PROJECT

“MESTRA”

……a message transfer application

DEVELOPED BY.

ANKIT KUMAR GUPTA
B.E. (CSE) 7th Semester
SG13302

MESTRA

MESTRA is a Linux console application developed for transferring messages from on system to the another system. It works on the basis of Berkeley Sockets. It was first introduced in the 4.2BSD systems. The application is based on the client server communication. A message receiving system will be the server and the message sending system will be the client. The application can be downloaded from <github.com/gepslyn/mestra>. It can be deployed to the system by the executing the MESTRA file.

LINUX

GNU/Linux has taken the world of computers by storm. At one time, personal computer users were forced to choose among proprietary operating environments and applications. Users had no way of fixing or improving these programs, could not look “under the hood,” and were often forced to accept restrictive licenses. GNU/Linux and other open source systems have changed that—now PC users, administrators, and developers can choose a free operating environment complete with tools, applications, and full source code.

GNU COMPILER COLLECTION (GCC)

The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Java, Ada, and Go, as well as libraries for these languages (libstdc++, libgcj,...). GCC was originally written as the compiler for the GNU operating system. The GNU system was developed to be 100% free software, free in the sense that it respects the user's freedom.

GNU DEBUGGER

GDB can do four main kinds of things (plus other things in support of these) to help you catch bugs in the act:

- Start your program, specifying anything that might affect its behavior.
- Make your program stop on specified conditions.
- Examine what has happened, when your program has stopped.
- Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another.

INTERPROCESS COMMUNICATION

Inter-process commination allows two related or unrelated processes to transfer data to each other. There are two types of processes; Independent processes which is executed without interrupted any other process, and Cooperative processes, which communicates to other process for the execution.

Advantages:

- Computation Speed-up
- Information Sharing
- Modularity
- Convenience

Types of IPC

- Shared Memory
- Mapped Memory
- Pipes
- FIFOs
- Sockets

The whole project is based on the sockets.

Sockets have two standards:

- Berkeley Sockets
- System V Transport Layer Interface

Berkeley Sockets are widely used in the communication system. Even all the internet and its services like ftp, telnet, servers, World Wide Web work on the basis of Berkeley Sockets.

SOCKETS

Outline

- *APIs – Motivation*
- *Sockets*
- *C Socket APIs*

What is an API?

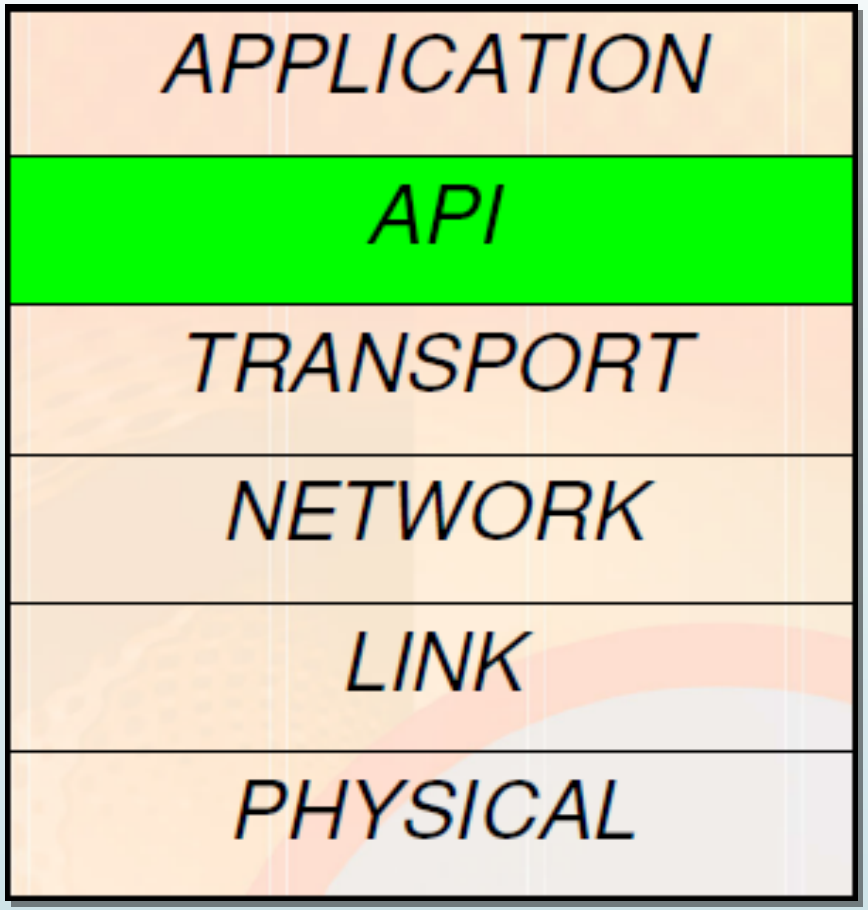
- *API – stands for Application Programming Interface.*

  Interface to what? – In our case, it is an interface to use the network.

