Network Message Bus

Submitted to: Prof K Hari Babu
Course: Network Programming
Assignment No: 1
Submitted By: Chetan Arora (2016A8PS0346P), Neethu Mariya Joy(2016A7PS0119P)

Message queues are a mechanism for communicating between two different processes in the same operating system. The kernel manages this queue and processes that have a handle to the queue can read and modify the queue according to the permissions provided. Network Message Bus is a similar module for communication between processes. However, the processes communicating could reside in different operating systems. This document underlines the design of the Network Message Bus(NMB) module.

When a client in a system (say A in system 1, Fig 1) needs to communicate with a client in another system (say D in system 2, Fig 1), Network Message Bus is used. This is achieved through communication between client A, local server B, local server C and client D.

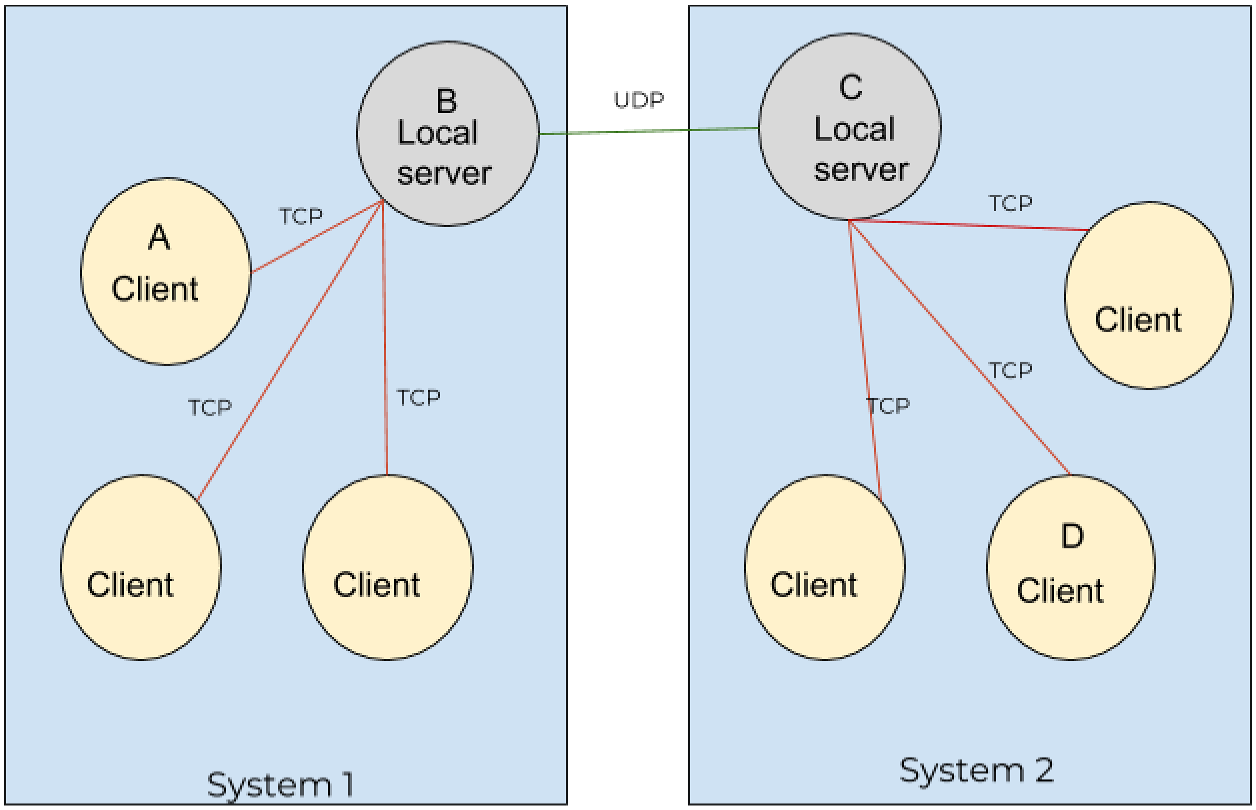


Fig 1. Communication channels between clients in two different operating systems.

