CS268 Homework Assignement #2: UNIX Networking Programming

April 5, 1996

Background

In this assigment you will familiarize yourself with the basics of socket-style network programming. In particular, you will be required to write socket code for multicasting and TCP connections. A great reference for this assignment, apart from the manual pages, is:

W. Richard Stevens, *Unix Network Programming*, 1990, Prentice Hall, ISBN 0-87692-749-5.

To get online infomation:

```
% man socket
```

will get you going in the right direction.

The Assignment

Somewhere on campus, there is a host which every 10 seconds sends a UDP packet to the multicast address 224.4.136.133 on port 14482. The packet will contain 8 bytes of data, organized in the following C structure:

```
struct keypkt {
        unsigned int key;
        unsigned short port;
        unsigned char pad[2];
};
```

All integral fields in this and subsequent structures are in network byte order (big endian). The `key` field of the packet is a 32 bit key which is valid for the next 10 seconds until the next key packet is sent. Within these 10 seconds, you must establish a TCP connection with the sending machine on the port designated by the `port` field in the multicast packet (the 10 second interval should not impose any problem since your code will do this orders of magnitude faster). In client-server terminology, the sending host is acting as a server listening on the given port and you must establish a client connection with it.

Once you've established the TCP connection, you should send a message consisting of the following structure:

```
struct request_pkt {
        unsigned int key;
        unsigned short port;
        char names[254];
};
```

The `key` field should contain the key that you received in the multicast UDP packet above. The `port` field should contain the port number on which your process will be listening so that the remote machine can connect to it with its reply. The `names` field should contain a null-terminated string of no more than 254 characters containing the names of the people doing the assignment. The names should be organized first name followed by last name with commas between names.

Finally, the remote machine will attempt to set up a client TCP connection with your machine on the port that was sent in the request packet. Once the connection is established, the remote machine will send you a message of the following form:

```
struct reply_pkt {
        unsigned int msg;
        unsigned int key;
};
```

If the `msg` field is 1, the key that you sent in the request packet did not match the one that was sent by the remote host. If the `msg` field is 0, then you've succeeded. You should record the value in the `key` field and hand that in as your "answer" to your assignment. You should print out the key field in hex.

The remote host will log the names from the `names` field of the request packet along with the key sent back to your host (which you are to hand in). In this way credit will be established.

Potential Problems

This section lists potential sources of problems in order to make your debugging life easier:

1. Ports should be in the range of 5000–65535. Ports below 5000 are often used by other software. Just to be sure keep your port numbers above 10000.
2. If you have access to the network monitoring tool *tcpdump*, use it. It will make your life easier.
3. You must do this assignment on campus. The multicast packets are configured to not leave the campus network.
4. **Beware of network order to host order conflicts!**. TCP/IP networks use big endian ordering, so little endian machines will wreak havoc on your network packets if your don't covert them to network order. Intel and Digital computers are examples of little endian host. Fortunately, there exist functions for converting longs and shorts between host and network order and vice versa: read the manpages on *htonl*, *htons*, *ntohl*, and *ntohs* to get details.
5. If you use a 64-bit architecture (e.g. DEC Alpha), be careful about the word sizes. On the Alpha, for example, a `long` is 64 bits, an `int` is 32 bits. On the Cray, most everything except characters are 64 bits.

Due Date

The remote host will continuously multicast packets from 3pm Wednesday April 10th until 3pm Wednesday April 17th. You should hand in your "key" in class on Wednesday April 17th.