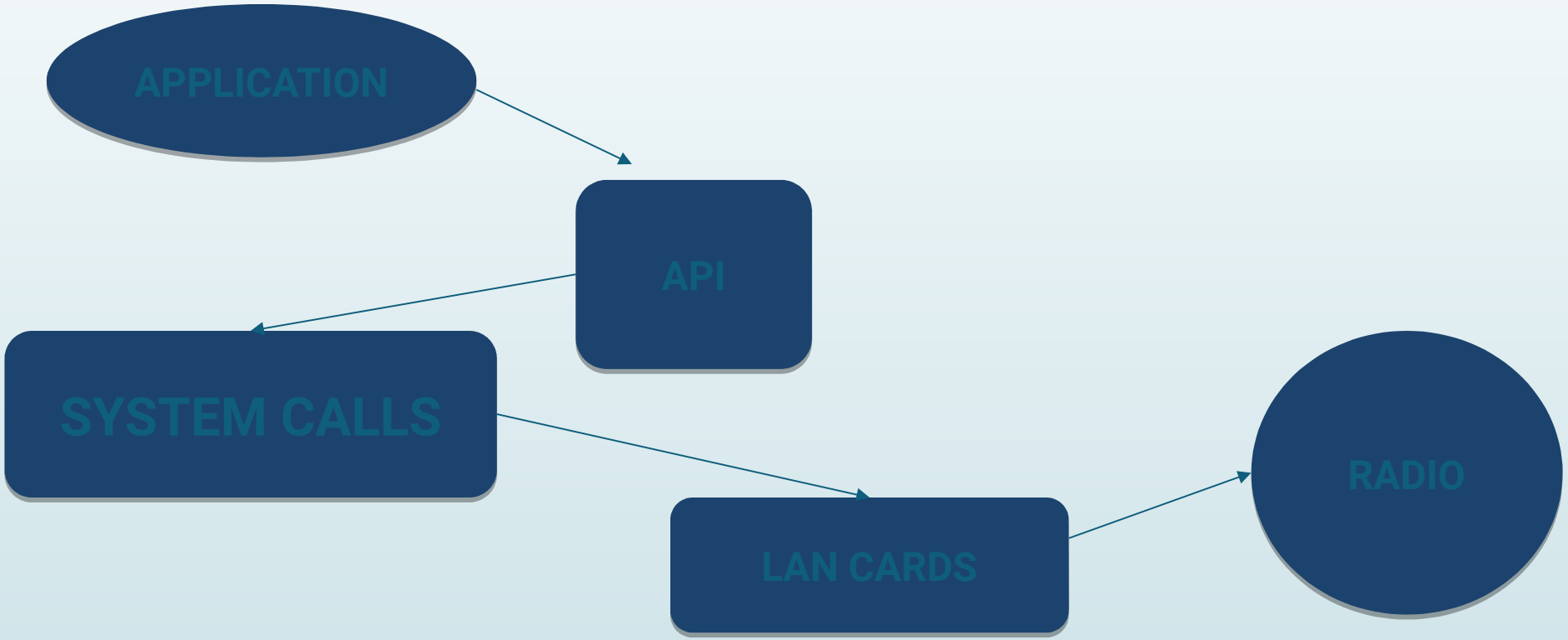
- *A connection to the transport layer.*

Need for API

- *One Word - Layering*
- *Functions at transport layer and below very complex.*
- *E.g. Imagine having to worry about errors on the wireless link and signals to be sent on the radio.*
- *Helps in code reuse.*


