Wi-Fi Communication

Mini Project

2170710 - Mobile Communication & Wireless Communication

Guided by:

Prof. Sunil A. Bakhru

Computer Engineering Department,

BVM Engineering College, Vallabh Vidyanagar

Group Members (E3/G4):

Rushi Patel (140070107040)

Siddharth Goswami (150073107002)

Rahul Mourya (150073107007)

Project Definition

Write a program that:

1. Identifies the Wi-Fi hot-spots in its wireless range.
2. Gives a counts of Wi-Fi hotspots.
3. Also performs file transfer between given two Wi-Fi devices –across different Wi-Fi hotspots.

Table of contents

Program 1 - *Objective*

Write a program to list all the Wi-Fi hotspots available and give total count of them using C programming.

Modules Covered :

1. Identifies the Wi-Fi hot-spots in its wireless range.
2. Gives a counts of Wi-Fi hotspots.

Program 1 - *File hierarchy*

Program

- `wifi_scan_all.c` — User Program

include — Include Folder (contains libraries)

- `interface_helper.c` — General Wi-Fi Interface Functions
- `wifi_scan.h` — Header File (contains necessary struct)

Code Explanation

wifi_scan_all.c

```c
#include "include/interface_helper.c"
#include "include/wifi_scan.h"
#include <stdio.h>  // printf, scanf
#include <string.h>

// convert bssid to printable hardware mac address
const char *bssid_to_string(const uint8_t bssid[BSSID_LENGTH], char bssid_string[BSSID_STRING_LENGTH])
{
    snprintf(bssid_string, BSSID_STRING_LENGTH, "%02x:%02x:%02x:%02x:%02x:%02x",
             bssid[0], bssid[1], bssid[2], bssid[3], bssid[4], bssid[5]);
    return bssid_string;
}

const int BSS_INFOS = 10; //the maximum amounts of APs (Access Points) we want to store

void Usage(char **argv);

int main(int argc, char **argv)
{
    struct wifi_scan *wifi=NULL;    // this stores all the library information
    struct bss_info bss[BSS_INFOS]; // this is where we are going to keep informatoin about APs (Access
    char mac[BSSID_STRING_LENGTH];  // a placeholder where we convert BSSID to printable hardware mac ad
    int status, i, count;

    if(argc != 2) // if the user doesn't know how to use, we display usage instructions
```

Code Explanation

wifi_scan_all.c

enum constants from **wifi_scan.h**

```c
#include "include/interface_helper.c"
#include "include/wifi_scan.h"
#include <stdio.h>  // printf, scanf
#include <string.h>

// convert bssid to printable hardware mac address
const char *bssid_to_string(const uint8_t bssid[BSSID_LENGTH], char bssid_string[BSSID_STRING_LENGTH])
{
```

wifi_scan.h

```c
// some constants - mac address length, mac adress string length, max length of wireless network id wit
enum wifi_constants {BSSID_LENGTH=6, BSSID_STRING_LENGTH=18, SSID_MAX_LENGTH_WITH_NULL=33};
// anything >=0 should mean that your are associated with the station
enum bss_status{BSS_NONE=-1, BSS_AUTHENTHICATED=0, BSS_ASSOCIATED=1, BSS_IBSS_JOINED=2};

// internal data used by the functions
struct wifi_scan;
```

```c
int main(int argc, char **argv)
{
    struct wifi_scan *wifi=NULL;    // this stores all the library information
    struct bss_info bss[BSS_INFOS]; // this is where we are going to keep informatoin about APs (Access
    char mac[BSSID_STRING_LENGTH];  // a placeholder where we convert BSSID to printable hardware mac ad
    int status, i, count;

    if(argc != 2) // if the user doesn't know how to use, we display usage instructions
```

Code Explanation

wifi_scan_all.c

```c
#include "include/interface_helper.c"
#include "include/wifi_scan.h"
#include <stdio.h>  // printf, scanf
#include <string.h>

// convert bssid to printable hardware mac address
const char *bssid_to_string(const uint8_t bssid[BSSID_LENGTH], char bssid_string[BSSID_STRING_LENGTH])
{
    snprintf(bssid_string, BSSID_STRING_LENGTH, "%02x:%02x:%02x:%02x:%02x:%02x",
             bssid[0], bssid[1], bssid[2], bssid[3], bssid[4], bssid[5]);
    return bssi
}

const int BSS_I                                                    ints) we want to store

void Usage(char **argv);

int main(int argc, char **argv)
{
    struct wifi_scan *wifi=NULL;     // this stores all the library information
    struct bss_info bss[BSS_INFOS]; // this is where we are going to keep informatoin about APs (Access
    char mac[BSSID_STRING_LENGTH];  // a placeholder where we convert BSSID to printable hardware mac ad
    int status, i, count;

    if(argc != 2) // if the user doesn't know how to use, we display usage instructions
```

