Prelab: 10
Demo: 40
Report: 46
Delay: 0

Total: 96

Real-Time Embedded Computing
Lab-5 Report

Haoxiang Zhang

Due date: November 4, 2019

46/50

Post Lab

Objectives and Lab Description

This project is about the communication between machines in a same network. There are one clients and several servers. At the beginning, the all of servers are not master. The Client program will ask all the servers "WHOIS" and the master server will reply that he is the master if he did is master. If there is no one said he is the master, the client program could ask "VOTE" to servers. The servers will get a random number between 0 to 9 and send it to broadcast. Then, compare each other to get the highest one as master. If they get the same number then the highest ip will be the master.

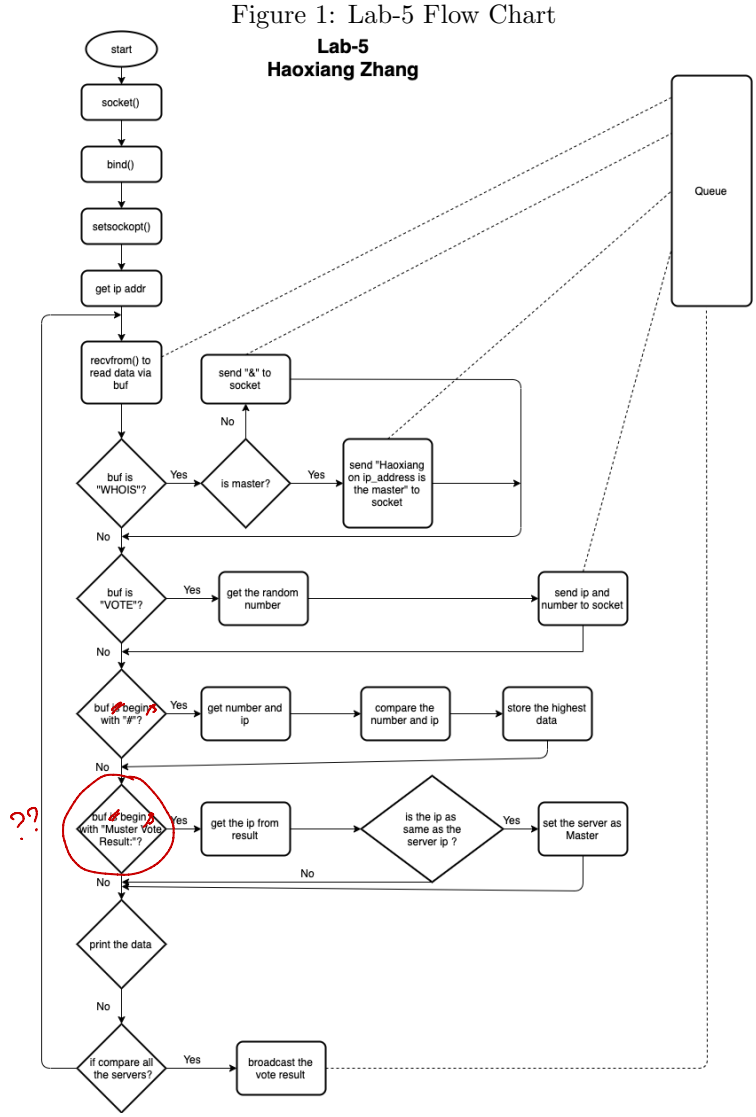
5/5

For the information sent to broadcast, when get the random number, send "# ip_address vote_number". And, if the server is the master, then send "Name on board ip_address is the master". All of the message, the size should be no more than 40 bytes.

Implementation

Flow Chart

Figure 1: Lab-5 Flow Chart



Sockets: Master/Slave Voting

At the beginning, I initialize the socket as UDP and set it as broadcast. Then, in order to get the local ip address, I declared a ifreq struct and copy the information from the socket that I set up last step, and then convert sockaddr_in to string with inet_ntoa() function.

In the while loop, first, I tried to read data from queue and store the data in buf. Secondly, check the client will ask "WHOIS" or not. If it does, check the current server is master nor not. if it is master then broadcast the message like this: "& Haoxiang: ip_address is Master", otherwise, send "&" to everyone.

This wasn't needed.

Then check the buf whether get "VOTE" or not. If it does, server will get a random number and broadcast the message like this "# ip_address vote_number".

10/10

Because there probably ask "VOTE" above. For the next is check the message that read whether is begin with "#'. First, I read the ip address and the vote number from the message. then set the highest vote number as the master. If the vote number are same then the higher ip address is the master.

For the last step , check if compared all of the servers then broadcast the final result. If get the final result then set the server as Master.

Experiments and Results

Sockets: Master/Slave Voting

When I create several servers and run them. The program will display the local ip address. When I type any thing in client program, all of the servers will get the message from client. When I type WHOIS, because there is no master, all of the servers send the "&" to broadcast. And then send the VOTE, in the client program, it will show the all the servers and their vote number. In the server programs, they will show the highest one. In the client program, then, will show the vote result and the master server program will shows "You are the Master now!".

