TW-1 (Message Queue)

Writer

```
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <stdlib.h>
#define MAX 50

struct msg_buffer {
    long mesg_type;
    char mesg_text[100];

}message;

int main() {
    key_t key;
    int msgid;

    key = ftok("progfile", 65);
    msgid = msgget(key, 0666 | IPC_CREAT);
    message.mesg_type = 1;
    printf("Write data: \n");
    fgets(message.mesg_text, MAX, stdin);
    msgsnd(msgid, &message, sizeof(message), 0);
    printf("Data sent is : %s\n", message.mesg_text);
    return 0;
}
```

Reader

```
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <stdlib.h>
#define MAX 50

struct msg_buffer {
    long mesg_type;
    char mesg_text[100];

}message;

int main() {
    key_t key;
    int msgid;
    key = ftok("progfile", 65);
    msgid = msgget(key, 0666 | IPC_CREAT);
    msgrcv(msgid, &message, sizeof(message), 1, 0);
    printf("Data read is: %s\n", message.mesg_text);
    msgctl(msgid, IPC_RMID, NULL);
    return 0;
}
```

Output

Writer

```
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-1$ gcc TW-1_writer.c
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-1$ ./a.out
Write data: Hello World
Data sent is : Hello World
```

Reader

```
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-1$ gcc TW-1_reader.c
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-1$ ./a.out
Data read is: Hello World
```

TW-1 (Pipe)

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>

int main() {
    int fd[2], n;
    char buffer[100];
    pid_t p;
    pipe(fd);
    p = fork();
    if (p > 0) {
      printf("Parent process pid: %d\n", getppid());
      printf("Child process pid: %d\n", p);
      printf("Passing value child\n");
      write(fd[1], "Hello World!\n", 13);
    }
    else {
      printf("Child process pid: %d\n", getpid());
      printf("Parent process pid: %d\n", getppid());
      n = read(fd[0], buffer, 100);
      printf("Data received by child process: \n");
      write(1, buffer, n);
    }
    return 0;
}
```

Output

```
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-1/pipe$ gcc TW-1.c
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-1/pipe$ ./a.out
Parent process pid: 7437
Child process pid: 7618
Passing value child
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-1/pipe$ Child process
pid: 7618
Parent process pid: 1511
Data received by child process:
Hello World!
```

TW-2

Server

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <string.h>
#define PORT 4444

int main() {
    int listenfd, connfd;
    struct sockaddr_in servAddr, cliAddr;
    socklen_t clilen;
    char buffer[1024];

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("[+] Server socket created successfully\n");

    bzero(&servAddr, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(PORT);
    servAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    bind(listenfd, (struct sockaddr *) &servAddr, sizeof(servAddr));
    printf("[+] Bind to PORT %d successful\n", PORT);

    listen(listenfd, 5);
    printf("[+] Listening...\n");

    connfd = accept(listenfd, (struct sockaddr *) &cliAddr, &clilen);

    strcpy(buffer, "Hello World!");
    send(connfd, buffer, strlen(buffer), 0);
    printf("[+] Data sent to client: %s\n", buffer);

    printf("[+] Closing the connection\n");
    return 0;
}
```

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define PORT 4444

int main() {
    int sockfd;
    struct sockaddr_in servAddr;
    char buffer[1024];

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("[+] Client socket created successfully\n");

    bzero(&servAddr, sizeof(servAddr));
    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(PORT);
    servAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

    connect(sockfd, (struct sockaddr *) &servAddr, sizeof(servAddr));
    printf("[+] Connected to server\n");

    recv(sockfd, buffer, 1024, 0);
    printf("[+] Data received from server: %s\n", buffer);
    printf("[+] Closing the connection\n");
    return 0;
}
```

Output

Server

```
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-2$ cc server.c
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-2$ ./a.out
[+]Server socket created successfully.
[+]Bind to port number 4444.
[+]Listening...
[+]Data sent to client: Hello.
[+]Closing the connection
```

Client

```
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-2$ cc client.c
lab2@lab2-virtual-machine:~/Aniket_NP-Lab/TW-2$ ./a.out
[+]Client socket created successfully.
[+]Connected to server.
[+]Data received: Hello
[+]Closing the connection.
```

TW-3

