NP Termworks with writeup

Termwork1

Message Queuing – This allows messages to be passed between processes using either a single queue or several message queue. This is managed by system kernel these messages are coordinated using an API.

Pipes (Same Process) – This allows flow of data in one direction only. Analogous to simplex systems (Keyboard). Data from the output is usually buffered until input process receives it which must have a common origin

Programs with sample input/output:

Message Queues:

Message 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;
}
```

Message Sender

```
#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;
```

```
}booboo@DESKTOP-NHD0C8T:~$ gcc n1mr.c -o n1mr
booboo@DESKTOP-NHD0C8T:~$ ./n1mr
Data read is: muh me lele
```

```
booboo@DESKTOP-NHD0C8T:~$ gcc n1ms.c -o n1ms
booboo@DESKTOP-NHD0C8T:~$ ./n1ms
Write data:
muh me lele
Data sent is : muh me lele
```

Pipes

```
booboo@DESKTOP-NHD0C8T:~$ gcc n1p.c -o n1p
booboo@DESKTOP-NHD0C8T:~$ ./n1p
Parent process pid: 427
Child process pid: 1443
Passing value child
Child process pid: 1443
Parent process pid: 1442
Data received by child process:
Hello World!
booboo@DESKTOP-NHD0C8T:~$
```

```
#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;

}
```

Termwork2

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server

Program with sample input/output:

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;

}
```

```
booboo@DESKTOP-NHD0C8T:~$ gcc n2s.c -o n2s
booboo@DESKTOP-NHD0C8T:~$ ./n2s
[+] Server socket created successfully
[+] Bind to PORT 4444 successful
[+] Listening...
[+] Data sent to client: Hello World!
[+] Closing the connection
booboo@DESKTOP-NHD0C8T:~$
```

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;

}
```

```
booboo@DESKTOP-NHD0C8T:~$ gcc n2c.c -o n2c
booboo@DESKTOP-NHD0C8T:~$ ./n2c
[+] Client socket created successfully
[+] Connected to server
[+] Data received from server: Hello World!
[+] Closing the connection
booboo@DESKTOP-NHD0C8T:~$
```

Termwork3

Routing Protocols: Distance vector routing is a class of routing protocols used in computer networks to determine the best path for data packets to travel from the source to the destination.

Vector Table Entries: Entries for each destination router contain preferred routes and estimated hop distances, enabling rapid routing adjustments

Program with sample input/output

```
#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;
      }
   }
}
```

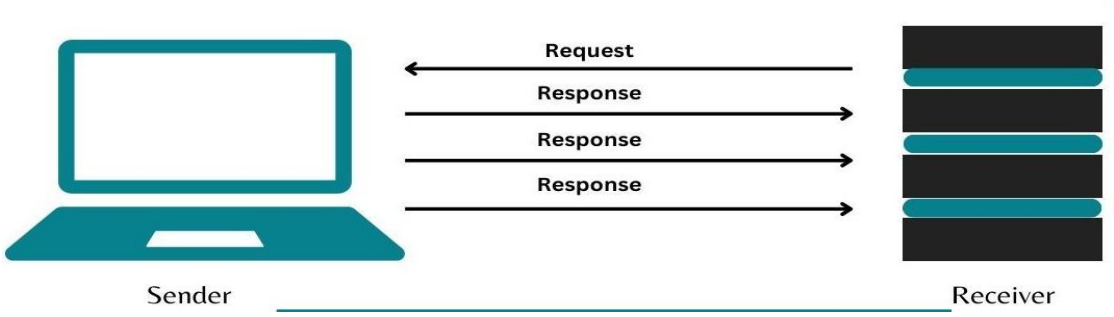
```
void display(int r) {
   int i;
   printf("\nThe routing table for node %d is as follows",
r);
   for (i = 1; i <= no; i++) {
      if (router[r].a[i][2] == 999)
         printf("\n%d \t no link \t no hop", router[r].a[i][1]);
      else
         printf("\n%d \t %d \t %d", router[r].a[i][1],
router[r].a[i][2], router[r].a[i][3]);
   }
}

void dv_algo(int r) {
   int i, j, z;
   for (i = 1; i <= no; i++) {
      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;

