|  |
| --- |
| **NATIONAL INSTITUTE OF TECHNOLOGY PATNA**  University Campus , Bihar-800005 |

**Department of COMPUTER SCIENCE& ENGINEERING**

A project report on

Client server based conversion (prefix infix postfix)  
application using UDP

**Course code: CSL5403**

**Course Title: Computer Networks**

**Branch: B-Tech CSE-A2**

Submitted by:

|  |  |
| --- | --- |
| **NAME** | **ROLL NUMBER** |
| SUDDAPU SANDEEP | 2006092 |
| YALLAMPATI HEMAVARDHINI | 2006074 |
| PARA VENKATA AKSHITH | 2006108 |

Page 1

**TABLE OF CONTENTS**

|  |
| --- |
| **S.NO CONTENTS PAGE NO**   1. **Introduction**   1.1 Socket Programming  1.2 Client  1.3 Server  1.4 Client server Model  1.5 Advantages of Client Server Model  1.6 Disadvantages of client server Model  1.7 TCP  1.8 UDP  1.9 Diff b/w TCP & UDP  1.10 INFIX PREFIX POSTFIX  **2. SYSTEM REQUIREMENTS**  **3. CODE**  **4. OUTPUT**  **5. OBSERVATION**  **6. CONCLUSION**  **7. LEARNING OUTCOMES**  **8. REFERENCES** |

Page 2

**LIST OF FIGURES**

|  |
| --- |
| **S.NO FIGURES PAGE NO.**  1.1 CLIENT SERVER MODEL  1.2 UDP HEADER FORMAT  1.3 UDP CLIENT SERVER MODEL |

Page 3

**AIM OF PROJECT**

Design and implement a Client Server based prefix to postfix, postfix to prefix, prefix to infix, infix to prefix, postfix to infix and infix to postfix conversion application using UDP.

1.INTRODUCTION

**1.1 Socket programming:**

* 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 the other socket reaches out to the other to form a connection.
* Sockets allow communication between two different processes on the same or different machines.
* Where is Socket used:

A Unix Socket is used in a client server application framework. A server is a process which does some function on request from a client. Most of the application level protocols like FTP, SMTP and POP3 make use of Sockets to establish connection between client and server and then for exchanging data.

* There are four types of sockets available to the users. The first two are most commonly used and last two are rarely used.
* Stream sockets
* Datagram sockets
* Raw socket
* Sequenced packet sockets

**1.2 Client**:

* A client is a program that runs on the local machine requesting service from the server. A client program is a finite program means that the service started by the user and terminates when the service is completed.
* A client is a machine or program sending requests to another client or a server in order to take action. A server is an entry points for multiple clients that will handle their requests
* How to make client: The system calls for establishing a connection are somewhat different for the client and the server, but both involve the basic construct of a socket. The two processes each establish their own sockets. The steps involved in establishing a socket on the client side are as follows:
* Create a socket with the socket() system call.
* Connect the socket to the address of the server using the connect() system call.
* Send and receive data. There are a number of ways to do this, but the simplest is to use the read() and write() system calls.

## 1.3 Server:

* A server is a program that runs on the remote machine providing services to the clients. When the client requests for a service, then the server opens the door for the incoming requests, but it never initiates the service.
* A server program is an infinite program means that when it starts, it runs infinitely unless the problem arises. The server waits for the incoming requests from the clients. When the request arrives at the server, then it responds to the request.

**Types of Server:** There are two types of servers

1. Iterative Server: This is the simplest form of server where a server process serves one client and after completing first request then it takes request from another client. Meanwhile another client keeps waiting.

2. Concurrent Servers: This type of server runs multiple concurrent processes to serve many requests at a time. Because one process may take longer and another client cannot wait for so long. The simplest way to write a concurrent server under Unix is to fork a child process to handle each client separately.

* How to make a server: The steps involved in establishing a socket on the server side are as follows:
* Create a socket with the socket() system call.
* Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.
* Listen for connections with the listen() system call.
* 4. Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.
* 5. Send and receive data using the read() and write() system calls

. .