```
#include <stdio.h>
#define NODES 10
#define NO_ROUTE 999
#define NO_HOP 1000

int no;

struct node {
    int a[NODES][4];
}router[NODES];


void init(int r) {
    int i;
    for (i = 1; i <= no; i++) {
      router[r].a[i][1] = i;
      router[r].a[i][2] = NO_ROUTE;
      router[r].a[i][3] = NO_HOP;
    }
    router[r].a[r][2] = 0;
    router[r].a[r][3] = r;
}

void inp(int r) {
    int i;
    printf("\nEnter distance from node %d to other nodes\n", r);
    printf("Enter 999 if there is no direct route\n");
    for (i = 1; i <= no; i++) {
      if (i != r) {
            printf("Enter distance to node %d: ", i);
            scanf("%d", &router[r].a[i][2]);
            router[r].a[i][3] = i;
      }
    }
}

void display(int r) {
    int i;
    printf("\nThe routing table for node %d is as follows", r);
    for (i = 1; i <= no; i++) {
      if (router[r].a[i][2] == 999)
            printf("\n%d \t no link \t no hop", router[r].a[i][1]);
      else
```

```
            printf("\n%d \t %d \t %d", router[r].a[i][1],
router[r].a[i][2], router[r].a[i][3]);
    }
}

void dv_algo(int r) {
    int i, j, z;
    for (i = 1; i <= no; i++) {
    // r → source router
    // i → step taken (via which router to reach the dest router)
    // j → destination router
    // cannot jump from the source router or to a router which is not
reachable or from the source router
      if (router[r].a[i][2] != 999 && router[r].a[i][2] != 0) {
            for (j = 1; j <= no; j++) {
                  z = router[r].a[i][2] + router[i].a[j][2];
                  if (z < router[r].a[j][2]) {
                        router[r].a[j][2] = z;
                        router[r].a[j][3] = i;
                  }
            }
      }
    }
}

int main() {
    int i, j, x, y;
    char choice = 'y';
    printf("Enter the number of nodes: ");
    scanf("%d", &no);
    for (i = 1; i <= no; i++) {
      init(i);
      inp(i);
    }
    printf("\nThe routing tables of nodes after initialization is as
follows");
    for (i = 1; i <= no; i++)
      display(i);
    printf("\n\nComputing shortest paths...\n");
    for (i = 1; i <= no; i++)
      dv_algo(i);
    printf("\nThe routing tables of nodes after computation of
shortest paths is as follows");
    for (i = 1; i <= no; i++)
      display(i);
```

```
    printf("\n");
    while (choice != 'n'){
      printf("\nEnter the nodes between which shortest distance is
to be found: ");
      scanf("%d %d", &x, &y);
      getchar();
      printf("The length of the shortest path between nodes %d and
%d is %d\n", x, y, router[x].a[y][2]);
      printf("Continue? (y/n): ");
      scanf("%c", &choice);
    }
    return 0;
}
```

Output

```
aniket@aniket-Lenovo-IdeaPad-S540-15IML-D:~/NP-Lab/TW-3$ gcc dvalgo.c
aniket@aniket-Lenovo-IdeaPad-S540-15IML-D:~/NP-Lab/TW-3$ ./a.out
Enter the number of nodes: 5

Enter distance from node 1 to other nodes
Enter 999 if there is no direct route
Enter distance to node 2: 1
Enter distance to node 3: 999
Enter distance to node 4: 999
Enter distance to node 5: 999

Enter distance from node 2 to other nodes
Enter 999 if there is no direct route
Enter distance to node 1: 1
Enter distance to node 3: 3
Enter distance to node 4: 4
Enter distance to node 5: 5

Enter distance from node 3 to other nodes
Enter 999 if there is no direct route
Enter distance to node 1: 999
Enter distance to node 2: 2
Enter distance to node 4: 3
Enter distance to node 5: 999