}
```

Refer output from journal ☺

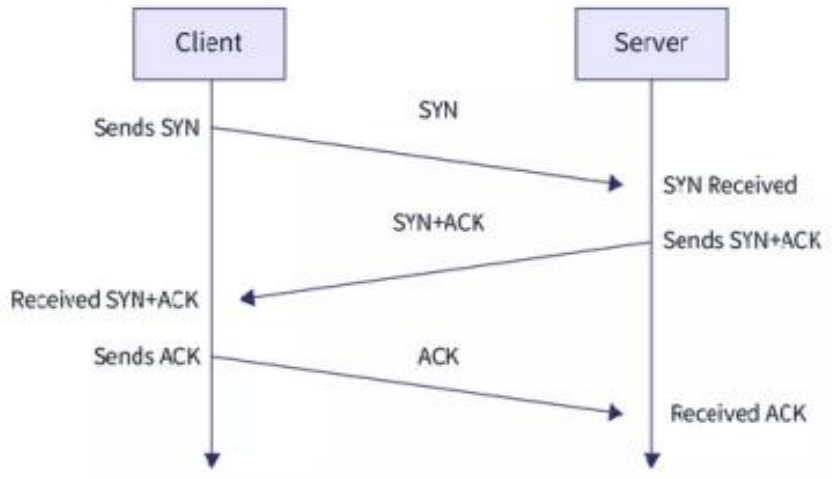
Termwork 4

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 go to edit in the menu bar and select find packet option or just press ***CTRL+F***.
4. In 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 indicating 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 UDP packet.
7. Go to statistics: Generate I/O Graph, Flow Graph and study and analyze both the graphs

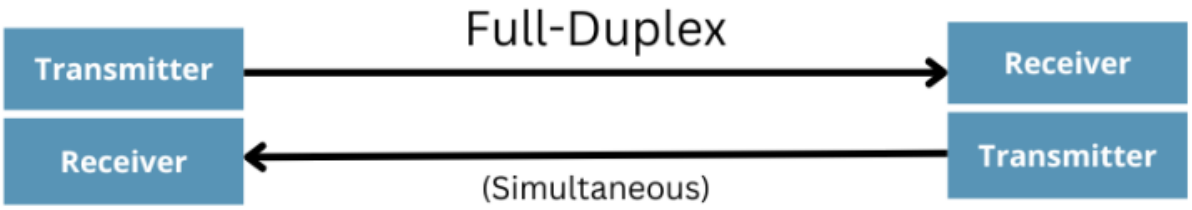


Termwork 5

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.
4. Type the following in, apply a filter column and hit-enter: **tcp.flags.fin==1 and tcp.flags.ack ==1**
5. Select any one of these listed packets, right-click and hover on conversation filter and select TCP.
6. Once done analyze the TCP Packets.
7. Go to statistics: Generate I/O Graph, Flow Graph and study and analyze both the graphs
8. Observe TCP 3-way Handshake mechanism, data transfer and connection termination through TCP

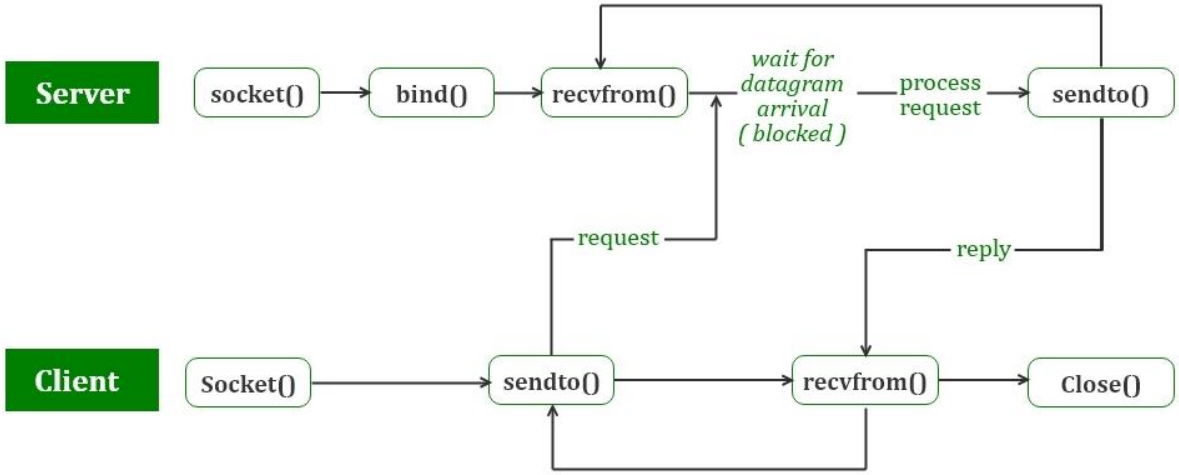


Termwork 6



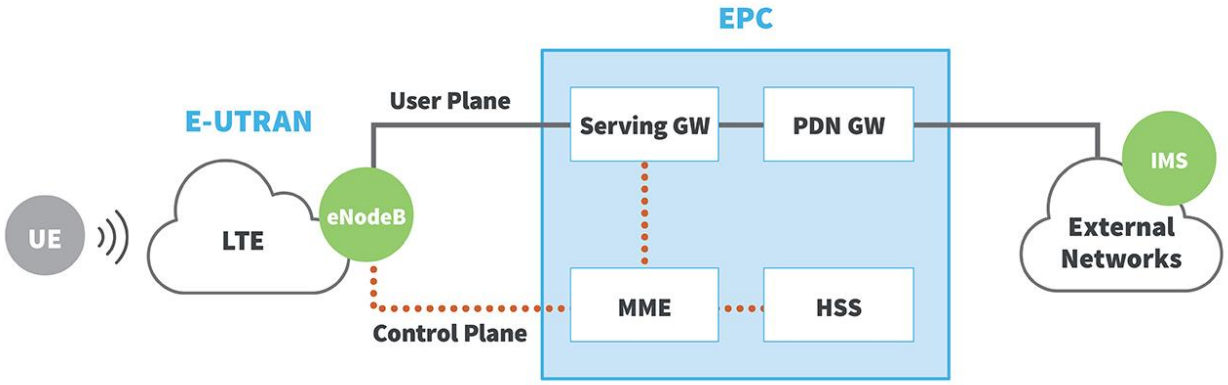
Step 1 : Open UBUNTU and locate and open **ns-allinone-3.28** folder on Desktop.
Step 2 : Go to ns-3.28 folder and open **examples->tutorial->first.cc**
Step 3 : In first.cc , include the following code. (Before – "Simulator::Run ();")
#include "ns3/netanim-module.h"
AnimationInterface anim("first, xml");
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("first.tr"));
pointToPoint.EnablePcapAll("first");
Step 4 : Copy first.cc and paste it in **ns-3.28->scratch** folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratchsimulator.cc and other files can be deleted.
Step 5 : Open terminal and change working directory to Desktop by **cd Desktop** and type following commands to go to location where scratch executable file is located i.e. scratch folder.