**1.4 Client Server Model:**

* A client and server networking model is a model in which computers such as servers provide the network services to the other computers such as clients to perform a user based tasks. This model is known as client-server networking model.
* An application program is known as a client program, running on the local machine that requests for a service from an application program known as a server program, running on the remote machine.

|  |
| --- |
| Fig 1.1: Client Server Model |

* A client program runs only when it requests for a service from the server while the server program runs all time as it does not know when its service is required.
* A server provides a service for many clients not just for a single client. Therefore, we can say that client-server follows the many-to-one relationship. Many clients can use the service of one server.
* Examples of Client-Server Model are Email, World Wide Web, etc.

### **1.5 Advantages of Client-server networks:**

* **Centralized:** Centralized back-up is possible in client-server networks, stored in server
* **Security:** These networks are more secure as all the shared resources are centrally administered.
* **Performance:** The use of the dedicated server increases the speed of sharing resources. This increases the performance of the overall system
* **Scalability:** We can increase the number of clients and servers separately, i.e., the new element can be added, or we can add a new node in a network at any time.

### **1.6 Disadvantages of Client-Server network:**

* **Traffic Congestion** is a big problem in Client/Server networks. When a large number of clients send requests to the same server may cause the problem of Traffic congestion.
* It does not have a robustness of a network, i.e., when the server is down, then the client requests cannot be met.

**1.7 TRANSMISSION CONTROL PROTOCOL(TCP):**

* TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between the different devices over a network.
* The main functionality of the TCP is to take the data from the application layer. Then it divides the data into a several packets, provides numbering to these packets, and finally transmits these packets to the destination. The TCP, on the other side, will reassemble the packets and transmits them to the application layer. As we know that TCP is a connection-oriented protocol, so the connection will remain established until the communication is not completed between the sender and the receiver.

### **Features of TCP protocol:**

* Transport layer protocol
* Reliable
* Order of data is maintained
* Error control
* Flow control
* Segment numbering system

**1.8 USER DATAGRAM PROTCOL (UDP):**

* **User Datagram Protocol (UDP)** is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an **unreliable and connectionless protocol.** So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network. The UDP enables process to process communication.
* The User Datagram Protocol (UDP) is a lightweight transport-layer protocol that works on top of IP. UDP provides a mechanism to detect corrupt data in packets, but it does not attempt to solve other problems that arise with packets, such as lost or out of order packets. That's why UDP is sometimes known as the Unreliable Data Protocol.
* UDP is a connectionless, unreliable, datagram protocol, quite unlike the connectionoriented, reliable byte stream provided by TCP. Nevertheless, there are instances when it makes sense to use UDP instead of TCP. Some popular applications are built using UDP: DNS, NFS, and SNMP.
* Here, UDP comes into the picture. For real-time services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets.

* Features:
* UDP is used when acknowledgement of data does not hold any significance.
* UDP is good protocol for data flowing in one direction.
* UDP is simple and suitable for Query based communications.
* UDP is not connection oriented.
* UDP does not provide congestion control mechanism.
* UDP does not guarantee ordered delivery of data.
* UDP is stateless.
* UDP is suitable protocol for streaming applications such as VoIP, multimedia streaming.

**UDP Header –**

* UDP header is an **8-bytes** fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contains all necessary header information and the remaining part consist of data.
* UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.

|  |
| --- |
| F **Fig 1.2: UDP Header Format** |

**The UDP header contains four fields:**

* **Source port number:** It is 16-bit information that identifies which port is going t send the packet.
* **Destination port number:** It identifies which port is going to accept the information. It is 16-bit information which is used to identify application-level service
* **Length:** It is 16-bit field that specifies the entire length of the UDP packet that includes the header also. The minimum value would be 8-byte as the size of the header is 8 bytes.
* **Checksum:** It is a 16-bits field, and it is an optional field. This checksum field checks whether the information is accurate or not as there is the possibility that the information can be corrupted while transmission.  In UDP, the checksum field is applied to the entire packet.