Enter distance from node 4 to other nodes
Enter 999 if there is no direct route
Enter distance to node 1: 999
Enter distance to node 2: 4
```

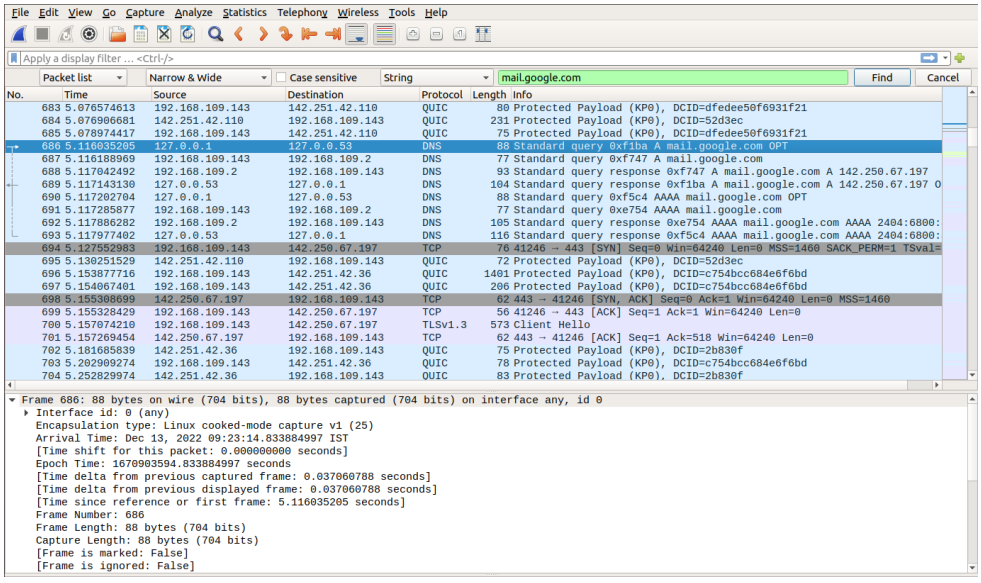
```
Enter distance to node 3: 3
Enter distance to node 5: 999

Enter distance from node 5 to other nodes
Enter 999 if there is no direct route
Enter distance to node 1: 999
Enter distance to node 2: 5
Enter distance to node 3: 999
Enter distance to node 4: 999

The routing tables of nodes after initialization is as follows
The routing table for node 1 is as follows
1       0       1
2       1       2
3       no link       no hop
4       no link       no hop
5       no link       no hop
The routing table for node 2 is as follows
1       1       1
2       0       2
3       3       3
4       4       4
5       5       5
The routing table for node 3 is as follows
1       no link       no hop
2       2       2
3       0       3
4       3       4
5       no link       no hop
The routing table for node 4 is as follows
1       no link       no hop
2       4       2
3       3       3
4       0       4
5       no link       no hop
The routing table for node 5 is as follows
1       no link       no hop
2       5       2
3       no link       no hop
4       no link       no hop
5       0       5

Computing shortest paths...
```

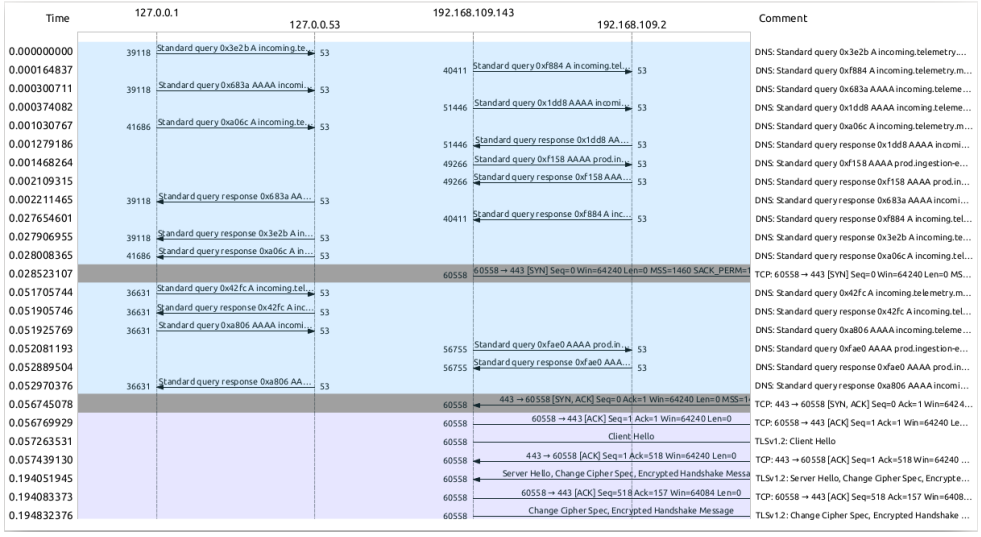
```
The routing tables of nodes after computation of shortest paths is as
follows
The routing table for node 1 is as follows
1       0       1
2       1       2
3       4       2
4       5       2
5       6       2
The routing table for node 2 is as follows
1       1       1
2       0       2
3       3       3
4       4       4
5       5       5
The routing table for node 3 is as follows
1       3       2
2       2       2
3       0       3
4       3       4
5       7       2
The routing table for node 4 is as follows
1       5       2
2       4       2
3       3       3
4       0       4
5       9       2
The routing table for node 5 is as follows
1       6       2
2       5       2
3       8       2
4       9       2
5       0       5

