**Vision**

To produce globally competitive professionals and socially sensitive computer engineers enriched with knowledge and power of innovation.

**Mission**

To impart **quality education** to the students at all levels that meets the changing needs of the industry.

To provide state-of-art facilities and resources to **solve real-world complex problems** and for **discovery of new knowledge through innovative research** that encourages **entrepreneurship** and economic development to benefit our **global society**.

To promote active **learning, critical thinking, engineering judgment and, professional capabilities of students** coupled with business and also educate and follow **ethical, social and environmentally responsible engineering practice**.

**How to Use This Manual**

This Manual assumes that the facilitators are aware of Collaborative Learning Methodologies.

This Manual will only provide them tool they may need to facilitate the session on Computer Organization module in collaborative learning environment.

The Facilitator is expected to refer this Manual before the session.

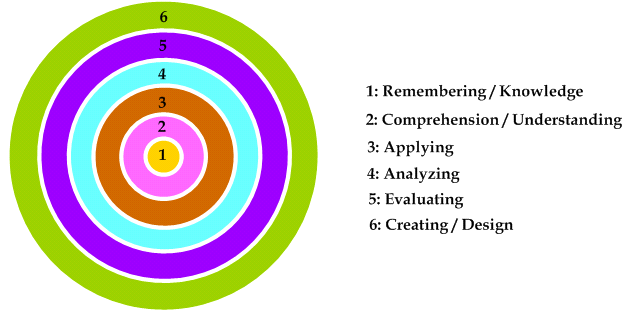
|  |  |  |  |
| --- | --- | --- | --- |
| **K**  Applying Knowledge  (PO:a) | **A**  Problem Analysis  (PO:b) | **D**  Design & Development  (PO:c) | **I**  Investigation of problems  (PO:d) |
| **M**  Modern Tool Usage  (PO:e) | **E**  Engineer & Society  (PO:f) | **E**  Environment Sustainability  (PO:h) | **T**  Ethics  (PO:i) |
| **T**  Individual & Team work  (PO:g) | **O**  Communication  (PO:k) | **M**  Project Management & Finance (PO:j) | **I**  Life Long Learning (PO:l) |

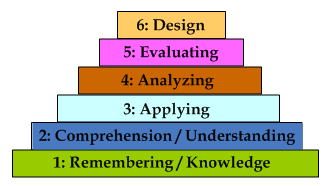
   

**Disk Approach- Digital Blooms Taxonomy**





This Manual uses icons as visual cues to the interactivities during the session.

|  |  |
| --- | --- |
| **Icons** | **Graduate Attributes** |
|  | Applying Knowledge |
|  | Problem Analysis |
|  | Design and Development |
|  | Investigation of Problem |
|  | Modern Tool Usage |
|  | Engineer and Society |
|  | Environment Sustainability |
|  | Ethics |
|  | Individual and Teamwork |
|  | Communication |
|  | Project Management and Finance |
|  | Lifelong Learning |
|  | **Blooms Taxonomy** |
|  | Remembering |
|  | Understanding |
|  | Applying |
|  | Analyzing |
|  | Evaluating |
|  | Creating |

|  |  |
| --- | --- |
|  | This icon is used to indicate instructions for faculties. |
|  | This icon is used to indicate a statement to be made by faculty. |
|  | This icon is used to indicate a list of additional resources. |
|  | This icon indicates an activity to be conducted. |
|  | This icon indicates questions to be asked by faculty. |

**Program Educational Objectives: -**

|  |  |
| --- | --- |
| **PEO I** | To prepare graduate for productive engineering career in industry and also to pursue **higher studies** and **research**. |
| **PEO II** | The graduate of program will have **solid foundation** in Computer Engineering. |
| **PEO III** | The graduate of program will have exposure to cutting edge technology, adequate training and opportunities to work as **team** with effective **communication skills** and **leadership qualities**. |
| **PEO IV** | The graduate of program will have skills to **identify, analyze, design, implement** and **manage** the software projects **using modern tools** for benefit our global **society** and promote them to disseminate it. |
| **PEO V** | The graduate of program will able to keep pace with continuous **upgrading technology** as well as aware of **social, environmental issues** and professional **ethics** and codes of professional practices. |

**Program Outcomes: -**

|  |  |
| --- | --- |
| a | An ability to apply knowledge of computing, mathematics, science and engineering fundamentals appropriate to Computer Engineering |
| a1 | Knowledge of basic science |
| a2 | Fundamental knowledge of Computer Engineering |
| b | An ability to analyze a problem, identify and define the computing requirement appropriate to the solution in Computer Engineering domain |
| b1 | identify the problem and requirements |
| b2 | define and analyze the problem requirements |
| c | An ability to design, implement and evaluate a system, process, component and program to meet desired needs within realistic constraints |
| c1 | design the computer system |
| c2 | implement the computer system |
| c3 | evaluate the computer system |
| d | An ability to investigate, formulates, analyze and provide appropriate solution to the complex engineering problems |
| d1 | investigate and formulate the complex problem |
| d2 | analyze and solve |
| e | An ability to use modern engineering tools, technologies, technique and skills necessary for engineering practices as a Computer Engineering |
| e1 | modern engineering tools |
| e2 | modern engineering technologies, technique and skills |
| f | Apply reasoning informed by the contextual knowledge of Computer field to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice |
| f1 | Apply knowledge of computer to societal and cultural issues |
| f2 | professional engineering practice |
| g | An ability to function effectively as an individual or as a team member to accomplish the goal |
| g1 | function effectively as an individual |
| g2 | work in a team |
| h | An ability to understand the environmental issues and provide the sustainable system |
| h1 | environmental issues |
| h2 | contemporary solutions |
| i | An ability to understand professional, financial, ethical, legal, security, and social issues and responsibility |
| i1 | professional, financial, and security |
| i2 | ethical, legal and social issues |
| j | An ability to apply knowledge of project management and finance |
| j1 | project management |
| j2 | finance |
| k | An ability to communicate effectively with engineering community at different levels |
| k1 | verbal communication |
| k2 | nonverbal communication |
| l | An ability to keep abreast with contemporary technologies through lifelong learning |
| l1 | contemporary technologies |
| l2 | lifelong learning |

**PO to PEO Mapping with the help of Articulation Matrix: -**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PO/GA | | PEO: 1 | PEO: 2 | PEO: 3 | PEO: 4 | PEO: 5 |
|  | a |  |  |  |  |  |
|  | b |  |  |  |  |  |
|  | c |  |  |  |  |  |
|  | d |  |  |  |  |  |
|  | e |  |  |  |  |  |
|  | f |  |  |  |  |  |
|  | h |  |  |  |  |  |
|  | i |  |  |  |  |  |
|  | g |  |  |  |  |  |
|  | k |  |  |  |  |  |
|  | j |  |  |  |  |  |
|  | l |  |  |  |  |  |

**Course Outcomes:**

CO1: To differentiate between wired and wireless services  

CO2: Ability to set-up, install and configure network.   

CO3: Ability to use different networking protocols and tools.  

CO4 :Ability to perform Concurrent programming for Networking and WSN.  

CO5: Ability to perform socket programming. 

CO6: Ability to set-up, install and configure WSN   

**CO to PO Mapping with the help of Articulation Matrix: -**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course Name | CO | PO | | | | | | | | | | | | | | | | | | | | | | | | |
|  | a | | b | | c | | | d | | e | | f | | g | | h | | i | | j | | k | | l | |
|  | a1 | a2 | b1 | b2 | c1 | c2 | c3 | d1 | d2 | e1 | e2 | f1 | f2 | g1 | g2 | h1 | h2 | i1 | i2 | j1 | j2 | k1 | k2 | l1 | l2 |
|  | CO1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| CO6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Laboratory Objectives:**

**LO1 :** To learn different services of network programming . 

**LO2 :** To analyze and configure wired Network . 

**LO 3 :** Analyze and implement the concept of different routing protocol . 

**LO4**: Ability to understand and use of different packet header format  

**LO 5**: Perform socket programming using different protocol 

**LO 6:** To analyze and configure wireless Network 

**LO to CO Mapping with the help of Articulation Matrix: -**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bloom Level | CO: 1 | CO: 2 | CO: 3 | CO: 4 | CO: 5 | CO: 6 |
| LO: 1 |  |  |  |  |  |  |  |
| LO: 2 |  |  |  |  |  |  |  |
| LO: 3 |  |  |  |  |  |  |  |
| LO: 4 |  |  |  |  |  |  |  |
| LO: 5 |  |  |  |  |  |  |  |
| LO: 6 |  |  |  |  |  |  |  |

**Experiment Learning Outcome: -**

**ELO1 :** Students should be able to implement packet sniffer program& identify header of each protocol using multicore programming**.** 

**ELO2 :** Student should be able to implement subnet network. 

**ELO3 :** Able to Execute http header analyzer program in c++. ****

**ELO4 :** Able to install, setup and configure DHCP SERVER ****

**ELO5 :** Explain and apply the concept of SAN in distubuted environment using exacting infrastructure **.** 

**ELO6 :** Explain and apply the concept of SAN for video server. 

**ELO7 :** Explain and apply the concept WSN for RIP routing**.** 

**ELO8 :** perform Linux networking commands emulation using Python or C++.

**ELO9 :** Able to Execute File Transfer Protocol (FTP) using socket programming. ****

**ELO10 :** Able to Execute Trivial File Transfer Protocol(TFTP) of UDP using socket programming. ****

**ELO11 :** Implement TCP/IP packet using standard TCP/IP include files and send it to the server in computer networks

**ELO12 :** . Able to execute congestion control algorithm in TCP. ****

**ELO13 :** .Perform concurrent Text Conferencing application using Python or Java . 

**ELO14:** Execute program of Concurrent Proxy server using Python or Java. 

**ELO15 :** Able to execute Multithreaded web server in LAN using NAT

**ELO16:** Able to apply IP Protocol to print a file on remote IP based printer. 

**ELO17 :** Explain and apply the concept of Sliding Window Protocol**. **

**ELO18 :** Able to Execute DVR using Bellman Ford Algorithm ****

**ELO19 :** Able to understand wireless network using ns3. 

**ELO to LO Mapping with the help of Articulation Matrix: -**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bloom Level | LO: 1 | LO: 2 | LO: 3 | LO: 4 | LO: 5 | LO: 6 |
| ELO: 1 |  |  |  |  |  |  |  |
| ELO: 2 |  |  |  |  |  |  |  |
| ELO: 3 |  |  |  |  |  |  |  |
| ELO: 4 |  |  |  |  |  |  |  |
| ELO: 5 |  |  |  |  |  |  |  |
| ELO: 6 |  |  |  |  |  |  |  |
| ELO: 7 |  |  |  |  |  |  |  |
| ELO: 8 |  |  |  |  |  |  |  |
| ELO: 9 |  |  |  |  |  |  |  |
| ELO: 10 |  |  |  |  |  |  |  |
| ELO: 11 |  |  |  |  |  |  |  |
| ELO: 12 |  |  |  |  |  |  |  |
| ELO: 13 |  |  |  |  |  |  |  |
| ELO: 14 |  |  |  |  |  |  |  |
| ELO: 15 |  |  |  |  |  |  |  |
| ELO: 16 |  |  |  |  |  |  |  |
| ELO: 17 |  |  |  |  |  |  |  |
| ELO: 18 |  |  |  |  |  |  |  |
| ELO: 19 |  |  |  |  |  |  |  |

**GROUP A**

**EXPERIMENT NO.1**

**Implementation of Packet sniffer.**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of packet sniffer | Demonstration, Chalk &Talk | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**TITLE:**  **Implementation of Packet sniffer. Program should identify header of each protocol.Use multi-core programming**

**OBJECTIVES:**

1. Students should be able to implement packet sniffer program.

2. Student should be able to identify header of each protocol using multicore programming**.**

**PROBLEM STATEMENT:**

Implementation of Packet sniffer. Program should identify header of each protocol.Use multi-core programming

**Hardware Requirement:**  8 GB RAM, 500GB/1TB HDD

**Software Requirement:** Latest version of 64 Bit Operating Systems Open Source Fedora-20

**INPUT:** Network traffic in a LAN

**OUTPUT:** Analysis and classification of network traffic in a LAN.

**THEORY:**

**How do Packet Sniffers Work?:** A packet sniffer, sometimes referred to as a network monitor or network analyzer, can be used legitimately by a network or system administrator to monitor and troubleshoot network traffic. Using the information captured by the packet sniffer an administrator can identify erroneous [packets](http://netsecurity.about.com/library/glossary/bldef-packet.htm) and use the data to pinpoint bottlenecks and help maintain efficient network data transmission.

In its simple form a packet sniffer simply captures all of the packets of data that pass through a given network interface. Typically, the packet sniffer would only capture packets that were intended for the machine in question. However, if placed into promiscuous mode, the packet sniffer is also capable of capturing **ALL** packets traversing the network regardless of destination.

**Packet Sniffing :** Packet sniffing is the act of capturing packets of data flowing across a computer network. The software or device used to do this is called a packet sniffer. Packet sniffing is to computer networks what wire tapping is to a telephone network.

Packet sniffing has legitimate uses to monitor network performance or troubleshoot problems with network communications.

**The protocols: IP, ICMP, UDP and TCP**

To fabricate our own packets, what we all need to know is the structures of the protocols that need to be included.  We can define our own protocol structure (packets’ header) then assign it with new values or we just assign new values for the standard built-in structures’ elements.   Below you will find detail information of the IP, ICMP, UDP and TCP headers.  Unix/Linux systems provide standard structures for the header files, so it is very useful in learning and understanding packets by fabricating our own packet by using a struct, so we have the flexibility in filling the packet headers.  We can always create our own struct, as long as the length of each field is correct.  In building our program later on, note also the little endian (Intel x86) notation and the big endian based machines (some processor architectures other than Intel x86 such as Motorola).  The following sections try to analyze header structures that will be used to construct our own packet in the program examples that follows, so that we know what values should be filled in and which meaning they have.  The data types that we need to use are: unsigned char (1 byte/8 bits), unsigned short int (2 bytes/16 bits) and unsigned int (4 bytes/32 bits).  Some of the information presented in the following sections might be a repetition from the previous one.

**IP**

The following figure is IP header format that will be used as our reference in the following discussion.

Figure 1.1: IP header format.

The **Internet** Protocol is the **network** layer protocol, used for routing the data from the source to its destination.  Every datagram contains an IP header followed by a transport layer protocol such as tcp or udp.  The following Table is a list of the IP header fields and their information.

**ICMP**

IP itself has no mechanism for establishing and maintaining a connection, or even containing data as a direct payload.  **Internet Control Messaging Protocol** is merely an **addition** to IP to carry error, routing and control messages and data, and is often considered as a protocol of the network layer.  The following is ICMP header format.

Figure:1.2 ICMP header format.

**UDP**

The **User Datagram Protocol** is a transport protocol for sessions that need to exchange data.  Both transport protocols, UDP and TCP provide 65535 (216) different standard and non standard source and destination ports.  The destination port is used to connect to a specific service on that port.  Unlike TCP, UDP is not reliable, since it doesn't use sequence numbers and stateful connections.  This means UDP datagrams can be spoofed, and might not be reliable (e.g. they can be lost unnoticed), since they are not acknowledged using replies and sequence numbers.  The following figure shows the UDP header format.

Figure 1.3: UDP header format.

**TCP**

The **Transmission Control Protocol** is the mostly used transport protocol that provides mechanisms to establish a reliable connection with some basic authentication, using connection states and sequence numbers.  The following is a TCP header format.

Figure 1.4: TCP header format.

**Program:**

 #include<stdio.h> //For standard things

#include<stdlib.h> //malloc

#include<string.h> //memset

#include<netinet/ip\_icmp.h> //Provides declarations for icmp header

#include<netinet/udp.h> //Provides declarations for udp header

#include<netinet/tcp.h> //Provides declarations for tcp header

#include<netinet/ip.h> //Provides declarations for ip header

#include<sys/socket.h>

#include<arpa/inet.h>

void ProcessPacket(unsigned char\* , int);

void print\_ip\_header(unsigned char\* , int);

void print\_tcp\_packet(unsigned char\* , int);

void PrintData (unsigned char\* , int);

int sock\_raw;

FILE \*logfile;

int tcp=0,udp=0,icmp=0,others=0,igmp=0,total=0,i,j;

struct sockaddr\_in source,dest;

//char IP\_src[100];

//int flg\_syn;

int main()

{

int saddr\_size , data\_size;

struct sockaddr saddr;

struct in\_addr in;

unsigned char \*buffer = (unsigned char \*)malloc(65536); //Its Big!

logfile=fopen("log.txt","w");

if(logfile==NULL) printf("Unable to create file.");

printf("Starting...\n");

//Create a raw socket that shall sniff

sock\_raw = socket(AF\_INET , SOCK\_RAW , IPPROTO\_TCP);

if(sock\_raw < 0)

{

printf("Socket Error\n");

return 1;

}

while(1)

{

saddr\_size = sizeof saddr;

//Receive a packet

data\_size = recvfrom(sock\_raw , buffer , 65536 , 0 , &saddr , &saddr\_size);

if(data\_size <0 )

{

printf("Recvfrom error , failed to get packets\n");

return 1;

}

//Now process the packet

ProcessPacket(buffer , data\_size);

}

close(sock\_raw);

printf("Finished");

return 0;

}

void ProcessPacket(unsigned char\* buffer, int size)

{

//Get the IP Header part of this packet

struct iphdr \*iph = (struct iphdr\*)buffer;

++total;

switch (iph->protocol) //Check the Protocol and do accordingly...

{

case 1: // ICMP Protocol

++icmp;

break;

case 2: // IGMP Protocol

++igmp;

break;

case 6: // TCP Protocol

++tcp;

print\_tcp\_packet(buffer , size);

break;

case 17://UDP Protocol

++udp;

break;

default: //Some Other Protocol like ARP etc.