print to fixed buffer of **char**

Code Explanation

wifi_scan_all.c

```c
#include "include/interface_helper.c"
#include "include/wifi_scan.h"
#include <stdio.h>  // printf, scanf
#include <string.h>

// convert bssid to printable hardware mac address
const char *bssid_to_string(const uint8_t bssid[BSSID_LENGTH], char bssid_string[BSSID_STRING_LENGTH])
{
    snprintf(bssid_string, BSSID_STRING_LENGTH, "%02x:%02x:%02x:%02x:%02x:%02x",
             bssid[0], bssid[1], bssid[2], bssid[3], bssid[4], bssid[5]);
    return bssid_string;
}

const int BSS_INFOS = 10;

void Usage(char **argv);

int main(int argc, char **argv)
{
    struct wifi_scan *wifi=NULL;     // this stores all the library information
    struct bss_info bss[BSS_INFOS]; // this is where we are going to keep informatoin about APs (Access
    char mac[BSSID_STRING_LENGTH];  // a placeholder where we convert BSSID to printable hardware mac ad
    int status, i, count;

    if(argc != 2) // if the user doesn't know how to use, we display usage instructions
```

constant **struct** size of **bss_info** array

Code Explanation

wifi_scan_all.c

```c
14  const int BSS_INFOS = 10; //the maximum amounts of APs (Access Points) we want to store
15
16  void Usage(char **argv);
17
18  int main(int argc, char **argv)
19  {
20      struct wifi_scan *wifi=NULL;     // this stores all the library information
21      struct bss_info bss[BSS_INFOS]; // this is where we are going to keep informatoin about APs (Access
22      char mac[BSSID_STRING_LENGTH];  // a placeholder where we convert BSSID to printable hardware mac ad
23      int status, i, count;
24
25      if(argc != 2) // if the user doesn't know how to use, we display usage instructions
26      {
27          Usage(argv);
28          return 0;
29      }
30
31      printf("Triggering scan needs permissions.\n");
32
33      printf("### Close the program with ctrl+c when you're done ###\n\n");
34
35      // initialize the library with network interface argv[1] (e.g. wlan0 / wlp3s0)
36      wifi = wifi_scan_init(argv[1]);
```

struct from **wifi_scan.h**

Rectangular Snip

Code Explanation

wifi_scan_all.c

```c
const int BSS_INFOS = 10; //the maximum amounts of APs (Access Points) we want to store

void Usage(char **argv);

int main(int argc, char **argv)
{
    struct wifi_scan *wifi=NULL;    // this stores all the library information
    struct bss_info bss[BSS_INFOS];
    char mac[BSSID_STRING_LENGTH];
    int status, i, count;

    if(argc != 2) // if the user c
    {
        Usage(argv);
        return 0;
    }

    printf("Triggering scan needs

    printf("### Close the program

    // initialize the library with network interface ar
    wifi = wifi_scan_init(argv[1]);
```

struct from **wifi_scan.h**

```c
// internal data used by the functions
struct wifi_scan;

// a single wireless network can have multiple BSSes working as
struct bss_info
{
    uint8_t bssid[BSSID_LENGTH]; //this is hardware mac address
    char ssid[SSID_MAX_LENGTH_WITH_NULL]; //this is the name of
    enum bss_status status;  //anything >=0 means that your are
    int32_t signal_mbm;  //signal strength in mBm, divide it by
    int32_t seen_ms_ago; //when the above information was collec
};
```

wifi_scan.h

Code Explanation

wifi_scan_all.c

```c
35      // initialize the library with network interface argv[1] (e.g. wlan0 / wlp3s0)
36      wifi = wifi_scan_init(argv[1]);
37
38      status = wifi_scan_all(wifi, bss, BSS_INF
39
40      // it may happen that device is unreachab                                    sn'
41      if(status < 0) {
42          perror("Unable to get scan data");
43      }
44      else { // wifi_scan_all returns the number of found stations, it may be greater than BSS_INFO
45          count = 0;
46      }
47
48      printf("MAC \t AP NAME \t RSS \t LAST SEEN \t STATUS \n");
49
50      for(i = 0; i < status && i < BSS_INFOS; ++i) {
51          printf("%s %s \t \t\t\t%d dBm \t%d \t%s\n",
52              bssid_to_string(bss[i].bssid, mac),                 // BSSID (MAC Address)
53              bss[i].ssid,                                        // ESSID (Access Point Name)
54              bss[i].signal_mbm/100,                              // RSS   (Recieved Signal Strengt
55              bss[i].seen_ms_ago,                                 // Last Seen (ms)
56              (bss[i].status==BSS_ASSOCIATED ? "associated" : "") // is connected
57          );
58          count++;
59      }
60
```

pass network interface given from command line argument as a parameter to **wifi_scan_init()** function in **wifi_scan.h**