Enter the nodes between which shortest distance is to be found: 1 5
The length of the shortest path between nodes 1 and 5 is 6
Continue? (y/n): y

Enter the nodes between which shortest distance is to be found: 1 4
The length of the shortest path between nodes 1 and 4 is 5
Continue? (y/n): n
```

TW-4

UDP Program execution steps

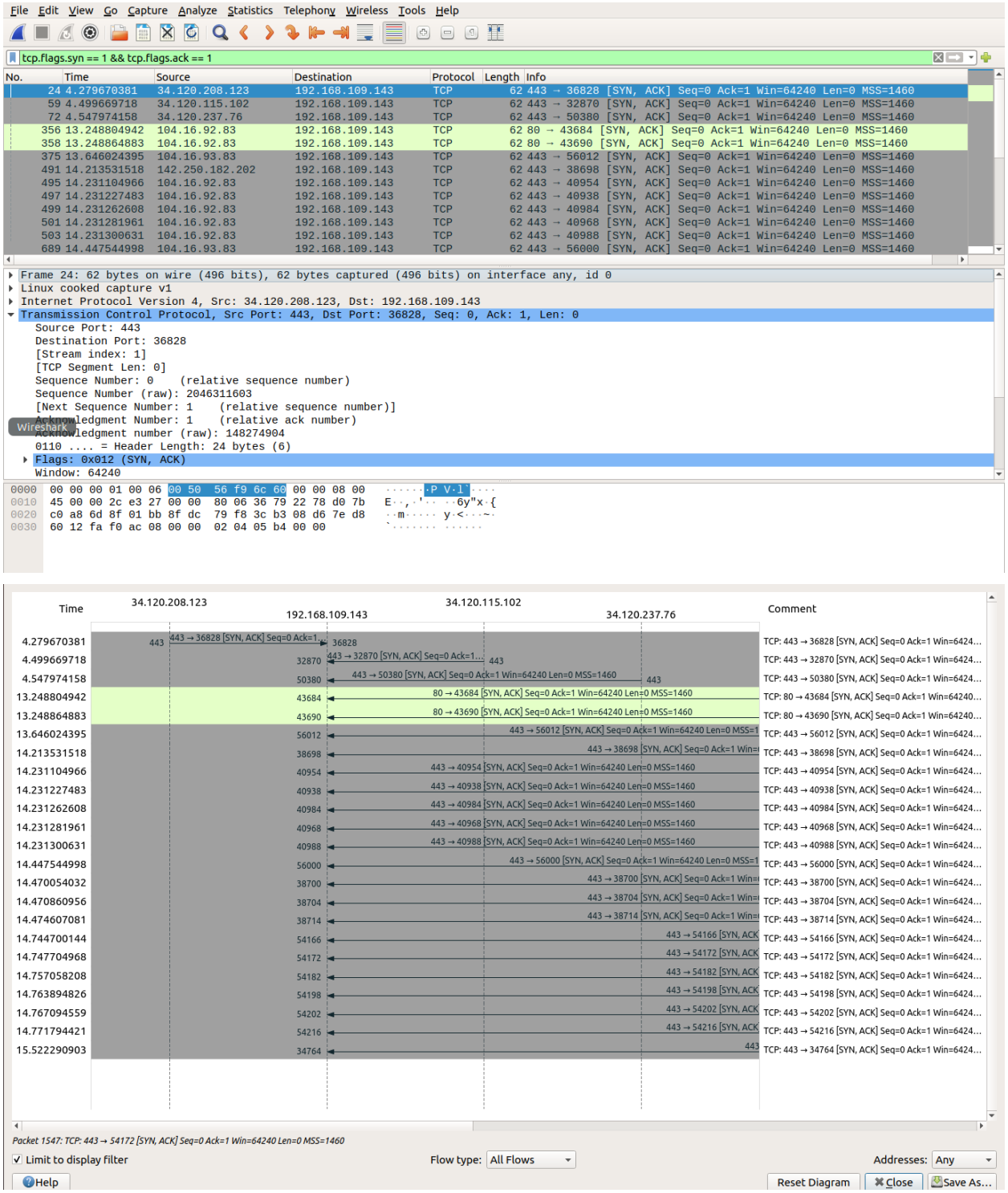
1. Open Wireshark and double click on any-interface to start the packet capture process.
2. Open the browser and enter any website's fully qualified domain name in the browser address bar and hit enter.
3. After the site is fully loaded, stop the capturing process in Wireshark, goto edit in the menu bar and select find packet option or just press Ctrl+F.