++others;

break;

}

printf("TCP : %d UDP : %d ICMP : %d IGMP : %d Others : %d Total : %d\r",tcp,udp,icmp,igmp,others,total);

}

void print\_ip\_header(unsigned char\* Buffer, int Size)

{

unsigned short iphdrlen;

struct iphdr \*iph = (struct iphdr \*)Buffer;

iphdrlen =iph->ihl\*4;

memset(&source, 0, sizeof(source));

source.sin\_addr.s\_addr = iph->saddr;

memset(&dest, 0, sizeof(dest));

dest.sin\_addr.s\_addr = iph->daddr;

fprintf(logfile,"\n");

fprintf(logfile,"IP Header\n");

fprintf(logfile," |-IP Version : %d\n",(unsigned int)iph->version);

fprintf(logfile," |-IP Header Length : %d DWORDS or %d Bytes\n",(unsigned int)iph->ihl,((unsigned int)(iph->ihl))\*4);

fprintf(logfile," |-Type Of Service : %d\n",(unsigned int)iph->tos);

fprintf(logfile," |-IP Total Length : %d Bytes(Size of Packet)\n",ntohs(iph->tot\_len));

fprintf(logfile," |-Identification : %d\n",ntohs(iph->id));

fprintf(logfile," |-TTL : %d\n",(unsigned int)iph->ttl);

fprintf(logfile," |-Protocol : %d\n",(unsigned int)iph->protocol);

fprintf(logfile," |-Checksum : %d\n",ntohs(iph->check));

fprintf(logfile," |-Source IP : %s\n",inet\_ntoa(source.sin\_addr));

fprintf(logfile," |-Destination IP : %s\n",inet\_ntoa(dest.sin\_addr));

// strcpy(IP\_src,inet\_ntoa(source.sin\_addr));

}

void print\_tcp\_packet(unsigned char\* Buffer, int Size)

{

unsigned short iphdrlen;

// flg\_syn=0;

struct iphdr \*iph = (struct iphdr \*)Buffer;

iphdrlen = iph->ihl\*4;

struct tcphdr \*tcph=(struct tcphdr\*)(Buffer + iphdrlen);

fprintf(logfile,"\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TCP Packet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

print\_ip\_header(Buffer,Size);

fprintf(logfile,"\n");

fprintf(logfile,"TCP Header\n");

/\*

\*

\*

\*

\*

\* ToDo: Printing TCP Header

\*

\*

\*

\*

\* \*/

fprintf(logfile,"\n");

fprintf(logfile," DATA Dump ");

fprintf(logfile,"\n");

fprintf(logfile,"IP Header\n");

PrintData(Buffer,iphdrlen);

fprintf(logfile,"TCP Header\n");

PrintData(Buffer+iphdrlen,tcph->doff\*4);

fprintf(logfile,"Data Payload\n");

PrintData(Buffer + iphdrlen + tcph->doff\*4 , (Size - tcph->doff\*4-iph->ihl\*4) );

fprintf(logfile,"\n###########################################################");

}

void PrintData (unsigned char\* data , int Size)

{

for(i=0 ; i < Size ; i++)

{

if( i!=0 && i%16==0) //if one line of hex printing is complete...

{

fprintf(logfile," ");

for(j=i-16 ; j<i ; j++)

{

if(data[j]>=32 && data[j]<=128)

fprintf(logfile,"%c",(unsigned char)data[j]); //if its a number or alphabet

else fprintf(logfile,"."); //otherwise print a dot

}

fprintf(logfile,"\n");

}

if(i%16==0) fprintf(logfile," ");

fprintf(logfile," %02X",(unsigned int)data[i]);

if( i==Size-1) //print the last spaces

{

for(j=0;j<15-i%16;j++) fprintf(logfile," "); //extra spaces

fprintf(logfile," ");

for(j=i-i%16 ; j<=i ; j++)

{

if(data[j]>=32 && data[j]<=128) fprintf(logfile,"%c",(unsigned char)data[j]);

else fprintf(logfile,".");

}

fprintf(logfile,"\n");

}

}

}

**CONCLUSION:**

Hence, we have successfully studied concept of packet sniffer using multicore programming.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO1:** Implement packet sniffer program& identify header of each protocol using multicore programming**.**

**QUESTIONS:**

* What do you mean by network protocol?
* Which are different network protocol?
* What is Protocol header?
* Explain Packet format.
* What is sniffing?
* What is sniffer?
* Explain Packet sniffer.
* Define IP header?
* Define TCP/UDP header.
* Explain ICMP header.
* Explain Network layer.
* Explain TCP/IP model.
* Explain OSI network model.
* Explain Advantages and disadvantages of packet sniffing.
* What is multicore programming?

**EXPERIMENT NO.2**

**Implementation of Subnet Network.**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of Subnet | Demonstration | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses, implementation | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusion | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**TITLE:**

Consider the network id 192.168.4.0 or such relevant IP and create four subnets namely

A, B, C, D. Assign the subnet mask. Write a Python \ C++ program to perform the Following operations (use overloading if applicable).

a) Ping the machine of same subnet.

b) Ping the machine in subnet A from machine of subnet B.

c) Analyze the output of the above sub assignments.

**Problem Statement:**

Consider the network id 192.168.4.0 or such relevant IP and create four subnets namely

A, B, C, D. Assign the subnet mask. Write a Python \ C++ program to perform the Following operations (use overloading if applicable).

a) Ping the machine of same subnet.

b) Ping the machine in subnet A from machine of subnet B.

c) Analyze the output of the above sub assignments.

**Objective**:

Student should be able to

a) Ping the machine of same subnet.

b) Ping the machine in subnet A from machine of subnet B.

c) Analyze the output of the above sub assignments.

**Hardware requirement:**

Computer, CAT 5 cables, Switches.

**Software requirement:**

Latest version of 64 Bit Operating Systems Open Source Fedora-20,

Latest versions of 64-Bit Programming languages such as Microsoft Visual Studio (ver. 12 or Higher) or equivalent open source, Eclipse 64-bit Platform.

**Theory**

**Description:**

**Classes of IP Address**

Internet Protocol hierarchy contains several classes of IP Addresses to be used efficiently in various situations as per the requirement of hosts per network. Broadly, the IPv4 Addressing system is divided into five classes of IP Addresses. All the five classes are identified by the first octet of IP Address.

Internet Corporation for Assigned Names and Numbers is responsible for assigning IP addresses.

The first octet referred here is the left most of all. The octets numbered as follows depicting dotted decimal notation of IP Address:

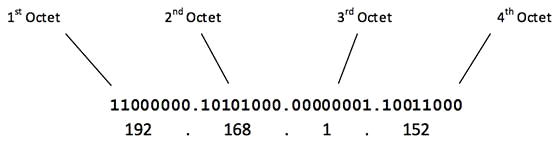


Fig. Group A 2.1

The number of networks and the number of hosts per class can be derived by this formula:



Fig. A 2.2

When calculating hosts' IP addresses, 2 IP addresses are decreased because they cannot be assigned to hosts, i.e. the first IP of a network is network number and the last IP is reserved for Broadcast IP.

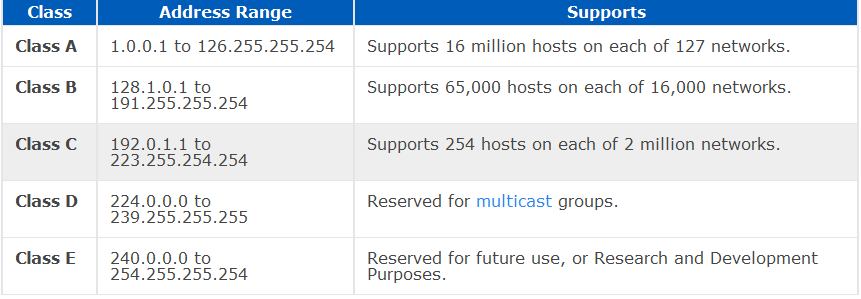


Fig. A 2.3

**Class A Address**

The first bit of the first octet is always set to 0 (zero). Thus the first octet ranges from 1 – 127, i.e.



Fig. A 2.4

Class A addresses only include IP starting from 1.x.x.x to 126.x.x.x only. The IP range 127.x.x.x is reserved for loopback IP addresses.

The default subnet mask for Class A IP address is 255.0.0.0 which implies that Class A addressing can have 126 networks (27-2) and 16777214 hosts (224-2).

Class A IP address format is thus: **0NNNNNNN**.HHHHHHHH.HHHHHHHH.HHHHHHHH

**Class B Address**

An IP address which belongs to class B has the first two bits in the first octet set to 10, i.e.



Fig. A 2.5

Class B IP Addresses range from 128.0.x.x to 191.255.x.x. The default subnet mask for Class B is 255.255.x.x.

Class B has 16384 (214) Network addresses and 65534 (216-2) Host addresses.

Class B IP address format is: **10NNNNNN.NNNNNNNN**.HHHHHHHH.HHHHHHHH

**Class C Address**

The first octet of Class C IP address has its first 3 bits set to 110, that is:



Fig. A 2.6

Class C IP addresses range from 192.0.0.x to 192.255.255.x. The default subnet mask for Class C is 255.255.255.x.

Class C gives 2097152 (221) Network addresses and 254 (28-2) Host addresses.

Class C IP address format is: **110NNNNN.NNNNNNNN.NNNNNNNN**.HHHHHHHH

**Class D Address**

Very first four bits of the first octet in Class D IP addresses are set to 1110, giving a range of:



Fig. A 2.7

Class D has IP address rage from 224.0.0.0 to 239.255.255.255. Class D is reserved for Multicasting. In multicasting data is not destined for a particular host, that is why there is no need to extract host address from the IP address, and Class D does not have any subnet mask.

**Class E Address**

This IP Class is reserved for experimental purposes only for R&D or Study. IP addresses in this class ranges from 240.0.0.0 to 255.255.255.254. Like Class D, this class too is not equipped with any subnet mask.

**What is mean by Subnet?**

A subnet (short for "sub network") is an identifiably separate part of an organization's network. Typically, a subnet may represent all the machines at one geographic location, in one building, or on the same local area network (LAN). Having an organization's network divided into subnets allows it to be connected to the Internet with a single shared network address. Without subnets, an organization could get multiple connections to the Internet, one for each of its physically separate sub networks, but this would require an unnecessary use of the limited number of network numbers the Internet has to assign. It would also require that Internet routing tables on gateways outside the organization would need to know about and have to manage routing that could and should be handled within an organization.

A portion of a network that shares a common address component. On TCP/IP networks, subnets are defined as all devices whose IP addresses have the same prefix. For example, all devices with IP address that start with 100.100.100. Would be part of the same subnet. Dividing a network into subnets is useful for both security and performance reasons. IP networks are divided using a subnet mask.

The Internet is a collection of networks whose users communicate with each other. Each communication carries the address of the source and destination networks and the particular machine within the network associated with the user or host computer at each end. This address is called the IP address (Internet Protocol address). This 32-bit IP address has two parts: one part identifies the network (with the network number) and the other part identifies the specific machine or host within the network (with the host number). An organization can use some of the bits in the machine or host part of the address to identify a specific subnet. Effectively, the IP address then contains three parts: the network number, the subnet number, and the machine number.

The standard procedure for creating and identifying subnets is provided in Internet Request for Comments 950.

The 32-bit IP address is often depicted as a dot address (also called dotted quad notation) - that is, four groups (or quads) of decimal numbers separated by periods. Here's an example:

130.5.5.25

Each of the decimal numbers represents a string of eight binary digits. Thus, the above IP address really is this string of 0s and 1s:

10000010.00000101.00000101.00011001

As you can see, we inserted periods between each eight-digit sequence just as we did for the decimal version of the IP address. Obviously, the decimal version of the IP address is easier to read and that's the form most commonly used.

Some portion of the IP address represents the network number or address and some portion represents the local machine address (also known as the host number or address). IP addresses can be one of several classes, each determining how many bits represent the network number and how many represent the host number. The most common class used by large organizations (Class B) allows 16 bits for the network number and 16 for the host number. Using the above example, here's how the IP address is divided:

<--Network address--><--Host address--> 130.5 . 5.25

If you wanted to add sub netting to this address, then some portion (in this example, eight bits) of the host address could be used for a subnet address. Thus:

<--Network address--><--Subnet address--><--Host address--> 130.5.5.25

To simplify this explanation, we've divided the subnet into a neat eight bits but an organization could choose some other scheme using only part of the third quad or even part of the fourth quad.

Once a packet has arrived at an organization's gateway or connection point with its unique network number, it can be routed within the organization's internal gateways using the subnet number. The router knows which bits to look at (and which not to look at) by looking at a subnet mask, which is a screen of numbers that tells you which numbers to look at underneath. In a binary mask, a "1" over a number says "Look at the number underneath"; a "0" says "Don't look." Using a mask saves the router having to handle the entire 32 bit address; it can simply look at the bits selected by the mask.

**What is mean by subnet mask?**

Sub network mask, a subnet mask, subnet, or sub netting is a method of dividing a network of IP addresses into groups. Sub netting allows each computer or networking device in its own subnet to communicate with each other and still allow communication between subnets by routing the traffic through the network router. By dividing a network into subnets, it can improve network security and keep overall network traffic balanced. A common example of a subnet mask for class C IP addresses is 255.255.255.0 and is the default subnet mask for many computers and network routers.

A mask used to determine what subnet an IP address belongs to. An IP address has two components, the network address and the host address. For example, consider the IP address 150.215.017.009. Assuming this is part of a Class B network, the first two numbers (150.215) represent the Class B network address, and the second two numbers (017.009) identify a particular host on this network.

**Sub netting**

Sub netting enables the network administrator to further divide the host part of the address into two or more subnets. In this case, a part of the host address is reserved to identify the particular subnet. This is easier to see if we show the IP address in binary format.

The full address is:

10010110.11010111.00010001.00001001

The Class B network part is:

10010110.11010111

and the host address is

00010001.00001001

If this network is divided into 14 subnets, however, then the first 4 bits of the host address (0001) are reserved for identifying the subnet.

The subnet mask is the network address plus the bits reserved for identifying the subnetwork -- by convention, the bits for the network address are all set to 1, though it would also work if the bits were set exactly as in the network address. In this case, therefore, the subnet mask would be 11111111.11111111.11110000.00000000. It's called a mask because it can be used to identify the subnet to which an IP address belongs by performing a bitwise AND operation on the mask and the IP address. The result is the subnetwork address:

Subnet Mask 255.255.240.000 11111111.11111111.11110000.00000000

IP Address 150.215.017.009 10010110.11010111.00010001.00001001

Subnet Address 150.215.016.000 10010110.11010111.00010000.00000000

The subnet address, therefore, is 150.215.016.000.

**Steps to perform the Experiment:**

* Computers for LAN Setup
* Cat 5 cable for connection
* Crimping of cables straight cabling
* Connection with the switch in star topology
* Configuring LAN and assigning IP addresses in class C
* Applying Subnet mask to form 4 subnets viz A.B,C,D
* Ping the machines in same subnet
* Ping the machines in different subnets
* Analyze the above outputs.

**CONCLUSION:**

Hence, we have successfully studied concept of subnet masking in computer networks.

**OUTCOME**

**Upon completion Students will be able to:**

**EL0:** Implement subnet network for four subnets namely

A, B, C, D. Assign the subnet mask and perform

a) Ping the machine of same subnet.

b) Ping the machine in subnet A from machine of subnet B.

c) Analyze the output of the above sub assignments.

**QUESTIONS:**

* What are different IP address classes?
* Difference between different IP Address classes.
* What is mean by subnet?
* What is mean by subnet masking?
* How can ping the machine within same subnet?
* What is mean by LAN?
* How can create LAN?
* How we can do masking with IP addresses?
* How we can ping the machine in Different subnet?
* What is the Difference between LAN, MAN, WAN?
* What is mean by pinging?
* How we can assign the IP addresses to the machine within same network?
* How we can assign the IP addresses to the machine in Different network?
* What are the ranges of different IP addresses?

**EXPERIMENT NO.3**

**Write a C++ program to read the HTTP header and analyze the parameters.**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of HTTP HEADER and analysis | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**OBJECTIVES:**

1. To understand the concept of http header format.

2. To read and implement http header analysis.

**PROBLEM STATEMENT:**

Write a C++ program to read the HTTP header and analyze the parameters

**SOFTWARE REQUIRED:** Latest version of 64 Bit Operating Systems Open Source Fedora-20, Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading, 8 GB RAM, 500GB/1TB HDD, Latest versions of 64-Bit Programming languages such as Microsoft Visual Studio(ver. 12 or Higher) or equivalent open source, Eclipse 64-bit Platform.

**INPUT:** packets in the format of http.

**OUTPUT:** http header analyzer of the packet.

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class Weather Report such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is constructor of the class

e = the end state or destructor of the class.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD- Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Weather\_Report:

s=initial state or constructor of the class weather()

s={weather()}- sets the default values for all five variables to respective values given in assignment.

Input i=(I1,I2)

I1={day\_of\_moth,hightemp,lowtemp,amount\_rain,amount\_snow}

Variables declared-NDD

Day\_of\_month={1,-------,n} n≠Ø. n=30 or 31.

Hightemp={1,------,n} n≠Ø. n=999.

lowtemp={-999,------,n} n≠Ø. n=6.

amount\_rain={1,------,n} n≠Ø. n=10.

amount\_snow={1,------,n} n≠Ø. n=10.

I2={99,999,-999,0,0}

Default values set all integers –Deterministic Data because of function weather(),memory requirement based on DD.

Hence let weather():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {display ( ), override ( ), avg ( )}

display ( )={details of all variable values}

override( )={ default values get overwritten and stored by values specified in this function}

I

I1 I2

( Default values) ( New values)

avg( )={sum of values / no. of days}

n

Σ/n

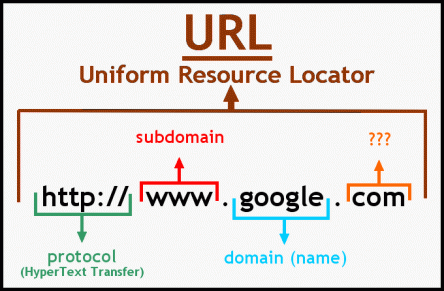
i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**THEORY:**

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.



*HTTP -URLs*

•URL

•Uniform Resource Locator

•protocol (http, ftp, news)

•host name (name.domainname)

•port (usually 80)

•directory path to the resource

•resource name

*HTTP -methods*

•Methods

•GET

•retrieve a URL from the server

•simple page request

•run a CGI program

•run a CGI with arguments attached to the URL

•POST

•preferred method for forms processing

•run a CGI program

•parameterized data in sysin

•more secure and private

*HTTP -methods*

•Methods (cont.)

•PUT

•Used to transfer a file from the client to the server

•HEAD

•requests URLs status header only

•used for conditional URL handling for performance enhancement schemes

•retrieve URL only if not in local cache or date is more recent than cached copy

*HTTP Request Packets*

•Sent from client to server

•Consists of HTTP header

•header is hidden in browser environment

•contains:

•content type / mime type

•content length

•user agent -browser issuing request

•content types user agent can handle

•and a URL

*HTTP Request Headers*

•Precede HTTP Method requests

•headers are terminated by a blank line

•Header Fields:

•From

•Accept

•Accept-Encoding

•Accept Language

•Referer

•Authorization

•Charge-To

•If-Modified-Since

•Pragma

***Status Codes***

•200 OK

•201 created

•202 accepted

•204 no content

•301 moved perm.

