International Journal of Computer Science and Informatics

Volume 2 | Issue 3 Article 11

January 2013

Multiplexed Chat Application

A. Vivekanand
Dept. of Computer Science & Engineering, CMR College of Engineering & Technology, Hyderabad, A.P, India, aelgani.vivekanand@gmail.com

B. Sivaiah
Dept. of Computer Science & Engineering, CMR College of Engineering & Technology, Hyderabad, A.P, India, sivabetld@gmail.com

SK. Khaja Shareef
Dept. of Computer Science & Engineering, CMR College of Engineering & Technology, Hyderabad, A.P, India, khaja.sk08@gmail.com

Multiplexed Chat Application

A.Vivekanand, B.Sivaiah, V.A.Narayana,D.Komali & SK. Khaja Shareef

Dept. of Computer Science & Engineering, CMR College of Engineering & Technology, Hyderabad, A.P, India
E-mail : aelgani.vivekanand@gmail.com, sivabetld@gmail.com, khaja.sk08@gmail.com

Abstract **-** Multiplexing in socket programming is the capability of handling input and output from different I/O channels. we can multiplex UDP and TCP sockets to build multiplexed chat application UDP is a connectionless transport layer protocol. Since TCP doesn't provide the feature of Multicasting UDP is a widely used protocol to implement it. UDP's stateless nature is useful for servers that answer small queries for large number of clients. Socket network programming is one of the most popular technologies used to build a chat application and establishing network communication between systems. Socket programming helps to implement the bottom level of network communication, using Application Programming Interface (API). In this paper we propose a method to make a chat room using socket based on User Datagram Protocol (UDP) which enables the feature of acknowledgments after every message sent and poll system call[1]. It is equivalent to a dedicated chat server having a Server and n number of Clients. After client and server set up to connect, you can achieve many machines to communicate through peer to peer communication, multicasting and File sending. During communication taking place there might be different system and network failures occurring, which we have discussed and proposed a convenient solution for that.

This system designed with BSD socket API achieves a satisfying and efficient communication between the users by experimental verification

Key words **-** *Chat Application, File sending, Multicasting, Transport Layer, UDP.*

I. INTRODUCTION

Open system interconnection (OSI) [2] model is a concept that describes how the data communication takes in a network. It consists of seven layers each performing some set of functions regarding how a process gets completed in the Network. OSI layer's consist of physical, data link, Network, Transport, Session, Application and presentation layers. The transport layer [3, 4] is the fourth layer of the OSI reference model which provides transparent transfer of data between end systems using the services of the network layer. It is layered just below the 'Session' and sits above the IP (Internet Protocol) in (OSI) model. This layer is interesting in a sense that it resides in the architectural center of the model. Accordingly, it represents an important transition point between the hardware-associated layers below it and the layers above that are more software-oriented and abstract. The transport layer helps layers above it, providing them cost effective data transfer and reliability. It enables end-to-end control and information transfer which can be implemented in all the End Systems (ES). In general there are two transport layer protocols TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). UDP is a connection less transport layer protocol which supports Network Application and establishes network communication [5].

TCP is connection Oriented protocol while UDP is connectionless protocol [6]. UDP protocol is a simple connectionless protocol that can be used to transfer datagram packets between sender and receiver in both the direction i.e. client and server can send and receive datagram packets to each other. User Datagram Protocol transports the data between the computers in the form of data packets, but it does not track the data and never shows the confirmation report as shown in Fig 1. Many of the streaming applications run using UDP, but they have built-in acknowledgments and retransmissions into the application in order reduce packet loss. And file sharing through UDP also alleviates the problems occurring in TCP. UDP [7] has a very basic and simplified header which has only 8 bytes of overhead as compared to 20 bytes in TCP (in order to guarantee reliability).

UDP is considered to be very useful in some circumstances, especially in speed and network management. The fact that UDP lacks built-in reliability mechanisms can be considered as an advantage from the application designer's point of view. That is, communication between application processes can be achieved reliably without being constrained by the transmission rate constraints imposed by TCP's congestion control mechanism. UDP has a lower latency as compared to TCP because it doesn't wait for any lost packets whereas TCP waits for the lost packets until the transmission timeout occurs as shown in Fig 2. The most important feature regarding UDP is that it supports Multicasting and Broadcasting which can't be implemented in TCP protocol. Now in further paper we will look at the past works and applications related to UDP.