4. In the Find Packet menu bar, select the String option in the display filter drop-down menu and enter the name of the website in the next box and click on find.
5. The arrow pointing towards the packet is the request packet, and the arrow coming out from the packet is the response packet.
6. Click on any request or response DNS packet and examine the UDP packet.





TW-5

TCP Program execution steps



TW-6

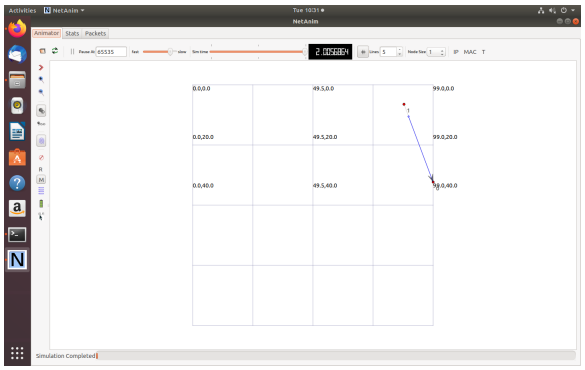
Load for
First.cc → TW6
Second.cc → TW7
Third.cc → TW8

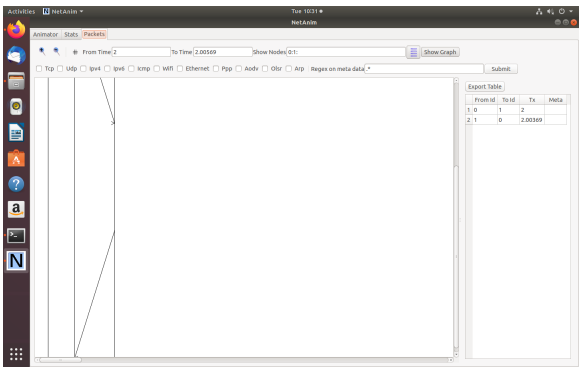
IMPORTANT - Load **NS3-WORKING** virtual machine

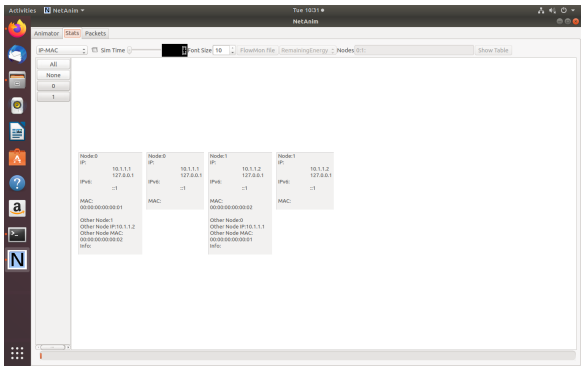
1) Copy the required TW file from examples/tutorial folder to scratch folder
2) Add the below four lines at the end of the program before “Simulator::Run()”