•302 moved temp

•304 not modified

•400 bad request

•401 unauthorized

•403 forbidden

•404 not found

•500 int. server error

•501 not impl.

•502 bad gateway

•503 svc not avail

***Statelessness***

Because of the Connect, Request, Response, Disconnect nature of HTTP it is said to be a stateless protocol

•i.e. from one web page to the next there is nothing in the protocol that allows a web program to maintain program “state” (like a desktop program).

•“state” can be maintained by “witchery” or “trickery” if it is needed

**CONCLUSION:**

Hence, we have successfully executed http header analyzer program in c++.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO:** Able to Execute http header analyzer program in c++. ****

**QUESTIONS:**

* What is HTTP?
* What is https?
* What is http port number?
* What are the header fields of http?
* Differentiate between http and https?
* Explain how http works?

**EXPERIMENT NO.4**

**Installing and configure DHCP server and write a program (C++\Python\Java) to install the software on remote machine.**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of DHCP installation and configuration. | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**OBJECTIVES:**

1. To understand the steps of installation and configuration of DHCP.

2. To learn step by step procedure of DHCP SERVER installation and configuration.

**PROBLEM STATEMENT:**

Installing and configure DHCP server and write a program (C++\Python\Java) to install the

software on remote machine.

**SOFTWARE REQUIRED:** Latest version of 64 Bit Operating Systems Open Source Fedora-20, Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading, 8 GB RAM, 500GB/1TB HDD, Latest versions of 64-Bit Programming languages such as Microsoft Visual Studio(ver. 12 or Higher) or equivalent open source, Eclipse 64-bit Platform.

**INPUT:** Systems in the network.

**OUTPUT:**  Automatic ip address allocation to computers in the network.

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class Weather Report such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is constructor of the class

e = the end state or destructor of the class.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD- Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Weather\_Report:

s=initial state or constructor of the class weather()

s={weather()}- sets the default values for all five variables to respective values given in assignment.

Input i=(I1,I2)

I1={day\_of\_moth,hightemp,lowtemp,amount\_rain,amount\_snow}

Variables declared-NDD

Day\_of\_month={1,-------,n} n≠Ø. n=30 or 31.

Hightemp={1,------,n} n≠Ø. n=999.

lowtemp={-999,------,n} n≠Ø. n=6.

amount\_rain={1,------,n} n≠Ø. n=10.

amount\_snow={1,------,n} n≠Ø. n=10.

I2={99,999,-999,0,0}

Default values set all integers –Deterministic Data because of function weather(),memory requirement based on DD.

Hence let weather():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {display ( ), override ( ), avg ( )}

display ( )={details of all variable values}

override( )={ default values get overwritten and stored by values specified in this function}

I

I1 I2

( Default values) ( New values)

avg( )={sum of values / no. of days}

n

Σ/n

i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**THEORY:**

The Dynamic Host Configuration Protocol (DHCP) is a standardized network protocol used on Internet Protocol (IP) networks for dynamically distributing network configuration parameters, such as IP addresses for interfaces and services.

When DHCP was created, its developers had a bit of an issue related to how exactly they should structure DHCP messages. BOOTP was already widely used, and maintaining compatibility between DHCP and BOOTP was an important goal. This meant that DHCP's designers needed to continue using the existing BOOTP message format. However, DHCP has more functionality than BOOTP, and this means more information than can easily be held in the limited BOOTP message format.

This apparent contradiction was resolved in two ways. First, the existing BOOTP message format was maintained for basic functionality, but DHCP clients and servers were programmed to use the BOOTP message fields in slightly different ways. Second, the BOOTP vendor extensions were formalized and became DHCP options, as we'll see in the next two topics. Despite the name “options”, some of these are in fact the additional fields needed for basic DHCP functionality, and are quite mandatory. With this dual approach, DHCP devices have access to the extra information they need. Meanwhile, the basic field format is unchanged, which allows DHCP servers to communicate with older BOOTP clients, which ignore the extra DHCP information that doesn't relate to them. See the topic on BOOTP/DHCP interoperability for more.

1. Operation Code: Specifies the general type of message. A value of 1 indicates a request message, while a value of 2 is a reply message.This code represents the general category of the DHCP message; a client sending a request to a server uses an Op code of 1, while a server replying uses a code of 2. So, for example, a DHCPREQUEST would be a request, while a DHCPACK or DHCPNAK is a reply. The actual specific type of DHCP message is encoded using the DHCP Message Type option.

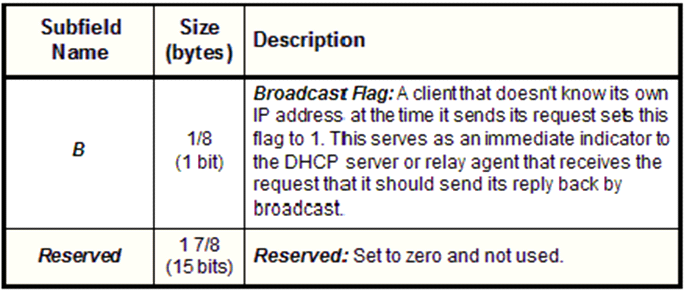
2. Hardware Address Length: Specifies how long hardware addresses are in this message. For Ethernet or other networks using IEEE 802 MAC addresses, the value is 6. This is also the same as a field in the ARP field format, HLN.

3. Hops: Set to 0 by a client before transmitting a request and used by relay agents to control the forwarding of BOOTP and/or DHCP messages.

4. Transaction Identifier: A 32-bit identification field generated by the client, to allow it to match up the request with replies received from DHCP servers.

5. Seconds: In BOOTP this field was vaguely defined and not always used. For DHCP, it is defined as the number of seconds elapsed since a client began an attempt to acquire or renew a lease. This may be used by a busy DHCP server to prioritize replies when multiple client requests are outstanding.

6. Flag:



7. Client IP Address: The client puts its own current IP address in this field if and only if it has a valid IP address while in the BOUND, RENEWING or REBINDING states; otherwise, it sets the field to 0. The client can only use this field when its address is actually valid and usable, not during the process of acquiring an address. Specifically, the client does not use this field to request a particular IP address in a lease; it uses the Requested IP Address DHCP option.

8. Client IP Address: The client puts its own current IP address in this field if and only if it has a valid IP address while in the BOUND, RENEWING or REBINDING states; otherwise, it sets the field to 0. The client can only use this field when its address is actually valid and usable, not during the process of acquiring an address. Specifically, the client does not use this field to request a particular IP address in a lease; it uses the Requested IP Address DHCP option.

9. “Your” IP Address: The IP address that the server is assigning to the client.

10. Server IP Address: The meaning of this field is slightly changed in DHCP. In BOOTP, it is the IP address of the BOOTP server sending a BOOTREPLY message. In DHCP, it is the address of the server that the client should use for the next step in the bootstrap process, which may or may not be the server sending this reply.

The sending server always includes its own IP address in the Server Identifier DHCP option.

11. Gateway IP Address: This field is used just as it is in BOOTP, to route BOOTP messages when BOOTP relay agents are involved to facilitate the communication of BOOTP requests and replies between a client and a server on different subnets or networks. See the topic on DHCP relaying. As with BOOTP, this field is not used by clients and does not represent the server giving the client the address of a default router (that's done using the Router DHCP option).

12. Client Hardware Address: The hardware (layer two) address of the client, which is used for identification and communication.

13. Server Name: The server sending a DHCPOFFER or DHCPACK message may optionally put its name in this field. This can be a simple text “nickname” or a fully-qualified DNS domain name (such as “myserver.organization.org”).

This field may also be used to carry DHCP options, using the “option overload” feature, indicated by the value of the DHCP Option Overload option.

14. Boot Filename: Optionally used by a client to request a particular type of boot file in a DHCPDISCOVER message. Used by a server in a DHCPOFFER to fully specify a boot file directory path and filename.

This field may also be used to carry DHCP options, using the “option overload” feature, indicated by the value of the DHCP Option Overload option.

15. Options: Holds DHCP options, including several parameters required for basic DHCP operation. Note that this field was fixed at 64 bytes in length in BOOTP but is variable in length in DHCP. See the next two topics for more information. This field may be used by both client and server.

**Implementation:**

**First we need to install a DHCP server Software**

**# apt-get install isc-dhcp-server**

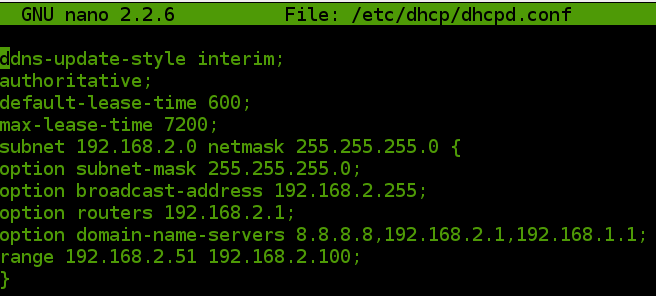


**Open the DHCP server configuration FILE.**

**# nano /etc/dhcp/dhcpd.conf**



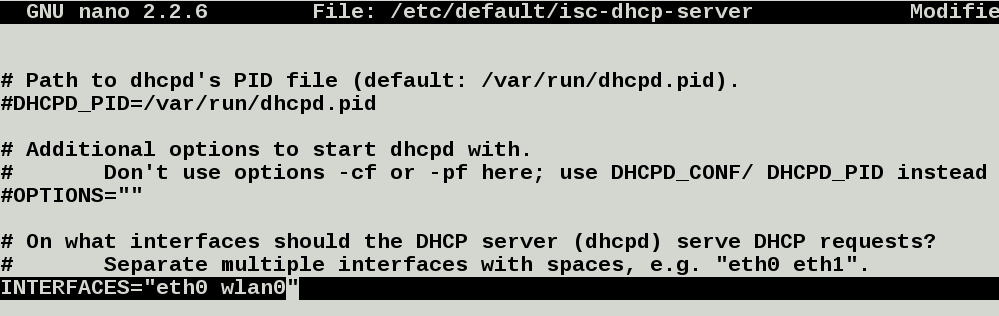
**Configure the DHCP server configuration FILE as per the required Settings.**



**Give the Interfaces on which**

**DHCP should work**





**Assign IP range to route the data packets**

**# route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.2.1**

**# dhcpd -cf /etc/dhcp/dhcpd.conf -pf /var/run/dhcpd.pid at0**

**Start DHCP server.**

**# /etc/init.d/isc-dhcp-server start**

**Install Bridging software  
brctl**

**apt-get install bridge-utils**

**Add a bridging interface.**

**Command::**

**brctl addbr BR1**



**Add your ethernet interface and at0 interface to**

**bridging interface.**

**Command::**

**# brctl addif BR1 eth0**

**# brctl addif BR1 at0**



**View your Bridge interfaces**

**Command::**

**brctl show**



**Change your ethernet interface address to null.**

**Command::**

**ifconfig eth0 0.0.0.0 up**



**Change your wireless interface(logical) address to null.**

**Command::**

**ifconfig at0 0.0.0.0 up**



**Allocate an IP address to Bridge interface**

**Command::**

**ifconfig BR1 192.168.1.20/24 up**



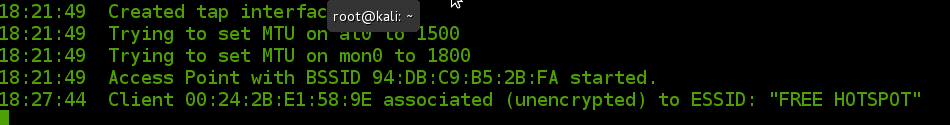
**Switch on port forwarding between interfaces**

**Command::**

**echo 1 > /proc/sys/net/ipv4/ip\_forward**



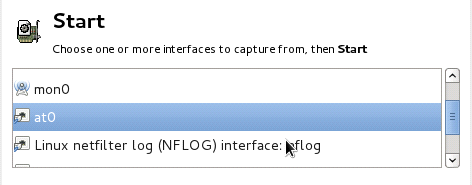
**Now Wait for a client to connect to your network.**



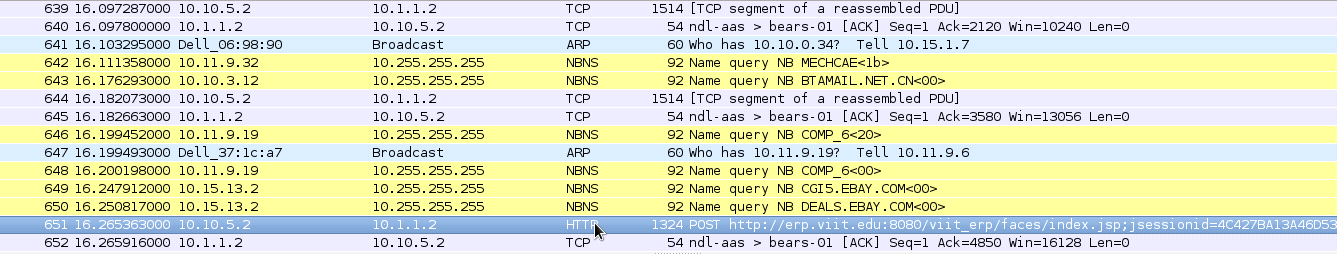
**Open wireshark on your machine**



**Select interface at0 and start monitoring on it**

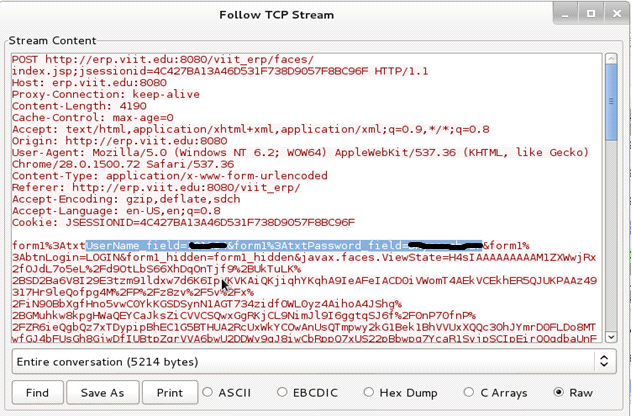


**Wireshark will start capturing packets**



**Select packet with HTTP post request.**

**Right click on it and follow TCP Stream.**



**CONCLUSION:**

Hence, we have successfully installed DHCP server and configured.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO1:** Able to install,setup and configure DHCP SERVER ****

**QUESTIONS:**

* What are minimum specifications for installation of the Windows 2003 Server operating system?
* What do you mean by DHCP/DNS?
* Define the terms: Domain, Forest, Tree.
* What is DCPROMO
* What are Domain users and Local users?
* What do you mean by complex password?
* What is static and dynamic IP address ?
* What do you mean by Organizational Units ?
* What is subnet mask?
* What is preferred DNS?
* What is alternate DNS ?

**EXPERIMENT NO.5**

**Design and Setup LAN with Star topology to access Storage Area Network (SAN). The SAN must have DSP data, Text Data, Multimedia Data available for the access.**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of iSCSI Initiator | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | How to Set up a Low Cost SAN with theLinux Software iSCSI Target | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**TITLE:** Design and Setup LAN with Star topology to access Storage Area Network (SAN). The SAN must have DSP data, Text Data, Multimedia Data available for the access

**OBJECTIVES:**

1. To learn network programming techniques iSCSI Initiator.

2. How to Set up a Low Cost SAN with the Linux Software iSCSI Target

**PROBLEM STATEMENT:**

Design and Setup LAN with Star topology to access Storage Area Network (SAN). The SAN must have DSP data, Text Data, Multimedia Data available for the access

**Hardware** **REQUIRED:**8 GB RAM,500GB HDD

**SOFTWARE REQUIRED:** *iSCSI* (Internet Small Computer System Interface).

**INPUT:** DSP data, Text Data, Multimedia Data available for the access.

**OUTPUT:** to access Storage Area Network (SAN)

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the Storage Area Network (SAN ) such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is *iSCSI*

e = the end state LUN .

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying *iSCSI* network or assignment functions.

NDD- Non deterministic data of the system to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For SAN:

s=initial state or network area

s={ Star topology ()}- sets the default netyworks for all five node to respective topology given in assignment.

Input i=(I1,I2)

I1={ iSCSI,lan,PC,LUN }

Variables declared-NDD

iSCSI ={1,-------,n} n≠Ø. n=8 port or 16.

lan={1,------,n} n≠Ø. n=999.

pc={999,------,n} n≠Ø. n=5.

LUN={1,------,n} n≠Ø. n=5.

I2={99,999,-999,0,0}

Default values set all integers –Deterministic Data because of function SAN(),memory requirement based on DD.

Hence let SAN ():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {star topology ( ), data ( )}

star topology ( )={details of star PC }

data( )={ default values star topology and stored by values specified in this function}

I

I1 I2

( Default values) ( New values)

SAN( )={star topology / DSP data}

n

Σ/n

i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**THEORY:**

The term SAN designates a new type of storage architecture in which the storage systems

are attached to a high speed network dedicated exclusively to storage. It involves a whole

new network totally distinct from existing communication networks, as is illustrated in

Figure1. The application servers (usually UNIX or Windows NT based) access the storage

resource through the SAN. Most of the local storage resources are off–loaded from the

applications servers, are managed separately from them, and are consolidated at the data

centre, site, or enterprise level.

