

TCP, UDP, and Sockets

INFO 341

1

Today's Topics

- TCP
- UDP
- UDP Sockets (TCP sockets last lecture)
- Review for the Mid-term exam

2

TCP

- Transmission Control Protocol
- RFC 793
- Provides
 - multiplexing (ports)
 - full duplex connection (handshakes)
 - reliable connection (buffers)
 - performance (window size, slow start)
- Socket API (Winsock, Berkeley)
- /etc/services documents port numbers

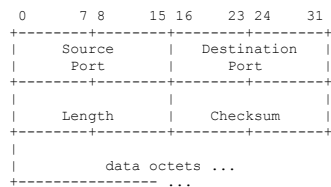
3

[UDP State Diagram]

- There is no state in UDP
- Client just sends data to the server
- Server system buffers arriving packets
- Server has to keep up with packets
- The client only knows if the server UDP port exists, otherwise an ICMP Destination Unreachable error is sensed.

10

[UDP Header RFC 798]



11

[Why sockets?]

- How was network programming done before?
 - Needed to have knowledge of the specific hosts network interface (NIC)
 - Needed to know hardware specific packet structure
 - The same program for different hosts were really totally different (because of the difference in underlying network hardware)

12

[Why sockets?]

- How was network programming done before?
 - Messy
- Sockets made it so that an “average” programmer could write network applications
- They were on every BSD Unix machine

13

[Compiling Lab 4 Example]

- Login to virgil
- Create directory (mkdir lab4)
- Change directory (cd lab4)
- Copy files over (ftp, or cut & paste)
- Compile
 - gcc -c cnailib.c
 - gcc lab4server.c cnailib.o -o lab4server
 - gcc lab4client.c cnailib.o -o lab4client

14

[Generic UDP client]

```
sock = socket(protocol_family,type,protocol);
while( interacting_with_server ) {
    sendto(sock,buffer,bufLen,flags,&server_address,server_addrlen);
    recvfrom(sock,buffer,bufLen,flags,&server_address,&server_addrlen);
}
close(sock)
```

15

Generic UDP server

```
server_sock = socket(protocol_family,type,protocol);
bind(server_sock,local_address,localaddr_len);
while( still_serving ) {
    recvfrom(sock,buffer,bufllen,flags,&client_address,&client_addrllen);
    sendto(sock,buffer,bufllen,flags,&client_address,client_addrllen);
}
close(server_sock)
```

16

UDP Client (almost)

```
struct sockaddr_in server_addr, client_addr;
struct hostent *hp;
int client_addrllen;
int sock;
int flags = MSG_WAITALL;
int port = 4567;
char data[128];
char buffer[128];

sock = socket(PF_INET, SOCK_DGRAM, 0);
hp = gethostbyname("www.ischool.washington.edu");
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
server_addr.sin_addr.s_addr = htonl(*(long*)hp->h_addr);
while( TRUE ) {
    // put some stuff in array data
    sendto(sock,data,strlen(data),0,&server_addr,sizeof(struct sockaddr_in));
    // wait for server to send response
    recvfrom(sock,buffer,sizeof(buffer),flags,&client_addr,&client_addrllen);
}
close(sock);
```

17

UDP Server (almost)

```
struct sockaddr_in server_addr, client_addr;
int client_addrllen;
int sock;
int flags = MSG_WAITALL;
int port = 4567;
char data[128];
char buffer[128];

sock = socket(PF_INET, SOCK_DGRAM, 0);
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
bind(sock,server_addr,sizeof(struct sockaddr_in));
while( still_serving ) {
    recvfrom(sock,buffer,sizeof(buffer),flags,&client_addr,&client_addrllen);
    // formulate response
    sendto(sock,data,strlen(data),0,&client_addr,sizeof(struct sockaddr_in));
}
close(sock)
```

18

[UDP Help - Lab 4]

Steps to Success ...

1. Make sure you understand the code for the TCP version & the library
2. Don't forget, 'man' can help you understand the functions `sendto()` `recvfrom()`
3. Modify incrementally, base your modifications on subroutines that you understand
4. Output can help you debug