APPLICATION
API
TRANSPORT
NETWORK
LINK
PHYSICAL

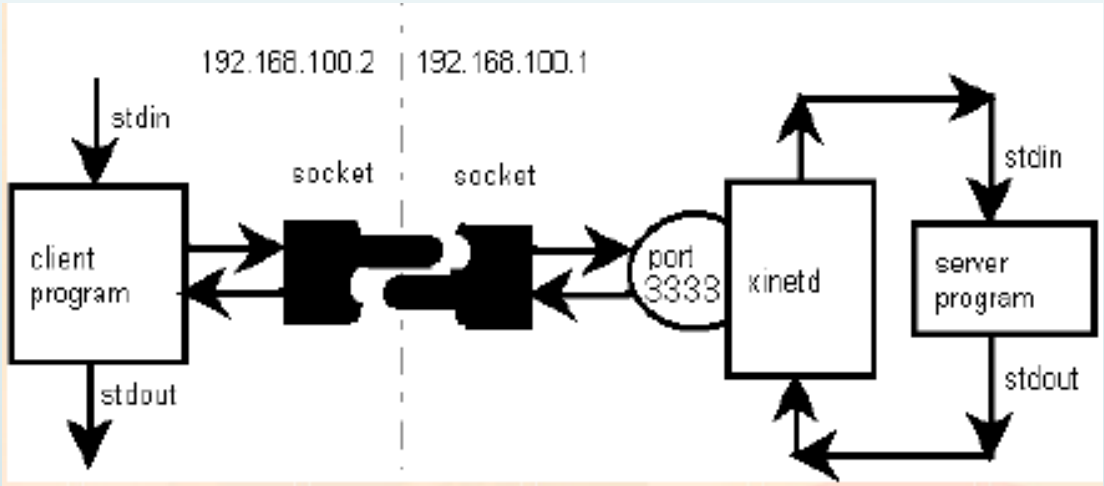
Layering Diagramatically



What is Socket?

- *It is an abstraction that is provided to an application programmer to send or receive data to another process.*
- *Data can be sent to or received from another process running on the same machine or a different machine.*
- *In short, it is an end point of a data connection.*

Socket – An Abstraction



Ports

- Sending process must identify the receiver
  - Address of the receiving end host
  - Plus identifier (port) that specifies the receiving process
- Receiving host
  - Destination address uniquely identifies the host
- Receiving process
  - Host may be running many different processes
- Destination port uniquely identifies the socket
  - Port number is a 16-bit quantity

Port Usage

- Popular applications have “well-known ports”
  - E.g., port 80 for Web and port 25 for e-mail
  - Well-known ports listed at http://www.iana.org
- Well-known vs. ephemeral ports
  - Server has a well-known port (e.g., port 80)
- By convention, between 0 and 1023; privileged
  - Client gets an unused “ephemeral” (i.e., temporary) port
  - By convention, between 1024 and 65535
- Flow identification
  - The two IP addresses plus the two port numbers
    - Sometimes called the “four-tuple”
  - Underlying transport protocol (e.g., TCP or UDP)
  - The “five-tuple”

Ports (Main Points)

- *Not related to the physical architecture of the computer.*
- *Just a number maintained by the operating system to identify the end point of a connection.*

