Error Control
- No physical link is perfect, bits will be corrupted
- We can either:
  - Detect errors and request retransmission
  - Or correct errors without retransmission
- Error Detection
  - Parity bits
  - Polynomial codes or checksums
  - Cyclic Redundancy Check (CRC)
    - Given a polynomial code and a message, what is the checksummed message?
    - Given a checksummed message, can you determine whether there are errors.

Ethernet - A Real Data-Link Layer
- What is the header of a typical Ethernet frame?
- How is Ethernet connected?
- What is the address used in Ethernet?
  - Example: 08:00:e4:b1:20
- What is the basic protocol? -- CSMA/CD
  - Multi-access (shared medium)
    - many hosts on 1 wire
  - Carrier sense:
    - can tell when another host is transmitting
  - Collision detection:
    - How can a device detect collision?
    - How to avoid two devices collide again?

Transmit Algorithm
- If line is idle…
  - send immediately
  - upper bound message size of 1500 bytes
  - must wait 9.6us between back-to-back frames
- If line is busy…
  - wait until idle and transmit immediately

Collisions
**Ethernet Backoff Algorithm**

- If collision,
  - How to detect collision?
  - Jam for 32 bits, then stop transmitting frame
  - Minimum frame is 64 bytes (header + 46 bytes of data)

**WHY?**

- Choose one slot randomly from \(2^k\) slots, where \(k\) is the number of collisions the frame has suffered.
- One contention slot length = 2 x end-to-end propagation delay
- If 16 backoffs occur, the transmission of the frame is considered a failure.

---

**IP - Network Layer**

- Provide delivery of packets from one host in the Internet to any other host in the Internet, even if the hosts are on different networks
  - **Connectionless** delivery (each datagram is treated individually).
  - **Unreliable** (delivery is not guaranteed).
  - Fragmentation / Reassembly (based on hardware MTU).
  - Routing.
  - Error detection

---

**IP Addresses**

- IP addresses are not the same as the underlying data-link (MAC) addresses. **WHY?**
  - IP addresses are *logical* addresses (not physical)
  - 32 bits.
  - Includes a network ID and a host ID.
  - When an organization applies for IP address, they get a network ID.

---

**The four formats of IP Addresses**

- 32 bits long: 129.252.138.8

**Class**

- **A**
  - 128 possible network IDs
  - Over 4 million host IDs per network ID

- **B**
  - 16K possible network IDs
  - 64K host IDs per network ID

- **C**
  - Over 2 million possible network IDs
  - About 256 host IDs per network ID

- **D**
  - Multicast Address
Host and Network Addresses
- A single network interface is assigned a single IP address called the host address.
- A host may have multiple interfaces, and therefore multiple host addresses.
- Hosts that share a network all have the same IP network address (the network ID).
- Display all network interface on a host
  - `ifconfig -a`
  - `netstat -i`

IP Routing
- Forwarding:
  - When each packet arrives, looking up the outgoing line to use for it in the routing table
  - Done on a hop-by-hop basis
  - If destination is directly connected or on a shared network, send IP datagram directly to destination
  - Otherwise send datagram to a default router

- Routing updates
  - filling in and updating the routing tables

Mapping IP Addresses to/from Hardware Addresses
- Address Resolution Protocol
  - How?
  - Why?
  - When?
- Reverse Address Resolution
  - How?
  - Why?
  - When?

IP Addresses
- Subnet Addressing
  - To make better use of class A and class B addresses, divide host ID into subnet ID and host ID

  ```
<table>
<thead>
<tr>
<th>Subnet</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>255.255.255.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0xFFFF FF 00</td>
<td></td>
</tr>
</tbody>
</table>
  ``

Example: A and B are class B addresses, using the same subnet mask.
- A = 165.230.82.52
- B = 165.230.24.93
- M = 255.255.255.0
- Same network?
- Same subnet?
Subnetting

- It is possible to have a single wire network with multiple subnets?

Variable length subnetting

- Subnet masks allow power of 2 subnets
- Use a hierarchy of routers to allow subnets to be divided with different subnet masks
- Another approach:
  - Variable length subnet masks
  - Allow a subnet to be defined by more than two masks
  - The router applies the masks one after another

