How to set up multicast between two beagle bone

By Late Nights group

Table of contents

1. Introduction

This guide's goal is to guide the user to create c code that allows both beagle bone to connect to the same multicast IP routing protocol. A multicast IP Routing protocol is used to distribute data for example audio/video streaming broadcast to multiple recipients. Multicast works by having a source send a single copy of a data into a multicast address, which is then distributed to an entire group of recipients. The code function can be put into one file e.g) network.c . This guide requires that both beagle bone to be able to connect to wifi and does not cover how to set up a wifi on the beagle bone. (see other guides posts on brian fraser's website).

2. Hardware & Pre Reqs

- 2x Beaglebone Green
- 2x Wifi Adapter
- Wifi Router with internet access

3. Master_function

Takes in no arguments.

1. **Create local variables for master to use.**
   ***See step 5 for get_local_ipaddress**

```
struct in_addr localInterface;
struct sockaddr_in groupSock;
int sd;

char databuf[20];
int datalen = sizeof(databuf);

get_local_ipaddress( "wlan0", databuf);
strncpy(LOCAL_IP_ADDR, databuf,datalen);
```

2. **Create a datagram socket on which to send**

```
sd = socket(AF_INET, SOCK_DGRAM, 0);

memset((char *) &groupSock, 0, sizeof(groupSock));
groupSock.sin_family = AF_INET;
groupSock.sin_addr.s_addr = inet_addr("239.0.0.0");
groupSock.sin_port = htons(12345);
```

3. **Set local interface for outbound multicast datagrams. The IP address specified must be associated with a local multicast capable interface**

```
localInterface.s_addr = inet_addr(databuf);

setsockopt(sd, IPPROTO_IP, IP_MULTICAST_IF, (char *)&localInterface,
sizeof(localInterface))
```

4. **Send a message to the multicast group specified by the groupSocket sockAddr structure**

```
sendto(sd, databuf, datalen, 0, (struct sockaddr*)&groupSock,
sizeof(groupSock));
```

5. **Return 1 for success or -1 for failure**

```
return 1;
```

4. Slave

Takes in no arguments

1. **Similarly to master, create local variables for slave to use**

```
struct sockaddr_in localSock;
int sd;
struct ip_mreq group;
int datalen;
char databuf[1024];
```

2. **Create a datagram socket on which to receive**

```
sd = socket(AF_INET, SOCK_DGRAM, 0);
```

3. **Enable SO_REUSEADDR to allow multiple instances of this application to receive copies of the multicast datagram.**

```
int reuse = 1;

setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, (char *)&reuse,
sizeof(reuse));
```

4. **Bind to proper port number with the IP address specified as INADDR_ANY.**

```
memset((char *) &localSock, 0, sizeof(localSock));
localSock.sin_family = AF_INET;
localSock.sin_port = htons(12345);
localSock.sin_addr.s_addr = INADDR_ANY;
```

5. **Join the multicast group x,x,x, on the local interface. Note that this IP_ADD_MEMBERHIP option must be called for each local interface over which the multicast datagram are to be received.**
6. ***See step 5 for get_local_ipaddress.**

```
char local_addr[20];
get_local_ipaddress("wlan0", local_addr);

group.imr_multiaddr.s_addr = inet_addr("239.0.0.0");
group.imr_interface.s_addr = inet_addr(local_addr);

strncpy(LOCAL_IP_ADDR, local_addr,sizeof(local_addr));
```

7. **Read data from the socket**

```
datalen = sizeof(databuf);
read(sd, databuf, datalen);

strncpy(PARTNER_ADDR, databuf,datalen);
```

8. **Return 1 for success or -1 for error**

```
return 1;
```

5. Get_local_ipaddress function.

Takes in char* interface_name, char* IP_address

1. **Initialize local variables for function**

```
struct ifaddrs *ifaddr, *ifa;
int family, s;
int max_address_lenth = 20;
char host[max_address_lenth];
getifaddrs(&ifaddr)
```

2. **Iterate through all interface address looking for ‘wlan0’**

```
for (ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
        if (ifa->ifa_addr == NULL)
                continue;

                s = getnameinfo(ifa->ifa_addr, sizeof(struct sockaddr_in) ,IP_address,
max_address_lenth, NULL, 0, NI_NUMERICHOST);

        if( (strcmp(ifa->ifa_name,"wlan0") == 0 ) &&
                (ifa->ifa_addr->sa_family      == AF_INET) ){
        //If we can’t find it exit
                if (s != 0){

                printf("getnameinfo() failed: %s\n", gai_strerror(s));
                exit(EXIT_FAILURE);
                }
        }
}
```

3. **Free ifaddr memory**

```
freeifaddrs(ifaddr);
```

6. Putting it all together

1. **Now that we have set up the multicast set up we can use it anyway we like. For example to exchange IP addresses with the two beagle bone we can do this: have slave send it's IP to the multi cast.**

```
int fd;
fd = socket(AF_INET, SOCK_DGRAM, 0)

struct sockaddr_in serveraddr;
memset( &serveraddr, 0, sizeof(serveraddr) );
serveraddr.sin_family = AF_INET;
serveraddr.sin_port = htons( 12340 );

 char local_addr[20];
 get_local_ipaddress("wlan0", local_addr); // get local address so it can be
transmitted to master

serveraddr.sin_addr.s_addr =  inet_addr(PARTNER_ADDR);

sendto( fd, local_addr, 20, 0, (struct sockaddr *)&serveraddr, sizeof(serveraddr))

close( fd );
```

2. **Likewise we will have master open a socket and wait for slave to send a message, this message will contain the slaves IP address.**

```
int fd;
```

```
fd = socket(AF_INET, SOCK_DGRAM, 0);

struct sockaddr_in serveraddr;
memset( &serveraddr, 0, sizeof(serveraddr) );
serveraddr.sin_family = AF_INET;

serveraddr.sin_port = htons( 12340 );
serveraddr.sin_addr.s_addr = htonl( INADDR_ANY );

bind(fd, (struct sockaddr *)&serveraddr, sizeof(serveraddr))

char buffer[20];
int length = recvfrom( fd, buffer, sizeof(buffer) - 1, 0, NULL, 0 );

strncpy(PARTNER_ADDR, buffer,length); // master will receivce slaves ip adress

close( fd );
```

3. **Make sure to run the code on slave first using ./slave then ./master on master on the command line.**

7. Troubleshooting

1. **Be sure to check that you've included all the libraries. For our project we had these libraries.**

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h>
#include <string.h>
#include <unistd.h>
#include <stdbool.h>
#include <ifaddrs.h>
```