

# cmsc 417 programming assignment two

February 13, 2006

Your task in this assignment will be to build and maintain a neighbor table from multicast “hello” messages.

## 1 Context

A “hello” message tells the receiver “I’m alive.” A recipient uses this hello message to construct *soft state*: a cached view of reality that can be reconstructed should the machine fail or reboot. (Contrast “hard” state written to disk.) This soft state will consist of a list or table of immediate neighbors and how to reach them.

## 2 Deadline

DUE: February 27. That’s a monday. The late deadline will be Friday, March 3rd; you have only one late deadline to play with.

## 3 Requirements

The program should do the following.

- Use as the source address on each packet the address: ( getpid() << 16 + last two digits of id << 8 + your favorite 8-bit number. )
- Send a message every 25 seconds to the multicast address. It can be a protocol one message with the payload “hello” if you like. It doesn’t matter, so long as the packet is valid.
- Maintain a list of known and valid neighbors.
- Expire neighbors after 2 minutes of inactivity.
- Upon the command “print neighbor table” on stdin from the command line, it should print, separated by commas and optional spaces, a table with the following fields in order:  
IP address, Port (integer), Network address (integer from the header), Account ID, Is Alive? (Y or N), Time Remaining (seconds)
- When stdin is closed, terminate.

NOTE: The command “print neighbor table” may be sent at any time. Your code must respond within one second.

NOTE: Ignore extra newlines.

NOTE: Ignore commands that are not “print neighbor table” unless you want to engineer your own interpreter.

NOTE: All printed values must be in host byte order.

NOTE: Your executable must be named “two” (for the testing scripts to operate correctly).

NOTE: Don’t even think about calling sleep().

## 4 Hints

Implement an event queue; at the moment, you have only one event (send the hello message each minute) and all other processing is input driven (receive a hello message or input from stdin). You may (hint, hint) have to implement more periodic events.

The select or poll system calls will be necessary for completing this assignment. While poll is more modern, select is more traditional. The file descriptor for stdin is 0.

A read() or fread() or fgets() or gets() on stdin will return differently if “end of file” has been reached. If so, exit.

## 5 Objectives

At the end of this assignment, you should understand:

- select.
- timers.
- Hello messages and the hello interval.