**UDP SERVER & CLIENT:**

* In UDP, the client does not form a connection with the server like in TCP and instead just sends a datagram. Similarly, the server need not accept a connection and just waits for datagrams to arrive. Datagrams upon arrival contain the address of the sender which the server uses to send data to the correct client.
* A UDP server is always listening. A UDP client is only listening after sending a message, for a response

**UDP Server:**

1. Create a UDP socket.
2. Bind the socket to the server address.
3. Wait until the datagram packet arrives from the client.
4. Process the datagram packet and send a reply to the client.
5. Go back to step 2.

**UDP Client:**

1. Create a UDP socket.

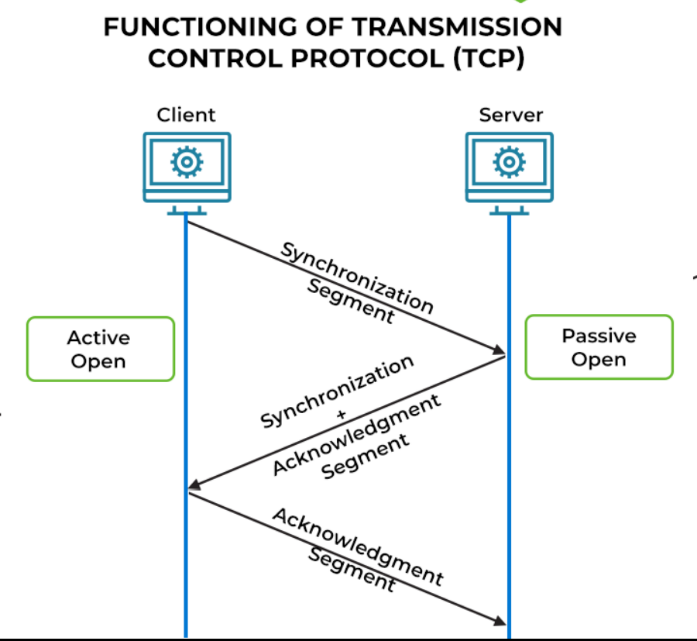
2. Send a message to the server.

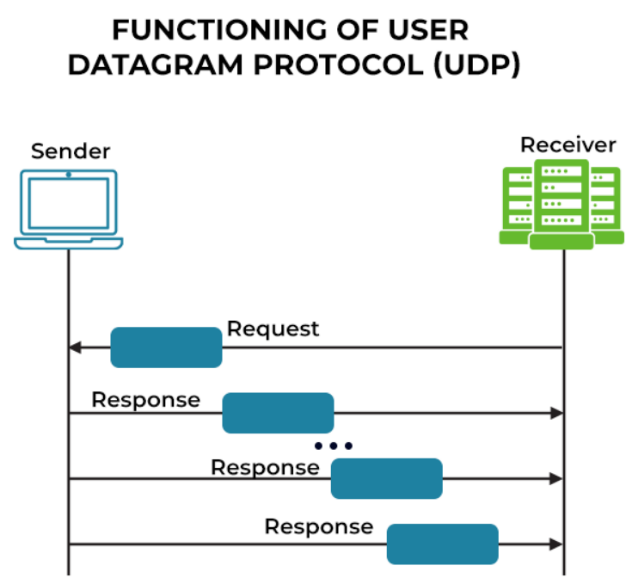
3. Wait until response from the server is received.