Step 6 : cd ns-allinone-3.28/ns-3.28
Step 7 : Run the first.cc by entering following command.
./waf –run scratch/first
Step 8 : Once build is successful, return to ns-allinone-3.28 folder with **cd ../** and enter into netanim3.108 with **cd netanim-3.108**
Step 9 : Now to see the animation, we have to open NetAnim software. So open by entering **./NetAnim** on terminal.
Step 10 : In NetAnim, open first.xml by clicking on open XML trace file icon.
Step 11 : Click on run option/icon to see the animation. To see the packet transfer, open Packets Tab.

Termwork 7



Step 1 : Open UBUNTU and locate and open **ns-allinone-3.28** folder on Desktop.
Step 2 : Go to ns-3.28 folder and open **examples->tutorial->first.cc**
Step 3 : In first.cc , include the following code. (Before – "Simulator::Run ();")
#include "ns3/netanim-module.h"
AnimationInterface anim("second, xml");
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("second.tr"));
pointToPoint.EnablePcapAll("second");
Step 4 : Copy first.cc and paste it in **ns-3.28->scratch** folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratchsimulator.cc and other files can be deleted.
Step 5 : Open terminal and change working directory to Desktop by **cd Desktop** and type following commands to go to location where scratch executable file is located i.e. scratch folder.
Step 6 : cd ns-allinone-3.28/ns-3.28
Step 7 : Run the first.cc by entering following command.
./waf –run scratch/second
Step 8 : Once build is successful, return to ns-allinone-3.28 folder with **cd ../** and enter into netanim3.108 with **cd netanim-3.108**
Step 9 : Now to see the animation, we have to open NetAnim software. So open by entering **./NetAnim** on terminal.
Step 10 : In NetAnim, open first.xml by clicking on open XML trace file icon.