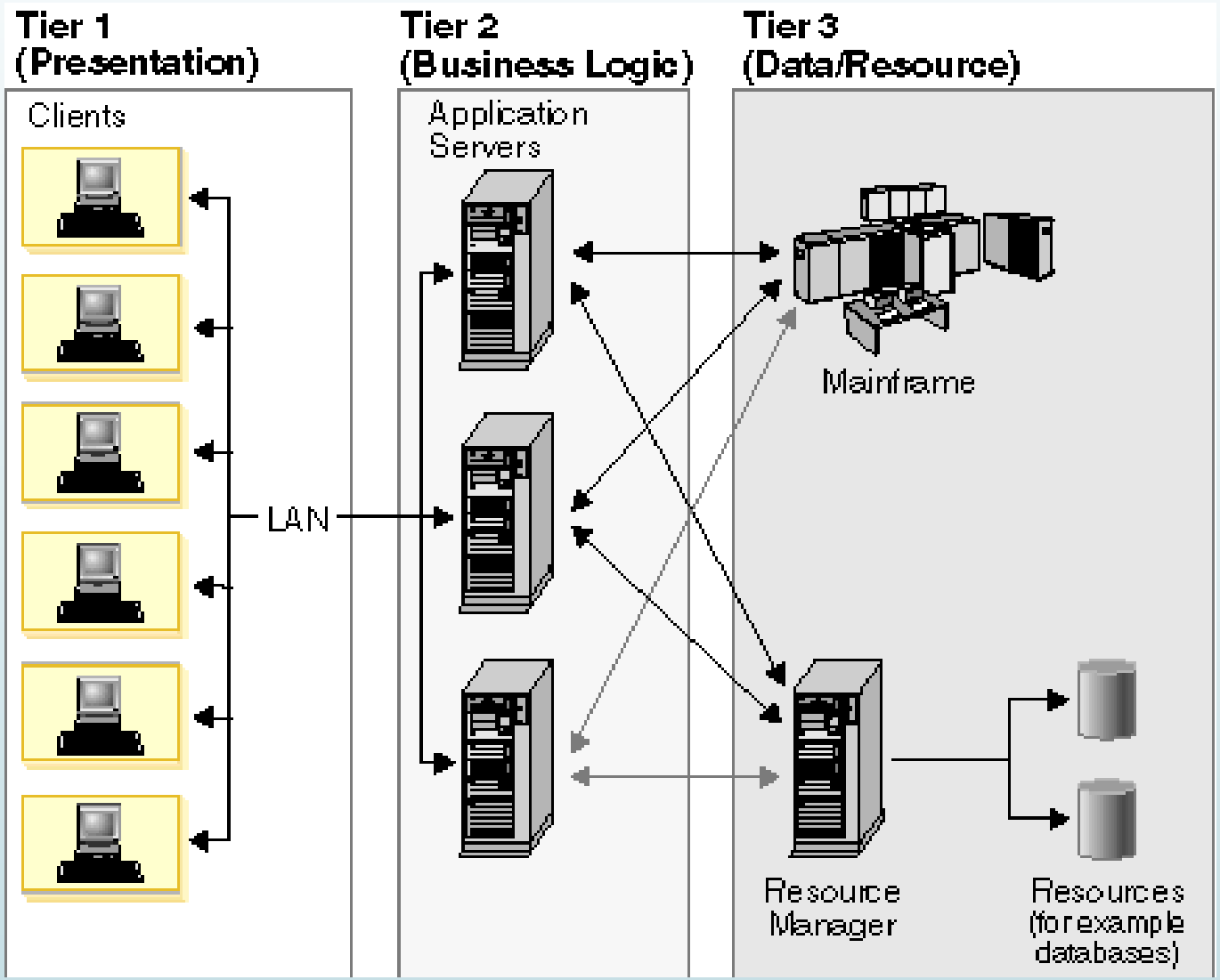
TCP (stream) sockets

- *Also known as SOCK_STREAM*
- *TCP is a connection-oriented byte-stream Protocol*
  - *During data packet. transmission, no packetization and address required by application.*
  - *Formatting has to be provided by application.*
  - *Two or more successive data sends on the pipe connected to socket may be combined together by TCP in a single packet.*

