

cmsc 417 programming assignment one

January 31, 2008

Your task in this assignment will be to write a single program that first acts as a multicast sender and then a multicast listener.

This assignment is to be done individually.

1 Context

Over the rest of the semester, we will build a virtual network using a combination of multicast frames (to emulate multi-access networks) and unicast tunnels to emulate a backbone network.

Building this application blends networked application development with simulations of network protocols, so you should be able to build both network protocols and network applications at the end of the semester.

2 Deadline

DUE: Feb 15. Take note! This is only approximately 120 lines of C code.

3 Requirements

The program should do two things in exactly the order below.

- First, send a message to a multicast address, specified below.
- Next, print all messages received at the multicast address until the process is interrupted (control-c).

4 Message format

Your message header should consist of the following:

```
uint8_t version;           /* set to 1; if you receive anything else, discard. */
uint8_t ttl;               /* set to 1; if you receive anything else, it's fine. */
uint16_t payload_length;  /* bytes following the header, not including trailing null of a c string */
uint32_t account_identifier; /* digits of your account name: unsigned int i; sscanf("cs417xxx", "cs417%u", &i); */
uint32_t source_address;  /* unused for now, set to 0 and ignore. */
uint32_t destination_address; /* unused for now, set to 0 and ignore. */
uint16_t checksum        /* unused for now, set to 0 and ignore. */
uint16_t protocol        /* must be 1; if you receive anything else, discard. */
```

- The types used above can be found in:

```
#include <inttypes.h>
```

- All values must be in network byte order.
- The header should be followed by a string, provided on the command line as:

```
./one "The message"
```

That is, the string, **if** given, will be in `argv[1]`. If `argc` is less than 2, either send a default message, or send no message at all.

- Your executable must be named “one” (for the testing scripts to operate correctly).
- The multicast address to be used is 224.0.50.112, port 41710.
- Do not send the null-terminating byte, do not expect to see the null terminating byte.

We will test whether your code sends a valid message with the given string once and only once, whether your code prints received messages to stdout, optionally with some heading, whether the system calls made by your code appear correct, and anything else we think of in the next few days.

5 Hints

- Functions required include `socket`, `bind`, `setsockopt` (with `IP_ADD_MEMBERSHIP` and `SO_REUSEADDR` options), `sendto`, and `recvfrom`.
- To print a string with a length, use `printf("%.*s", length, string)`
- man pages, books, wikipedia, and google are your friends. But please credit your sources.

6 Warnings

- Do not increase the TTL of multicast messages from the default of one; that is, do not use `IP_MULTICAST_TTL`.
- If your call to `bind()` fails, it is because you or another student has not properly set `SO_REUSEADDR`; temporarily use a different port.
- Ensure that you can run more than one instance (process) of your program at a time by using `SO_REUSEADDR`.
- The maximum size of an IP packet is 65,535 bytes.
- Your code may appear to work when it shouldn't, if another working implementation has invoked `IP_ADD_MEMBERSHIP`.

7 Objectives

At the end of this assignment, you should understand:

- Basic socket function calls for connectionless (datagram) service: `socket`, `bind`, `sendto`, `recvfrom`.
- Messages as C datatypes.
- Host and network byte ordering, `htons`, `htonl`, `ntohs`, `ntohl`.
- Sockets, `sockaddr`, `inet_addr`, `inet_ntop`,