The SAN architecture is represented below:

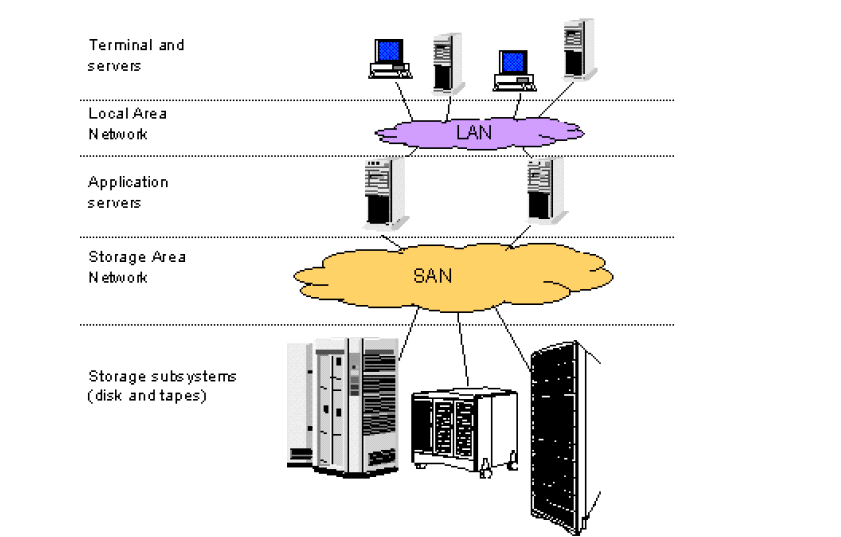


Fig 6.1 SAN architecture

In this the term SAN refers both to the high speed network infrastructure and the whole storage architecture, including servers, storage subsystems and management software.Fibre Channel is currently the preferred technology for implementing SAN architecture. The technology is open, widely accepted, and is defined by ANSI standards. To simplify the migration and integration of legacy equipment, SAN infrastructure based on Fibre Channeltechnology supports multiple protocols. For example, the infrastructure can convey SCSI protocols, widely used in UNIX and Intel based servers, ESCON for IBM mainframes, and IP to offer networking capability. But the purpose of SAN is not to replace LANs. All these protocols can simultaneously use the same cables. This new storage architecture is very different from traditional architectures, where each storage system is connected to a single or sometimes to a limited group of servers. That is

**Concepts of SAN:**

**iSCSI Initiator**

*iSCSI* (Internet Small Computer System Interface) is a protocol that allows SCSI commands to be transmitted over a network. Typically iSCSI is implemented in a SAN (Storage Area Network) to allow servers to access a large store of hard drive space. The iSCSI protocol refers to clients as *initiators* and iSCSI servers as *targets*.Ubuntu Server can be configured as both an iSCSI initiator and a target. This guide provides commands and configuration options to setup aniSCSI initiator. It is assumed that you already have an iSCSI target on your local network and have the appropriate rights to connect to it. The

instructions for setting up a target vary greatly between hardware providers, so consult your vendor documentation to configure your specific iSCSI target.

**iSCSI Initiator Install**

To configure Ubuntu Server as an iSCSI initiator install the *open-iscsi* package. In a terminal enter:

sudo apt-get install open-iscsi

**iSCSI Initiator Configuration**

Once the *open-iscsi* package is installed, edit /etc/iscsi/iscsid.conf changing the following:

node.startup = automatic

You can check which targets are available by using the *iscsiadm* utility. Enter the following in a terminal:

sudo iscsiadm -m discovery -t st -p 192.168.0.10

1. *-m:* determines the mode that iscsiadm executes in.

2. *-t:* specifies the type of discovery.

3. *-p:* option indicates the target IP address.

Change example *192.168.0.10* to the target IP address on your network.

If the target is available you should see output similar to the following:

192.168.0.10:3260,1 iqn.1992-05.com.emc:sl7b92030000520000-2

The *iqn* number and IP address above will vary depending on your hardware.

You should now be able to connect to the iSCSI target, and depending on your target setup you may have to enter user credentials. Login to the

**iSCSI node:**

sudo iscsiadm -m node --login

Check to make sure that the new disk has been detected using *dmesg*

dmesg | grep sd

[ 4.322384] sd 2:0:0:0: Attached scsi generic sg1 type 0

[ 4.322797] sd 2:0:0:0: [sda] 41943040 512-byte logical blocks: (21.4 GB/20.0 GiB)

[ 4.322843] sd 2:0:0:0: [sda] Write Protect is off

[ 4.322846] sd 2:0:0:0: [sda] Mode Sense: 03 00 00 00

[ 4.322896] sd 2:0:0:0: [sda] Cache data unavailable

[ 4.322899] sd 2:0:0:0: [sda] Assuming drive cache: write through

[ 4.323230] sd 2:0:0:0: [sda] Cache data unavailable

[ 4.323233] sd 2:0:0:0: [sda] Assuming drive cache: write through

[ 4.325312] sda: sda1 sda2 < sda5 >

[ 4.325729] sd 2:0:0:0: [sda] Cache data unavailable

[ 4.325729] sd 2:0:0:0: [sda] Cache data unavailable

[ 4.325732] sd 2:0:0:0: [sda] Assuming drive cache: write through

[ 4.325735] sd 2:0:0:0: [sda] Attached SCSI disk

[ 2486.941805] sd 4:0:0:3: Attached scsi generic sg3 type 0

[ 2486.952093] sd 4:0:0:3: [sdb] 1126400000 512-byte logical blocks: (576 GB/537 GiB)

[ 2486.954195] sd 4:0:0:3: [sdb] Write Protect is off

[ 2486.954200] sd 4:0:0:3: [sdb] Mode Sense: 8f 00 00 08

[ 2486.954692] sd 4:0:0:3: [sdb] Write cache: disabled, read cache: enabled, doesn't

support DPO or FUA

[ 2486.960577] sdb: sdb1

[ 2486.964862] sd 4:0:0:3: [sdb] Attached SCSI disk

In the output above *sdb* is the new iSCSI disk. Remember this is just an example; the output you see on your screen will vary.

Next, create a partition, format the file system, and mount the new iSCSI disk. In a terminal enter:

sudo fdisk /dev/sdb

n

p

enter

w

The above commands are from inside the *fdisk* utility; see man fdisk for more detailed instructions. Also, the *cfdisk* utility is sometimes

more user friendly.

Now format the file system and mount it to /srv as an example:

sudo mkfs.ext4 /dev/sdb1

sudo mount /dev/sdb1 /srv

Finally, add an entry to /etc/fstab to mount the iSCSI drive during boot:

/dev/sdb1 /srv ext4 defaults,auto,\_netdev 0 0

It is a good idea to make sure everything is working as expected by rebooting the server.

**StepbyStep: How to Set up a Low Cost SAN with the Linux Software iSCSI Target**

A software iSCSI target can be a great way to set up shared storage when you don’t have enough dough toafford pricey SAN hardware. The iSCSI target acts just like a real hardware iSCSI array, except it’s just a piece of software running on a traditional server (or even a VM!). Setting up an iSCSI target is an easy and low costway to get the shared storage you need, whether you’re using a clustering product like Microsoft WindowsServer Failover Clustering (WSFC), a cluster filesystem such as GFS or OCFS, or even if you’re wanting to getthe most out of your virtualization platform (be it VMware, XenServer, or HyperV)by enabling storage pooling and live migration.

**About LioTarget**

Recently, the Linux kernel has adopted LIOTarget as the standard iSCSI target for Linux. LIO Targetis available in Linux kernels 3.1 and higher. LIOTarget supports SCSI3 Persistent Reservations, which are required by Windows Server Failover Clustering, VMware vSphere, and other clustering products. The LUNs (disks) presented by the iSCSI target can be entire disks, partitions, or even just plain old files on the filesystem. LIO Targetsupports all of these options.Below, we’ll walk through the steps to configure LIO Target

on an Ubuntu 12.04 server (other recent distros will probably work also, but the steps may vary slightly).

**Configuration Steps**

**First, install the Liotarget packages:**

# aptget install –noinstallrecommends targetcli pythonurwid

**Liotarget is controlled using the targetcli command line utility.**

**The first step is to create the backing store for the LUN. In this example, we’ll use a filebacked**

**LUN, which is just a normal file on the filesystem of the iSCSI target server.**

# targetcli

/> cd backstores/

/backstores> ls

o-backstores…………………………………………………… […]

o-fileio…………………………………………. [0 Storage Object]

o-iblock…………………………………………. [0 Storage Object]

o-pscsi………………………………………….. [0 Storage Object]

o-rd\_dr ………………………………………….. [0 Storage Object]

o-rd\_mcp …………………………………………. [0 Storage Object]

**Now the LUN is created. Next we’ll set up the target so client systems can access the storage.**

/backstores/fileio/lun0> cd /iscsi

/iscsi> create ***(create iqn and target port group)***

Created target iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1.

Selected TPG Tag 1.

Successfully created TPG 1.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1

/iscsi/iqn.20…a672ba1/tpgt1> set attribute authentication=0 ***(turn off chap auth)***

/iscsi/iqn.20…a672ba1/tpgt1> cd luns

/iscsi/iqn.20…a1/tpgt1/luns> create /backstores/fileio/lun0 ***(create the target LUN)***

Selected LUN 0.

Successfully created LUN 0.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1/luns/lun0

/iscsi/iqn.20…gt1/luns/lun0> cd ../../portals

**iSCSI traffic can consume a lot of bandwidth, so you’ll probably want the iSCSI traffic to be on a dedicated (or SAN) network, rather than your public network.**

/iscsi/iqn.20…tpgt1/portals> create 10.10.102.164 *(create portal to listen for connections)*

Using default IP port 3260

Successfully created network portal 10.10.102.164:3260.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1/portals/10.10.102.164:3260

/iscsi/iqn.20….102.164:3260> cd ..

/iscsi/iqn.20…tpgt1/portals> create 10.11.102.164

Using default IP port 3260

Successfully created network portal 10.11.102.164:3260.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1/portals/10.11.102.164:3260

/iscsi/iqn.20…102.164:3260> cd ../../acls

**Now, you’ll just need to register the iSCSI initiators (client systems). To do this, you’ll need to find theinitiator names of the systems. For Linux, this will usually be in /etc/iscsi/initiatorname.iscsi. For**

**Windows, the initiator name is found in the iSCSI Initiator Properties Panel in the Configuration Tab.**

/iscsi/iqn.20…a1/tpgt1/acls> create iqn.199405.

com.redhat:f5b312caf756 *(register initiator — this IQN is the*

*IQN of the initiator — do this for each initiator that will access the target)*

Successfully created Node ACL for iqn.199405.com.redhat:f5b312caf756

Created mapped LUN 0.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1/acls/iqn.199405.com.redhat:f5b312caf756

/iscsi/iqn.20….102.164:3260> cd /

**Now, remember to save the configuration. Without this step, the configuration will not be persistent.**

/> saveconfig ***(SAVE the configuration!)***

/> exit

You’ll now need to connect your initiators to the target. You’ll generally need to provide the IP address of the target to connect to it. After the connection is made, the client systems will see a new disk. The disk will need to be formatted before use. And that’s it! You’re ready to use your new SAN. Have fun!

**CONCLUSION:**

Hence, we have successfully Set up a Low Cost SAN with the Linux Software iSCSI Target

**OUTCOME**

**Upon completion Students will be able to:**

**ELO:** Explain and apply the concept of SAN in distubuted environment using exacting infrastuture **. **

**QUESTIONS:**

* What is a SAN?
* Name 3 Limitations of DAS Technology?
* Name at least 4 of the Hardware component in a SAN?
* How could implementation of a SAN benefit LAN performance?
* Name 3 key features of Fibre Channel?
* How is a SAN Superior to SCSI?
* What is a Point-to-Point Topology?
* What is FC-AL?
* What is Switched or Switched Fabric Topology?
* Where does SAN fit into Today’s Networking Environment?
* What type of san switch you are using?
* What is the command to measure the speed of the port?
* What you mean by lun masking?
* What is the difference between Raid 1 and Raid 5?
* What is the difference between Lun masking and Zoning?
* Give examples of few CLI commands of the Storage controller?
* What is the meaning of IOPS? Where it plays important role?
* Give an example of the Database configurations for Disk layout?
* Name at least 3 ways consolidating storage improves operational efficiencies. ?
* What advantages do LAN-free back up and restore deliver?
* Name at least 3 ways a SAN provides a High Availability Infrastructure.
* What are the steps involved to accomplish Raid best configurations?
* What is the functionality of multi pathing?

**PRACTICE ASSIGNMENTS:**

* Create a SAN to store DSP data, Text Data, Multimedia Data available for the access

**GROUP B**

**EXPERIMENT NO.1**

**Implementation of streaming video server and displaying video at client side using Java. Videos are stored using SAN (BIGDATA)**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of iSCSI Initiator | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | How to Set up a Low Cost SAN with theLinux Software iSCSI Target | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**TITLE:** Design and Setup LAN with Star topology to access Storage Area Network (SAN). The SAN must have DSP data, Text Data, Multimedia Data available for the access

**OBJECTIVES:**

1. To learn network programming techniques iSCSI Initiator.

2. How to Set up a Low Cost SAN with the Linux Software iSCSI Target

**PROBLEM STATEMENT:**

Design and Setup LAN with Star topology to access Storage Area Network (SAN). The SAN must have DSP data, Text Data, Multimedia Data available for the access

**Hardware** **REQUIRED:**8 GB RAM,500GB HDD

**SOFTWARE REQUIRED:** *iSCSI* (Internet Small Computer System Interface).

**INPUT:** DSP data, Text Data, Multimedia Data available for the access.

**OUTPUT:** to access Storage Area Network (SAN)

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the Storage Area Network (SAN ) such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is *iSCSI*

e = the end state LUN .

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying *iSCSI* network or assignment functions.

NDD- Non deterministic data of the system to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For SAN:

s=initial state or network area

s={ Star topology ()}- sets the default netyworks for all five node to respective topology given in assignment.

Input i=(I1,I2)

I1={ iSCSI,lan,PC,LUN }

Variables declared-NDD

iSCSI ={1,-------,n} n≠Ø. n=8 port or 16.

lan={1,------,n} n≠Ø. n=999.

pc={999,------,n} n≠Ø. n=5.

LUN={1,------,n} n≠Ø. n=5.

I2={99,999,-999,0,0}

Default values set all integers –Deterministic Data because of function SAN(),memory requirement based on DD.

Hence let SAN ():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {star topology ( ), data ( )}

star topology ( )={details of star PC }

data( )={ default values star topology and stored by values specified in this function}

I

I1 I2

( Default values) ( New values)

SAN( )={star topology / DSP data}

n

Σ/n

i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**THEORY:**

The term SAN designates a new type of storage architecture in which the storage systems

are attached to a high speed network dedicated exclusively to storage. It involves a whole

new network totally distinct from existing communication networks, as is illustrated in

Figure1. The application servers (usually UNIX or Windows NT based) access the storage

resource through the SAN. Most of the local storage resources are off–loaded from the

applications servers, are managed separately from them, and are consolidated at the data

centre, site, or enterprise level.

The SAN architecture is represented below:

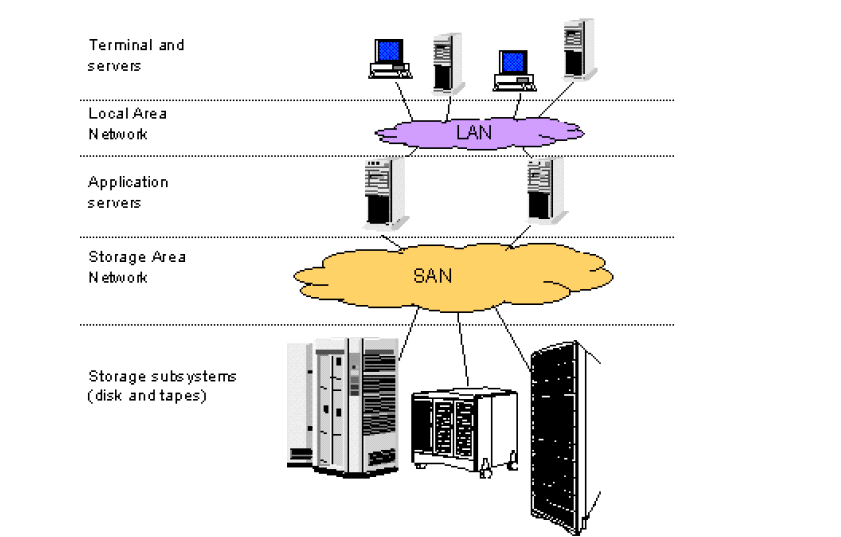


Fig 6.1 SAN architecture

In this the term SAN refers both to the high speed network infrastructure and the whole storage architecture, including servers, storage subsystems and management software.Fibre Channel is currently the preferred technology for implementing SAN architecture. The technology is open, widely accepted, and is defined by ANSI standards. To simplify the migration and integration of legacy equipment, SAN infrastructure based on Fibre Channeltechnology supports multiple protocols. For example, the infrastructure can convey SCSI protocols, widely used in UNIX and Intel based servers, ESCON for IBM mainframes, and IP to offer networking capability. But the purpose of SAN is not to replace LANs. All these protocols can simultaneously use the same cables. This new storage architecture is very different from traditional architectures, where each storage system is connected to a single or sometimes to a limited group of servers. That is

**Concepts of SAN:**

**iSCSI Initiator**

*iSCSI* (Internet Small Computer System Interface) is a protocol that allows SCSI commands to be transmitted over a network. Typically iSCSI is implemented in a SAN (Storage Area Network) to allow servers to access a large store of hard drive space. The iSCSI protocol refers to clients as *initiators* and iSCSI servers as *targets*.Ubuntu Server can be configured as both an iSCSI initiator and a target. This guide provides commands and configuration options to setup aniSCSI initiator. It is assumed that you already have an iSCSI target on your local network and have the appropriate rights to connect to it. The

instructions for setting up a target vary greatly between hardware providers, so consult your vendor documentation to configure your specific iSCSI target.

**iSCSI Initiator Install**

To configure Ubuntu Server as an iSCSI initiator install the *open-iscsi* package. In a terminal enter:

sudo apt-get install open-iscsi

**iSCSI Initiator Configuration**

Once the *open-iscsi* package is installed, edit /etc/iscsi/iscsid.conf changing the following:

node.startup = automatic

You can check which targets are available by using the *iscsiadm* utility. Enter the following in a terminal:

sudo iscsiadm -m discovery -t st -p 192.168.0.10

1. *-m:* determines the mode that iscsiadm executes in.

2. *-t:* specifies the type of discovery.

3. *-p:* option indicates the target IP address.

Change example *192.168.0.10* to the target IP address on your network.

If the target is available you should see output similar to the following:

192.168.0.10:3260,1 iqn.1992-05.com.emc:sl7b92030000520000-2

The *iqn* number and IP address above will vary depending on your hardware.