```
mask: 255.255.255.128
subnet 1
mask: 255.255.255.192
subnet 2,3
mask: 255.255.255.128
subnet: 221.2.3.0
mask: 255.255.255.192
subnet: 221.2.3.128
subnet: 221.2.3.192
```

Question

- If an ISP has a 203.6.8.0 Network, he has 5 customers who in turn has a network of 60, 60, 60, 30, 30 hosts. If the ISP wants to assign a subnet to each customer
  - What should the subnet mask be?
  - What is the address range in each subnet?
  - What should the routing table entries be?

CIDR - (classless Inter domain routing)

- Original addressing schemes (class-based):
  - 32 bits divided into 2 parts:
    - Class A
    - Class B
    - Class C
  - Class C address has max of 254 hosts
  - Not enough for many organizations
  - Too many class C addresses
    - huge routing tables
    - Use CIDR address mask to aggregate
- CIDR introduced to solve 2 problems:
  - exhaustion of IP address space
  - size and growth rate of routing table

```
CIDR Block Prefix # Equivalent Class C # of Host Addresses
/27 1/8th of a Class C 32 hosts
/26 1/4th of a Class C 64 hosts
/25 1/2 of a Class C 128 hosts
/24 1 Class C 256 hosts
/23 2 Class C 512 hosts
...
/15 512 Class C 131,072 hosts
/14 1,024 Class C 262,144 hosts
/13 2,048 Class C 524,288 hosts
```

Address Arithmetic: Address Blocks

- Address format <IP address/prefix P>.
  - The prefix denotes the upper P bits of the IP address.
  - Can be used to specify arbitrary blocks of addresses
- The <address/prefix> pair defines an address block:
  - Examples:
    - 200.15.0.0/16 => [ 200.15.0.0 - 200.15.255.255 ]
    - 192.24.0.0/13 => [ 192.24.0.0 - 192.31.255.255 ]