11/15

After vote a master, type WHOIS in client program, then will show's that the master ip is the master.

Should have put the screenshots.

-4

Discussion and Post-lab questions

1. I tried to let the program know when does it finish. I used a boolean variable comparing to check, but it is not a good way to check.

   - Finally, I found an another way to get it. First, When I ask WHOIS. I account how many message begin with "&". Include master and slaves. Then I set the value of amount to variable "remain". Once compare one server, remain minus one. Until the remain is equal 0, send the result to everyone.

15/15

2. I tried to read ip address step by step. to get the variable sa_data in struct sockaddr, and then convert string to long and convert long to string, but it does not work.

   - I tried use memcpy() to point the pointer of the ifr_addr in struct ifreq to socket directly. Then just convert the long to string will be okay.

Comments : 5/5

Coding Section 40/40

Main Function

Listing 1: Main thread to read GPS and Push Button and interpolate the Event position

```
/*
    Project: RT_Embedded_computing Lab-5
     Author: Haoxiang Zhang
  Description: Server(UDP)
*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <strings.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <sys/ioctl.h>
#include <net/if.h>
#include <inttypes.h>
#include <stdbool.h>
#include <time.h>

#define MSG_SIZE 40



bool isMaster = false;
int master_num = -1;  // master's vote number
char master_ip[14]; // master ip address
int people = 0;   // number of servers
int remain = -1;  // check how many server should be compare

void error(const char *msg){
  perror(msg);
  exit(0);
}

int main(int argc, char *argv[]){
```

```c
// check the argument, have to have port number
if (argc < 2) {
  printf("ERROR: no port find\n");
  exit(0);
}

srand(time(NULL));


// ========================   Declare Area ========================
//                  ===
int server_fd;     // socket       ===
int sa_in_len;     // length of struct sockaddr_in   ===
socklen_t fromlen;  // length of sockaddr_in (socklen_t)  ===
int msg_check;     // check the receive and send   ===
int option = 1;    // socket option      ===
struct ifreq ifr;     //      ===
struct sockaddr_in server;  //        ===
struct sockaddr_in from;  //        ===
char buf[MSG_SIZE]; // buffer       ===
char* ip_addr;     // ip address for this server   ===
//                  ===
// ========================   Declare Area ========================


// ============================   INITIALIZATION ============================

// setup socket
if ((server_fd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
error("Socket Failed");
}

// erase the data from memory
sa_in_len =  sizeof(server);  // size: 16
bzero(&server, sa_in_len);    // erase the size of memory

// setup server socket
// INADDR_ANY tells socket to listen on all available interfaces
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons(atoi(argv[1]));

// bind socket
if (bind(server_fd, (struct sockaddr*)&server, sizeof(server)) < 0) {
error("Bind Failed");
}