Code Explanation

wifi_scan_all.c

```c
35      // initialize the library with network interface argv[1] (e.g. wlan0 / wlp3s0)
36      wifi = wifi_scan_init(argv[1]);
37
38      status = wifi_scan_all(wifi, bss, BSS_INFOS);
39
40      // it may happen that device is unreachable (e.g.
41      if(status < 0) {
42          perror("Unable to get scan data");
43      }
44      else { // wifi_scan_all returns the number of found stations, it may be greater than BSS_INFO
45          count = 0;
46      }
47
48      printf("MAC \t AP NAME \t RSS \t LAST SEEN \t STATUS \n");
49
50      for(i = 0; i < status && i < BSS_INFOS; ++i) {
51          printf("%s %s \t \t\t\t%d dBm \t%d \t%s\n",
52              bssid_to_string(bss[i].bssid, mac),                 // BSSID (MAC Address)
53              bss[i].ssid,                                        // ESSID (Access Point Name)
54              bss[i].signal_mbm/100,                              // RSS   (Recieved Signal Strengt
55              bss[i].seen_ms_ago,                                 // Last Seen (ms)
56              (bss[i].status==BSS_ASSOCIATED ? "associated" : "") // is connected
57          );
58          count++;
59      }
```

pass wifi_scan struct, bss and BSS_INFOS as parameter to **wifi_scan_all()** function in **wifi_scan.h**

Code Explanation

wifi_scan_all.c

```c
35      // initialize the library with network interface argv[1] (e.g. wlan0 / wlp3s0)
36      wifi = wifi_scan_init(argv[1]);
37
38      status = wifi_scan_all(wifi, bss, BSS_INFOS);
39
40      // it may happen that device is unreachable (e.g.
41      if(status < 0) {
42          perror("Unable to get scan data");
43      }
44      else { // wifi_scan_all returns the number of four
45          count = 0;
46      }
47
48      printf("MAC \t AP NAME \t RSS \t LAST SEEN \t STATUS \n");
49
50      for(i = 0; i < status && i < BSS_INFOS; ++i) {
51          printf("%s %s \t \t\t\t%d dBm \t%d \t%s\n",
52              bssid_to_string(bss[i].bssid, mac),                 // BSSID (MAC Address)
53              bss[i].ssid,                                        // ESSID (Access Point Name)
54              bss[i].signal_mbm/100,                              // RSS   (Recieved Signal Strengt
55              bss[i].seen_ms_ago,                                 // Last Seen (ms)
56              (bss[i].status==BSS_ASSOCIATED ? "associated" : "") // is connected
57          );
58          count++;
59      }
```

- **wifi_scan_all()** makes a passive scan of all networks around.
- Function requires permissions (sudo).
- returns -1 if errors, else count of stations found.

Code Explanation

wifi_scan_all.c

```c
35      // initialize the library with network interface argv[1] (e.g. wlan0 / wlp3s0)
36      wifi = wifi_scan_init(argv[1]);
37
38      status = wifi_scan_all(wifi, bss, BSS_INFOS);
39
40      // it may happen that device is unreachable
41      if(status < 0) {
42          perror("Unable to get scan data");
43      }
44      else { // wifi_scan_all returns the number
45          count = 0;
46      }
47
48      printf("MAC \t AP NAME \t RSS \t LAST SEEN \t STATUS \n");
49
50      for(i = 0; i < status && i < BSS_INFOS; ++i) {
51          printf("%s %s \t \t\t\t%d dBm \t%d \t%s\n",
52              bssid_to_string(bss[i].bssid, mac),                 // BSSID (MAC Address)
53              bss[i].ssid,                                        // ESSID (Access Point Name)
54              bss[i].signal_mbm/100,                              // RSS   (Recieved Signal Strengt
55              bss[i].seen_ms_ago,                                 // Last Seen (ms)
56              (bss[i].status==BSS_ASSOCIATED ? "associated" : "") // is connected
57          );
58          count++;
59      }
```

- use minimum value of i considering **status** and **BSS_INFOS** declared earlier.
- In other words we want to limit the number of BSS to display by either **status** or **BSS_INFOS**, whichever is less.

Code Explanation

wifi_scan_all.c

```c
    // initialize the library with network interface argv[1] (e.g. wlan0 / wlp3s0)
    wifi = wifi_scan_init(argv[1]);

    status = wifi_scan_all(wifi, bss, BSS_INFOS);

    // it may happen that device is unreachable (e.g. the device works in such way that it doesn'
    if(status < 0) {
        perror("Unable to get scan data");
    }
    else { // wifi_scan_all returns the number of
        count = 0;
    }

    printf("MAC \t AP NAME \t RSS \t LAST SEEN \t STATUS \n");

    for(i = 0; i < status && i < BSS_INFOS; ++i) {
        printf("%s %s \t \t\t\t%d dBm \t%d \t%s\n",
            bssid_to_string(bss[i].bssid, mac),                 // BSSID (MAC Address)
            bss[i].ssid,                                        // ESSID (Access Point Name)
            bss[i].signal_mbm/100,                              // RSS   (Recieved Signal Strengt
            bss[i].seen_ms_ago,                                 // Last Seen (ms)
            (bss[i].status==BSS_ASSOCIATED ? "associated" : "") // is connected
        );
        count++;
    }
```

- using dot operator to access values of **bss_info** struct.