You should now be able to connect to the iSCSI target, and depending on your target setup you may have to enter user credentials. Login to the

**iSCSI node:**

sudo iscsiadm -m node --login

Check to make sure that the new disk has been detected using *dmesg*

dmesg | grep sd

[ 4.322384] sd 2:0:0:0: Attached scsi generic sg1 type 0

[ 4.322797] sd 2:0:0:0: [sda] 41943040 512-byte logical blocks: (21.4 GB/20.0 GiB)

[ 4.322843] sd 2:0:0:0: [sda] Write Protect is off

[ 4.322846] sd 2:0:0:0: [sda] Mode Sense: 03 00 00 00

[ 4.322896] sd 2:0:0:0: [sda] Cache data unavailable

[ 4.322899] sd 2:0:0:0: [sda] Assuming drive cache: write through

[ 4.323230] sd 2:0:0:0: [sda] Cache data unavailable

[ 4.323233] sd 2:0:0:0: [sda] Assuming drive cache: write through

[ 4.325312] sda: sda1 sda2 < sda5 >

[ 4.325729] sd 2:0:0:0: [sda] Cache data unavailable

[ 4.325729] sd 2:0:0:0: [sda] Cache data unavailable

[ 4.325732] sd 2:0:0:0: [sda] Assuming drive cache: write through

[ 4.325735] sd 2:0:0:0: [sda] Attached SCSI disk

[ 2486.941805] sd 4:0:0:3: Attached scsi generic sg3 type 0

[ 2486.952093] sd 4:0:0:3: [sdb] 1126400000 512-byte logical blocks: (576 GB/537 GiB)

[ 2486.954195] sd 4:0:0:3: [sdb] Write Protect is off

[ 2486.954200] sd 4:0:0:3: [sdb] Mode Sense: 8f 00 00 08

[ 2486.954692] sd 4:0:0:3: [sdb] Write cache: disabled, read cache: enabled, doesn't

support DPO or FUA

[ 2486.960577] sdb: sdb1

[ 2486.964862] sd 4:0:0:3: [sdb] Attached SCSI disk

In the output above *sdb* is the new iSCSI disk. Remember this is just an example; the output you see on your screen will vary.

Next, create a partition, format the file system, and mount the new iSCSI disk. In a terminal enter:

sudo fdisk /dev/sdb

n

p

enter

w

The above commands are from inside the *fdisk* utility; see man fdisk for more detailed instructions. Also, the *cfdisk* utility is sometimes

more user friendly.

Now format the file system and mount it to /srv as an example:

sudo mkfs.ext4 /dev/sdb1

sudo mount /dev/sdb1 /srv

Finally, add an entry to /etc/fstab to mount the iSCSI drive during boot:

/dev/sdb1 /srv ext4 defaults,auto,\_netdev 0 0

It is a good idea to make sure everything is working as expected by rebooting the server.

**StepbyStep: How to Set up a Low Cost SAN with the Linux Software iSCSI Target**

A software iSCSI target can be a great way to set up shared storage when you don’t have enough dough toafford pricey SAN hardware. The iSCSI target acts just like a real hardware iSCSI array, except it’s just a piece of software running on a traditional server (or even a VM!). Setting up an iSCSI target is an easy and low costway to get the shared storage you need, whether you’re using a clustering product like Microsoft WindowsServer Failover Clustering (WSFC), a cluster filesystem such as GFS or OCFS, or even if you’re wanting to getthe most out of your virtualization platform (be it VMware, XenServer, or HyperV)by enabling storage pooling and live migration.

**About LioTarget**

Recently, the Linux kernel has adopted LIOTarget as the standard iSCSI target for Linux. LIO Targetis available in Linux kernels 3.1 and higher. LIOTarget supports SCSI3 Persistent Reservations, which are required by Windows Server Failover Clustering, VMware vSphere, and other clustering products. The LUNs (disks) presented by the iSCSI target can be entire disks, partitions, or even just plain old files on the filesystem. LIO Targetsupports all of these options.Below, we’ll walk through the steps to configure LIO Target

on an Ubuntu 12.04 server (other recent distros will probably work also, but the steps may vary slightly).

**Configuration Steps**

**First, install the Liotarget packages:**

# aptget install –noinstallrecommends targetcli pythonurwid

**Liotarget is controlled using the targetcli command line utility.**

**The first step is to create the backing store for the LUN. In this example, we’ll use a filebacked**

**LUN, which is just a normal file on the filesystem of the iSCSI target server.**

# targetcli

/> cd backstores/

/backstores> ls

o-backstores…………………………………………………… […]

o-fileio…………………………………………. [0 Storage Object]

o-iblock…………………………………………. [0 Storage Object]

o-pscsi………………………………………….. [0 Storage Object]

o-rd\_dr ………………………………………….. [0 Storage Object]

o-rd\_mcp …………………………………………. [0 Storage Object]

**Now the LUN is created. Next we’ll set up the target so client systems can access the storage.**

/backstores/fileio/lun0> cd /iscsi

/iscsi> create ***(create iqn and target port group)***

Created target iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1.

Selected TPG Tag 1.

Successfully created TPG 1.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1

/iscsi/iqn.20…a672ba1/tpgt1> set attribute authentication=0 ***(turn off chap auth)***

/iscsi/iqn.20…a672ba1/tpgt1> cd luns

/iscsi/iqn.20…a1/tpgt1/luns> create /backstores/fileio/lun0 ***(create the target LUN)***

Selected LUN 0.

Successfully created LUN 0.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1/luns/lun0

/iscsi/iqn.20…gt1/luns/lun0> cd ../../portals

**iSCSI traffic can consume a lot of bandwidth, so you’ll probably want the iSCSI traffic to be on a dedicated (or SAN) network, rather than your public network.**

/iscsi/iqn.20…tpgt1/portals> create 10.10.102.164 *(create portal to listen for connections)*

Using default IP port 3260

Successfully created network portal 10.10.102.164:3260.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1/portals/10.10.102.164:3260

/iscsi/iqn.20….102.164:3260> cd ..

/iscsi/iqn.20…tpgt1/portals> create 10.11.102.164

Using default IP port 3260

Successfully created network portal 10.11.102.164:3260.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1/portals/10.11.102.164:3260

/iscsi/iqn.20…102.164:3260> cd ../../acls

**Now, you’ll just need to register the iSCSI initiators (client systems). To do this, you’ll need to find theinitiator names of the systems. For Linux, this will usually be in /etc/iscsi/initiatorname.iscsi. For**

**Windows, the initiator name is found in the iSCSI Initiator Properties Panel in the Configuration Tab.**

/iscsi/iqn.20…a1/tpgt1/acls> create iqn.199405.

com.redhat:f5b312caf756 *(register initiator — this IQN is the*

*IQN of the initiator — do this for each initiator that will access the target)*

Successfully created Node ACL for iqn.199405.com.redhat:f5b312caf756

Created mapped LUN 0.

Entering new node /iscsi/iqn.200301.org.linuxiscsi.murray.x8664:sn.31fc1a672ba1/tpgt1/acls/iqn.199405.com.redhat:f5b312caf756

/iscsi/iqn.20….102.164:3260> cd /

**Now, remember to save the configuration. Without this step, the configuration will not be persistent.**

/> saveconfig ***(SAVE the configuration!)***

/> exit

You’ll now need to connect your initiators to the target. You’ll generally need to provide the IP address of the target to connect to it. After the connection is made, the client systems will see a new disk. The disk will need to be formatted before use. And that’s it! You’re ready to use your new SAN. Have fun!

**CONCLUSION:**

Hence, we have successfully Set up a Low Cost SAN with the Linux Software iSCSI Target

**OUTCOME**

**Upon completion Students will be able to:**

**ELO:** Explain and apply the concept of SAN for video server.

**QUESTIONS:**

* What is a SAN?
* Name 3 Limitations of DAS Technology?
* Name at least 4 of the Hardware component in a SAN?
* How could implementation of a SAN benefit LAN performance?
* Name 3 key features of Fibre Channel?
* How is a SAN Superior to SCSI?

**EXPERIMENT NO.2**

**Simulation of WAN**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of WAN & RIP | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of Problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**TITLE:**  **Simulation of WAN**

**OBJECTIVES:**

* To learn a WAN.
* To configure RIP routing protocol.
* To simulate WAN using RIP routing.

**PROBLEM STATEMENT:**

Simulation of WAN (RIP) using packet tracer/Network Simulator 3 (NS3) or higher equivalent.

**REQUIREMENTS:**

Software: 64 bit NS3.

**INPUT:** Any user defined data.

**OUTPUT:** Data after traversing through WAN using RIP routing.

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class WAN Simulation such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is initial data.

e = the end state.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD - Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Simulation\_WAN:

s=initial state or constructor of the class WANRIP()

s= {WANRIP()}- sets the default values for all five variables to respective values given in assignment.

Input i=(I1,I2)

I1={Node, PC}

Variables declared-NDD

Node = {1,-------,n} n≠Ø. n=999.

PC={1,------,n} n≠Ø. n=2.

I2={999,2}

Default values set all integers –Deterministic Data because of function WANRIP(),memory requirement based on DD.

Hence let WANRIP():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {create(), display ( )}

create( )={ creation of nodes}

display ( )={details of all nodes}

I

I1 I2

(Default values) (New values)

avg( )={sum of values / no. of days}

n

Σ/n

i=1

Success- desired output is generated WAN.

Failure- desired output is not generated in WAN.

**Theory:**

**WAN:**

A **wide area network** (**WAN**) is a network that covers a broad area (i.e., any telecommunications network that links across metropolitan, regional, national or international boundaries) using leased telecommunication lines. Business and government entities utilize WANs to relay data among employees, clients, buyers, and suppliers from various geographical locations. In essence, this mode of telecommunication allows a business to effectively carry out its daily function regardless of location. The Internet can be considered a WAN as well, and is used by businesses, governments, organizations, and individuals for almost any purpose imaginable.

The textbook definition of a WAN is a computer network spanning regions, countries, or even the world. However, in terms of the application of computer networking protocols and concepts, it may be best to view WANs as computer networking technologies used to transmit data over long distances, and between different LANs, MANs and other localized computer networking architectures. This distinction stems from the fact that common LAN technologies operating at Layer 1/2 (such as the forms of Ethernet or Wi-Fi) are often geared towards physically localized networks, and thus cannot transmit data over tens, hundreds or even thousands of miles or kilometers.

WANs do not just necessarily connect physically disparate LANs. A CAN, for example, may have a localized backbone of a WAN technology, which connects different LANs within a campus. This could be to facilitate higher bandwidth applications, or provide better functionality for users in the CAN.

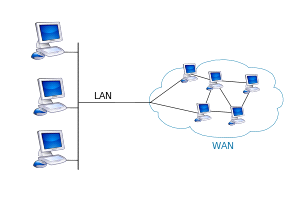


Fig. 2.1 WAN

WANs are used to connect LANs and other types of networks together, so that users and computers in one location can communicate with users and computers in other locations. Many WANs are built for one particular organization and are private. Others, built by Internet service providers, provide connections from an organization's LAN to the Internet. WANs are often built using leased lines. At each end of the leased line, a router connects the LAN on one side with a second router within the LAN on the other. Leased lines can be very expensive. Instead of using leased lines, WANs can also be built using less costly circuit switching or packet switching methods. Network protocols including TCP/IP deliver transport and addressing functions. Protocols including Packet over SONET/SDH, MPLS, ATM and Frame Relay are often used by service providers to deliver the links that are used in WANs. X.25 was an important early WAN protocol, and is often considered to be the "grandfather" of Frame Relay as many of the underlying protocols and functions of X.25 are still in use today (with upgrades) by Frame Relay.

**RIP:**

A routing protocol is a set of rules or standard that determines how routers on a network communicate with each other and exchange information to enable them select best routes to a remote network while each router has knowledge only of networks attached to it directly. Routers running routing protocol share this information first among immediate neighbors, then throughout the entire network. Thus, routers gain insight knowledge of the topology of the network. Routing protocols perform several activities, including:

•Network discovery

•Updating and maintaining routing tables

The router which sits at the base of a network maintains a routing table, which is a list of networks known by the router. The routing table includes network addresses for its own interfaces which are the directly connected networks, as well as network addresses for remote networks. A remote network is a network that can only be reached by forwarding the packet to another router. Remote networks are added to the routing table in two ways:

•By the network administrator manually configuring static routes.

•By implementing a dynamic routing protocol.

Dynamic Routing protocols are used by routers to share information about the reachability and status of remote networks.

Several types and classification of protocols exist, but the focus is on two classes especially, routed protocols and routing protocols.

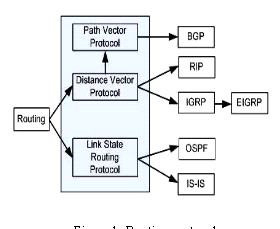


Fig. 2.2 Routing Protocols

RIP is intended for use within the IP-based Internet. The Internet is organized into a number of networks connected by gateways. Hosts and gateways are presented with IP datagrams addressed to some host. Routing is the method by which the host or gateway decides where to send the datagram. It may be able to send the datagram directly to the destination, if that destination is on one of the networks that are directly connected to the host or gateway. However, the interesting case is when the destination is not directly reachable. In this case, the host or gateway attempts to send the datagram to a gateway that is nearer the destination

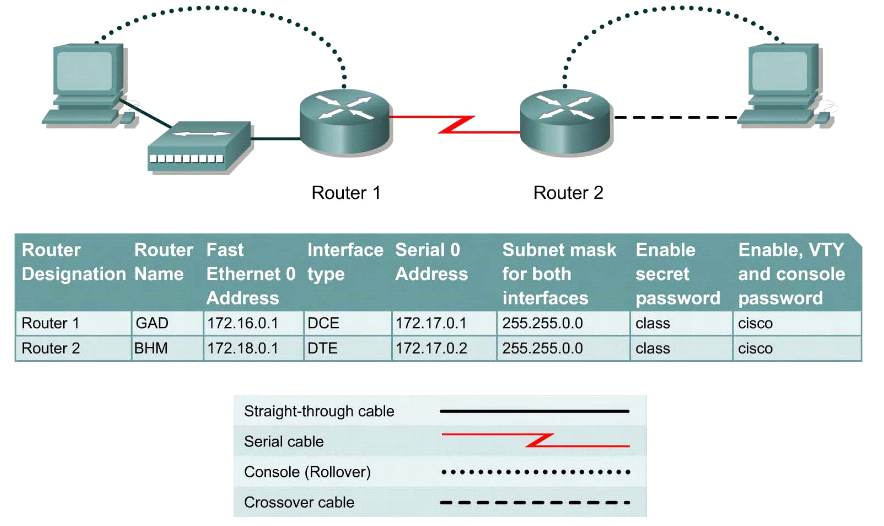
**Configuring RIP**

Fig. 2.3 Configuring RIP

*Background/Preparation*

Setup a network similar to the one in the diagram. Any router that meets the interface requirements displayed in the above diagram, such as 800, 1600, 1700, 2500, 2600 routers, or a combination, may be used. Please refer to the chart at the end of the lab to correctly identify the interface identifiers to be used based on the equipment in the lab. The configuration output used in this lab is produced from 1721 series routers. Any other router used may produce a slightly different output.

The following steps are intended to be executed on each router unless specifically instructed otherwise.

Start a HyperTerminal session as performed in the Establishing a HyperTerminal session lab.

Step 1 Configure the routers

From the global configuration mode, configure the hostname as shown in the chart. Then configure the console, virtual terminal, and enable passwords. If there is a problem doing this, refer to the configuring router passwords lab. Next, configure the interfaces according to the chart. Refer to the Configuring Host Tables lab for assistance.

Step 2 Check the routing table entries

a. Using the command show ip route, view the IP routing table for GAD.

GAD>show ip route

output eliminated

Gateway of last resort is not set

C 172.16.0.0/24 is directly connected, FastEthernet0

C 172.17.0.0/24 is directly connected, Serial0

b. Using the command show ip route, view the IP routing table for BHM.

BHM>show ip route

output eliminated

Gateway of last resort is not set

C 172.17.0.0/24 is directly connected, Serial0

C 172.18.0.0/24 is directly connected, FastEthernet0

Step 3 Configure the routing protocol on the Gadsden router

a. From the global configuration mode, enter the following:

GAD(config)#router rip

GAD(config-router)#network 172.16.0.0

GAD(config-router)#network 172.17.0.0

GAD(config-router)#exit

Step 4 Save the Gadsden router configuration

GAD#copy running-config startup-config

Step 5 Configure the routing protocol on the Birmingham router

a. From the global configuration mode, enter the following:

BHM(config)#router rip

BHM(config-router)#network 172.17.0.0

BHM(config-router)#network 172.18.0.0

BHM(config-router)#exit

BHM(config)#Exit

Step 6 Save the Birmingham router configuration

BHM#copy running-config startup-config

Step 7 Configure the hosts with the proper IP address, subnet mask and default gateway

Step 8 Verify that the internetwork is functioning by pinging the Fast Ethernet interface of the other router

* From the host attached to GAD, is it possible to ping the BHM router Fast

Ethernet interface?

* From the host attached to BHM, is it possible to ping the GAD router Fast

Ethernet interface?

* If the answer is no for either question, troubleshoot the router configurations to find the error.

Then do the pings again until the answer to both questions is yes.

Step 9 Show the routing tables for each router

* From the enable or privileged EXEC mode, examine the routing table entries using the show ip route command on each router.

b. What are the entries in the GAD routing table?

c. What are the entries in the BHM routing table?

**NETWORK SIMULATION TOOL**

Simulation modeling is becoming popular method for network performance analysis because: of its Capability to simulate complex architectures and topologies. It is user friendly (GUI) and designers can test their new ideas and carry out performance related studies and it is free from the burden of the "trial and error" hardware implementations. Therefore, keeping in view the above mentioned advantages we have preferred to use OPNETTM (Optimized Network Engineering Tool), key features being:

• Modeling and Simulation Cycle: OPNET provides powerful tools to assist user to go through important three phases in a design circle (i.e. the building of models, the execution of a simulation and the analysis of the output data)

• Hierarchical Modeling: OPNET employs a hierarchical structure to modeling. Each level of the hierarchy describes different aspects of the complete model being simulated.

• Specialized in communication networks: Detailed library models provide support for existing protocols and allow researchers and developers to either modify these existing models or develop new models of their own.

• Automatic simulation generation: OPNET models can be compiled into executable code. An executable discrete-event simulation can be debugged or simply executed, resulting in output data.

**Conclusion:**

Hence, we have successfully studied & implemented simulation of WAN using RIP routing.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO1:** Explain and apply the concept WSN for RIP routing**. **

**QUESTIONS:**

* What is WAN?
* Difference between a WAN, MAN.
* What is LAN, PAN, and CAN?
* Applications of WAN.
* Difference between a WAN, LAN.
* Difference between a WAN, PAN.
* Difference between a WAN, CAN.
* What is Speed in WAN?
* What is range of WAN?
* What is WAN Optimization?
* Explain Wide area file services.
* Use of WAN.
* Limitations of WAN.
* What are basic requirements of WAN?
* What are WAN routing protocols?
* What is RIP?

**Program B3:**

// Routing Information Protocol Implmentation

// Kaitec Solutions Pvt. Ltd.