```
#include "ns3/netanim-module.h"
AnimationInterface anim("first, xml");
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("first.tr"))
;
pointToPoint.EnablePcapAll("first");
```

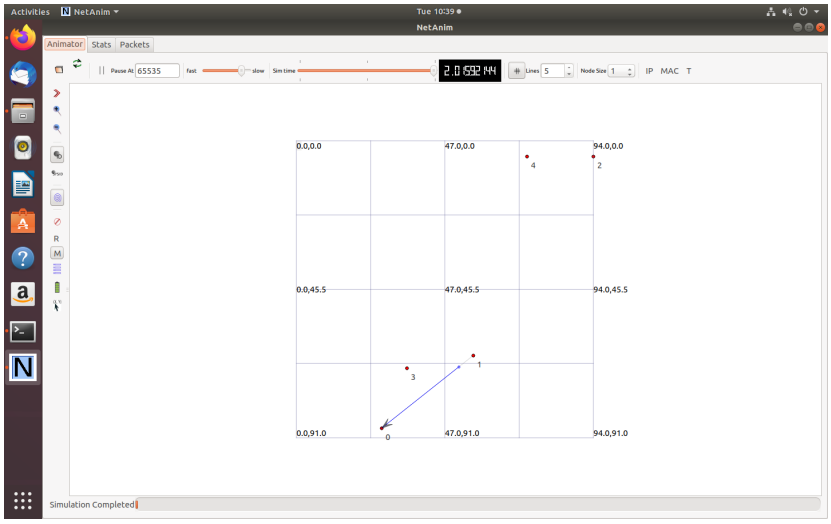
3) Open terminal in **ns-3.28** and run command **./waf build**
4) Run command **./waf --run scratch/[first/second/third]**
5) **cd .. (to move back a folder)**
6) Change directory to **netanim-3.1** and run the command **./NetAnim** to run the network animator
7) Load the XML file **[first.xml /second.xml / third.xml]** and run it

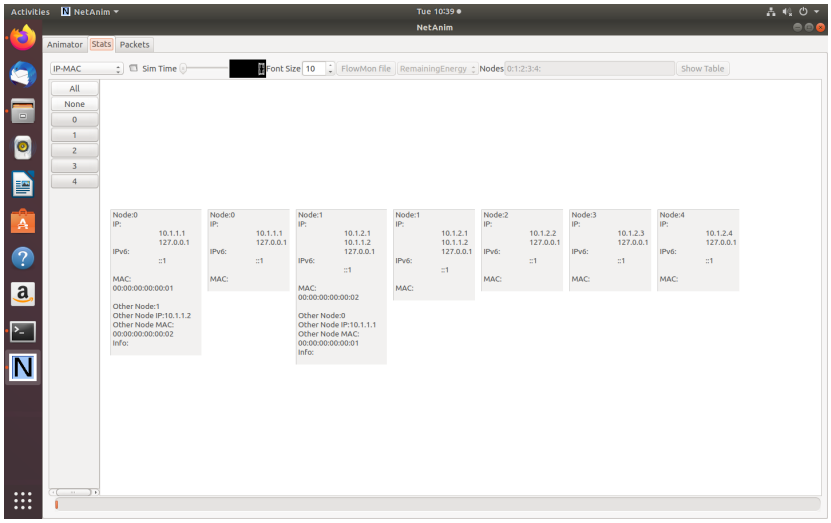


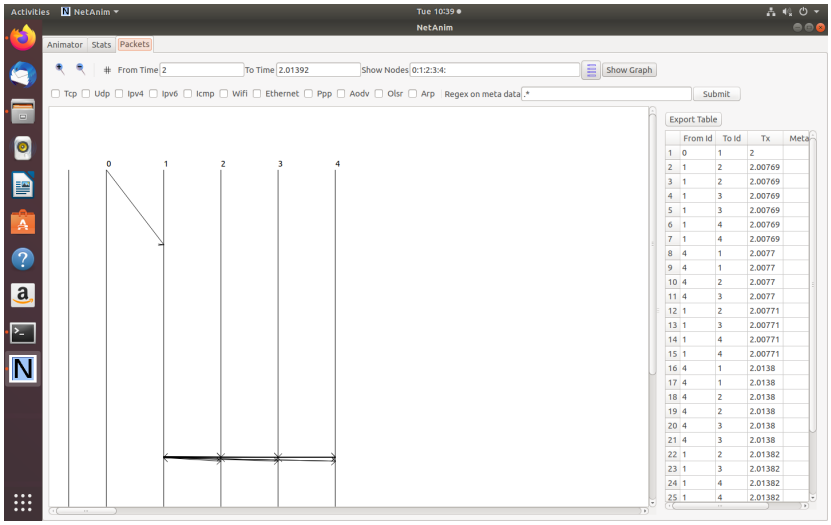




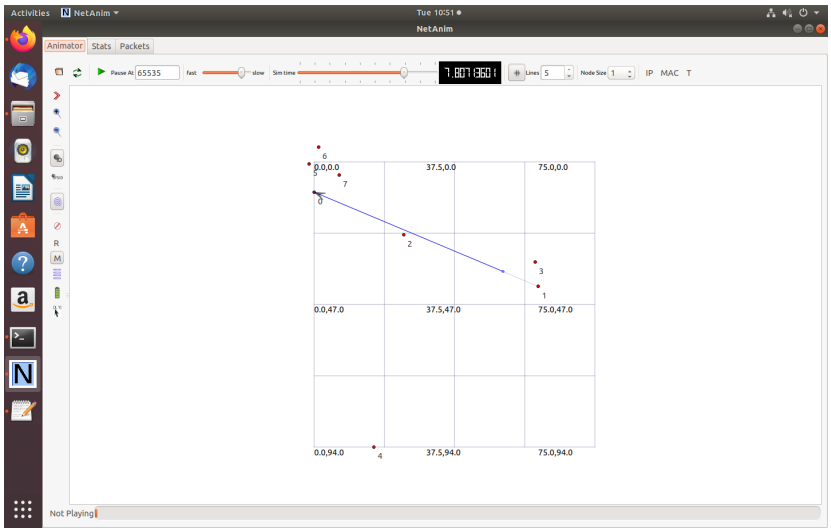
TW-7

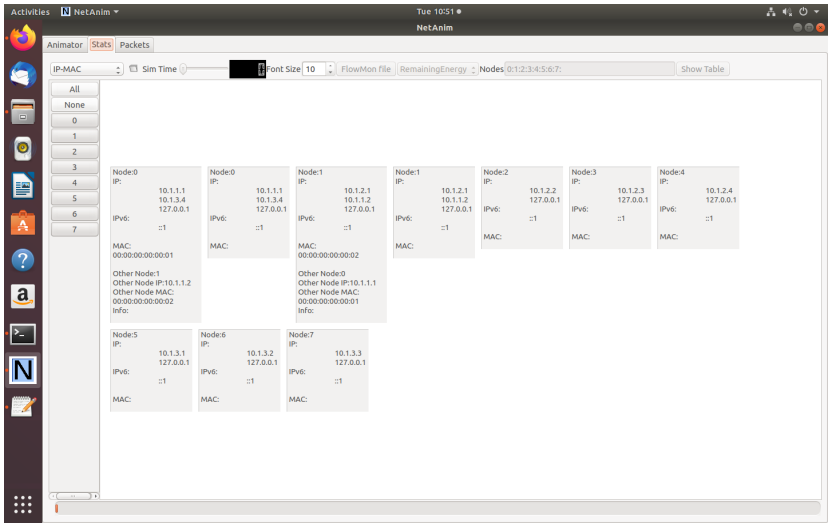

Activities
NetAnim
Tue 10:39
Animator
Stats
Packets
0.0,0.0
47.0,0.0
94.0,0.0
0.0,45.5
47.0,45.5
94.0,45.5
0.0,91.0
47.0,91.0
94.0,91.0
Simulation Completed

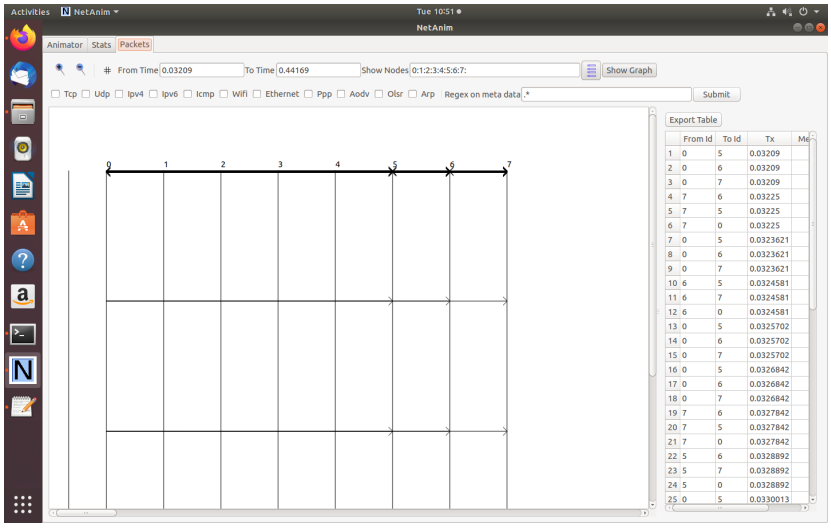

Activities
NetAnim
Tue 10:39
Animator
Stats
Packets
IP-MAC
Sim Time
Font Size
FlowMon File
RemainingEnergy
Nodes
Show Table
All
None
0
1
2
3
4


Activities
NetAnim
Tue 10:39
Animator
Stats
Packets