Code Explanation

wifi_scan_all.c

```
47
48      printf("MAC \t AP NAME \t RSS \t LAST SEEN \t STATUS \n");
49
50      for(i = 0; i < status && i < BSS_INFOS; ++i) {
51          printf("%s %s \t \t\t\t%d dBm \t%d \t%s\n",
52              bssid_to_string(bss[i].bssid, mac),                // BSSID (MAC Address)
53              bss[i].ssid,                                       // ESSID (Access Point Name)
54              bss[i].signal_mbm/100,
55              bss[i].seen_ms_ago,
56              (bss[i].status==BSS_ASSOCIATED ? "associ
57          );
58          count++;
59      }
60
61      printf("\n----------------------------------------\n");
62      printf("Total hotspot(s) count: %d", count);
63      printf("\n----------------------------------------\n");
64
65      // free the library resources
66      wifi_scan_close(wifi);
67
68      return 0;
69  }
70
```

Finally we close library resource by **wifi_scan_close()** function

Compiling the program

```
$> gcc wifi_scan_all.c -lmnl -o wifi_scan_all
```

GNU Compiler

C Program

lib-mnl (minimalistic Netlink communication library)

-o Output File Name

Compiling the program

```
$> sudo ./wifi_scan_all wlp3s0
```

Elevated Permissions

Executable generated by compiler

Network Interface Name (find by **ifconfig**) [1]

Extra bits on Network Interface Names [2]

`wlp3s0`

`wl`	`p3`	`s0`
wlan	PCI Bus 3	Slot 0

Output

MCWC_Program_WIFI_AP: ./wifi_scan_all

...IFI_AP: ./wifi_scan_all

```
sidx1024@sidx1024-X450JN:~/Documents/MCWC_Program_WIFI_AP$ gcc wifi_scan_all.c -lmnl -o wifi_scan_all
sidx1024@sidx1024-X450JN:~/Documents/MCWC_Program_WIFI_AP$ sudo ./wifi_scan_all wlp3s0
Triggering scan needs permissions.
### Close the program with ctrl+c when you're done ###

MAC ADDRESS          AP NAME                    RSS(dbm)   LAST SEEN  STATUS
cc:d3:1e:50:4a:18    Jio of Siddharth           -58        128        associated
90:8d:78:70:b9:47    nirav                      -45        88
f0:d7:aa:aa:20:ac    ADYYTW90b0coNSlQbHVz       -58        2656
a8:6b:ad:36:95:d4    R 2 D 2                    -40        2632
90:8d:78:cd:da:2f    Bring Beer and get access  -89        28972
c8:3a:35:52:00:c8    Don't ask for wifi         -86        28324
98:de:d0:bd:1c:62                               -88        1676
7c:91:22:8f:04:48    Don                        -95        26564
f4:f2:6d:47:46:7c    Prohibited                 -84        3944

----------------------------------------
Total hotspot(s) count: 9
----------------------------------------
sidx1024@sidx1024-X450JN:~/Documents/MCWC_Program_WIFI_AP$
```

Program 2 - *Objective*

Write a program to transfer a file between two Wi-Fi devices over any available WiFi hotspot, using C programming.

Modules Covered :

3. Also performs file transfer between given two Wi-Fi devices –across different Wi-Fi hotspots.

Program 2 - *File hierarchy*

`Program`	
`server`	Server Folder
• `server.c`	Server C Program
`client`	Client Folder
• `client.c`	Client C Program

Code Explanation (Cont.)

```
1   #include <stdio.h>
2   #include <string.h>
3   #include <unistd.h> // read(sockfd, fname, 256);
4   #include <arpa/inet.h> // inet_xxx, ntohs
5
6   int main(int argc, char *argv[]) {
7
8     int sockfd = 0;
9     int bytesReceived = 0;
10    char recvBuff[1024];
11
12    memset(recvBuff, '0', sizeof(recvB
13    struct sockaddr_in serv_addr;
14
15    /* Create a socket first */
16    if ((sockfd = socket(AF_INET, SOCK
17      printf("\nError : Could not crea
18      return 1;
19    }
20
21    /* Initialize sockaddr_in data structure */
22    serv_addr.sin_family = AF_INET;
```

sockfd -> socket file descriptor

In Unix and related computers operating systems, a file descriptor is an abstract indicator used to access a file or other input/output resource, such as a pipe or network connection.