```
192.00011000.0.0
13th bits fixed Variable
```
CIDR: Classless Inter-Domain Routing

- Q: Say an ISP has 192.5.48.0, 192.5.49.0, 192.5.50.0, 192.51.0, what should the IP address advertised be?
  A: 192.5.48.0/22

- Q: Say an ISP has 200.8.4/24 address, how many addresses are included?
  A: 256 addresses

- Q: If a customer needs only 4 addresses from 200.8.4.24, then what address should be specified?
  A: 200.8.4.24/30

Other Developments: NAT

- NAT - Network address translation
- Why NAT?
- How NAT work?

  - Hosts need not have unique global IP address
  - Hosts are assigned private addresses
  - 10.0, 172.16 to 172.31 and 192.168. Are allocated for private hosts (Hmmm, what if those addresses appear on the Internet themselves?)

  - Packets from private hosts are replaced with source address of NAT gateway, use port# to uniquely do the reverse translation.

UDP vs. TCP

- What is the difference between UDP & TCP?
  - Connection oriented VS. Connectionless
  - many others......

- Q: Which protocol is better?
  A: It depends on the application.

  TCP provides a connection-oriented, reliable, byte stream service (lots of overhead).

  UDP offers minimal datagram delivery service (as little overhead as possible).
TCP
- TCP provides the end-to-end reliable connection that IP alone cannot support.
- The TCP protocol
  - Frame format
  - Connection Creation
  - Flow control
  - Congestion control
  - Connection termination

Addressing in TCP/IP
- Each TCP/IP address includes:
  - Internet Address
  - Protocol (UDP or TCP)
  - Port Number

NOTE: TCP/IP is a protocol suite that includes IP, TCP and UDP.

Hmmmmm. TCP or UDP?
- Electronic commerce?
- Video server?
- File transfer?
- Email?
- Chat groups?
- Robotic surgery controlled remotely over a network?

TCP Segment Format
- There are a bunch of control flags:
  - URG: urgent data included.
  - ACK: this segment is (among other things) an acknowledgement.
  - RST: error - abort the session.
  - SYN: Used to establish connection; synchronize Sequence Numbers (setup)
  - FIN: polite connection termination.

TCP Connection Establishment
- Three-way handshake
  - Client
  - Server

01 5 1 6 3 1
20 bytes

TCP segment

TCP Connection Establishment
- Three-way handshake
  - Client
  - Server

SYN ISN=X
1

SYN ISN=Y ACK=X+1
2

ACK=Y+1
3

"I want to talk, and I'm starting with byte number X+1."

"OK, I'm here and I'll talk. My first byte will be called number Y+1, and I know your first byte will be number X+1."

"Got it - you start at byte number Y+1."
Why 3-Way?
- Why is the third message necessary?

**HINTS:**
- TCP is a reliable service.
- IP delivers each TCP segment.
- IP is not reliable.

TCP Flow Control

TCP Termination

Test Questions
- Why is a 3-way handshake necessary?
- Who sends the first FIN - the server or the client?
- Once the connection is established, what is the difference between the operation of the server’s TCP layer and the client’s TCP layer?
- What happens if a bad guy can guess ISNs?

Socket?
- What is Socket?
  - Network API, developed by Berkeley
- Between which two layers do the socket sit?
- What is the wish list of a socket?
- What functions should the socket provide?
- What are the elements of a Socket?
- What is a socket descriptor?
- What are the two typical socket types?
  - Stream sockets
  - Datagram sockets
Network API

- OSI model
  - Application
  - Presentation
  - Session
  - Transport
  - Network
  - Data link
  - Physical

- Internet protocol suite
  - Application
  - Transport
  - Network
  - Data link
  - Physical

Socket Descriptor Data Structure

- Descriptor Table

<table>
<thead>
<tr>
<th>Index</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- Family: PF_INET
- Service: SOCK_STREAM
- Local IP: 111.22.3.4
- Remote IP: 123.45.6.78
- Local Port: 2249
- Remote Port: 3726

Socket system calls

- General Use
  - read()
  - write()
  - close()

- Connection-oriented (TCP)
  - socket()
  - connect()
  - listen()
  - accept()

- Connectionless (UDP)
  - send()
  - recv()
  - sendto()
  - recvfrom()

You should know...

- What does each system call do?
- Can each system call be used in TCP/UDP socket?
- I do not expect you to remember the sequence of each parameter, but you should know:
  - How to set each parameter?
  - What is the typical return value?
- Can you understand and explain:
  - myaddr.sin_addr.s_addr = htonl(INADDR_ANY);
- Practical issues
  - How is endpoint address specified?
  - and...

Network Byte Order

- What is network byte order?
- Why do we need Network Byte Order?
- When and how should we use the network byte functions?
- What are network byte order functions:
  - 'h': host byte order
  - 'n': network byte order
  - 's': short (16bit)
  - 'l': long (32bit)