The communication protocols used are as below:

1. Client A to local server B using TCP
2. Local server B to local server C through UDP
3. Local server C to client D through TCP

The interface provided to the client resembles that of a message queue. The following functions are provided:

- Msgget_nmb similar to msgget
- Msgsnd_nmb similar to msgsnd
- Msgrcv_nmb similar to msgrcv
- Msgrem_nmb for closing the queue

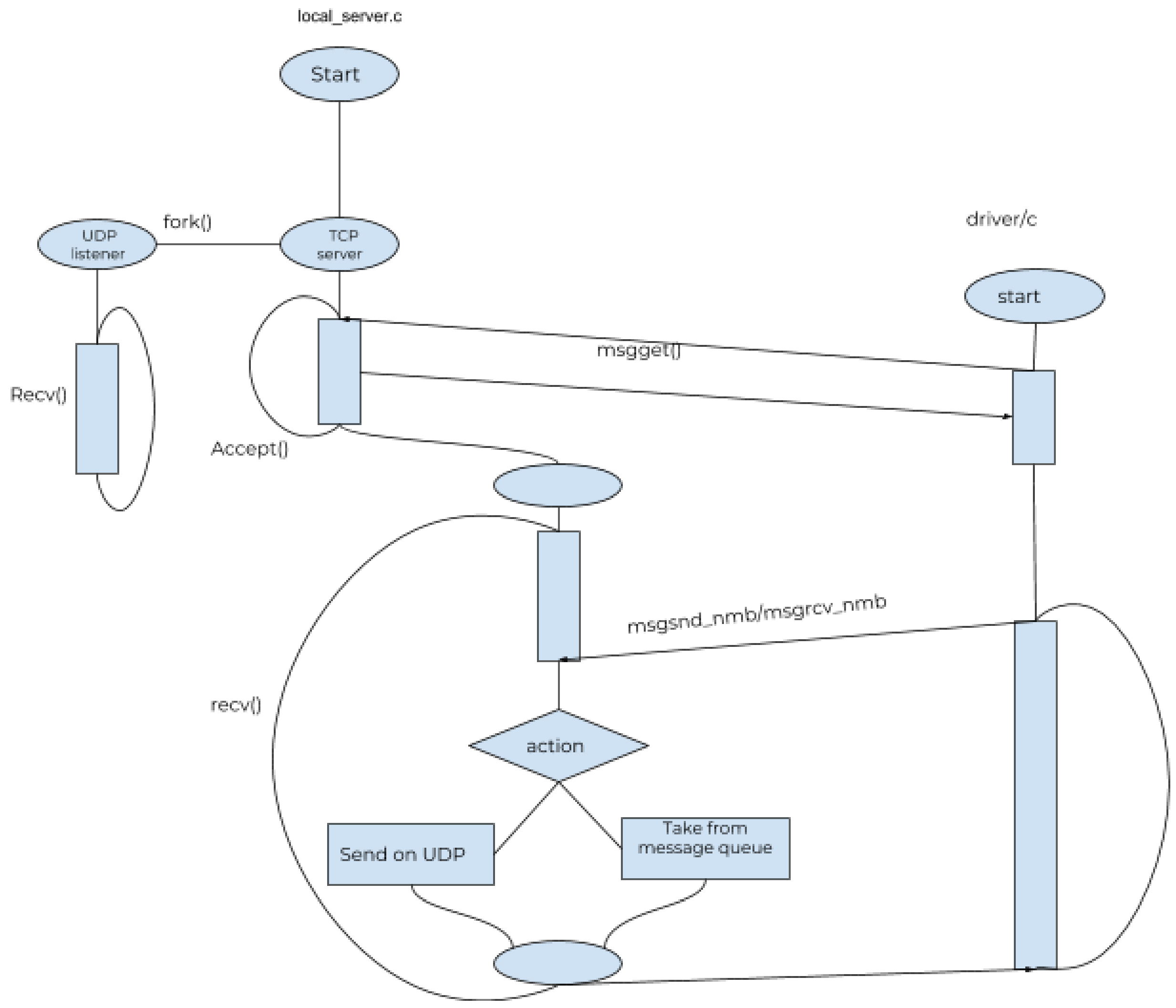


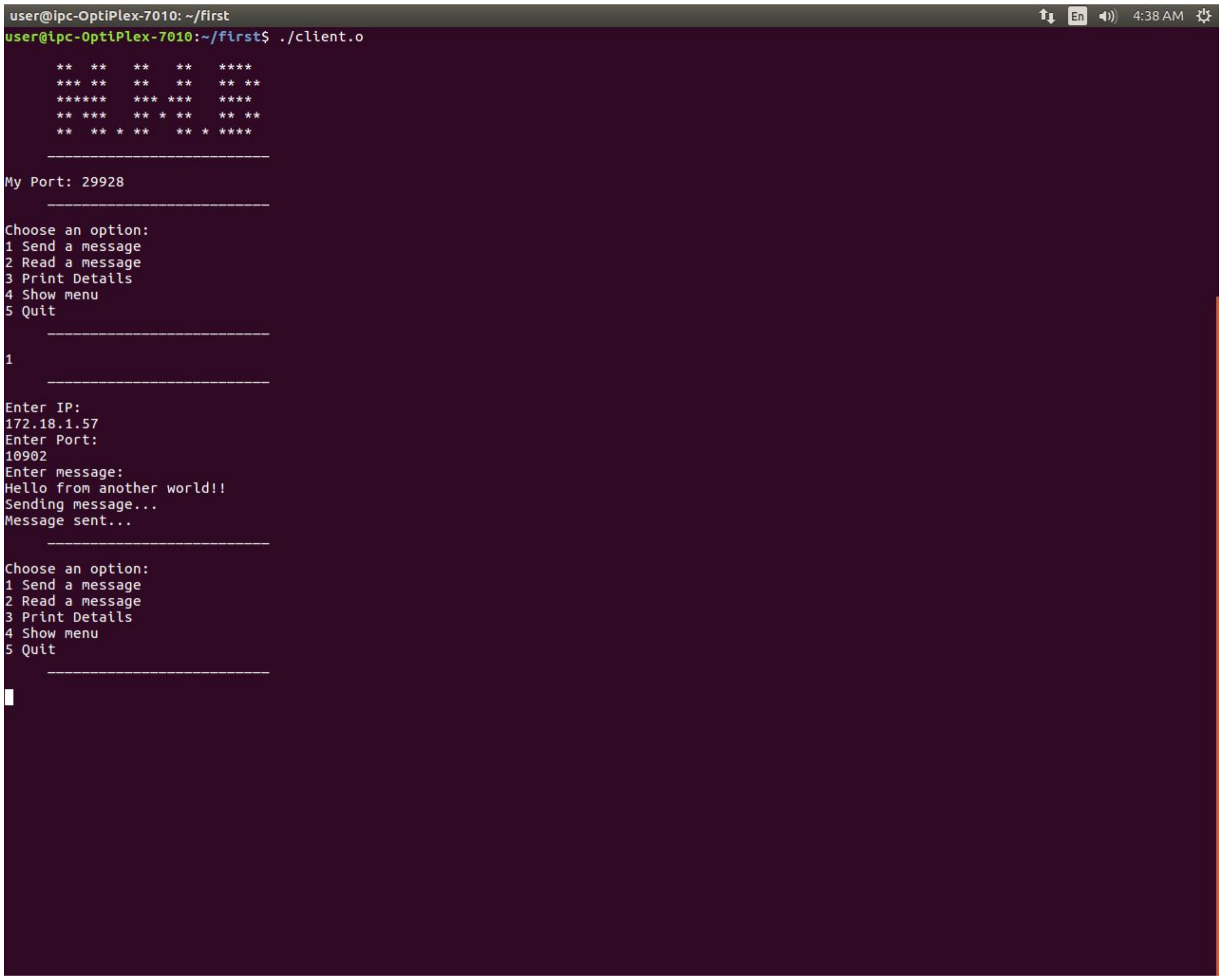
Fig 2: Flow chart of the system

The local server works in the following manner:

- On start, the local server initialises a message queue, say Q.
- Then the process creates a child. The child works as a UDP listener and the parent works as a TCP listener.
- When the UDP listener receives a message , it inserts it into Q.
- If a local client connects through TCP, the parent creates a new child to handle the tcp connection. This child listens through recv( ) function.
- When a client sends a message, the NMB interface sends an action variable along with the message depending upon the function called.
- On receiving message from the nmb interface, the tcp connection acts according to the action variable.
- If the action variable is MSG_SND, the tcp child created sends the message through UDP to the desired system. Initially, the message type in set to be in format of 4 bytes of IP_ADDR and 2 bytes for PORT. Before sending the UDP sets this field to be the port address of the destination client.
- If the action variable is MSG_RCV, the tcp child checks Q to find any message that matches the client port.