File descriptors are part of the POSIX application programming interface.

A file descriptor is a non-negative integer, represented in C programming language as the type int.

Code Explanation (Cont.)

```
1   #include <stdio.h>
2   #include <string.h>
3   #include <unistd.h> // read(sockfd, fname, 256);
4   #include <arpa/inet.h> // inet_xxx, ntohs
5
6   int main(int argc, char *argv[]) {
7
8     int sockfd = 0;
9     int bytesReceived = 0;
10    char recvBuff[1024];
11
12    memset(recvBuff, '0', sizeof(recvB
13    struct sockaddr_in serv_addr;
14
15    /* Create a socket first */
16    if ((sockfd = socket(AF_INET, SOCK
17      printf("\nError : Could not crea
18      return 1;
19    }
20
21    /* Initialize sockaddr_in data structure */
22    serv_addr.sin_family = AF_INET;
```

sockfd -> socket file descriptor

In Unix and related computers operating systems, a file descriptor is an abstract indicator used to access a file or other input/output resource, such as a pipe or network connection.

File descriptors are part of the POSIX application programming interface.

A file descriptor is a non-negative integer, represented in C programming language as the type int.

Code Explanation (Cont.)

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h> // read(sockfd, fname, 256);
#include <arpa/inet.h> // inet_xxx, ntohs

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

  int sockfd = 0;
  int bytesReceived = 0;
  char recvBuff[1024];

  memset(recvBuff, '0', sizeof(recvBuff));
  struct sockaddr_in serv_addr;

  /* Create a socket first */
  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("\nError : Could not create socket \n");
    return 1;
  }

  /* Initialize sockaddr_in data structure */
  serv_addr.sin_family = AF_INET;
```

recvBuff -> receive buffer, char array of 1024 bytes

memset (*array*, replacement, sizeof *array*) -> clear **recvBuff** and fill with 0 zeros.

Code Explanation (Cont.)

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h> // read(sockfd, fname, 256);
#include <arpa/inet.h> // inet_xxx, ntohs

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

  int sockfd = 0;
  int bytesReceived = 0;
  char recvBuff[1024];

  memset(recvBuff, '0', sizeof(recvBuff));
  struct sockaddr_in serv_addr;

  /* Create a socket first */
  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("\nError : Could not create socket \n");
    return 1;
  }

  /* Initialize sockaddr_in data structure */
  serv_addr.sin_family = AF_INET;
```

sockaddr_in -> inbuilt-struct provided by inet header file

memset (*array*, replacement, sizeof *array*) -> clear **recvBuff** and fill with 0 zeros.

Code Explanation (Cont.)

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h> // read(sockfd, fname, 256);
```

```c
// sockaddr_in

#include <netinet/in.h>

struct sockaddr_in {
    short            sin_family;   // e.g. AF_INET
    unsigned short   sin_port;     // e.g. htons(3490)
    struct in_addr   sin_addr;     // see struct in_addr, below
    char             sin_zero[8];  // zero this if you want to
};

struct in_addr {
    unsigned long s_addr;  // load with inet_aton()
};
```

provided by inet

t, sizeof *array*) ->
0 zeros.

```c
serv_addr.sin_family = AF_INET;
```

Code Explanation (Cont.)

```
#include <stdio.h>
#include <string.h>
#include <unistd.h> // read(sockfd, fname, 256);
#
i
```

```
// sockaddr_in

#include <netinet/in.h>

struct sockaddr_in {
    short           sin_family;   // e.g. AF_INET
    unsigned short  sin_port;     // e.g. htons(3490)
    struct in_addr  sin_addr;     // see struct in_addr, below
    char            sin_zero[8];  // zero this if you want to
};

struct in_addr {
    unsigned long s_addr;  // load with
};
```

s_addr ->stores IP address of 4 bytes.

```
/* Initialize sockaddr_in data structure */
serv_addr.sin_family = AF_INET;
```

Code Explanation (Cont.)

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h> // read(sockfd, fname, 256);
#include <arpa/inet.h> // inet_xxx, ntohs

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

  int sockfd = 0;
  int bytesReceived = 0;
  char recvBuff[1024];

  memset(recvBuff, '0', sizeof(recvBuff));
  struct sockaddr_in serv_addr;

  /* Create a socket first */
  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0))
    printf("\nError : Could not create socket \n
    return 1;
  }

  /* Initialize sockaddr_in data structure */
```

socket(int domain, int type, int protocol**)** [3]

- Creates an endpoint for communication
- Returns a file descriptor

- domain: **AF_INET** (Constant IPV4)
  - Specifies the communications domain in which a socket is to be created.
- type: **SOCK_STREAM** (Provides sequenced, reliable, two-way, connection-based byte streams.)
  - Specifies the type of socket to be created.
- protocol: **0**
  - Specifies a particular protocol to be used with the socket. Specifying a protocol of 0 causes socket() to use an unspecified default protocol appropriate for the requested socket type.

