**Indian Institute of Information Technology, Allahabad.**

**B.Tech 5th Semester**

**Computer Networks**

**Lab Assignment #3**

***Instructor: Dr. V K Chaurasiya & Dr. S Verma***

1. The next thing that you should add to your network programming skills is the use of the utility functions that make it easier to write robust general purpose programs. Two of these are:

gethostbyname( )

getservbyname( )

Convert your previous program to use these system calls. The idea is that you could run the server anywhere, and by giving the correct name in the arguments, the client will connect to the server on that host.

gethostbyname will convert a name into a structure that contains the IP address of the host. Similarly, getservbyname will convert a service name to a structure that contains the port number. Take a look at the man pages for these system calls to learn more about them. It's good practice. Functionally, the only thing you should change is that the server should print out a message for each connection with the following general form:

Request received from host\_name

where host\_name is the name of the host making the request.

1. You need to build a very simple client-server operation using the stream capabilities. You are to build a client and server pair, to implement your own little directory service. Suppose that the server has the following database

**0 Bob**

**3 Anne**

**5 Barb**

**7 Ray**

**9 Denbigh**

**10 Terri**

**104 John**

This can be in a file that is read. The client is to read a request, which is a numeric address to be sent to the server. The servers look up the matching name and send it back to the client to be printed out. If the address isn't found, the server should send back an error message. For example, "Address not found".

1. In this lab, you will convert your previous program. The only logical change you need to make is to implement your exchange of messages as a protocol.

The protocol will be structured as follows (note, a string of digits followed by a b means a binary number):

Requests:

Byte content

---- -------

1st request code: 0000001(b) = name & 00001001(b) = number

1 to n request data

n+1 end-of-request, 00000011(b) (ETX)

Replies:

byte content

---- -------

0-n reply data

n+1 end-of-reply, 00000011(b) (ETX)

Note that the requests allow for both name and number matching, so you need to also be able to match a name. Also note that the character string can contain any type of byte oriented data, including binary representations of numbers. For example,

short val;

char \*message;

message = &val;

send (sock, message, 2, 0);