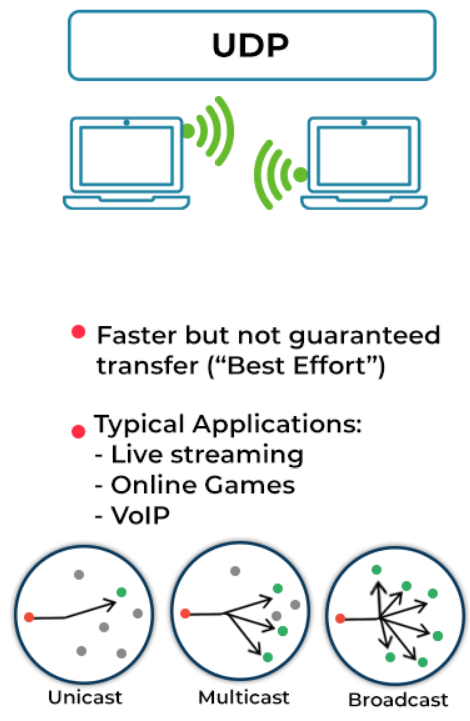
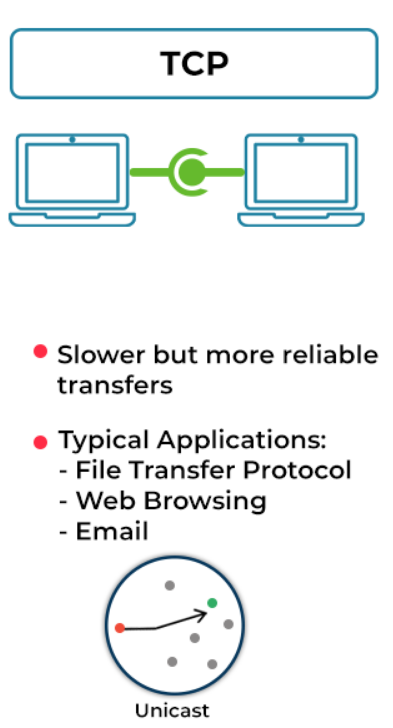
4. Process reply and go back to step 2, if necessary.

|  |
| --- |
| **Fig 1.3: UDP CLIENT SERVER** |

5. Close socket descriptor and exit

**1.9 DIFFERNECE BETWEEN TCP AND UDP:**





**Connection oriented Connection less**

**Retransmissions possible No Retransmissions**

**Error checking and error recovery Only error checking**

Page 10

**1.10 INFIX PREFIX POSTFIX**

* **INFIX NOTATION:** When the operator is written in between the operands, then it is known as **infix notation**. Operand does not have to be always a constant or a variable; it can also be an expression itself.

**Syntax of infix notation is given below:**

**<operand> <operator> <operand>**

* **PREFIX NOTATION:** A prefix notation is another form of expression but it does not require other information such as precedence and associativity, whereas an infix notation requires information of precedence and associativity. It is also known as **polish notation**. In prefix notation, an operator comes before the operands.

**The syntax of prefix notation is given below:**

**<operator> <operand> <operand>**

* **POSTFIX NOTATION:** If we move the operators after the operands then it is known as a postfix expression. In other words, postfix expression can be defined as an expression in which all the operators are present after the operands.

**The syntax of postfix notation is given below:**

**<operand> <operand> <operator>**

Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Infix** | **Prefix** | **Postfix** | **Notes** |
| A \* B + C / D | + \* A B / C D | A B \* C D / + | multiply A and B, divide C by D, add the results |
| A \* (B + C) / D | / \* A + B C D | A B C + \* D / | add B and C, multiply by A, divide by D |
| A \* (B + C / D) | \* A + B / C D | A B C D / + \* | divide C by D, add B, multiply by A |

Page 11

**2. SYSTEM REQUIREMENTS**

**Hardware Requirements:**

* 2 GHz x 86 processor
* 256 MB of system memory (RAM)
* 100 MB of hard drive space
* Keyboard/Mouse for data input

**Software Requirements:**

* Java compiler Command prompt
* MS Word (Documentation)

Page 12

**3.CODE**

**5.OBSERVATIONS**

# 5.1 Java Socket Programming:

* Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connection-less.
* Socket and Server Socket classes are used for connection-oriented socket programming and Datagram Socket and Datagram Packet classes are used for connection-less socket programming.
* The client in socket programming must know two information:

**Creating Server:**

1. Server Socket ss=**new** Server Socket (6666);
2. Socket s=ss. Accept () ;//establishes connection and waits for the client

**Creating Client:**

1. Socket s=**new** Socket("localhost",6666);

5.2Java Datagram Packet Class:

* **Java Datagram Packet** is a message that can be sent or received. It is a data container. If you send multiple Packet, it may arrive in any order. Additionally, packet delivery is not guaranteed.