Code Explanation (Cont.)

```
12  memset(recvBuff, '0', sizeof(recvBuff));
13  struct sockaddr_in serv_addr;
14
15  /* Create a socket first */
16  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
17    printf("\nError : Could not create socket \n");
18    return 1;
19  }
20
21  /* Initialize sockaddr_in data structure */
22  serv_addr.sin_family = AF_INET;
23  serv_addr.sin_port = htons(5000); // port
24  char ip[50];
25
26  /* Ask IP address from user if not entere
27  if (argc < 2) {
28    printf("Enter IP address to connect: ")
29    scanf("%s", ip);
30  }
31  else {
32    strcpy(ip, argv[1]);
33  }
```

htons() -> function converts the unsigned short integer **hostshort** from **host byte** order to **network byte** order.

On the i386 the host byte order is Least Significant Byte first, whereas the network byte order, as used on the Internet, is Most Significant Byte first.

Code Explanation (Cont.)

```
21  /* Initialize sockaddr_in data structure */
22  serv_addr.sin_family = AF_INET;
23  serv_addr.sin_port = htons(5000); // port
24  char ip[50];
25
26  /* Ask IP address from user if not entered in command line arguments */
27  if (argc < 2) {
28    printf("Enter IP address to c
29    scanf("%s", ip);
30  }
31  else {
32    strcpy(ip, argv[1]);
33  }
34
35  serv_addr.sin_addr.s_addr = inet_addr(ip);
36
37  /* Attempt a connection */
38  if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
39    printf("\n Error : Connect Failed \n");
40    return 1;
```

Inet_addr() -> converts IP Address in char to unsigned long

Code Explanation (Cont.)

```
21  /* Initialize sockaddr_in data structure */
22  serv_addr.sin_family = AF_INET;
23  serv_addr.sin_port = htons(5000); // port
24  char ip[50];
25
26  /* Ask IP address from user if not entered in command line arguments */
27  if (argc < 2) {
28    printf("Enter IP address to connect: ");
29    scanf("%s", ip);
30  }
31  else {
32    strcpy(ip, argv
33  }
34
35  serv_addr.sin_addr.s_addr = inet_addr(ip);
36
37  /* Attempt a connection */
38  if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
39    printf("\n Error : Connect Failed \n");
40    return 1;
```

connect() -> System call connects the socket referred to by the file descriptor **sockfd** to the address specified by **addr**. The **addrlen** argument specifies the size of **addr**.

Code Explanation (Cont.)

```
21  /* Initialize sockaddr_in data structure */
22  serv_addr.sin_family = AF_INET;
23  serv_addr.sin_port = htons(5000); // port
24  char ip[50];
25
26  /* Ask IP address from user if not entered in command line arguments */
27  if (argc < 2) {
28    printf("Enter IP address to connect: ");
29    scanf("%s", ip);
30  }
31  else {
32    strcpy(ip, argv
33  }
34
35  serv_addr.sin_addr.s_addr = inet_addr(ip);
36
37  /* Attempt a connection */
38  if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
39    printf("\n Error : Connect Failed \n");
40    return 1;
```

serv_addr() -> socket struct created earlier.

Code Explanation (Cont.)

```
41  }
42
43  printf("Connected to ip: %s : %d\n", inet_ntoa(serv_addr.sin_addr), ntohs(serv_addr.sin_port));
44
45  /* Create file where data will be stored */
46  FILE *fp;
47  char fname[100];
48  read(sockfd, fname, 256);
49  printf("File Name: %s\n", fname);
50  printf("Receiving file...");
51  fp = fopen(fname, "ab");
52  if (NULL == fp) {
53    printf("Error opening file");
54    return 1;
55  }
56  long double sz = 1;
57
58  /* Receive data in chunks of 1024 bytes */
59  while ((bytesReceived = read(sockfd, recvBuff, 1024)) > 0) {
60    sz++;
61    printf("Received: %Le Mb", (sz / 1024));
62    fflush(stdout);
63    fwrite(recvBuff, 1, bytesReceived, fp);
64  }
65
```

inet_ntoa() -> function converts the IP address to **const char**

ntohs() -> convert port number from network byte order to host order

Code Explanation (Cont.)

```
41  }
42
43  printf("Connected to ip: %s : %d\n", inet_ntoa(serv_addr.sin_addr), ntohs(serv_addr.sin_port));
44
45  /* Create file where data will be stored */
46  FILE *fp;
47  char fname[100];
48  read(sockfd, fname, 256);
49  printf("File Name: %s\n", fname);
50  printf("Receiving file...");
51  fp = fopen(fname, "ab");
52  if (NULL == fp) {
53    printf("Error opening file");
54    return 1;
55  }
56  long double sz = 1;
57
58  /* Receive data in chunks of 1024 bytes */
59  while ((bytesReceived = read(sockfd, recvBuff, 1024)) > 0) {
60    sz++;
61    printf("Received: %Le Mb", (sz / 1024));
62    fflush(stdout);
63    fwrite(recvBuff, 1, bytesReceived, fp);
64  }
65
```