  ```c
  uint16_t htons(uint16_t);
  uint16_t ntohs(uint_16_t);
  uint32_t htonl(uint32_t);
  uint32_t ntohl(uint32_t);
  ```

TCP Sockets Programming

- How to create a TCP socket
  ```c
  sock = socket(PF_INET, SOCK_STREAM, 0);
  ```
- What the typical work flow for a server?
  - How to establish an passive mode TCP socket?
  - In which function is the following procedure implemented?
    - Tell the kernel to accept incoming connection requests directed at the socket address. 3-way handshake
    - Tell the kernel to queue incoming connections for us.
    - How to send/receive data.
    - How to terminate a connection.
      - close()
      - reading EOF
- What the typical work flow for a client?
  - Where is 3-way handshake done?
Client-Server Communication (TCP)

TCP Client
- `socket()`
- `bind()`
- `connect()`
- `read()`
- `write()`
- `close()`

TCP Server
- `socket()`
- `bind()`
- `listen()`
- `accept()`
- process request
- `read()`
- `write()`
- `close()`

TCP Client connects to a well-known port of the server, which blocks until a connection from the client is established. After connection establishment, data is transmitted in both directions and finally, the connection is closed.

Client-Server Communication (UDP)

UDP Client
- `socket()`
- `bind()`
- `sendto()`
- `recvfrom()`
- `close()`

UDP Server
- `socket()`
- `bind()`
- `recvfrom()`
- `sendto()`
- process request
- `close()`

UDP sockets are used for datagram communication, where each message is sent as a separate datagram, and the receiver determines the source of the data. Connections are not established, and messages are sent directly without establishing a connection first.

UDP Sockets Programming

- How to create UDP sockets?
  ```c
  sock = socket(PF_INET, SOCK_DGRAM, 0);
  ```
- What is the typical workflow for:
  - Client
  - Server
  - How to send data? `sendto()`
  - How to receive data? `recvfrom()`
    - `recvfrom()` blocks until a datagram is received from the client.
    - `recvfrom()` can be interrupted with `alarm()`.
- Connected mode?
  - Why?
  - How?
  - Who can?

Signal handling:
```c
static void sig_alrm(int signo)
{
    return;
}
```

Timeout for `recvfrom()`:
- `alarm()` to set a time limit for receiving a datagram.
- `alarm kısa();` to check if there was an error or if the timeout was reached.
- If `recvfrom()` times out, `errno` is set to `EINTR`.

There are some other (better) ways to do this - check out section 14.2.
**Connected mode**
- A UDP socket can be used in a call to.
- What will OS do after `connect()` is called?
  - Register the address of the peer in OS
  - No handshake
  - No data is sent
- Once a UDP socket is connected:
  - can use `write()` and `send()`
  - can use `read()` and `recv()`
    - only datagrams from the peer will be returned.

**Questions**
- Can UDP socket connected to a broadcast address?
  - A: yes, a connected UDP socket exchanges datagrams with only one IP address
- Server A is connected to a broadcast address, so….
  - Can this UDP socket send?
  - Can this UDP socket receive?

**I/O Multiplexing**
- Why do we need I/O Multiplexing?
  - need to be able to monitor multiple descriptors
- What are the options to achieve I/O Multiplexing?
  - What are the cons and pros for each option?
    - Use nonblocking I/O.
      - use `fcntl()` to set O_NONBLOCK
    - Use alarm and signal handler to interrupt slow system calls.
    - Use multiple processes/threads.
    - Use functions that support checking of multiple input sources at the same time.

**Using select()**
- What does `select()` do?
  - system call allows us to use blocking I/O on a set of descriptors (file, socket, …).
- How to use `select()`
  - Create `fd_set`
  - Clear the whole thing with `FD_ZERO`
  - Add each descriptor you want to watch using `FD_SET`.
  - Call `select` when `select` returns, use `FD_ISSET` to see if I/O is possible on each descriptor.
Advanced programming

- JAVA RMI
- Daemons
  - daemon initiation
  - system message output mechanism
  - inetd
- Multicast socket programming

Application list

- TELNET
- RLOGIN
- FTP
- TFTP
- HTTP
- DNS
- SMTP, POP3

Q: What are their usage?
  - Remote login? File transfer?
Q: which transportation protocol used?
  - TCP, UDP?
Q: How is data being transferred between a server and a client?
Q: What are the challenges/design issues for each application? How have those issues been solved?

TELNET Client and Server

TELNET -- Sorcerer’s Apprentice Syndrome

send DATA[n]  receive DATA[n]  send ACK[n]
(time out)   send DATA[n]  receive ACK[n]
retransmit DATA[n]   receive DATA[n] (dup)  send ACK[n] (dup)
receive ACK[n] (dup)  send DATA[n+1]  receive DATA[n+1] (dup)
send DATA[n+1] (dup)  receive ACK[n+1]  send ACK[n+1] (dup)

...  receive DATA[n+1] (dup)  send ACK[n+1] (dup)

The Fix

- Sender should not resend a data packet in response to a duplicate ACK.
- If sender receives ACK[n] - don’t send DATA[n+1] if the ACK was a duplicate.
Sample question 1

- Show the execution of Dijkstra's algorithm on the following graph, with vertex A as the source vertex. At each step, show the value of the chosen vertex (w) and the updates to the distance and parent vectors.

![Graph Image]

Sample question 2

- Java port scanner. Write a port scan program. In particular, you are asked to write a program that will find out (print out) which of the first 1024 ports seem to be hosting TCP servers on a host "broad.cse.sc.edu"

Sample question 3

- Please explain the packet captured in wireshark.

![Wireshark Packet Image]