From Time
To Time
Show Nodes
Show Graph
Tcp
Udp
Ipv4
Ipv6
Icmp
Wifi
Ethernet
Ppp
Aodv
Olsr
Arp
Regex on meta data
Submit
Export Table
From Id
To Id
Tx
Meta

TW-8


Activities
NetAnim
Tue 10:51
NetAnim
Animator
Stats
Packets
Not Playing
0.0,0.0
37.5,0.0
75.0,0.0
0.0,47.0
37.5,47.0
75.0,47.0
0.0,94.0
37.5,94.0
75.0,94.0


Activities
NetAnim
Tue 10:51
NetAnim
Animator
Stats
Packets
IP-MAC
Sim Time
Font Size
FlowMon file
RemainingEnergy
Nodes
Show Table
All
None


Activities
NetAnim
Tue 10:51
NetAnim
Animator
Stats
Packets
From Time
To Time
Show Nodes
Show Graph
Tcp
Udp
Ipv4
Ipv6
Icmp
Wifi
Ethernet
Ppp
Aodv
Olsr
Arp
Regex on meta data
Submit
Export Table
From Id
To Id
Tx

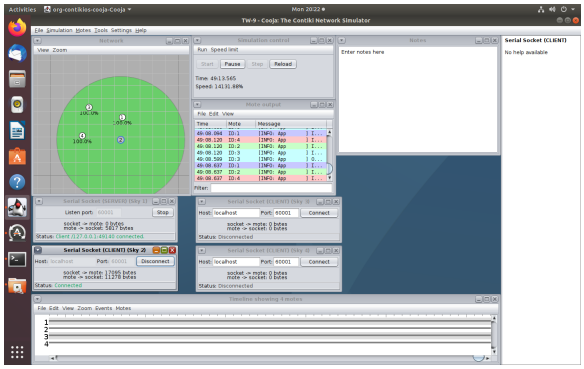
TW-9

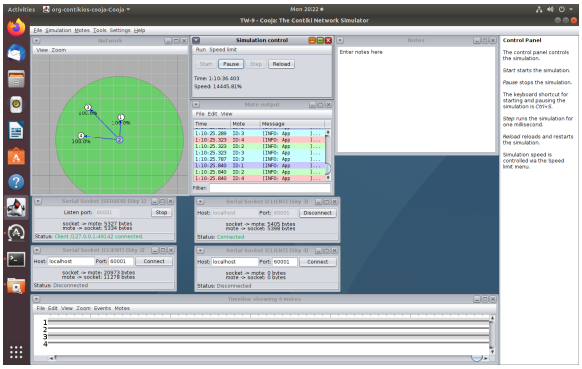
Steps to open the cooja simulator

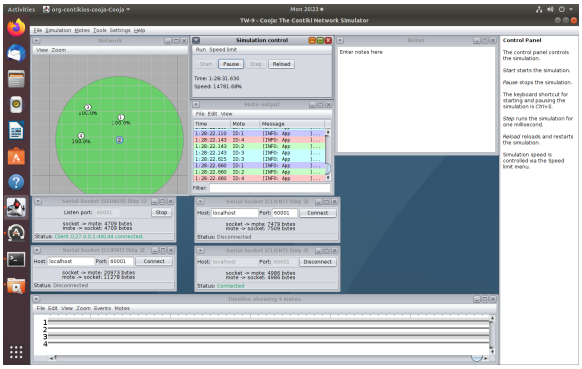
1. Goto root directory
2. cd contiki-ng
3. cd tools
4. cd cooja
5. ant run

Steps to create motes and configure them as server and client

1. Goto File -> New Simulation
2. Name the simulation and click on create
3. Click on Motes -> Add motes -> Create a new mote type -> Sky mote
4. Click on Browse and select ipv6-hooks.c (/contiki-ng/examples/libs/ipv6-hooks)
5. Click on open and then on compile and then on create
6. Enter the number of motes as 4 and click on Add motes
7. Place all motes close to each other such that the coverage is 100% for each of them
8. Right click on mote 1 and then click More tools for Sky 1 and then on Serial Socket (SERVER). Mote 1 has been configured as Server.
9. Similarly, configure motes 2, 3 and 4 as clients.
10. Copy the server's listening port number and paste it as the port number for all clients.
11. Start the server and connect the client to the server.
12. Run the simulation by clicking on Simulation -> Run Simulation







TW-10

Follow the exact same steps to create 2 motes (client and server).

1. Click on Browse and select rpl-udp(/contiki-ng/examples/libs/rpl-udp)
2. Create udp-server.c and add 1 mote by clicking Motes -> Add new Mote -> Browse
3. Create udp-client.c and add 1 mote
4. Place both the motes close to each other
5. Configure 1 as server and 2 as client
6. Copy the server's port number to the client.
7. Start the server and connect the client.
8. Run the simulation.