read() -> read from a file or socket

Code Explanation (Cont.)

```
41  }
42
43  printf("Connected to ip: %s : %d\n", inet_ntoa(serv_addr.sin_addr), ntohs(serv_addr.sin_port));
44
45  /* Create file where data will be stored */
46  FILE *fp;
47  char fname[100];
48  read(sockfd, fname, 256);
49  printf("File Name: %s\n", fname);
50  printf("Receiving file...");
51  fp = fopen(fname, "ab");
52  if (NULL == fp) {
53    printf("Error opening file");
54    return 1;
55  }
56  long double sz = 1;
57
58  /* Receive data in chunks of 1024 bytes */
59  while ((bytesReceived = read(sockfd, recvBuff, 1024)) > 0) {
60    sz++;
61    printf("Received: %Le Mb", (sz / 1024));
62    fflush(stdout);
63    fwrite(recvBuff, 1, bytesReceived, fp);
64  }
65
```

Code Explanation (Cont.)

```
41  }
42
43  printf("Connected to ip: %s : %d\n", inet_ntoa(serv_addr.sin_addr), ntohs(serv_addr.sin_port));
44
45  /* Create file where data will be stored */
46  FILE *fp;
47  char fname[100];
48  read(sockfd, fname, 256);
49  printf("File Name: %s\n", fname);
50  printf("Receiving file...");
51  fp = fopen(fname, "ab");
52  if (NULL == fp) {
53    printf("Error opening file");
54    return 1;
55  }
56  long double sz = 1;
57
58  /* Receive data in chunks of 1024 bytes */
59  while ((bytesReceived = read(sockfd, recvBuff, 1024)) > 0) {
60    sz++;
61    printf("Received: %Le Mb", (sz / 1024));
62    fflush(stdout);
63    fwrite(recvBuff, 1, bytesReceived, fp);
64  }
65
```

read() -> read from a file or socket

Code Explanation (Cont.)

```
52    if (NULL == fp) {
53      printf("Error opening file");
54      return 1;
55    }
56    long double sz = 1;
57
58    /* Receive data in chunks of 1024 bytes */
59    while ((bytesReceived = read(sockfd, recvBuff, 1024)) > 0) {
60      sz++;
61      printf("Received: %Le Mb", (sz / 1024));
62      fflush(stdout);
63      fwrite(recvBuff, 1, bytesReceived, fp);
64    }
65
66    if (bytesReceived < 0) {
67      printf("\n Read Error \n");
68    }
69
70    printf("\nFile OK....Completed\n");
71    return 0;
72  }
73
```

Code Explanation (Cont.) - Server

```c
pthread_t tid;
struct sockaddr_in serv_addr;
int listenfd = 0, ret;
char sendBuff[1024];
int numrv;
size_t clen = 0, slen = 0;

listenfd = socket(AF_INET, SOCK_STREAM, 0);
if (listenfd < 0) {
  printf("Error in socket creation\n");
  return -1;
}

printf("Socket retrieve success\n");

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(5000);
slen = sizeof(serv_addr);
ret = bind(listenfd, (struct sockaddr *) &serv_addr, slen);
if (ret < 0) {
  printf("Error in bind\n");
  return -1;
}

if (listen(listenfd, 10) == -1) {
  printf("Failed to listen\n");
  return -1;
}
```

bind() -> assigns a local protocol address to a socket.

Code Explanation (Cont.) - Server

```c
pthread_t tid;
struct sockaddr_in serv_addr;
int listenfd = 0, ret;
char sendBuff[1024];
int numrv;
size_t clen = 0, slen = 0;

listenfd = socket(AF_INET, SOCK_STREAM, 0);
if (listenfd < 0) {
  printf("Error in socket creation\n");
  return -1;
}

printf("Socket retrieve success\n");

serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(5000);
slen = sizeof(serv_addr);
ret = bind(listenfd, (struct sockaddr *) &serv_addr, slen);
if (ret < 0) {
  printf("Error in bind\n");
  return -1;
}

if (listen(listenfd, 10) == -1) {
  printf("Failed to listen\n");
  return -1;
}
```

listen() -> file descriptor and number of connections to allow

Code Explanation (Cont.) - Server

```
87
88    while (1) {
89      clen = sizeof(c_addr);
90      printf("Waiting...\n");
91      connfd = accept(listenfd, (struct sockaddr *) &c_addr, (socklen_t *) &clen);
92      if (connfd < 0) {
93        printf("Error in accept\n");
94        continue;
95      }
96      err = pthread_create( &tid, NULL, &SendFileToClient, &connfd);
97      if (err != 0)
98        printf("\ncan't create thread :[%s]", strerror(err));
99    }
100   close(connfd);
101   return 0;
102 }
103
```

