Program Assignment 3

CMSC 417 Spring 2012

February 29

1 Deadline

March 14, 2012.

2 Introduction

In this project, you will write a simple multicast program. Your task is to write a single program that first acts as a multicast sender and then as a multicast listener. Some of the next projects will build upon this one. Don't start late!

3 Team Project

You should build a team of 3 from this project.

Once you've built a project team, please send the TA an email with the names of team members.

4 Requirements

1. Implementation

   - The multicast address to be used is **224.0.50.112, port 5050.**
   - Your source code file must be named **"one.c"** for the testing scripts to operate correctly.
   - The program should do two things in exactly the order below.
     (a) **Create two sockets for multicast sending and multicast receiving.**

(b) **Send a message to the multicast address specified above.**

(c) **Print all messages received at the multicast address to stdout until the process is interrupted (control-c). You should be able to receive the multicast message that you've sent.**

- You should use funtion **setsockopt() with SO_REUSEADDR** so that port 5050 can be used by other students at the same time.
- Your program will send a message provided as a command line argument.
  Example: ./one "my message"
- Do not send the null-terminating byte and do not expect to receive the null terminating byte.

2. Program Operation

   - You should send a message with valid header to the specified multicast address exactly once.
   - When you receive a message with **valid header**, print out its **payload**.
   - If you receive a message with **invalid header, DO NOT** print out its payload. You may print an error message. But you won't be penalized if you don't.

5 Message format

A message consists of "HEADER" followed by "PAYLOAD".

- Your message header should consist of the following:

```
uint8_t version;          /* must be 1. If you receive anything else, discard*/
uint8_t ttl;              /* must be 1. If you receive anything else, discard*/
uint16_t payload_length;      /* bytes following the header */
uint32_t account_identifier;  /* digits of your account name */
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*/
```

- All values must be in network byte order.
- The header should be followed by a payload, text string, which will be provided as a command line argument. If no argument is provided, send a default hello message

  Example: ./one "my message"

6 Hints

- Functions required include: socket, bind, setsockopt (with IP_ADD_MEMBERSHIP and SO_REUSEADDR options), sendto and recvfrom.
- You may use any references you choose (e.g., man pages, books, Wikipedia, and Google).

  One of the references is here:

  `http://tack.ch/multicast/#lan`
- You may receive messages from other students, if you are lucky.
- The TA will answer general questions/confusions, and is not supposed to debug for you. Please try your best to debug your programs yourself!

7 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

8 Submission

- Please submit your code to the Submit Server (https://submit.cs.umd.edu/).
- You should upload a zip file which contains the file **one.c**