#include <iostream>

#include <string>

#include <map>

#include <time.h>

#include <unistd.h>

using namespace std;

// Routing Tables are stored using map

typedef map<string, int> innermap; // in, distance

typedef map<int, innermap> mainmap; // Router number , innermap(ip, distance)

mainmap m;

// mysleep function

void mysleep(unsigned int mseconds)

{

clock\_t goal = mseconds + clock();

while (goal > clock()) ;

}

// function to display intial router configurations

void displayIntialRoutingTbl() {

map<int, innermap >::iterator it;

map<string, int>::iterator inner\_it;

for ( it=m.begin() ; it != m.end(); it++ ) {

cout << "\n\n Routing Table of the Router = " << (\*it).first << endl;

for( inner\_it=(\*it).second.begin(); inner\_it != (\*it).second.end(); inner\_it++)

cout << (\*inner\_it).first << " => " << (\*inner\_it).second << endl;

}

}

// Exchange table between neighbours

int exchangeTables() {

//R1

//exchange routing table between neighbours.

// Modify following code to send/receive packets Using UDP protcol on Port 520

map<int, innermap >::iterator it, it2;

map<string, int>::iterator inner\_it, temp\_it;

it = m.find(2);

if (it != m.end()) {

for(inner\_it=(\*it).second.begin(); inner\_it != (\*it).second.end(); inner\_it++) {

it2 = m.find(1);

temp\_it = (\*it2).second.find((\*inner\_it).first);

if (temp\_it != (\*it2).second.end()) {

// cout << "\n Key Found! = " << (\*inner\_it).first << endl;

} else {

// cout << "\n Key Not Found! = " << (\*inner\_it).first << endl;

m[1].insert(make\_pair((\*inner\_it).first, ((\*inner\_it).second+1)));

}

//cout << (\*inner\_it).first << " => " << (\*inner\_it).second << endl;

}

}

//R2

//exchange routing table between neighbours.

// Modify following code to send/receive packets Using UDP protcol on Port 520

// TODO1----Table received from R1 by R2

/\*

\*

\*

\*

\*/

// TODO2------ Table received from R3 by R2

/\*

\*

\*

\*

\*

\*/

//R3

//exchange routing table between neighbours.

// Modify following code to send/receive packets Using UDP protcol on Port 520

// Table received from R2 by R3

it = m.find(2);

if (it != m.end()) {

for(inner\_it=(\*it).second.begin(); inner\_it != (\*it).second.end(); inner\_it++) {

it2 = m.find(3);

temp\_it = (\*it2).second.find((\*inner\_it).first);

if (temp\_it != (\*it2).second.end()) {

// cout << "\n Key Found! = " << (\*inner\_it).first << endl;

} else {

// cout << "\n Key Not Found! = " << (\*inner\_it).first << endl;

m[3].insert(make\_pair((\*inner\_it).first, ((\*inner\_it).second+1)));

}

//cout << (\*inner\_it).first << " => " << (\*inner\_it).second << endl;

}

}

// Check for convergence of Routing Tables

map<int, innermap >::iterator itR1, itR2, itR3;

itR1 = m.find(1);

itR2 = m.find(2);

itR3 = m.find(3);

if ( ((\*itR1).second.size() == (\*itR2).second.size()) && (((\*itR3).second.size() == (\*itR2).second.size())) )

return 1;

return 0;

}

int main (int argc, char \*argv[]) {

string tempip;

cout << "\n Network Topology is given below:";

cout << "\n 'LAN1'----'R1'---------'R2'---------'R3'------'LAN2'";

m.insert(make\_pair(1, innermap())); // Router 1 Table

m.insert(make\_pair(2, innermap())); // Router 2 Table

m.insert(make\_pair(3, innermap())); // Router 3 Table

cout << "\n Enter Network addr for 'LAN1----R1' = ";

cin >> tempip;

m[1].insert(make\_pair(tempip, 0));

cout << "\n Enter Network addr for 'R1----R2' = ";

cin >> tempip;

m[1].insert(make\_pair(tempip, 0));

m[2].insert(make\_pair(tempip, 0));

cout << "\n Enter Network addr for 'R2-----R3' = ";

cin >> tempip;

m[2].insert(make\_pair(tempip, 0));

m[3].insert(make\_pair(tempip, 0));

cout << "\n Enter Network addr for 'R3----LAN2' = ";

cin >> tempip;

m[3].insert(make\_pair(tempip, 0));

displayIntialRoutingTbl();

int converge = 0;

// Sleep for 5 seconds

while (1) {

sleep(2); // input is in seconds

converge = exchangeTables();

if (converge == 1)

break; // terminate while loop and end program

}

cout << "\n Routing Tables converged!!!";

displayIntialRoutingTbl();

return 0;

}

**Output:**

[root@173-16-13-250 jspm]# g++ rip.cpp

[root@173-16-13-250 jspm]# ./a.out

Network Topology is given below:

'LAN1'----'R1'---------'R2'---------'R3'------'LAN2'

Enter Network addr for 'LAN1----R1' = 173.16.12.41

Enter Network addr for 'R1----R2' = 173.16.8.1

Enter Network addr for 'R2-----R3' = 173.16.13.100

Enter Network addr for 'R3----LAN2' = 173.16.16.5

Routing Table of the Router = 1

173.16.12.41 => 0

173.16.8.1 => 0

Routing Table of the Router = 2

173.16.13.100 => 0

173.16.8.1 => 0

Routing Table of the Router = 3

173.16.13.100 => 0

173.16.16.5 => 0

**EXPERIMENT NO.3**

**Study and perform Linux networking commands emulation using Python or C++**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of linux networking commands | Demonstration | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**TITLE:** Study and perform Linux networking commands emulation using Python or C++.

**OBJECTIVES:**

1.Students should be able to perform Linux networking commands emulation using Python or C++.

**PROBLEM STATEMENT:**

Study and perform Linux networking commands emulation using Python or C++.

**Hardware Requirement:**  8 GB RAM, 500GB/1TB HDD

**Software Requirement:** Latest version of 64 Bit Operating Systems Open Source Fedora-20

**INPUT: networking commands in a LAN**

**OUTPUT: Output of networking commands in a LAN**

**THEORY:**

**An emulation** is a system that behaves *exactly like* something else, and abides by all of the rules of the system being emulated. It is effectively a complete replication of another system, right down to being binary compatible with the emulated system's inputs and outputs, but operating in a different environment to the environment of the original emulated system. The rules are fixed, and cannot be changed or the system fails.

**Example of Networking commands in Unix**

let's see some example of various networking command in Unix and Linux. Some of them are quite basic  e.g. ping and telnet and some are more powerful e.g. nslookup and netstat. When you used these commands in combination of find and grep you can get anything you are looking for e.g. hostname, connection end points, connection status etc.

**hostname** *with no options displays the machines host name*  
**hostname –d** *displays the domain name the machine belongs to*  
**hostname –f** *displays the fully qualified host and domain name*  
**hostname –i** *displays the IP address for the current machine*  
  
  
**ping**  
It sends packets of information to the user-defined source. If the packets are received, the destination device sends packets back. Ping can be used for two purposes  
  
1. To ensure that a network connection can be established.  
2. Timing information as to the speed of the connection.  
  
If you **do ping** [**www.yahoo.com**](www.yahoo.com) it will display its IP address. Use ctrl+C to stop the test.   
  
**ifconfig**  
View network configuration, it displays the current network adapter configuration. It is handy to determine if you are getting transmit (TX) or receive (RX) errors.  
  
  
**netstat**  
Most useful and very versatile for finding connection to and from the host. You can find out all the multicast groups (network) subscribed by this host by issuing **"netstat -g"**  
  
**netstat -nap | grep port** *will display process id of application which is using that port*  
**netstat -a  or netstat –all** *will display all connections including TCP  and UDP*    
**netstat --tcp  or netstat –t** *will display only TCP  connection*  
**netstat --udp or netstat –u** *will display only UDP  connection*  
**netstat -g** *will display all multicast network subscribed by this host.*  
  
**nslookup**  
If you know the IP address it will display hostname. To find all the IP addresses for a given domain name, the command nslookup is used. You must have a connection to the internet for this utility to be useful.  
E.g. **nslookup blogger.com**  
  
You can also use nslookup to [convert hostname to IP Address](http://javarevisited.blogspot.com/2011/09/find-hostname-from-ip-address-to.html) and from IP Address from hostname.  
  
**traceroute**  
A handy utility to view the number of hops and response time to get to a remote system or web site is traceroute. Again you need an internet connection to make use of this tool.

#Getting IP address from Host Name

# TODO: dest\_addr = // write code to Get IP from HOST

port = 33434

max\_hops = 30

**finger**  
View user information, displays a user’s login name, real name, terminal name and write status. this is pretty old unix command and rarely used now days.  
  
**telnet**  
Connects destination host via telnet protocol, if telnet connection establish on any port means connectivity between two hosts is working fine.  
**telnet hostname port**   *will telnet hostname with the port specified. Normally it is used to see whether host is alive and network connection is fine or not.*

#sending UDP and Receiving ICMP packets

recv\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, icmp)

send\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM, udp)

#setting sender packet TTL field to value of the ttl variable.

### TODO : Write your Code here

recv\_socket.bind(("", port))

# TODO Write a code to send pkt using send Socket

curr\_addr = None

*curr\_name = None*

**Program:**

#!/usr/bin/python

import socket

def main(dest\_name):

#Getting IP address from Host Name

# TODO: dest\_addr = // write code to Get IP from HOST

port = 33434

max\_hops = 30

#initializing receiver and sender packets

icmp = socket.getprotobyname('icmp')

udp = socket.getprotobyname('udp')

ttl = 1

while True:

#sending UDP and Receiving ICMP packets

recv\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, icmp)

send\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM, udp)

#setting sender packet TTL field to value of the ttl variable.

### TODO : Write your Code here

recv\_socket.bind(("", port))

# TODO Write a code to send pkt using send Socket

curr\_addr = None

curr\_name = None

try:

#retrieving the router address from packet

\_, curr\_addr = recv\_socket.recvfrom(512)

curr\_addr = curr\_addr[0]

try:

#getting hostname of the retrieved address

curr\_name = socket.gethostbyaddr(curr\_addr)[0]

except socket.error:

curr\_name = curr\_addr

except socket.error:

pass

finally:

send\_socket.close()

recv\_socket.close()

#Printing data in the terminal

if curr\_addr is not None:

curr\_host = "%s (%s)" % (curr\_name, curr\_addr)

else:

curr\_host = "\*"

print "%d\t%s" % (ttl, curr\_host)

ttl += 1

if curr\_addr == dest\_addr or ttl > max\_hops:

break

if \_\_name\_\_ == "\_\_main\_\_":

main('www.google.com')

**CONCLUSION:**

Hence, we have successfully studied Linux networking commands emulation using Python or C++.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO:** 1.perform Linux networking commands emulation using Python or C++.

**QUESTIONS:**

* What is Emulation?
* What are different Linux networking commands?
* finding host/domain name and IP address – **hostname**
* • test network connection – **ping**
* • getting network configuration – **ifconfig**
* • Network connections, routing tables, interface statistics – **netstat**
* • query DNS lookup name – **nslookup**
* • communicate with other hostname – **telnet**
* • outing steps that packets take to get to network host – **traceroute**
* • view user information – **finger**
* • checking status of destination host - **telnet**

**EXPERIMENT NO.4**

**Write FTP/Telnet program using socket programming for TCP using C++**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of FTP,Socket Programming and TCP | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**Title: Write FTP/Telnet program using socket programming for TCP using C++**

**OBJECTIVES:**

1. To Understand the concept of socket programming

2. To implement ftp using Socket Programmming.

**PROBLEM STATEMENT:**

Write FTP/Telnet program using socket programming for TCP using C++

**SOFTWARE REQUIRED:** Latest version of 64 Bit Operating Systems Open Source Fedora-20, Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading, 8 GB RAM, 500GB/1TB HDD, Latest versions of 64-Bit Programming languages such as Microsoft Visual Studio(ver. 12 or Higher) or equivalent open source, Eclipse 64-bit Platform.

**INPUT:** Input text file from user.

**OUTPUT:** file sent and file received status. And contents of the file.

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class Weather Report such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is constructor of the class

e = the end state or destructor of the class.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD- Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Weather\_Report:

s=initial state or constructor of the class weather()

s={weather()}- sets the default values for all five variables to respective values given in assignment.

Input i=(I1,I2)

I1={day\_of\_moth,hightemp,lowtemp,amount\_rain,amount\_snow}

Variables declared-NDD

Day\_of\_month={1,-------,n} n≠Ø. n=30 or 31.

Hightemp={1,------,n} n≠Ø. n=999.

lowtemp={-999,------,n} n≠Ø. n=6.

amount\_rain={1,------,n} n≠Ø. n=10.

amount\_snow={1,------,n} n≠Ø. n=10.

I2={99,999,-999,0,0}

Default values set all integers –Deterministic Data because of function weather(),memory requirement based on DD.

Hence let weather():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {display ( ), override ( ), avg ( )}

display ( )={details of all variable values}

override( )={ default values get overwritten and stored by values specified in this function}

I

I1 I2

( Default values) ( New values)

avg( )={sum of values / no. of days}

n

Σ/n

i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**THEORY:**

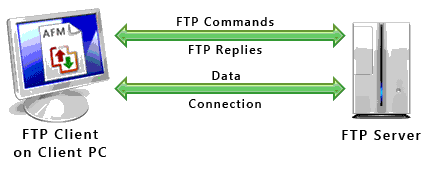
The File Transfer Protocol (FTP) is a standard network protocol used to transfer computer files from one host to another host over a TCP-based network, such as the Internet. FTP is built on a client-server architecture and uses separate control and data connections between the client and the server.

**Understanding How FTP Works**

Data exchange has been important from the early days of computing.  A popular means of data exchange is connecting computers to one another.  
  
The File Transfer Protocol (FTP) is used to transfer files between two computers over a network and Internet.  In this article we will look at how to work with a FTP client.  [Auto FTP Manager](http://www.deskshare.com/ftp-client.aspx) is an advanced FTP client that automates file transfers between your computer and the FTP server.

**What is FTP?**

When you want to copy files between two computers that are on the same local network, often you can simply "share" a drive or folder, and copy the files the same way you would copy files from one place to another on your own PC.  
  
What if you want to copy files from one computer to another that is halfway around the world?  You would probably use your Internet connection.  However, for security reasons, it is very uncommon to share folders over the Internet.  File transfers over the Internet use special techniques, of which one of the oldest and most widely-used is [FTP.  **FTP,**](FTP.  FTP, )short for**"File Transfer Protocol,"**can transfer files between any computers that have an Internet connection, and also works between computers using totally different operating systems.  
  
Transferring files from a client computer to a server computer is called **"uploading"** and transferring from a server to a client is **"downloading".**



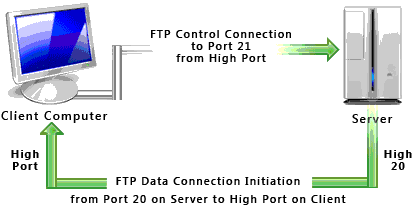
**Requirements for using FTP**

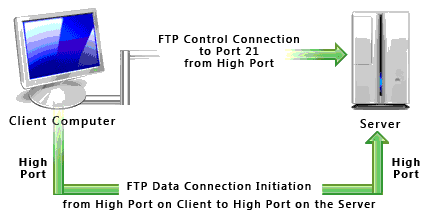
* An FTP client like Auto FTP Manager installed on your computer
* Certain information about the FTP server you want to connect to:  
  a.  The **FTP server address.** This looks a lot like the addresses you type to browse web sites.Example : Server address is "<ftp.videodesk.net>".  
  Sometimes the server address will be given as a numeric address, like "64.185.225.87".  
  b.  A user name and password.  Some FTP servers let you connect to them anonymously.  
  For anonymous connections, you do not need a user name and password.

To transfer files, provide your client software (Auto FTP Manager) with the server address, user name, and password.  After connecting to the FTP server, you can use Auto FTP Manager's **File Manager**to upload, download and delete files.  Using the File Manager is a lot like working with Windows Explorer.  
  
**FTP and Internet Connections**  
FTP uses one connection for commands and the other for sending and receiving data.  FTP has a standard port number on which the FTP server "listens" for connections.  A port is a "logical connection point" for communicating using the Internet Protocol (IP).  The standard port number used by FTP servers is 21 and is used only for sending commands.  Since port 21 is used exclusively for sending commands, this port is referred to as a **command port.** For example, to get a list of folders and files present on the FTP server, the FTP Client issues a "LIST" command.  The FTP server then sends a list of all folders and files back to the FTP Client.  So what about the internet connection used to send and receive data?  The port that is used for transferring data is referred to as a **data port.** The number of the data port will vary depending on the "mode" of the connection.  (See below for Active and Passive modes.)

**Active and Passive Connection Mode**

The FTP server may support **Active** or **Passive**connections, or both.  In an Active FTP connection, the client opens a port and listens and the server actively connects to it.  In a Passive FTP connection, the server opens a port and listens (passively) and the client connects to it.  You must grant Auto FTP Manager access to the Internet and to choose the right type of FTP ConnectionMode.  
  
Most FTP client programs select passive connection mode by default because server administrators prefer it as a safety measure.  Firewalls generally block connections that are "initiated" from the outside.  Using passive mode, the FTP client (like Auto FTP Manager) is "reaching out" to the server to make the connection.  The firewall will allow these outgoing connections, meaning that no special adjustments to firewall settings are required.  
  
If you are connecting to the FTP server using **Active mode**of connection you must set your firewall to accept connections to the port that your FTP client will open.  However, many Internet service providers block incoming connections to all ports above 1024.  Active FTP servers generally use port 20 as their data port.



It's a good idea to use **Passive mode** to connect to an FTP server.  Most FTP servers support the Passive mode.  For Passive FTP connection to succeed, the FTP server administrator must set his / her firewall to accept all connections to any ports that the FTP server may open.  However, this is the server administrator's problem (and standard practice for servers).  You can go ahead, make and use FTP connections.  
  


Once the FTP Client manages to open the internet connections, one for command and one for data, it starts communicating with the FTP server.  You are now ready to transfer your files and folders between the two connected computers with [Auto FTP Manager](http://www.deskshare.com/ftp-client.aspx).

**CONCLUSION:**

Hence, we have successfully executed File Transfer Protocols(FTP) using socket Programming.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO1:** Able to Execute File Transfer Protocol (FTP) using socket programming. ****

**QUESTIONS:**

* What is file transfer protocol (FTP)?
* What is FTP port number?
* What is TCP?
* Differentiate between TCP and UDP?
* Explain the Socket Programming?
* What are the system calls of Socket Programming?
* Explain connection establishment of socket programming ?
* Differentiate between FTP and TFTP?