II. RELATED WORK

UDP is used only a few percent in a typical network. There are some internet applications that use UDP such as Domain Name System (DNS) [8], Simple Network Management Protocol (SNMP) [8], Dynamic Host Configuration Protocol (DHCP) [8] and the Routing Information Protocol (RIP) [8]. Many chat room based application have been proposed like outlook, win popup, akeni but mostly all of them are implemented using TCP protocol which is considered reliable but doesn't provide the speed and Multicasting as such. For real time voice and video streaming application UDP is preferred over TCP because of its higher speed. This application doesn't have any impact on loss of a single packet.

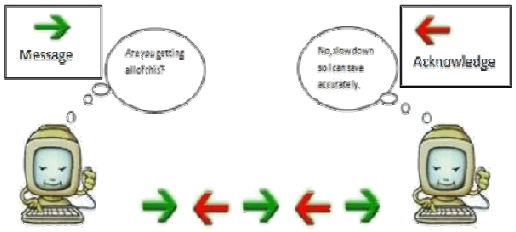

Fig. 1 : TCP

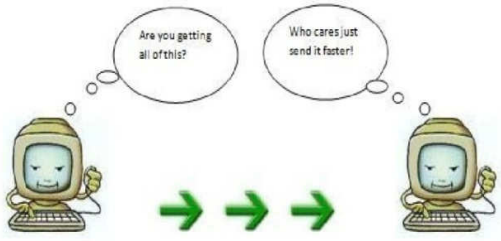

Fig. 2 : UDP

Chat applications using UDP socket has been made, but that doesn't have features of sending files and real time audio, video chat from one peer to another. Most of the modern multiplayer games [9] use UDP broadcast to start game, send messages and information to other players. Another popular use of UDP is a tunneling protocol [10]. A tunnel endpoint encapsulates the packets into UDP and transmits them to tunnel endpoint which encapsulates the UDP datagram's and forward the original packets. Tunnels are used to create virtual links that looks like a direct connection between two locations that are distant in the physical internet topology.

III. TRANSMISSION METHODS

This section introduces you how to deliver or transmit data in network and describes the fundamental of various techniques like: unicast, broadcast and multicast. There are numerous methods for delivering or transmitting data in the network as discussed below.

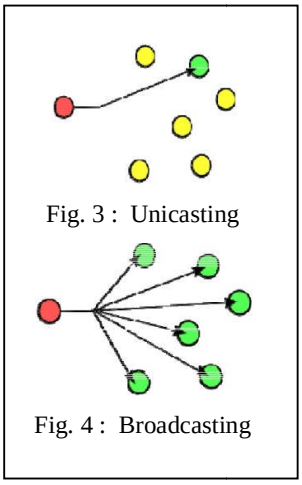

Fig. 3 : Unicasting

Fig. 4 : Broadcasting

A. *Unicasting*

Unicast transmission is the sending of messages to a single network destination host as shown in Fig 3. This method solves network traffic problems to deliver messages from one host to another. So, we can say that this is one-to-one data delivering method. For better understanding, consider a situation where one computer's request for a site is received by all connected to the Internet. This way you are actually flooding the Internet with such requests. Hence, you need Unicast transmission [12] for small and large networks. Examples of Unicast transmission are HTTP, SMTP, etc and it is still predominant in LANs.

B. *Broadcasting*

Broadcast [11, 12] is a type of transmission where data sent by one host is received by every host present in the network as shown in Fig 4. Taking the case of

Dynamic Host Configuration Protocol in which when a system boots up and requests for an IP from DHCP server which system is not aware of so it broadcasts the request of an IP throughout the network. It is used to send same message to all computers in LANs, e.g. Address Resolution Protocol (ARP) uses this technique to send an address resolution query to all. Most routers designed these days' blocks IP broadcast traffic and restrict it to the local subnet.

C. Multicasting

In this method, you can transmit data or messages to all destination host machines, which has been interested an appropriate multicast group. The sender generates only one data stream but it delivered to all destination hosts in the group as shown in Fig 5. It supports one-to-many data delivering networks.

Multicast [11, 12] delivers a data or information simultaneously to all interested destination hosts machines. Multicast is placed in a class 'D' IP address which has addresses space from 224.0.0.0 to 239.255.255.255. In our implementation we have generated a group IP for every multicast group to communicate in a group. Whenever a client wants to create a group, server would provide an IP in the above given pool of IP's and store the information (IP and Port Number) of all the clients in the group. One good example of Multicast based network is video transmission in which one computer needs to transmit video channel to a specific group of computers. This way, other computers which are part of this Multicast IP network will be able to receive same set of data at the same time. Multicast offers savings on bandwidth and is the preferred way of data communication when data is to be transmitted to a set of computers. Unlike broadcast transmission, multicast clients receive stream of packets only if they have previously elected to do so (by joining the specific multicast group address). Membership of any group is controlled by the receivers and is dynamic in nature.