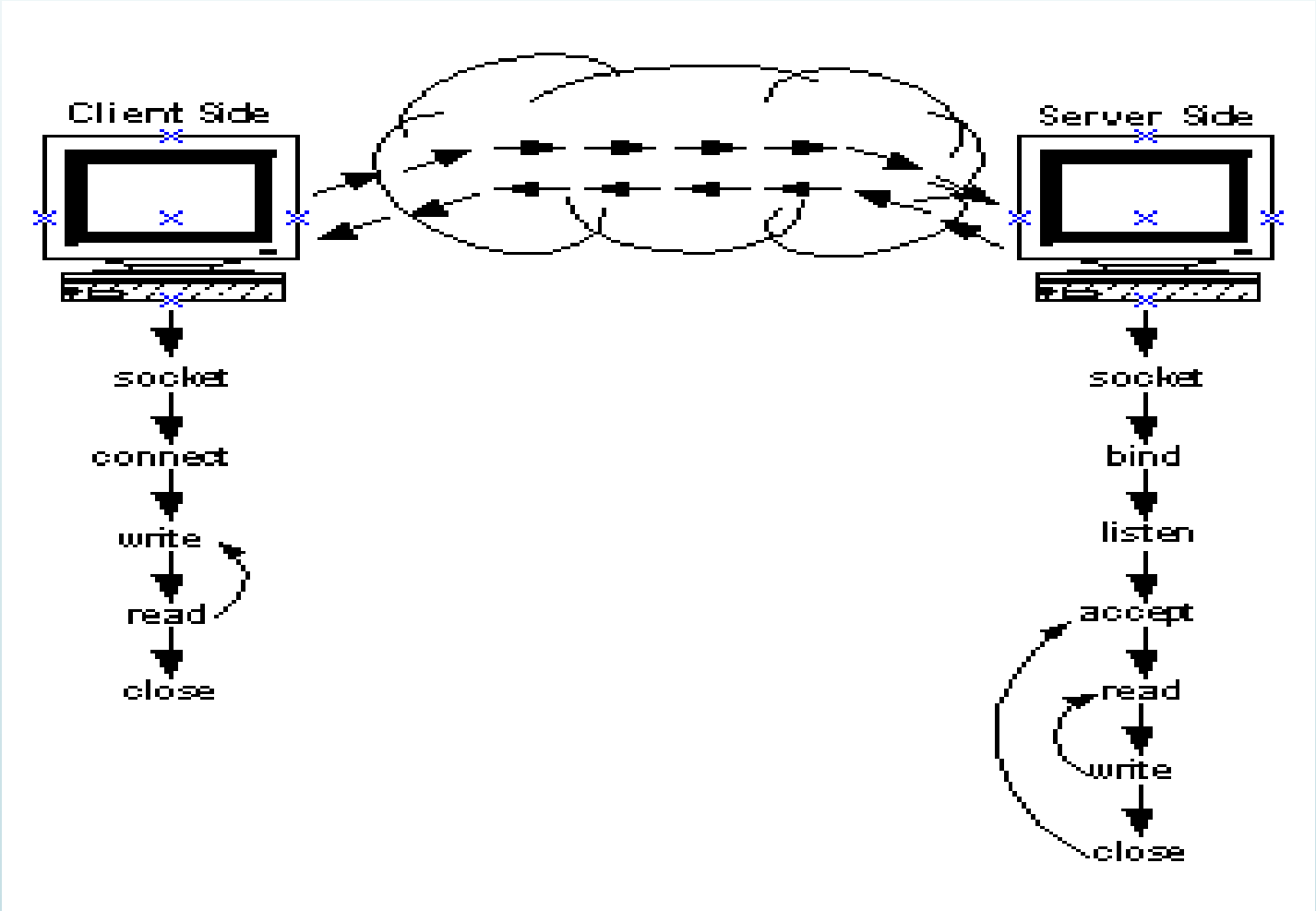
UDP (datagram) sockets

- *Also known as SOCK_DGRAM*
- *UDP is connectionless and packet-oriented.*
  - *Info sent in packet format as needed by app.*
  - *Every packet requires address information.*
  - *Lightweight, no connection required.*
  - *Overhead of adding destination address with each packet at the application layer. (Can be eliminated by "connecting")*
- *Distinction in the way these sockets are used by different hosts – client and server.*