**EXPERIMENT NO.5**

**Write TFTP program using socket programming for UDP using C++**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of TFTP,Socket Programming and UDP | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**Title: Write TFTP program using socket programming for UDP using C++**

**OBJECTIVES:**

1. To perform TFTP using socket programming.

2. To execute and analyze TFTP using socket programming.

**PROBLEM STATEMENT:**

Write TFTP program using socket programming for UDP using C++

**SOFTWARE REQUIRED:** Latest version of 64 Bit Operating Systems Open Source Fedora-20, Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading, 8 GB RAM, 500GB/1TB HDD, Latest versions of 64-Bit Programming languages such as Microsoft Visual Studio(ver. 12 or Higher) or equivalent open source, Eclipse 64-bit Platform.

**INPUT:** Input text file from user.

**OUTPUT:** file sent and file received status. And contents of the file.

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class Weather Report such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is constructor of the class

e = the end state or destructor of the class.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD- Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Weather\_Report:

s=initial state or constructor of the class weather()

s={weather()}- sets the default values for all five variables to respective values given in assignment.

Input i=(I1,I2)

I1={day\_of\_moth,hightemp,lowtemp,amount\_rain,amount\_snow}

Variables declared-NDD

Day\_of\_month={1,-------,n} n≠Ø. n=30 or 31.

Hightemp={1,------,n} n≠Ø. n=999.

lowtemp={-999,------,n} n≠Ø. n=6.

amount\_rain={1,------,n} n≠Ø. n=10.

amount\_snow={1,------,n} n≠Ø. n=10.

I2={99,999,-999,0,0}

Default values set all integers –Deterministic Data because of function weather(),memory requirement based on DD.

Hence let weather():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {display ( ), override ( ), avg ( )}

display ( )={details of all variable values}

override( )={ default values get overwritten and stored by values specified in this function}

I

I1 I2

( Default values) ( New values)

avg( )={sum of values / no. of days}

n

Σ/n

i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**THEORY:**

**TFTP**, or **Trivial File Transfer Protocol**, is a simple high-level protocol for transferring data servers use to boot diskless workstations, X-terminals, and routers by using User Data Protocol (UDP).

**How TFTP works?**

Trivial File Transfer Protocol (TFTP) is a simple light weight file transfer protocol, used for transferring files over the network. This protocol is similar to FTP but supports much lesser features and hence comes with a smaller foot print.

**What TFTP provides**

* Faster file transfer, as it uses UDP as the transport layer protocol
* Lesser Code size or foot print
* Ascii and binary modes of file transfer

**What TFTP does not provide**

* does not provide authentication
* does not support a rich set of user interface commands

**Use of TFTP**

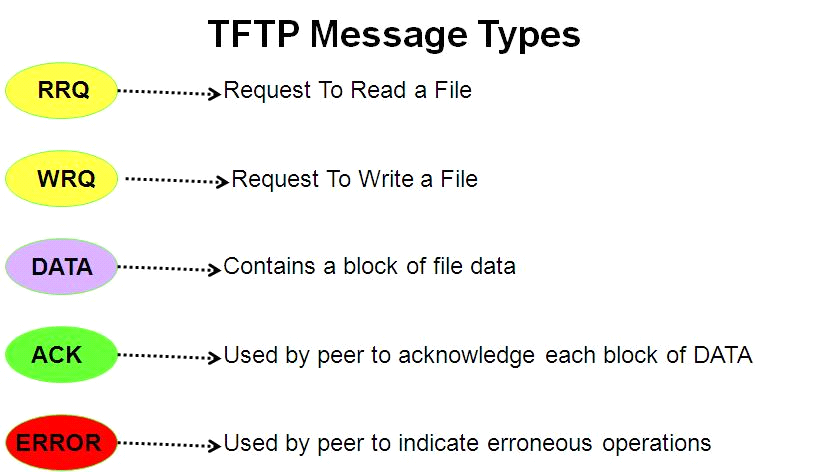
* TFTP is mainly used during device bootstrap process for downloading device OS/firmware and configuration files. It is typically used for copying bootstrap and configuration files between nodes belonging to the same LAN.
* TFTP is used in situations where all the features of a full file transfer protocol like FTP are not needed.
* It is used along with boot protocols like BOOTP and DHCP to initialize devices. Whenever an IP enabled node boots up, it gets its IP address and other device and network related parameters through BOOTP or DHCP. As part of these parameters, the client also receives the TFTP server address, bootstrap file and configuration file details (file name and directory location). The client then uses the TFTP protocol to download the bootstrap image and configuration files from the TFTP server.

**Basic Theory of Operation**

* TFTP is a client-server, application layer protocol, with TFTP clients running theTFTP client software and TFTP servers running the TFTP server software.
* TFTP uses UDP as the underlying transport layer protocol. Since UDP is much simpler when compared to the complicated TCP, it requires much lesser code space and hence TFTP can fit even inside small boot ROMs.
* TFTP servers wait on the well known UDP port number 69. A TFTP client, that wishes to send or receive files from the server, establishes a UDP connection to the server, by opening a UDP socket to the server’s IP address on port 69.
* The TFTP client then sends a read request (RRQ) to the server if it wants to get a file or sends a write request (WRQ) if it wants to transfer a file onto the server.
* TFTP splits a file, to be transferred, into blocks of size 512 bytes and transfers it as TFTP DATA messages. Each TFTP DATA block is numbered and carried inside separate UDP messages.
* The last block of a file is always sent with a size lesser than 512. When the peer receives a block with size less than 512 bytes, it treats that block as the last block of the file that is being transferred. Even if the file size happens to be an exact multiple of 512 bytes, TFTP sends a block with zero bytes as the final block, to indicate to the peer that the file transfer is over.
* **Reliability:** Each block is numbered and sent inside a separate UDP message. Since TFTP uses UDP, reliabile delivery of each block is not guaranteed by the underlying network protocols. So, TFTP itself takes care of reliability by requiring the peer to acknowledge each successfully received block.
* **Flow Control:**TFTP sends data block by block. After sending a block, the sending end starts a block timer. If an acknowledgment is received for the block from the peer before the timer expires, then the next block of the file is sent. Otherwise, the current block is resent as soon as the block timer expires and the whole process repeats itself till the block is successfully acknowledged. Hence, TFTP is basically a **stop and wait protocol** and flow control is achieved by the sender sending atmost one outstanding block at any instant of time.

TFTP messages

The TFTP protocol has basically 5 types of messages as given in the diagram below:



* The **RRQ** and **WRQ** messages are used by the client to request the server to start reading or writing a file respectively. Both these messages send the file name and transfer mode (ascii or binary) as additional parameters.
* The **DATA** messages carry the actual file blocks, with each message carrying a block of data. Each block has a sequence number field indicating the block number.
* The **ACK** message is used to acknowledge successfully received data blocks. It has the sequence number as the additional parameter, indicating the block number that was successfully received. Whenever a block is received error free (indicated by the UDP checksum), then the receiving TFTP node immediately acknowledges the block to the peer, by sending an ACK message.
* The **ERROR** message is sent to the peer whenever some operation could not be performed (e.g. invalid file name, file does not have read/write permissions etc).
* TFTP protocol has been enhanced to allow for additional option negotiations like initial sequence number, block size etc.

**CONCLUSION:**

Hence, we have successfully executed TFTP of UDP using socket programming.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO:** Able to Execute Trivial File Transfer Protocol(TFTP) of UDP using socket programming. ****

**QUESTIONS:**

* What is Trivial file transfer protocol (TFTP)?
* What is FTP port number?
* What is TCP?
* Differentiate between TCP and UDP?
* Explain the Socket Programming?
* What are the system calls of Socket Programming?
* Explain connection establishment of socket programming ?
* Differentiate between FTP and TFTP?

**EXPERIMENT NO.6**

**Implement any congestion control algorithm for TCP using Python**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of congestion control algorithm in TCP. | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**Title: Implement any congestion control algorithm for TCP using Python**

**OBJECTIVES:**

1. To Understand the concept of congestion control in TCP.

2. To implement congestion control using TCP.

**PROBLEM STATEMENT:**

Implement any congestion control algorithm for TCP using Python.

**SOFTWARE REQUIRED:** Latest version of 64 Bit Operating Systems Open Source Fedora-20, Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading, 8 GB RAM, 500GB/1TB HDD, Latest versions of 64-Bit Programming languages such as Microsoft Visual Studio(ver. 12 or Higher) or equivalent open source, Eclipse 64-bit Platform.

**INPUT:** Incoming packets stream .

**OUTPUT:** Congestion control of packets stream.

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class Weather Report such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is constructor of the class

e = the end state or destructor of the class.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD- Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Weather\_Report:

s=initial state or constructor of the class weather()

s={weather()}- sets the default values for all five variables to respective values given in assignment.

Input i=(I1,I2)

I1={day\_of\_moth,hightemp,lowtemp,amount\_rain,amount\_snow}

Variables declared-NDD

Day\_of\_month={1,-------,n} n≠Ø. n=30 or 31.

Hightemp={1,------,n} n≠Ø. n=999.

lowtemp={-999,------,n} n≠Ø. n=6.

amount\_rain={1,------,n} n≠Ø. n=10.

amount\_snow={1,------,n} n≠Ø. n=10.

I2={99,999,-999,0,0}

Default values set all integers –Deterministic Data because of function weather(),memory requirement based on DD.

Hence let weather():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {display ( ), override ( ), avg ( )}

display ( )={details of all variable values}

override( )={ default values get overwritten and stored by values specified in this function}

I

I1 I2

( Default values) ( New values)

avg( )={sum of values / no. of days}

n

Σ/n

i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**THEORY:**

• Problem: When too many packets are transmitted through a network, congestion occurs at very high traffic, performance collapses completely, and almost no packets are delivered

• Causes: bursty nature of traffic is the root cause → When part of the network no longer can cope a sudden increase of traffic, congestion builds upon. Other factors, such as lack of bandwidth, ill-configuration and slow routers can also bring up congestion Maximum carrying  
 capacity of subnet Packets delivered Perfect Desirable Congested Packets sent.

• Solution: congestion control, and two basic approaches – Open-loop: try to prevent congestion occurring by good design – Closed-loop: monitor the system to detect congestion, pass this information to where action canbe taken, and adjust system operation to correct the problem (detect, feedback and correct)

• Differences between congestion control and flow control: – Congestion control try to make sure subnet can carry offered traffic, a global issue involving all the hosts and routers. It can be open-loop based or involving feedback – Flow control is related to point-to-point traffic between given sender and receiver, it always involves direct feedback from receiver to sender.

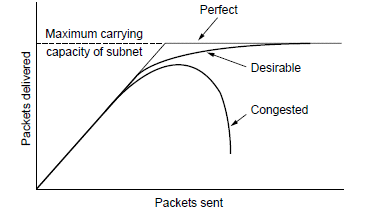


Fig: Congestion control mechanism.

**Standard TCP Congestion Control Algorithms.**

The standard fare in TCP implementations today can be found in RFC 2581 [2]. This reference document specifies four standard congestion control algorithms that are now incommon use. Each of the algorithms noted within that document was actually designed long before the standard was published . Their usefulness has passed the test of time.

The four algorithms, Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery are described below.

**Slow Start**

Slow Start, a requirement for TCP software implementations is a mechanism used by the sender to control the transmission rate, otherwise known as sender-based flow control. This is accomplished through the return rate of acknowledgements from the receiver. Inother words, the rate of acknowledgements returned by the receiver determine the rate at which the sender can transmit data.

When a TCP connection first begins, the Slow Start algorithm initializes a congestion window to one segment, which is the maximum segment size (MSS) initialized by the receiver during the connection establishment phase. When acknowledgements are returned by the receiver, the congestion window increases by one segment for each acknowledgement returned. Thus, the sender can transmit the minimum of the congestion window and the advertised window of the receiver, which is simply called the transmission window. Slow Start is actually not very slow when the network is not congested and network response time is good. For example, the first successful transmission and acknowledgement of a TCP segment increases the window to two segments. After successful transmission of these two segments and acknowledgements completes, the window is increased to four segments. Then eight segments, then sixteen segments and so on, doubling from there on out up to the maximum window size advertised by the receiver or until congestion finally does occur.

At some point the congestion window may become too large for the network or network

Conditions may change such that packets may be dropped. Packets lost will trigger a timeout at the sender. When this happens, the sender goes into congestion avoidance mode as described in the next section.

**Congestion Avoidance**

During the initial data transfer phase of a TCP connection the Slow Start algorithm is used. However, there may be a point during Slow Start that the network is forced to drop one or more packets due to overload or congestion. If this happens, Congestion Avoidance is used to slow the transmission rate. However, Slow Start is used in conjunction with Congestion Avoidance as the means to get the data transfer going again so it doesn’t slow down and stay slow. In the Congestion Avoidance algorithm a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver’s advertised window size), but to at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start mode. If congestion was indicated by duplicate ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked (see below).

As data is received during Congestion Avoidance, the congestion window is increased. However, Slow Start is only used up to the halfway point where congestion originallyoccured. This halfway point was recorded earlier as the new transmission window. After this halfway point, the congestion window is increased by one segment for all segments in the transmission window that are acknowledged. This mechanism will force the sender to more slowly grow its transmission rate, as it will approach the point where congestion had previously been detected.

**Fast Retransmit**

When a duplicate ACK is received, the sender does not know if it is because a TCP segment was lost or simply that a segment was delayed and received out of order at the receiver. If the receiver can re-order segments, it should not be long before the receiver sends the latest expected acknowledgement. Typically no more than one or two duplicate ACKs should be received when simple out of order conditions exist. If however more than two duplicate ACKs are received by the sender, it is a strong indication that at least one segment has been lost. The TCP sender will assume enough time has lapsed for all segments to be properly re-ordered by the fact that the receiver had enough time to send three duplicate ACKs.

When three or more duplicate ACKs are received, the sender does not even wait for a

retransmission timer to expire before retransmitting the segment (as indicated by the position of the duplicate ACK in the byte stream). This process is called the Fast Retransmit algorithm. Immediately following Fast Retransmit is the Fast Recovery algorithm.

**3.4 Fast Recovery**

Since the Fast Retransmit algorithm is used when duplicate ACKs are being received, the TCP sender has implicit knowledge that there is data still flowing to the receiver. Why? The reason is because duplicate ACKs can only be generated when a segment is received. This is a strong indication that serious network congestion may not exist and that the lost segment was a rare event. So instead of reducing the flow of data abruptly by going all the way into Slow Start, the sender only enters Congestion Avoidance mode. Rather than start at a window of one segment as in Slow Start mode, the sender resumes transmission with a larger window, incrementing as if in Congestion Avoidance mode.

This allows for higher throughput under the condition of only moderate congestion. To summarize this section of the paper, figure 2 below depicts what a typical TCP data transfer phase using TCP congestion control might look like. Notice the periods of exponential window size increase, linear increase and drop-off. Each of these scenarios depicts the sender’s response to implicit or explicit signals it receives about network conditions.

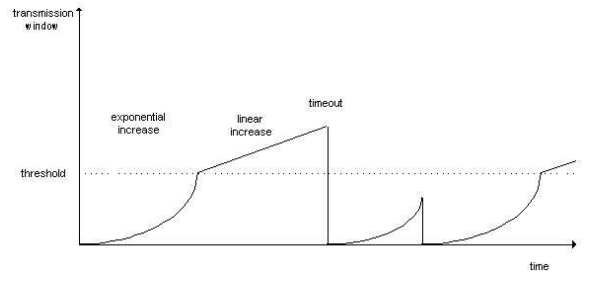


Fig :Congestion Control Overview

**CONCLUSION:**

Hence, we have successfully executed congestion control in TCP and remove the traffic hence solve the problem of congestion.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO:** Able to Execute congestion control algorithm in TCP. ****

**QUESTIONS:**

* What is congestion control in TCP?
* How congestion occurs?
* What are the different methods to solve the congestion control ?
* What are advantages of congestion control algorithm.
* What is congestion avoidance ?

**EXPERIMENT NO.8**

**Implementation of concurrent text conferencing application using python or Java**

**Session Plan**

**TITLE:** Implementation of Concurrent Text Conferencing application using Python or Java

**OBJECTIVES:**

1.Students should be able to implement concurrent Text Conferencing application using Python or Java .

**PROBLEM STATEMENT:**

Implementation of Concurrent Text Conferencing application using Python or Java

**Hardware Requirement:**  8 GB RAM, 500GB/1TB HDD

**Software Requirement:** Latest version of 64 Bit Operating Systems Open Source Fedora-20, NetBeans 8.0

**INPUT:** Text conferencing data/packets from different clients

**OUTPUT:** Interaction of Text conferencing data/packets among different clients through server in a LAN.

**THEORY:**

The Concurrent text conferencing include the LAN Chat Server is a program that functions as the server side of the chat software for the local area network.

The server side of the program allows all the clients of the software to connect with one to each other. In order to operate this program correctly, you will need the client side of this program (different distribution).

**Text Messaging**

It combines everything necessary for text chat between users: chatting in channels, message exchange, animated emoticons, and a bulletin board. All text information is transferred in Unicode encoding.

**Image Exchange**

It allows image exchange that is quick, comfortable and effective. Images inserted into public channels, private channels or messages are automatically compressed to ensure maximum comfort and minimum network load.

**File Transfer**

It enables users to exchange files in the fastest and most comfortable way. Just drag and drop the necessary files or folders onto the recipient’s image in the list and the program will begin the file transfer. Advanced buffering mechanisms make the most effective possible use of networks with a bandwidth from 32 Kbit/s to 1 000 Mbit/s. Resumed downloads allow users to finish a file transfer even in cases where the connection is broken for up to 5 minutes.

**Program:**

package com.socket;

import java.io.Serializable;

public class Message implements Serializable{

private static final long serialVersionUID = 1L;

public String type, sender, content, recipient;

public Message(String type, String sender, String content, String recipient){

this.type = type; this.sender = sender; this.content = content; this.recipient = recipient;

}

@Override

public String toString(){

return "{type='"+type+"', sender='"+sender+"', content='"+content+"', recipient='"+recipient+"'}";

}

}

**CONCLUSION:**

Hence, we have successfully studied concurrent Text Conferencing application using Python or Java .

**OUTCOME**

**Upon completion Students will be able to:**

**ELO:** perform concurrent Text Conferencing application using Python or Java .

**QUESTIONS:**

**What is Text conferencing in LAN?**

**Explain the concept of concurrent Text Conferencing?**

**What are different steps for performing Concurrent text conferencing?**

**EXPERIMENT NO.9**

**Implementation of concurrent Proxy Server**

**Session Plan**

**TITLE:** Concurrent Proxy server

**PROBLEM STATEMENT**

Implementation of Concurrent Proxy server program using Python or Java

**OBJECTIVE**

Student should be able to execute program of Concurrent Proxy server using Python or Java .