Step 11 : Click on run option/icon to see the animation. To see the packet transfer, open Packets Tab.

Termwork8

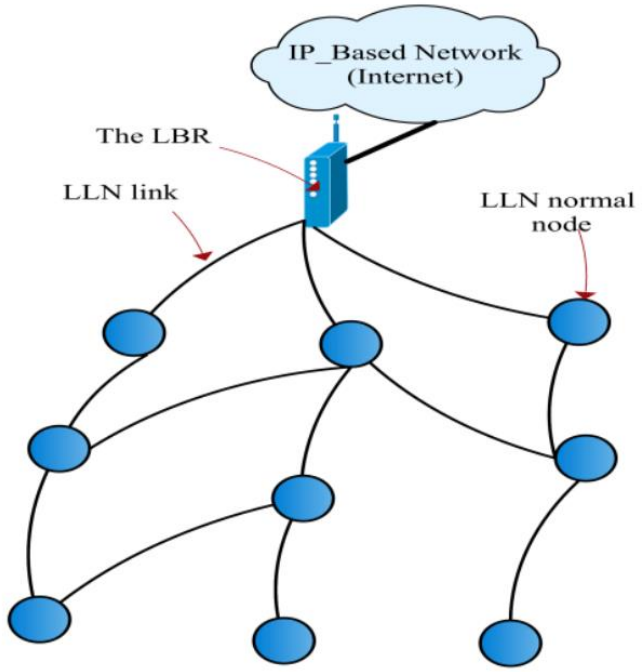


Step 1 : Open UBUNTU and locate and open **ns-allinone-3.28** folder on Desktop.
Step 2 : Go to ns-3.28 folder and open **examples->tutorial->first.cc**
Step 3 : In first.cc , include the following code. (Before – "Simulator::Run ();")
#include "ns3/netanim-module.h"
AnimationInterface anim("third, xml");
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll(ascii.CreateFileStream("third.tr"));
pointToPoint.EnablePcapAll("third");
Step 4 : Copy first.cc and paste it in **ns-3.28->scratch** folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratchsimulator.cc and other files can be deleted.
Step 5 : Open terminal and change working directory to Desktop by **cd Desktop** and type following commands to go to location where scratch executable file is located i.e. scratch folder.
Step 6 : cd ns-allinone-3.28/ns-3.28
Step 7 : Run the first.cc by entering following command.
./waf –run scratch/third
Step 8 : Once build is successful, return to ns-allinone-3.28 folder with **cd ../** and enter into netanim3.108 with **cd netanim-3.108**
Step 9 : Now to see the animation, we have to open NetAnim software. So open by entering **./NetAnim** on terminal.
Step 10 : In NetAnim, open first.xml by clicking on open XML trace file icon.
Step 11 : Click on run option/icon to see the animation. To see the packet transfer, open Packets Tab.

Termwork9

Go to the Location contiki-ng/tools/cooja/ with command
cd contiki-ng/tools.cooja
Run the cooja simulator with **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



Termwork10

Go to the Location contiki-ng/tools/cooja/ with command
cd contiki-ng/tools.cooja
Run the cooja simulator with **ant run**

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