Here, commonly used Constructors of Datagram Packet class

* **Datagram Packet (byte [] Barr, int length):** it creates a datagram packet. This constructor is used to receive the packets.
* **Datagram Packet (byte [] Barr, int length, Inet Address address, int port):** it creates a datagram packet. This constructor is used to send the packets
  1. **UDP SOCKETS:**
* Implementation of conversions on postfix prefix infix using UDP by creating datagram packets in a socket.
* UDP socket routines enable simple IP communication using the user datagram protocol (UDP).
* The [User Datagram Protocol (UDP)](http://en.wikipedia.org/wiki/User_Datagram_Protocol) runs on top of the Internet Protocol (IP) and was developed for applications that do not require reliability, acknowledgment, or flow control features at the transport layer. This **simple protocol** provides transport layer addressing in the form of UDP ports and an optional checksum capability.

**6.CONCLUSION**

This mini project details the socket programming.  A Socket is a communications connection point (endpoint) that you can name and address in a network. Socket programming shows how to use socket APIs to establish communication links between remote and local processes. This project includes the how the client and server works explained by the client server model. Normally, client server architecture is a basic arrangement where clients are located at the workstations while servers are located far away on the powerful machines in the network. This model is beneficial in the office environments where the clients and servers usually perform routine tasks.

In this project using UDP for conversions, for sure, the development of UDP (User Datagram Protocol) is revolutionary. It allows fast delivery, which is highly valuable for a number of applications. The main purpose of the UDP protocol enables computer applications to send messages, referred to as datagrams, to other systems on an IP (Internet Protocol) network. UDP is a minimal, connectionless protocol and doesn't require any handshake communication before setting up communication channels or data routes. Using client must specify the server's IP address and port number for the call to sendto. using UDP converting infix to postfix, infix to prefix, postfix to prefix, postfix to infix, prefix to postfix, prefix to infix based on client server architecture. Stack data structure used in this Conversions.

This study helps us to realize the advantages and disadvantages of client server model and TCP vs UDP.UDP can form duplicates but in TCP does not possible for duplicates. The solution and implementation for this project is based on UDP client server model. That we were able to execute the conversions of postfix, prefix, infix.

7.LEARNING OUTCOMES

Following are the learning outcomes of us by this mini project that is” implement a Client-Server based prefix to postfix, postfix to prefix, prefix to infix, infix to prefix, postfix to infix and infix to postfix conversion application using UDP.

* After this experiment, we are familiar with Socket programming it’s different types of sockets, where is socket used in programming.
* We are also able to understand client server model, how it’s working in TCP and UDP.
* We are able to classify the differences between TCP and UDP.
* Before this Experiment mostly we used TCP client Server architecture for implementing programs on socket but, after this experiment we are able to implement codes in TCP as well as UDP.
* We got an idea on how UDP client & UDP server works how they get connected.
* After this project we are able to understand the how infix converted to prefix, postfix converted to infix, prefix converted to postfix ….. using user datagram protocol.
* Finally, we learnt UDP client server working model and have a detailed knowledge of the UDP Sockets.

8.REFERENCES

* **UNIX® Network Programming Volume 1, Third Edition**: The Sockets Networking API By W. Richard Stevens, Bill Fenner, Andrew M. Rudoff.
* Lectures on distributed system.pdf: Client-server communication UDP/TCP By Paul Krzyzanowski.
* Geeks For Geeks: <https://www.geeksforgeeks.org/udp-server-client-implementation-c/>
* Website: <https://www.cs.man.ac.uk/~pjj/cs212/fix.html>
* TCP/IP illustrated, volume 1: The protocols, 2nd Edition: **By** [**Kevin R. Fall**](https://www.oreilly.com/beta-search?q=author:%22Kevin%20R.%20Fall%22)**and** [**Author: W. Richard Stevens**](https://www.oreilly.com/beta-search?q=author:%22W.%20Richard%20Stevens%22)**.**