Client – Server Architecture



Flow in client-server model



MESTRA

……..a console message transfer application

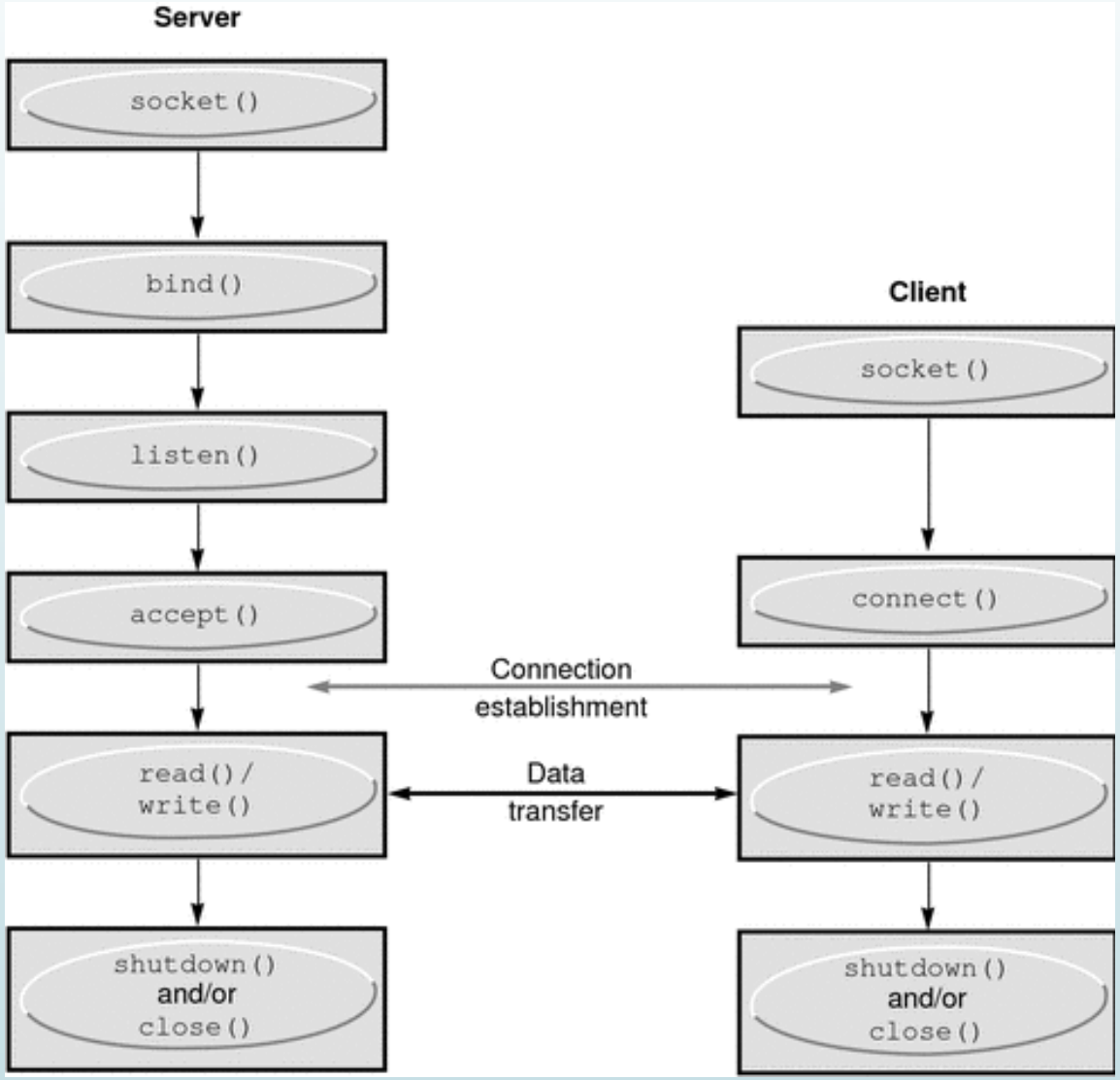
INTRODUCTION

MESTRA is a Linux console application developed for transferring messages from on system to the another system. It works on the basis of Berkeley Sockets. It was first introduced in the 4.2BSD systems. The application is based on the client server communication. A message receiving system will be the server and the message sending system will be the client. The application can be downloaded from <github.com/gepslyn/tranif>. It can be deployed to the system by the executing the MESTRA file.

INTERNAL ALGORITHM

- For the transfer of messages the activity begins with the receiving system.
- The receiving system will create a socket having the domain AF_INET. It is an internet domain socket. The socket will be of the type SOCK_STREAM. First of all connection is established then the data will be transferred.
- Now the server system will create an address structure. This structure has three fields; (a) sin_family, which defines the domain, is AF_INET, (b) sin_port, which defines the port, is **9784**, and (c) sin_address which takes the address, is the local system address.
- Now the address will be bound with the socket to make it available for the communication. It will be now visible by the client.
- Server system creates a queue in which the connection requests will wait. It is done with listen () system call.
- Now the server is waiting for the client's request for connection.

INTERNAL ALGORITHM



INTERNAL ALGORITHM

- At the client side, its own socket will be created of the domain AF_INET. It is an internet domain socket. The protocol will be zero. It is of the type **SOCK_STREAM.**
- The address structure for the client socket will be created. The same port will be used for the client also.
- Now the client will request for the communication with connect system call.
- The client will write the data to its socket.
- Socket is connected through the port.
- Port is associated with the network.
- The data is transferred to the server's socket.
- Now the server receive the data through its socket and it will be stored in the file as the name is specified by the server system user.

DESIGN

The complete software has been written in C. It has been compiled with gcc compiler. The complete software is designed into three file. On file contains the code for management of the flow control throughout the program. It is the main C file. Rest of the codes is developed into two files. One of them contains, the code for the server program and another contains the client program.

DESIGN

- **Main Program**

This is the control program. It's main task is to get the choice of the user to initiate other Local or Internet server/client program. It transfers the program control to the other functions

DESIGN

Server Program

- Socket for the server will be created.
- Address structure for the server socket will be defined.
- The port **9784** will be used for the communication.
- Binding of the address to the socket will be done.
- A connection queue will be created.
- If there is a connection request, it will be accepted by the server.
- Then the client socket file descriptor will be received. By this file descriptor, the data will be received.
- Now the client socket's file descriptor will the closed.
- Now, the data will be shown to the receiver.
- If the communication has been done, the server socket will be closed.

DESIGN

Client Program

- The socket for the client side system will be created.
- Now the address for the client socket will be defined. The Port Number for client side also has to be **9784**.
- Now, the will request to the server for connection.
- If the connection has been established, then client will write the data of buffer to the client side socket.
- The data will be taken from the user. This data now will be sent to the server side socket through the network.

FUTURE SCOPE OF THE PROJECT

- My previous developed product “ENDECS_v_1.1” (can be downloaded from github)will be integrated will this message transfer application. This product will provide encryption and compression.
- Basically ENDECS is a cryptography tool, which is used to compress and encrypt the original information into cipher text file to send the data from sender to the receiver side. At the receiver side the cipher text and the encryption key is used to get the plain text. This process is used to transfer the sensitive information from any medium (wireless or wired network). ENDECS is a console application which can be operated only in the terminal mode of operating system. This application is basically designed for GNU Linux operating systems. It is very fast in order to generate the cipher and plain text

How to use?

