Applications

- Goal: Look at typical network applications so that we can appreciate why we go through the complexities of building networks (Internet)
- Client-server applications:
  - Servers “wait” at well known locations (ports)
  - Clients “connect” to servers. Clients “know” these well known locations
  - Some applications are both servers and clients
  - E.g. Web, Email, ftp,
- Streaming multimedia applications
  - Windows media, Real, Quicktime ..
- Low latency applications
  - E.g. VOIP, Games, chat

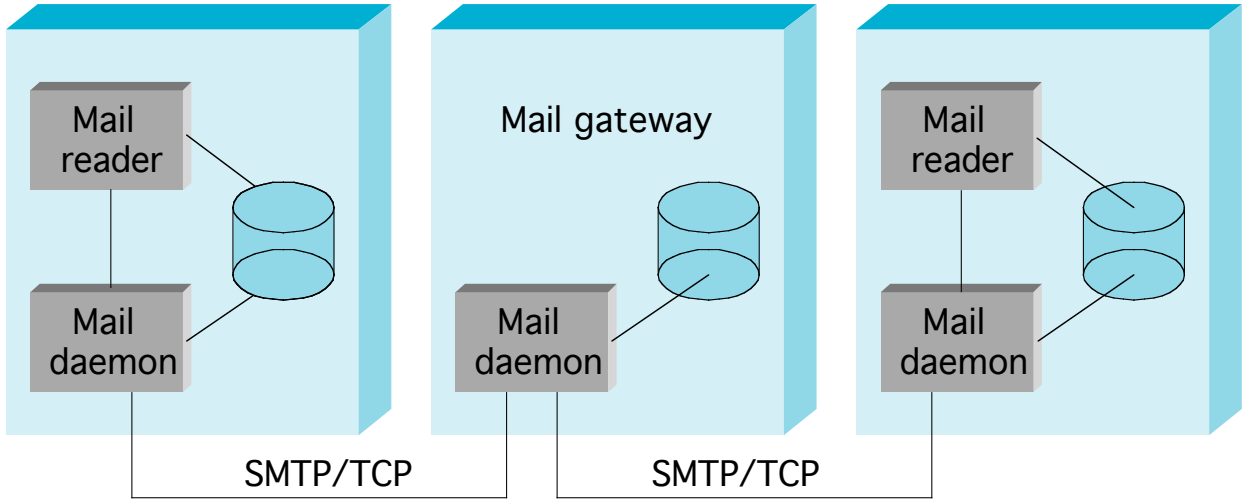
Client-Server applications

- The server end “waits” for clients to connect to
  - In socket() interface, we use the accept() call
- Once the client connects to the server, the server and client talk an application specific protocol
  - E.g. SMTP, HTTP, IMAP etc. (more later)
- Client end “talks” to the waiting server
  - In socket() interface, we will use the connect() call
- Client needs to know where the server is waiting
  - Use well known ports and domain name service (DNS) to help with the location problem
  - For example, SMTP waits in port 25

Email using SMTP/IMAP/POP

- Email consists of two components
  - Simple Mail Transfer Protocol (SMTP) for email clients to send out emails (e.g. smtp.nd.edu, port 25)
  - Internet Message Access Protocol (IMAP) for email clients to receive your emails (e.g. imap.nd.edu, port 143)
  - You can use telnet to "talk" to these servers directly
  - E.g. type 'telnet smtp.nd.edu 25' and then type 'help'
  - For SMTP, you client (say Outlook), connects to smtp.nd.edu and then delivers an email destined for friend@aol.com. Smtp.nd.edu then locates the SMTP server responsible for AOL. These servers may delegate to other SMTP servers. Eventually it reaches friend@aol.com
  - Friend will use IMAP to retrieve this email

Email


Mail reader
Mail daemon
Mail gateway
Mail daemon
Mail reader
Mail daemon
SMTP/TCP
SMTP/TCP

Web server

- Web servers wait on port 80
  - Try ‘telnet www.nd.edu 80’ and then type ‘GET / HTTP/1.0’
- We can also use web proxy servers that forward your request
  - E.g. ‘telnet sys.cse.nd.edu 3128’ and then type ‘GET http://www.yahoo.com/ HTTP/1.0’

File transfer protocol (ftp)

- ftp can act both as a server and client (active mode)
- Client connects to ftp server to send control
- Data (files) are actually transferred in one of two ways:
  - Active mode: Client acts as the ‘server’ and asks the ftp server to connect to its own port and transfer file
  - Passive mode: Client connects again to the ‘server’ data port nnd transfers the data

Nature of client server applications

- Once the connect is “established”, both ends can read() and write() data. Whatever is “written” in one end appears to be “read” on the other end. (actual code example ‘morrow)
- In this class, we do not focus on the application protocol used over this “line” (SMTP, IMAP …). We will develop technologies that make it look like a line, even though the wire may go through my cell phone, might connect to servers half way across the globe
- Note that we make no guarantees on “when” the data actually appears (and in some cases, if at all)

