# CSE 589 Fall 2015
# Programming Assignment 2
# Routing Protocols

**Name :** Dhiren Patel
**UBitName :** dhirenbh
**UBit number :** 50170084

If you are typing command and same time if you receive packet from some neighbor, then command string will be broken as received message will be displayed, but keep typing command from where it was left. (Don't type command again from beginning in this case)

**Program name :** dhirenbh_proj2.c

**How to run :** ./a.out -t <topology-file-name> -i <routing-update-interval>

- **Structure for Routing Table : At line no. 21 in program**

```
struct DV
{
        int ur_id;
        int neigh_id;
        int next_hop;
        char *cost;
};
```

- **Structure for Packet :** I have used file for creating packet. In file I'm inserting binary data after converting all required data into proper format.

  fp = fopen("**packet.bin**","wb");

  This file will be created. To check whether it is in proper format or not, No. of bytes of file should be counted. In our specified packet format, It will be of **68 bytes**.

  **This is how I'm populating data into it.**

  In **void make_packet()** function **at line no. 755 in program**

  For cost, ID, 0 etc. variable will be same at port. So, instead of creating new variables for each , I have used same variable port_t.

  IP will also be converted into 32 bits.

```
uint32_t ip;
uint16_t port;
```

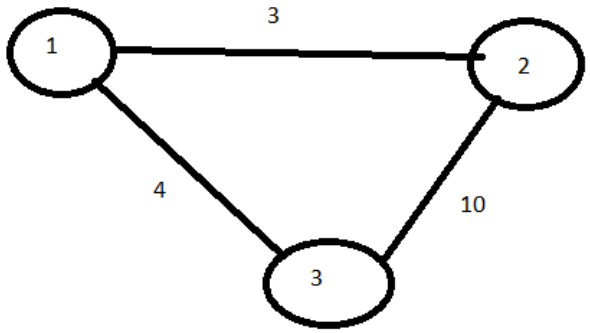
```
struct  in_addr  ton;

        port_t = htons(5);      //No. of update fields
        fwrite(&port_t,sizeof(port_t),1,fp);
        port_t = htons(self_port);   //Port no
        fwrite(&port_t,sizeof(port_t),1,fp);
        inet_pton(AF_INET,self_ip,&ton); //own IP  address
        ip_t = ton.s_addr;
        fwrite(&ip_t,sizeof(ip_t),1,fp);
        if(checkpoint == 0)
        {
        temp = 0;
        while(temp<list_index)    //As many no. of fields to include in packet
        {
                inet_pton(AF_INET,list[temp].ip,&ton);  // IP address - others
                ip_t = ton.s_addr;
                fwrite(&ip_t,sizeof(ip_t),1,fp);
                port_t = htons(list[temp].port);  // Port - others
                fwrite(&port_t,sizeof(port_t),1,fp);
                port_t = htons(0);  // 0 field
                fwrite(&port_t,sizeof(port_t),1,fp);
                port_t = htons(list[temp].id);    // Others Id
                fwrite(&port_t,sizeof(port_t),1,fp);
                if(strcmp(d[temp].cost,"Inf")==0)
                        port_t = htons(10000);
                else
                port_t = htons(atoi(d[temp].cost));   // Cost of link
                fwrite(&port_t,sizeof(port_t),1,fp);
                temp++;
        }
```

## How commands work :

1) **Update :** Suppose Topology is as below.

I will explain working implementation of this command with below example.




**At server 1 : update 1 2 5 or update 1 2 inf**

First it will check whether 2 is 1's neighbor or not.

At this case at server 1 , link cost for link (1-2) will be updated as 5 or inf.
At server 2, link cost for link (2-1) will be updated as 5 or inf.

(And both server's routing table will also be updated with this new cost, as packets from server 1 to server 2 goes directly to sever 2 and vice versa)
Server 1 will send this link change info to server 2 for this.

**( In case of update 1 2 inf , at server 1 all paths in routing table which passes through 2 will be set to infinity. )**

**At server 2 : update 2 3 12 or update 2 3 inf**

In this case, packets from 2 to 3 or 3 to 2 passes through 1.
So, link cost at both servers will be updated but routing table will not be updated with this new link as there is shortest path from 2 to 3 via 1.

But later suppose cost of path (1-2) increases to 20, then Distance vector table at both server 2 and 3 will be updated with 12.

## 2) Disable

In above figure, if user types

**At Server 1 : disable 2**

1) It will check whether 2 is it's neighbor or not.
2) It will update link cost to inf and will remove it from it's neighbor list and no longer sends routing information to 1.
3) When 2 will not receive any packets from 1 in 3 consecutive timeout intervals, it will also update it's link cost of 1 to inf and will remove it from it's neighbor list and will no longer send any packets to 1.

## 3) Display

**At Server 1 : display**
It will show routing table as below.

| Source-Id | Destination-Id | Next-Hop | Cost of Path |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 1 | 2 | 2 | 7 |
| 1 | 3 | 3 | 2 |
| 1 | 4 | 4 | Inf |
| 1 | 5 | 5 | Inf |

## 4) Crash

If crash command is types at server 1, then all cost in routing table will be set to inf. All neighbor's link will be removed. So, it will not send any packets to any server. So, other server will not receive packets from server 1 for 3 consecutive timeout, so they will set that link path to inf. And will stop sending packets to server 1.

**Step** and **packets** commands are working as described in project document.

## How Timeout is working

**Timerfd is created and added in select readfds.**

```
int timer_fd;
time_t start,end;
struct itimerspec new_value;
struct timespec now;
start = time(NULL);
end = start + interval;


new_value.it_value.tv_sec =  (end-start) ;
   new_value.it_value.tv_nsec = 0;
new_value.it_interval.tv_sec = 0;
new_value.it_interval.tv_nsec = 0;
timer_fd = timerfd_create(CLOCK_MONOTONIC, 0);
if (timer_fd == -1)
printf("Timer fd create error");
if (timerfd_settime(timer_fd, 0, &new_value, 0) == -1)
{
       printf("Timer fd settime error");
}
FD_SET(timer_fd,&readfds);
if(FD_ISSET(timer_fd,&readfds))
{
start = time(NULL);
end = start + interval;
//write ur timeout code here
}
```

## My program contains following functions :

1) **void separate_words(char *str)** - is used to separate words from line using strtok.
2) **int totalline(FILE *fp)** – gives total number of lines in topology file.
3) **char * readline(int n,FILE *fp)** -- Reads particular line from topology file given by n
4) **void populate_initial_DV_and_neighbour_tables(FILE *fp)** – At program startups it populates initial distance vector and neighbour's table
5) **void display_routing_table() –** is used to display routing table
6) **void update_routing_table(char *str,int dummy_list_index) –** updates routing table after processing input packet
7) **void process_received_packet() –** Processes received input packet ( like separating various fields like ip and port etc. )
8) **void disable_link(int temp) –** is used to disable link between neighbours
9) **void server(FILE *fp) –** creates UDP socket connection at program startup
10) **void client(int ,int) –** is used to send UDP messages to other servers
11) **void make_packet(int checkpoint) --** is used to make packet before sending