pthread_create(process id, attributes, function, function argument**)**

Code Explanation (Cont.) - Server

```c
void SendFileToClient(int *arg) {
  int connfd = (int) *arg;
  printf("Connection accepted and id: %d\n", connfd);
  printf("Connected to Clent: %s:%d\n", inet_ntoa(c_addr.sin_addr), ntohs(c_addr.sin_port));
  write(connfd, fname, 256);

  FILE *fp = fopen(fname, "rb");
  if (fp == NULL) {
    printf("File opern error");
  }

  /*Read data from file and send it */
  while (1) {
    /*First read file in chunks of 256 bytes */
    unsigned char buff[1024] = {
      0
    };
    int nread = fread(buff, 1, 1024, fp);
    printf("Bytes read %d \n", nread);

    /*If read was success, send data. */
    if (nread > 0) {
      printf("Sending \n");
      write(connfd, buff, nread);
    }
    if (nread < 1024) {
      if (feof(fp)) {
        printf("End of file\n");
        printf("File transfer completed for id: %d\n", connfd);
      }
      if (ferror(fp))
        printf("Error reading\n");
      break;
    }
  }
  printf("Closing Connection for id: %d\n", connfd);
  close(connfd);
```

Compiling the program

```
$> gcc server.c -lpthread -o server
```

Link – pthread library (required in linking phase)

-o Output File Name

Compiling the program

```
$> ./server bird.jpg
```

Compiling the program

```
$> gcc client.c -o client
```

-o Output File Name

Compiling the program

```
$> ./client 192.168.15.131
```

Output - Server

```
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/server$ ./server
Socket retrieve success
Enter file name to send: bird.jpg
Waiting...
Waiting...
Connection accepted and id: 4
Connected to Clent: 192.168.15.131:44286
Segmentation fault (core dumped)
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/server$
```

Output - Client

```
client: ./client
Connected to ip: 192.168.15.131 : 5000
File Name: bird.jpg
Receiving file...
File OK....Completed
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/client$
```

Some useful Linux commands - 1

```
$> sudo iw dev wlp3s0 scan | grep SSID
```

iw is a 802.11 configuration utility for wireless devices.

OBJECT name (dev / phy)

Global regular expression print

Output – iw

```
sidx1024@sidx1024-X450JN:~/Documents$ sudo iw dev wlp3s0 scan | grep SSID
        SSID: Jio of Siddharth
        SSID: nirav
        SSID: VISHESH
        SSID: R 2 D 2\x20
        SSID: ADYYTW90b0coNSlQbHVz
        SSID: Bring Beer and get access\x20
        SSID: \x20   \x20
sidx1024@sidx1024-X450JN:~/Documents$
```

Some useful Linux commands - 2

```
$> nmcli dev wifi
```

nmcli – NetworkManager CLI

OBJECT name (dev / phy)

Output – nmcli

Documents: nmcli

Documents: nmcli

```
sidx1024@sidx1024-X450JN:~/Documents$ nmcli dev wifi
*  SSID                        MODE   CHAN  RATE        SIGNAL  BARS  SECURITY
   R 2 D 2                     Infra  5     54 Mbit/s   100     ▂▄▆▇  WPA1 WPA2
   nirav                       Infra  11    54 Mbit/s   92      ▂▄▆▇  WPA1 WPA2
   ADYYTW90b0coNSlQbHVz        Infra  5     54 Mbit/s   69      ▂▄▆_
*  Jio of Siddharth            Infra  1     54 Mbit/s   66      ▂▄▆_  WPA2
   Don't ask for wifi          Infra  8     54 Mbit/s   30      ▂___  WPA1
   Prohibited                  Infra  2     54 Mbit/s   27      ▂___  WPA1 WPA2
                               Infra  9     54 Mbit/s   27      ▂___  WPA1 WPA2
   Bring Beer and get access   Infra  6     54 Mbit/s   25      ▂___  WPA1 WPA2
   VISHESH                     Infra  1     54 Mbit/s   19      ▂___  WPA2
sidx1024@sidx1024-X450JN:~/Documents$
```

References

[1] Network Interface Names, https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/

[2] systemd/src/udev/**udev-builtin-net_id.c**
https://github.com/systemd/systemd/blob/master/src/udev/udev-builtin-net_id.c#L20

[3] Socket Documentation

http://man7.org/linux/man-pages/man2/socket.2.html

[4] htons documentation

https://linux.die.net/man/3/htons

Libraries used

[1] C/C++ library for monitoring signal strength of WiFi networks,

https://github.com/bmegli/wifi-scan

Code for both programs can be downloaded at :

https://github.com/siddharth1024/MCWC-WiFi-MiniProject

12th October 2017

Thank you