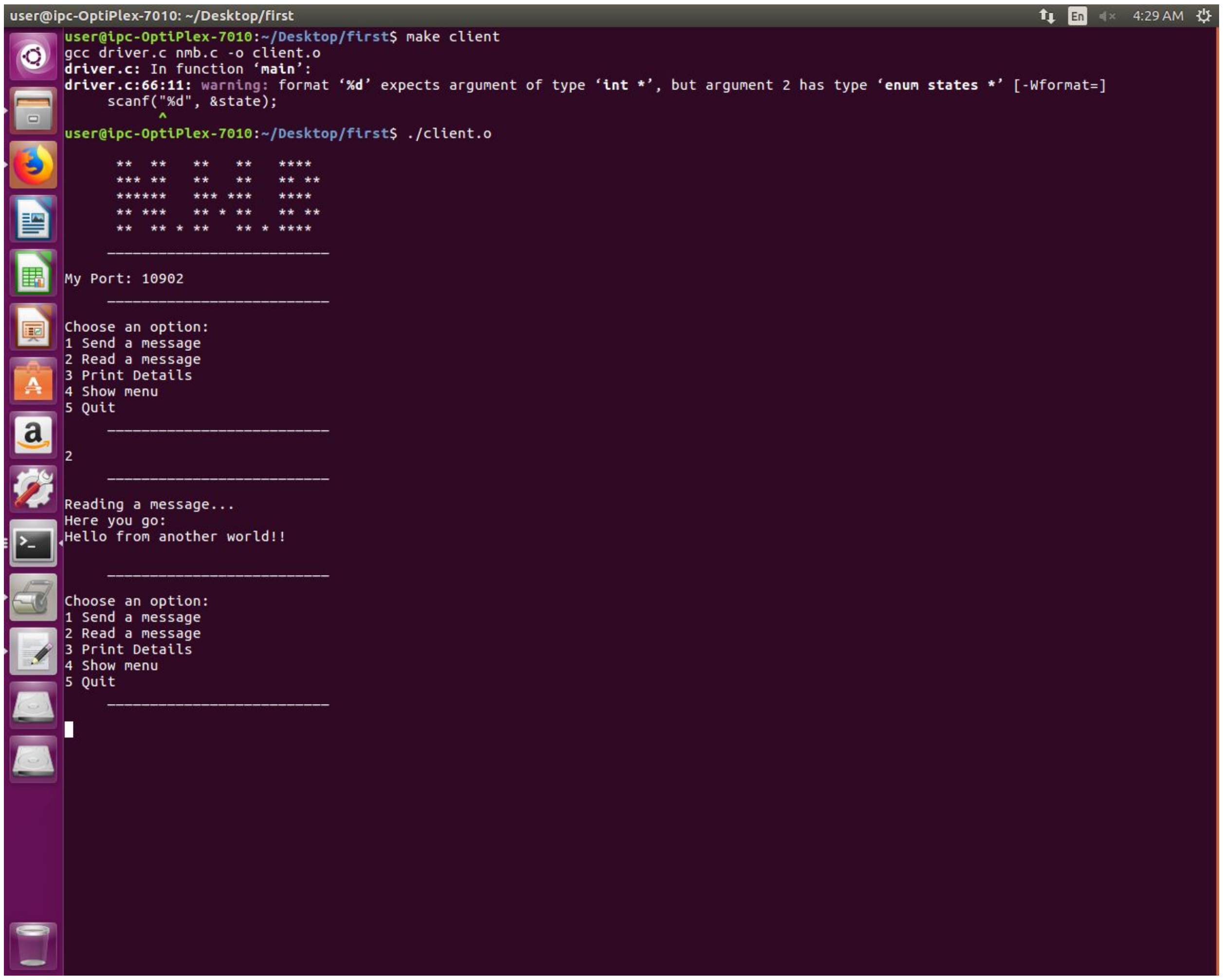
Usage

To use the client-server system, run:
```
cd path/to/file
make server
make client
./server.o
````
And in another tab, run :
```
cd path/to/file
./client.o
```
Screenshots of the communication between two clients running on different pcs are shown below.



```
user@ipc-OptiPlex-7010: ~/first
user@ipc-OptiPlex-7010:~/first$ ./client.o

      **  **   **   **   ****
      *** **   **   **   ** **
      ******   *** ***   ****
      ** ***   ** * **   ** **
      **  ** * **   ** * ****
     __________________________

My Port: 29928
     __________________________

Choose an option:
1 Send a message
2 Read a message
3 Print Details
4 Show menu
5 Quit
     __________________________

1
     __________________________

Enter IP:
172.18.1.57
Enter Port:
10902
Enter message:
Hello from another world!!
Sending message...
Message sent...
     __________________________

Choose an option:
1 Send a message
2 Read a message
3 Print Details
4 Show menu
5 Quit
     __________________________

```

Fig 3: Sender client



```
user@ipc-OptiPlex-7010:~/Desktop/first$ make client
gcc driver.c nmb.c -o client.o
driver.c: In function ‘main’:
driver.c:66:11: warning: format ‘%d’ expects argument of type ‘int *’, but argument 2 has type ‘enum states *’ [-Wformat=]
     scanf("%d", &state);
           ^
user@ipc-OptiPlex-7010:~/Desktop/first$ ./client.o

      **  **   **   **   ****
      *** **   **   **   ** **
      ******   *** ***   ****
      ** ***   ** * **   ** **
      **  ** * **   ** * ****

     __________________________

My Port: 10902
     __________________________

Choose an option:
1 Send a message
2 Read a message
3 Print Details
4 Show menu
5 Quit
     __________________________

2
     __________________________

Reading a message...
Here you go:
Hello from another world!!

     __________________________

Choose an option:
1 Send a message
2 Read a message
3 Print Details
4 Show menu
5 Quit
     __________________________

```

Fig 4: receiver client