- Download the binary file of the application from the link < https://github.com/gepslyn/mestra>
- You will get the application for x-86 system.
- Make the binary file executable. Execute the command.:

  chmod +x mestra
- Copy and paste the application to the /bin directory of the system, as

  cp mestra /bin.

How to use?

- Run the command *mestra*, you will get the following information.:

```
gepslyn:/home/ankit/mestra # cp mestra /bin
gepslyn:/home/ankit/mestra # mestra

                                        MESTRA
                    Copyright (C) 2016 version.1.0 <ankitgupta.cs40@gmail.com>

gepslyn:/home/ankit/mestra # _
```

How to use?

- For any help run the command *mestra –help* or *mestra -h*.

```
gepslyn:/home/ankit/mestra # mestra --help

                                        MESTRA
                Copyright (C) 2016 version.1.0 <ankitgupta.cs40@gmail.com>

Usage of mestra
Syntax.:
mestra -option [client/server IP ADDRESS or local file name for socket.............]

Options.:

                -h      --help                  for help
                -s      --server                for internet server program
                -c      --client                for internet client program
                -r        --local_server        for local server program
                -t        --local_client        for local client program
gepslyn:/home/ankit/mestra # _
```

How to use?

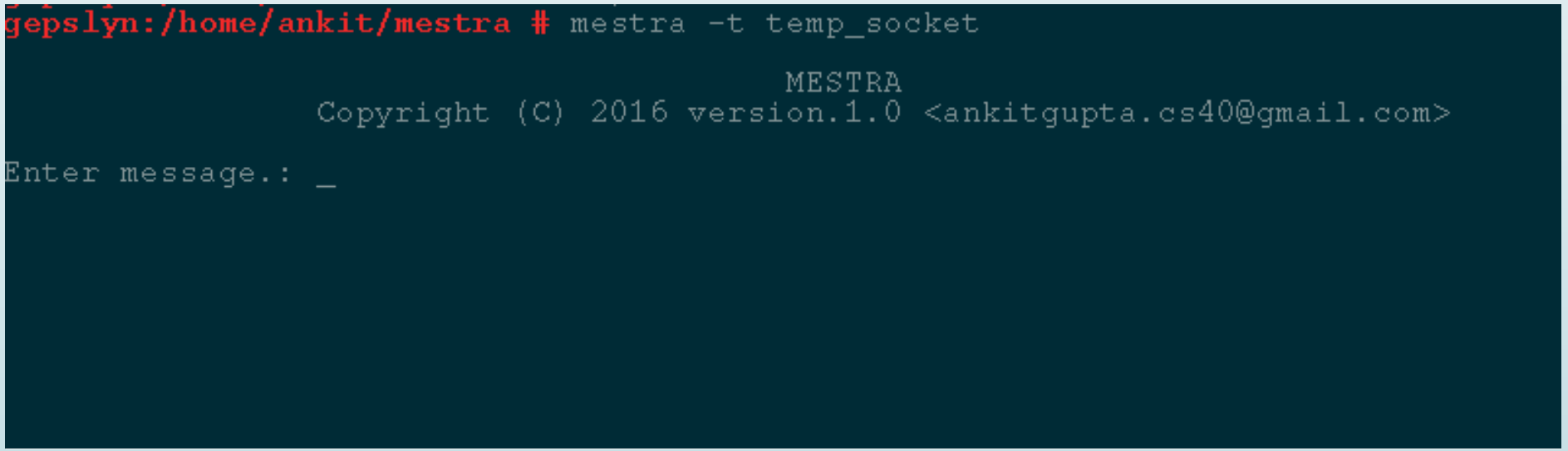
LOCAL COMMUNICATION

- For the local communication use local sockets (local server and local client).
- Use -r option for server and -t option for client system.
- To create a local socket for the server use the command-line