IV. FILE SHARING

Some Peer-to-Peer (P2P) file sharing operation models over asymmetric networks have several shortcomings that may have a significant impact to the system and network performance.

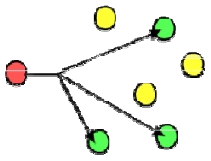

Fig. 5 : Multicasting

Paths on which data transmission occurs are highly redundant, thereby wasting a lot of bandwidth. The downloading throughput of the peer node is limited by the upward bandwidth of other peer nodes. Since in TCP each and every packet is being followed by an acknowledgement packet on the upward channel which results in the deteriorate of performance in sending a file .These shortcomings severely effects the efficiency of Peer to Peer file sharing as well as network performance. Peer to Peer File transfer between the client's will be in the various number of packets.Whole file would be broken into chunks of binary data which would be send to the destination where it would be combined to form a complete file.

V. PROPOSED SOLUTION

We have implemented this using Java platform which would be through Socket Programming. Our solution includes features like peer to peer communication, multicasting and files sharing. The chat application is a client server model as shown in figure 6.

Server: We maintain a thread at server side which receives various incoming clients request. It stores the information of the clients i.e. IP address and name in a list. This list is then broadcasted to all the users. Contrary to this, whenever a client logs out, it deletes that particular client entry from its list and updates it accordingly. It also keeps track of various chat rooms, providing a group IP to each multicast group.

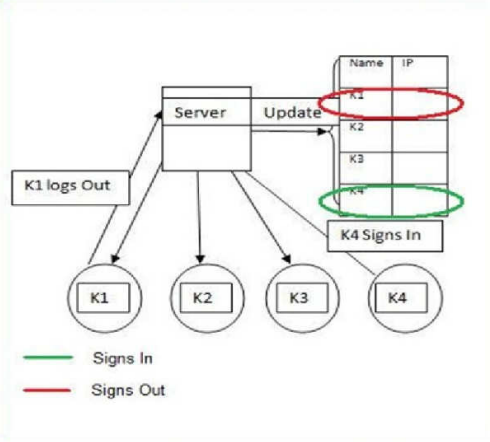

Fig. 6 : Server-Client Model

Client: It first registers itself by sending its username to the server and starts a thread which receives list of all online users from server. Each user has an option of establishing a peer to peer communication with any of the online user, sending a file and creates a room where users selected by him interact with each other.

A. Peer to Peer Communication

This module refers to communication between any two clients. Sender sends the request containing its name and a generated random port to the receiver. After receiving the sender's request at the receiver's end communication is established by starting a thread on both sides. When a network gets established between peers, a counter will be initialized at both the sides which will depict the sequence number of the packet. For every packet sent on either side there will be an acknowledgment from the other side which will increase the counter with 1. So if any kind of failure occurs as discussed later can be identified and a packet may be sent again. For example: If A sends a message with sequence number 10 to B, B will reply back with an acknowledgement with sequence number 11.

B. Multicasting

This feature can be implemented using the multicasting concepts. When a user wants to create a chat room for multiple users, it sends a request to the server. Server then acknowledges this request by providing a group IP and sends the information about the room to all the concerned users. Now, in this room whenever a user wants to send a message it will be multicast to all the users through the group IP.

Server-Client Model as depicted in the figure 6 shows client's k1, k2, k3 establishing connection with the server. When another client k4 sends a request to join, server processes its request by adding its information in the table i.e. Name and IP. Similarly, as client k1 logs out, server deletes the entry related to k1 and correspondingly updates the table.

C. File Sending

In this module there would be feature in our chat window which would allow the user to send a file through the network to the other user. When any user wants to send a file to other user a message would appear on the receiver's side that whether he/she wants to accept it or not. After the acceptance from the receiver whole file would be read and send it to the receiver in the segments of 65,000 bytes and finally at the receiver's end the segments would be received and stored in a buffer and clubbed according to the sequence number. There would be confirmation message after receiving the file.

D. Failure Model

During our implementation, we incurred various kinds of failures for which we proposed the solution in the following manner. As depicted in the Fig 7, we have discussed both system and network failure. In system failure, either a server, sender or receiver may get failed due to some unforeseen reasons.

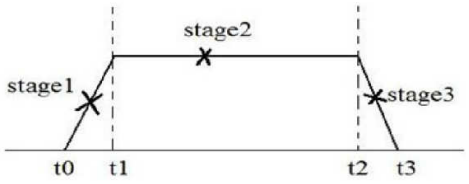

Fig. 7: Failure Model

We have discussed following stages in our failure model-

Stage 1: It belongs from time t0 to t1 which corresponds to buffering time of the packet at sender side. During this stage, if a sender gets failed, then packet would not be sent and gets stored in the queue. This packet will be send only when sender side recovers back. Next, if receiver gets failed, then packet will not be received.