```

```
// set broadcast option
if (setsockopt(server_fd, SOL_SOCKET, SO_BROADCAST, &option, sizeof(option)) < 0) {
error("Socket option sets Failed");
}

// define ifreq
ifr.ifr_addr.sa_family = AF_INET;
strcpy(ifr.ifr_name, "enp0s25"); // this is the local network name

// get the address from sock
ioctl(server_fd, SIOCGIFADDR, &ifr);
memcpy(&server, &ifr.ifr_addr, sizeof(server)); // copy the memory to address
ip_addr = inet_ntoa(server.sin_addr);
printf("Your IP address: %s\n", ip_addr);
strcpy(master_ip, ip_addr);




// ========================= START WORKING =========================

// receiving and sending
fromlen = sizeof(struct sockaddr_in);
while(1){
  // erase buffer
  bzero(buf, MSG_SIZE);

  // receiving
  if (recvfrom(server_fd, buf, MSG_SIZE, 0, (struct sockaddr*)&from, &fromlen) < 0){ error("recvfrom"); }

  if (strncmp(buf, "&", 1) == 0){ people++; }


  // print receive
  if (strncmp(buf, "WHOIS", 5) == 0) {
  // ========================= WHOIS =========================
    // initial number of servers
    people = 0;
        printf("Someone: Asking who is master.....\n");
    // if is master, then tell broadcast
    // & symbol is for checking number of servers
    if (isMaster){
      printf("System : You are the Master!!!!\n");
      char msg[MSG_SIZE];
      sprintf(msg, "& Haoxiang:%s is Master\n", ip_addr);
      from.sin_addr.s_addr = inet_addr("128.206.19.255");
      if (sendto(server_fd, msg, sizeof(msg), 0, (struct sockaddr*)&from, fromlen) < 0) { error("sendto"); }
    } else {
      printf("System : You are not master...\n");
      from.sin_addr.s_addr = inet_addr("128.206.19.255");
```

```
        if (sendto(server_fd, "&", 1, 0, (struct sockaddr*)&from, fromlen) < 0) { error("sendto"); }
      }

    } else if (strncmp(buf, "VOTE", 4) == 0){
    // ========================= VOTE ==============================
      // initial
      isMaster = false;
      remain = people;
      master_num = -1;
      bzero(master_ip, 14);
      printf("Start Voting...\n");
      int vote_num = -1;
          char msg[MSG_SIZE];
      // get random number
      vote_num = rand() % 10;
      // # symbol is for finding vote numbers
      sprintf(msg, "# %s %d\n", ip_addr, vote_num);
      from.sin_addr.s_addr = inet_addr("128.206.19.255");
      if (sendto(server_fd, msg, 40, 0, (struct sockaddr*)&from, fromlen) < 0) { error("sendto"); }

    } else if(strncmp(buf, "#", 1) == 0){
    // ========================= compare vote number ==============================
          remain = remain -1;
      char delim[] = " ";
      char* ptr = strtok(buf, delim);
      char* arr[3];
      int i = 0;
      while (ptr != NULL){
        arr[i++] = ptr;
        ptr = strtok (NULL, delim);
      }
      // higher vote number win
      if(atoi(arr[2]) > master_num) {
        master_num = atoi(arr[2]);
        strcpy(master_ip, arr[1]);
      // if vote number are same, higher ip win
      } else if (atoi(arr[2]) == master_num) {
        if (strncmp(arr[1], master_ip, 14) > 0){
          master_num = atoi(arr[2]);
          strcpy(master_ip, arr[1]);
        }
      }


    } else if (strncmp(buf, "Master Vote Result:", 19) == 0) {
    // ========================= set the high number as master ==============================
      char delim[] = ":";
      char* ptr = strtok(buf, delim);
      char* a[2];
      int i = 0;
      while (ptr != NULL){
        a[i++] = ptr;
        ptr = strtok (NULL, delim);
```

```
      }
      // printf("%s\n", a[1]);

      // if this machine is master, then set as master
      if(strncmp(a[1], ip_addr, 14) == 0){
        isMaster = true;
        printf("You are the Master now!\n");
      }

    } else {
          printf("Received a data: %s\n", buf);
    }

    // if compare all of servers, then broadcast result
    if (remain == 0){
      remain = -1;
      printf("IP: %s\n", master_ip);
      printf("VA: %d\n", master_num);
      char msg[MSG_SIZE];
      sprintf(msg, "Master Vote Result:%s\n", master_ip);
      from.sin_addr.s_addr = inet_addr("128.206.19.255");
      if (sendto(server_fd, msg, 40, 0, (struct sockaddr*)&from,
    fromlen) < 0) { error("sendto"); }
    }
  }

  return 0;
}
// The IP address: 128.206.19.machine_num
// The destination address: 204.159.253.118




/*
==================================================================
                              Task note
==================================================================

One master computer and several slaves comupter
implement server on Raspberri Pi
a client will ask all the student( include master ) "WHOIS"
if no master, the clients can ask "VOTE" to vote a new master

at the begining no one is master

"WHOIS" - ask who is master
"VOTE"  - ask everyone to vote a new master

to vote:
  each client sent a broadcast "# ip_addresss vote_number"
  highest number win
  if number are same then check highest ip win
```

```
dynamically get ip
message size: 41
vote range [1,10]
the port should be an argument of the program

==================================================================
                    Understanding about the project
==================================================================

1. Server and clients should be run under same network.
2. In order to communicate with other device, should enter in a
    same port.
3. To set as TCP (Connection based), shoud set socket as
    SOCK_STREAM type
4. To set as UDP (Connectionless), should set socket as SOCK_DGRAM
    tyep

Socket:
  Scoket Structures:
    sockaddr_in
      struct sockaddr_in{
        short sin_family;
        unsigned short sin_port;
        struct in_addr sin_addr;
        char sin_zero[8];
      }
    in_addr
      struct in_addr{
        unsigned long s_addr; // address
      }
    sockaddr
      struct sockaddr{
        unsigned short sa_family;
        char sa_data[14];
      }

------------------------------------------------------------------

  Useful Functions:
    in_addr inet_addr(char addr)  // IPV4 format
      same type  with struct in_addr, convert string to long
    char* inet_ntoa(struct in_addr in)
      convert long to string

------------------------------------------------------------------

  Setting up a Socket
    int socket(int domain, int type, int protocol)
          AF_INET     TCP/UDP     default:0
      set up a socket's attributes
    int bind(int desc, struct sockaddr* addr, int addrlen)

------------------------------------------------------------------

  sendto() and recvfrom() is same as write() and read()
```

```
It is better to add a error() function to print error if there is a
    error

void error(char* msg){
  perror(msg);
  exit(1);
}

-------------------------------------------------------------------------
#include <strings.h>
bzero(&memory_addr, mem_size);
bzero() erase the mem_size data from memory start from memory_addr


=========================================================================
                          Ways to copy ip address
=========================================================================

  struct ifreq ifr;
  struct sockaddr_in sin;
1.
  memcpy(&sin, &ifr.ifr_addr, sizeof(sin));
  *inet_ntoa(sin.sin_addr);
2.
  struct sockaddr_in* ipaddr = (struct sockaddr_in*)&ifr.ifr_addr;
  *inet_ntoa(sin.sin_addr);
*/
```