The “line” network protocol

- We will describe Transmission Control Protocol (TCP)
  - TCP provides reliable, in order, at most once semantics
- TCP is built on top of Internet Protocol (IP)
  - IP provides best effort service. “packets” can be delivered out of order, more than once or delivered at all
- IP has such low requirements that any network technology can implement it

Multimedia - Isochronous traffic

- Multimedia traffic requires a new constant and predictable data delivery
- Lets take video. Broadcast TV uses 29.97 frames/second. If you send each frame in a packet, then you have to receive a frame every 1/30th of a second. If not, you will notice lower quality.
  - Also, it is okay to lose frames than to wait for them to be sent slowly
  - We don't need reliable delivery, you either want the next frame in 1/30 sec or skip to the next frame

Multimedia demo for 24 fps and 8 fps

Network design

- Different applications require different stuff from the network
  - TCP provides reliable deliver service
  - Multimedia requires delay guarantees
  - Underlying networks can be unreliable, drop data, duplicate data, out of order data.

Network programming in C

- Client and Server end of a network connection
  - Server end waits for connection requests
  - Client end connects to server end
  - Network server can infact be a client to other services
  - Each network connection end point is identified by a IP and port number

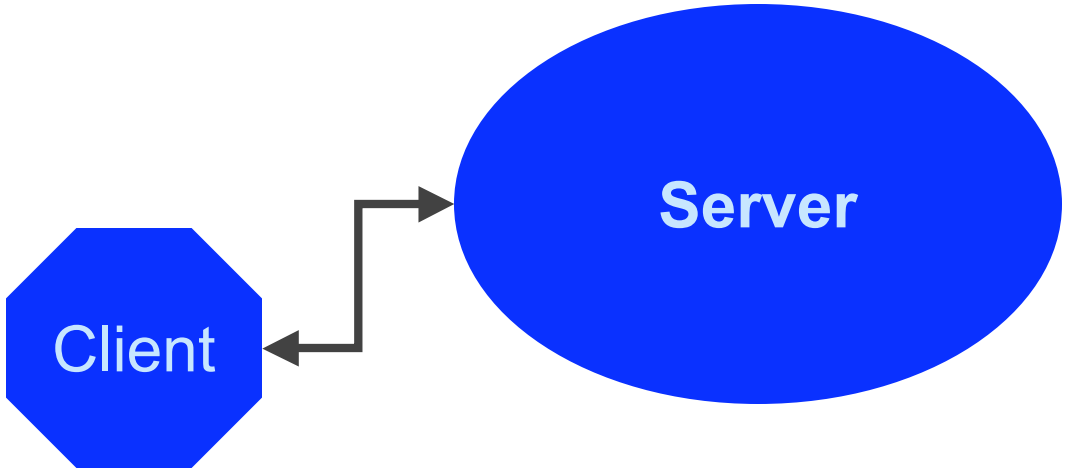
Sockets

- Communications mechanism
- Behaves like a pipe – data sent on one end is received on the other end

Sender (client end) Receiver (server end)

- On a server, you can bind the socket to a port so that it listens for connection requests on that port
- On the client, you can connect to a server socket

Central server based system



- Simple central server based approach
  - Server bind and waits on a well known port for requests
  - Clients connect to server using well known port

Connectionless

soc = socket(**`AF_INET, SOCK_STREAM, IP`**)

sendto(soc, messageBuffer, messageLen, flags,
destinationSockaddr, len)

recvfrom (soc, messageBuffer, messageLen, flags,
sourceAddr, len)

Client

```
soc = socket(AF_INET, SOCK_STREAM, IP);
bzero((void *) &sAddr, sizeof(sAddr));
sAddr.sin_family = AF_INET;
sAddr.sin_addr = SERVER ADDRESS;
sAddr.sin_port = SERVER_PORT;
connect(soc, &sAddr, sizeof(sAddr))
←-----------END--------------→
write(soc, .. , .. )
read(soc, .. , .. )
close(soc)
```

Server

```
soc = socket(AF_INET, SOCK_STREAM, IP);
bzero((void *) &sAddr, sizeof(sAddr));
sAddr.sin_family = AF_INET;
sAddr.sin_addr = INADDR_ANY;
sAddr.sin_port = SERVER_PORT;
bind(soc, &sAddr, sizeof(sAddr))
socNew = accept(soc, .. , ..)
←----------END----------→
write(socNew, .. , .. )
read(socNew, .. , .. )
close(socNew)
```

Useful tools

- Tcpdump
  - Dumps network packets
- Netstat
  - Shows active connections
- Ping and traceroute
  - Verifies that “packets” can get to a machine
- Host/dig/nslookup
  - Hostname->IP mapping