  mestra -r temp_socket

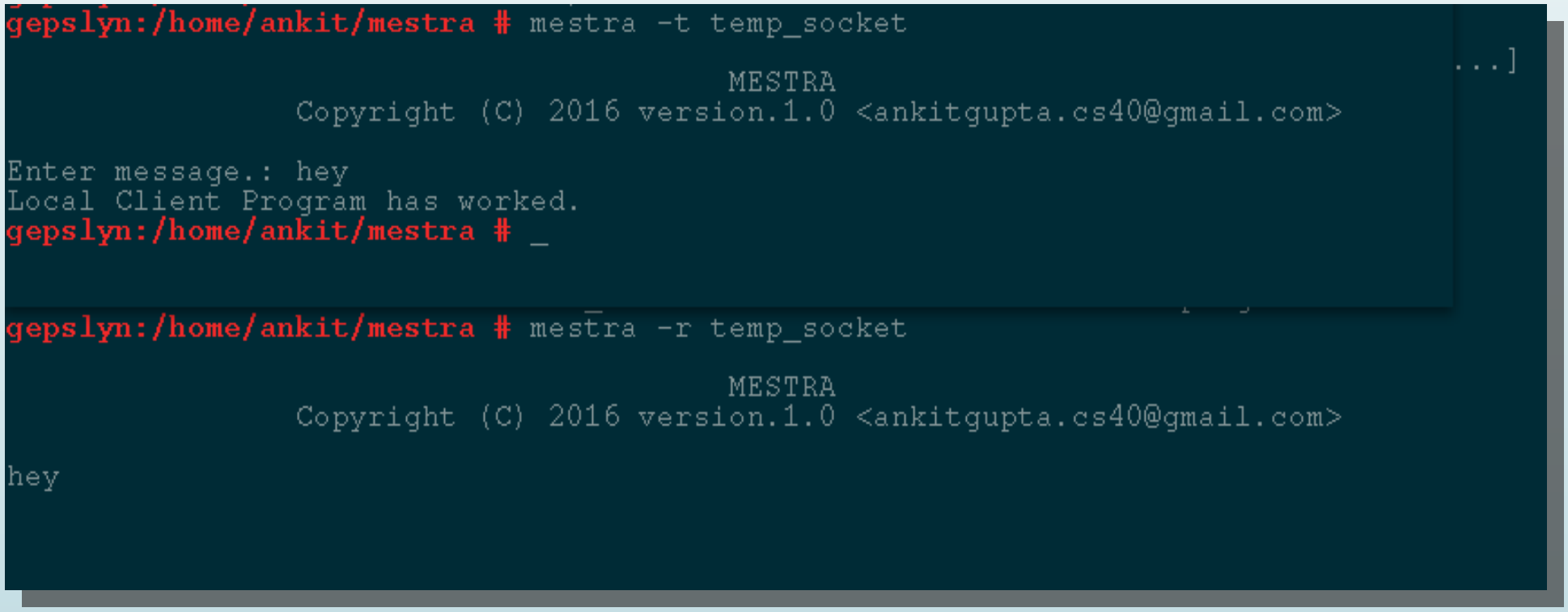
How to use?

- Now the receiver is waiting for the messages from the client.
- Open another terminal and run the client program for sending messages.
- Run the command.
- *mestra -t temp_socket*



How to use?

- You will be asked to enter the messages to send. Enter the messages and *enter*.
- Message will be sent.



How to use?

- To close the client socket send a message *quit* to the server.

```
gepslyn:/home/ankit/mestra # mestra -t temp_socket

                                        MESTRA
                Copyright (C) 2016 version.1.0 <ankitgupta.cs40@gmail.com>

Enter message.: quit
Local Client Program has worked.
gepslyn:/home/ankit/mestra # _
gepslyn:/home/ankit/mestra # mestra -r temp_socket

                                        MESTRA
                Copyright (C) 2016 version.1.0 <ankitgupta.cs40@gmail.com>

hey
Local server Program has worked.
gepslyn:/home/ankit/mestra # _
```

How to use?

INTERNET COMMUNICATION

- For the internet communication use internet sockets (internet server and internet client).
- Use -s option for server and -c option for client system.
- To create an internet socket for the server use the command-line

mestra -s [ip address of the server]

```
Ranjan_PC:/home/Ranjan # mestra

                                    MESTRA
                Copyright (C) 2016 version.1.0 <ankitgupta.cs40@gmail.com>

Ranjan_PC:/home/Ranjan # mestra -s 172.16.3.74

                                    MESTRA
                Copyright (C) 2016 version.1.0 <ankitgupta.cs40@gmail.com>

hey
_
```

How to use?

- Now the receiver is waiting for the messages from the client.
- Open another terminal and run the client program for sending messages.
- Run the command.
- *mestra -c [ip address of the receiver]*

How to use?

- You will be asked to enter the messages to send. Enter the messages and *enter*.
- Message will be sent.

```
gepslyn:/home/ankit/mestra # mestra -c 172.16.3.74

                                        MESTRA
                Copyright (C) 2016 version.1.0 <ankitgupta.cs40@gmail.com>

Enter message.: hey
Internet Client Program has worked.
gepslyn:/home/ankit/mestra # _
```

How to use?

- To close the server socket send a message *quit* to the server.

```
Ranjan_PC:/home/Ranjan # mestra -s 172.16.3.74

                                        MESTRA
                    Copyright (C) 2016 version.1.0 <ankitgupta.cs40@gmail.com>

hey
Internet Server Program has worked.
Ranjan_PC:/home/Ranjan # _
```


THANK
YOU