Stage 2: It belongs from time t1 to t2 which corresponds to the time, packet is in the network i.e. the packet is sent from the buffer of sender but not yet received. If at this stage, sender gets failed, then no matter what message will be received. Next, if receiver gets failed, then packet will not be received. But this information about receiver failure is not known to sender. For this problem, we have implemented acknowledgment of each packet we send from sender side. If we don't receive the acknowledgement at the sender side, then it means receiver has failed. By this, we do know about receiver status at the sender side.

Stage 3: It belongs from time t2 to t3 in which packet has now arrived in the buffer of the receiver. At this stage, failure at sender won't have any impact on the packet because it has already arrived in the receiver buffer. Next, if receiver gets failed, then this packet would be recovered when the receiver comes up again.

During all these stages, if server gets failed then a message informing about server failure would be shown on each client in the network.

```
Algorithm Chat_Main()
{
        lfd=socket();//tcp sockets
        ufd=socket();//udp sockets

       while(1){

       pfd[0].fd=lfd;
       pfd[0].events=POLLRDNORM;
       pfd[1].fd=udpfd;
```

```
    pfd[1].events=POLLIN;

    poll(pfd,2,-1);

   if(pfd[0].revents & POLLRDNORM){

    confd=accept();

   if((childpid=fork())==0){
        close(lfd);
     handle stream clients
        exit(0);
    }
     close(connfd);
   }

  if(pfd[1].revents & POLLIN)  {

        Handle udp clients
     }

 }//while

}
```

VI. CONCLUSION AND FUTURE WORK

UDP can be used in peer to peer communication and multicasting. There is a system that acts as a server and it also stores the user's information (username and their IP address). Server sends this information to all the online users. A user can have conversation to more than one user simultaneously. One user can send a file to other user. Server can multicast the data to the users. Problem occurred during the implementation was to identify the packets which we handled by accompanying the packet with a particular type of string. Cryptographic algorithms [13] can be used to secure the data and files that are being sent from one peer to another. To increase the robustness we have used acknowledgments with every packet being sent. This design has some basic requirements of a chat room like instant messaging, multicasting and file sending. We get messages in order which we sent.

ACKNOWLEDGMENT

The authors wish to express sincere thanks to the Secretary CMRCET Mr.Ch.Gopal Reddy and our Head of the Department Mr.K.Srinivas for providing excellent computational infrastructure and stimulating environment in accomplishing this research work.

REFERENCES

[1] Stevens ,Fenner,Ruddof, "UNIX network Programming",Edition 3rd ,PHI,2010 pp 153-191 Volume I.

[2] Hubert Zimmermann The OSI Reference Model. IEEE TRANSACTIONS ON COMMUNICATIONS, VOL. COM-28, NO. 4, APRIL 1980

[3] Andrew S Tannenbaum, "*Computer Network*", Edition 4th, Dorling Kindersley, 2006, pp 487-557.

[4] TCP/IP Transport Layer Protocols. [http:// www.tcpipguide.com/free/t_TransportLayerProtocols.htm] (01/02/2010).

[5] Transport Layer Protocol. [http:// tools.ietf.org/ html/rfc793] 03/02/2010).

[6] Zhao-na Zheng and Peng Sun, Design and Implementation of Chat room using UDP, FCC 2009, CCIS Volume 34 pp 235-239, 2009.

[7] What are the advantages of UDP over TCP? [http://wiki.answers.com/Q/What_are_the_advantages_of_UDP_over _TCP] (05/02/2010).

[8] UDP, User Datagram Protocol, Network Sorcery Inc, [http:// www.networksorcery.com/ enp/ protocol/udp.htm] (26/01/2010).

[9] RFC768 , User Datagram Protocol [http:// www.faqs.org/rfcs/rfc768.html]

[10] Anthony Volodkin, Playing Multiplayer Games over a VPN. [http://non-standard.net/ freebsd/game-vpn/game-vpn.html] (22/01/2010).

[11] Unicast, Broadcast and Multicast. [http:// www.erg.abdn.ac.uk/users/ gorry/eg3561/intro-pages/uni-b-mcast.html](03/02/2010).

[12] James Wang, CpSc 360: Distributed and Network Programming, Chapter 20, 21 Broadcasting and Multicasting. [http://www.cs.clemson.edu/~jzwang/0808360/cpsc36015.pdf] (01/02/2010).

[13] Johannes Buchmann "An Introduction to Cryptography ",Edition 2nd Springer,2004,pp127-272.

❑❑❑