**Hardware requirement:**

Computer

**Software requirement:**

Latest version of 64 Bit Operating Systems Open Source Fedora-20,

Latest versions of 64-Bit Programming languages, Eclipse 64-bit Platform C++, gcc compiler.

**Theory**

**Description:**

**Server:**

A server is a running instance of an application (software) capable of accepting requests from the client and giving responses accordingly. Servers can run on any computer including dedicated computers, which individually are also often referred to as "the server". In many cases, a computer can provide several services and have several servers running. The advantage of running servers on a dedicated computer is security. For this reason most of the servers are daemon processes and designed in that they can be run on specific computer(s).

Servers operate within client-server architecture. Servers are computer programs running to serve the requests of other programs, the clients. Thus, the server performs some tasks on behalf of clients. It facilitates the clients to share data, information or any hardware and software resources. The clients typically connect to the server through the network but may run on the same computer. In the context of Internet Protocol (IP) networking, a server is a program that operates as a socket listener.

Servers often provide essential services across a network, either to private users inside a large organization or to public users via the Internet. Typical computing servers are database server, file server, mail server, print server, web server, gaming server, and application server.

**Proxy Server**

In computer networks, a proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource available from a different server and the proxy server evaluates the request as a way to simplify and control its complexity. Proxies were invented to add structure and encapsulation to distributed systems. Today, most proxies are web proxies, facilitating access to content on the World Wide Web and providing anonymity.

A proxy server is a hardware or software system that acts as an intermediary between an endpoint device and another server from which that device is requesting a service.

In the enterprise, a proxy server is used to facilitate security, administrative control or caching service, among other possibilities. In a personal computing context, proxy servers are used to enable user privacy and anonymous surfing. Proxy servers are used for both legal and illegal purposes.

On corporate networks, a proxy server is associated with -- or is part of -- a gateway server that separates the network from external networks (typically the Internet) and a firewall that protects the network from outside intrusion. A proxy server may exist in the same machine with a firewall server or it may be on a separate server and forward requests through the firewall. Proxy servers are used for both legal and illegal purposes.

When a proxy server receives a request for an Internet service (such as a Web page request), it looks in its local cache of previously downloaded Web pages. If it finds the page, it returns it to the user without needing to forward the request to the Internet. If the page is not in the cache, the proxy server, acting as a client on behalf of the user, uses one of its own IP addresses to request the page from the server out on the Internet. When the page is returned, the proxy server relates it to the original request and forwards it on to the user.

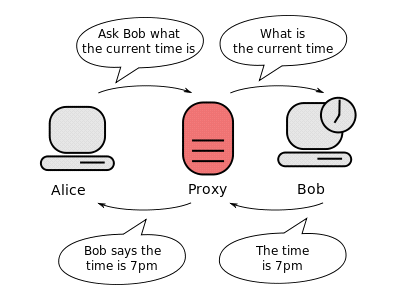


Fig.9.1: Proxy Server

Communication between two computers (shown in grey) connected through a third computer (shown in red) acting as a proxy. Bob does not know whom the information is going to, which is why proxies can be used to protect privacy.

**Proxy servers have two main purposes:**

* **Improve Performance:** Proxy servers can dramatically improve performance for groups of users. This is because it saves the results of all requests for a certain amount of time. Consider the case where both user X and user Y access the World Wide Web through a proxy server. First user X requests a certain Web page, which we'll call Page 1. Sometime later, user Y requests the same page. Instead of forwarding the request to the Web server where Page 1 resides, which can be a time-consuming operation, the proxy server simply returns the Page 1 that it already fetched for user X. Since the proxy server is often on the same network as the user, this is a much faster operation. Real proxy servers support hundreds or thousands of users. The major online services such as America Online, MSN and Yahoo, for example, employ an array of proxy servers.
* **Filter Requests:** Proxy servers can also be used to filter requests. For example, a company might use a proxy server to prevent its employees from accessing a specific set of Web sites.

**Types of proxy**

A proxy server may reside on the user's local computer, or at various points between the user's computer and destination servers on the Internet.

* A proxy server that passes requests and responses unmodified is usually called a [gateway](http://en.wikipedia.org/wiki/Gateway_(computer_networking)) or sometimes a tunneling proxy.
* A forward proxy is an Internet-facing proxy used to retrieve from a wide range of sources (in most cases anywhere on the Internet).
* A [reverse proxy](http://en.wikipedia.org/wiki/Reverse_proxy) is usually an Internet-facing proxy used as a front-end to control and protect access to a server on a private network. A reverse proxy commonly also performs tasks such as load-balancing, authentication, decryption or caching.

**Open proxies**

An open proxy is a forwarding proxy server that is accessible by any Internet user. Gordon Lyon estimates there are "hundreds of thousands" of open proxies on the Internet. An anonymous open proxy allows users to conceal their IP address while browsing the Web or using other Internet services. There are varying degrees of anonymity however, as well as a number of methods of 'tricking' the client into revealing itself regardless of the proxy being used.

**Reverse proxies**

A reverse proxy (or surrogate) is a proxy server that appears to clients to be an ordinary server. Requests are forwarded to one or more proxy servers which handle the request. The response from the proxy server is returned as if it came directly from the origin server, leaving the client no knowledge of the origin servers.[3] Reverse proxies are installed in the neighborhood of one or more web servers. All traffic coming from the Internet and with a destination of one of the neighborhood's web servers goes through the proxy server. The use of "reverse" originates in its counterpart "forward proxy" since the reverse proxy sits closer to the web server and serves only a restricted set of websites. There are several reasons for installing reverse proxy servers:

* Encryption / SSL acceleration: when secure web sites are created, the SSL encryption is often not done by the web server itself, but by a reverse proxy that is equipped with SSL acceleration hardware. See [Secure Sockets Layer](http://en.wikipedia.org/wiki/Secure_Sockets_Layer). Furthermore, a host can provide a single "SSL proxy" to provide SSL encryption for an arbitrary number of hosts; removing the need for a separate SSL Server Certificate for each host, with the downside that all hosts behind the SSL proxy have to share a common DNS name or IP address for SSL connections. This problem can partly be overcome by using the SubjectAltName feature of [X.509](http://en.wikipedia.org/wiki/X.509) certificates.
* [Load balancing](http://en.wikipedia.org/wiki/Load_balancing_(computing)): the reverse proxy can distribute the load to several web servers, each web server serving its own application area. In such a case, the reverse proxy may need to rewrite the URLs in each web page (translation from externally known URLs to the internal locations).
* Serve/cache static content: A reverse proxy can offload the web servers by caching static content like pictures and other static graphical content.
* Compression: the proxy server can optimize and compress the content to speed up the load time.
* Spoon feeding: reduces resource usage caused by slow clients on the web servers by caching the content the web server sent and slowly "spoon feeding" it to the client. This especially benefits dynamically generated pages.
* Security: the proxy server is an additional layer of defense and can protect against some OS and Web Server specific attacks. However, it does not provide any protection from attacks against the web application or service itself, which is generally considered the larger threat.
* Extranet Publishing: a reverse proxy server facing the Internet can be used to communicate to a firewall server internal to an organization, providing extranet access to some functions while keeping the servers behind the firewalls. If used in this way, security measures should be considered to protect the rest of your infrastructure in case this server is compromised, as its web application is exposed to attack from the Internet.

**Advantages of Proxy Server:**

An advantage of a proxy server is that its cache can serve all users. If one or more Internet sites are frequently requested, these are likely to be in the proxy's cache, which will improve user response time. A proxy can also log its interactions, which can be helpful for troubleshooting.

**Concurrent Proxy server:**

**ProxyServer.Java : -**

package proxy;

import java.net.\*;

import java.io.\*;

public class ProxyServer {

public static void main(String[] args) throws IOException {

ServerSocket serverSocket = null;

boolean listening = true;

int port = 10000; //default

try {

port = Integer.parseInt(args[0]);

} catch (Exception e) {

//ignore me

}

/\*\*\* TODO: 1. Create a proxy server socket to listen request from client\*\*\*/

while (listening)

{

new ProxyThread(serverSocket.accept()).start();

}

serverSocket.close();

}

}

**ProxyThread.java-**

package proxy;

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class ProxyThread extends Thread {

private Socket socket = null;

private static final int BUFFER\_SIZE = 32768;

public ProxyThread(Socket socket) {

super("ProxyThread");

this.socket = socket;

}

public void run() {

//get input from user

//send request to server

//get response from server

//send response to user

try {

DataOutputStream out =

new DataOutputStream(socket.getOutputStream());

BufferedReader in = new BufferedReader(

new InputStreamReader(socket.getInputStream()));

String inputLine, outputLine;

int cnt = 0;

String urlToCall = "";

///////////////////////////////////

//begin get request from client

while ((inputLine = in.readLine()) != null) {

try {

StringTokenizer tok = new StringTokenizer(inputLine);

tok.nextToken();

} catch (Exception e) {

break;

}

//parse the first line of the request to find the url

if (cnt == 0) {

String[] tokens = inputLine.split(" ");

urlToCall = tokens[1];

//can redirect this to output log

System.out.println("Request for : " + urlToCall);

}

cnt++;

}

//end get request from client

///////////////////////////////////

BufferedReader rd = null;

try {

//begin send request to server, get response from server

/\* \* \* TODO: 2. Create a URL object and assign it with urlToCall.\* \* \*/

conn.setDoInput(true);

//not doing HTTP posts

conn.setDoOutput(false);

// Get the response

InputStream is = null;

HttpURLConnection huc = (HttpURLConnection)conn;

if (conn.getContentLength() > 0) {

try {

is = conn.getInputStream();

rd = new BufferedReader(new InputStreamReader(is));

} catch (IOException ioe) {

System.out.println("\*\*\*\*\*\*\*\*\* IO EXCEPTION \*\*\*\*\*\*\*\*\*\*: " + ioe);

}

}

//end send request to server, get response from server

///////////////////////////////////

///////////////////////////////////

//begin send response to client

byte by[] = new byte[ BUFFER\_SIZE ];

int index = is.read( by, 0, BUFFER\_SIZE );

while ( index != -1 )

{

out.write( by, 0, index );

index = is.read( by, 0, BUFFER\_SIZE );

}

out.flush();

//end send response to client

///////////////////////////////////

} catch (Exception e) {

//can redirect this to error log

System.err.println("Encountered exception: " + e);

//encountered error - just send nothing back, so

//processing can continue

out.writeBytes("");

}

//close out all resources

if (rd != null) {

rd.close();

}

if (out != null) {

out.close();

}

if (in != null) {

in.close();

}

if (socket != null) {

socket.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**CONCLUSION:**

Hence, we have successfully studied and executed concept concurrent proxy server using Java.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO:** Execute program of Concurrent Proxy server using Java.

**QUESTIONS:**

* What is mean by Server?
* What is mean by proxy server?
* What is the difference between server and proxy server?
* What is mean by concurrent proxy server?
* What are types of proxy server?
* What are the different advantages of proxy server?
* What is the main purpose of proxy server?
* How we implement the concurrent proxy server in programming?
* What is the difference between proxy server and concurrent proxy server?
* How concurrent proxy server works in network

**EXPERIMENT NO.10**

**Implementation of Multithreaded web Server**

**EXPERIMENT NO.11**

**Implement a program for remote print manager to print documents on remote printer**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of Remote print manager & IPP | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**TITLE:**  **Remote print manager to print documents on remote printer**

**OBJECTIVES:**

* To learn how Remote Print Manager (RPM) Works.
* To use RPM to print file on remote IP based printer.

**PROBLEM STATEMENT:**

Implement a program for remote print manager to print documents on remote printer. (IP based printer) using Python.

**REQUIREMENTS:**

Hardware: IP Based Printer

**INPUT:** Any user defined file to print.

**OUTPUT:** File after printing.

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class WAN Simulation such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is initial data.

e = the end state.

i= input of the system.

o=output of the system.

DD- Deterministic data it helps identifying the load store functions or assignment functions.

NDD - Non deterministic data of the system S to be solved.

Success- Desired outcome generated.

Failure- Desired outcome not generated or forced exit due to system error.

For class IP\_Printer:

s=initial state()

s= {IP()}- sets the default values for all four variables to respective values given in assignment.

Input i= (I1, I2)

I1= {LAN, PC, Printer, Router}

Variables declared-NDD

LAN= {1,-------,n} n≠Ø. n=999.

PC= {1,------,n} n≠Ø. n=99.

Priter= {1,------,n} n≠Ø. n=1.

Router={1,------,n} n≠Ø. n=10.

I2={999,99,1,10}

Default values set all integers –Deterministic Data because of function IP(),memory requirement based on DD.

Hence let IP():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= { Insert( ), Print ( )}

Insert( ) = { Select file to send to printer}

Print ( ) = {details of all files to print}

I

I1 I2

(Default values) (New values)

avg( )={sum of values / no. of days}

n

Σ/n

i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**Theory:**

**Remote Printing**

Standard TCP/IP Port Monitor (SPM) is designed for Windows 2000 print servers that communicate with shared printers using TCP/IP. This includes network-ready printers, network adapters like the Hewlett-Packard JetDirect, and external network boxes like the Intel NetPort. SPM can support many printers on one server and is faster and easier to configure than the LPR Port Monitor. SPM is also compatible with RFC 1759, the standard for the Simple Network Management Protocol (SNMP) As a result; SPM provides much more detailed status than LPR.

**Prerequisites**

To use SPM, TCP/IP must be installed on the print server so it can talk to the print device. Clients do not need TCP/IP; any common network protocol such as SMB, NetWare Core Protocol (NCP), LPR, AppleTalk, or NetBEUI can be used. The transport protocol is not important because only the Windows 2000 print server communicates with the print device.

For example, if both the client and the server have the Internetwork Packet Exchange (IPX) protocol, the client can send the document using IPX. The server can send the document to the printer over TCP/IP. Figure B.12.1 illustrates these connections.

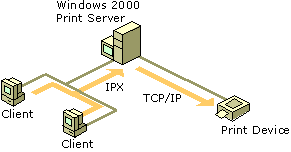


Fig. 11.1 **Protocol Links for Remote Printing**

**Internet Printing Protocol:**

In computing, the **Internet Printing Protocol** (**IPP**) provides a standard network protocol for remote printing as well as for managing print jobs, media size, resolution, and so forth. Like all IP-based protocols, IPP can run locally or over the Internet. Unlike other printing protocols, IPP also supports access control, authentication, and encryption, making it a much more capable and secure printing mechanism than older ones.

IPP began as a proposal by Novell for the creation of an Internet printing protocol project in 1996. The result was a draft written by Novell and Xerox called the Lightweight Document Printing Application (LDPA). At about the same time, Lexmark publicly proposed something called the Hypertext Printing Protocol (HTPP), and both HP and Microsoft had started work on new print services for what became MS Windows 2000. Each of the companies chose to start a common Internet Printing Protocol project in the Printer Working Group (PWG) and negotiated an IPP birds-of-a-feather (BOF) session with the Application Area Directors in the Internet Engineering Task Force (IETF). The BOF session in December 1996(?) showed sufficient interest in developing a printing protocol, leading to the creation of the IETF Internet Printing Protocol (ipp) working group. IPP/1.0 was published as a series of experimental documents (RFC 2565, RFC 2566, RFC 2567, RFC 2568, RFC 2569, and RFC 2639) in 1999. IPP/1.1 followed as a draft standard in 2000 with support documents in 2001 and 2003 (RFC 2910, RFC 2911, RFC 3196, RFC 3510]). Additional extensions to IPP were published as RFCs until 2005 when the IETF IPP working group was concluded.

Work on IPP continues in the PWG with the publication of 12 candidate standards providing extensions to IPP and definition of IPP/2.0, IPP/2.1, and now IPP/2.2 representing different categories or classes of printers. A new IPP Everywhere project began in July 2010 to define an IPP profile and extensions required to support driverless printing, with a focus on non-traditional platforms such as netbooks and mobile Internet devices. The new project also expands the scope of IPP standards to include printer discovery and standard document formats.

**Implementation**

IPP is implemented using the Hypertext Transfer Protocol (HTTP) and inherits all of the HTTP streaming and security features. For example, authorization can take place via HTTP's Digest access authentication mechanism, GSSAPI, or via public key certificates. Encryption is provided using the SSL/TLS protocol-layer, either in the traditional always-on mode used by HTTPS or using the HTTP Upgrade extension to HTTP (RFC 2817). Streaming is supported using HTTP chunking.

IPP uses the traditional client-server model, with clients sending IPP request messages with the MIME media type "application/ipp" in HTTP POST requests to an IPP printer. IPP request messages consist of key/value pairs using a custom binary encoding followed by an "end of attributes" tag and any document data required for the request. The IPP response is sent back to the client in the HTTP POST response, again using the "application/ipp" MIME media type.

Among other things, IPP allows a client to:

* Query a printer's capabilities
* Submit print jobs to a printer
* Query the status of a printer
* Query the status of one or more print jobs
* Cancel previously submitted jobs

IPP uses TCP with port 631 as its well-known port. IPP implementations such as CUPS also use UDP with port 631 for IPP printer discovery. Products using the Internet Printing Protocol include, among others, CUPS which is part of Apple Mac OS X and many BSD and Linux distributions and is the reference implementation for IPP/2.0 and IPP/2.1, Novell [iPrint](http://en.wikipedia.org/wiki/IPrint), and Microsoft Windows, starting with MS Windows 2000. MS Windows XP and Windows Server 2003 offer IPP printing via HTTPS. MS Windows Vista, Windows 7, Windows Server 2008 and 2008 R2 also support IPP printing over RPC in the "Medium-Low" security zone.

**Conclusion:**

Hence, we have successfully studied & implemented Remote print Manager to print on IP based printer.

**OUTCOME**

**Upon completion Students will be:**

**ELO:** Able to explain Internet protocol**. **

**ELO:** Able to apply IP Protocol to print a file on remote IP based printer. 

**QUESTIONS:**

* How remote printing is done?
* Difference between Remote & Local Printing.
* What is Cloud Printing?
* Applications of Remote Printing.
* Difference between Remote & Internet Printing.
* Difference between Remote & Cloud Printing.
* Explain IPP.
* Can we use remote printing for specific area only?
* What is range over which Remote printing been used?
* Procedure for Remote Printing.

**Program B12:**

import subprocess, os

printer='Deskjet\_1510'

pdffile='output.pdf'

cmd = 'lp -d %s %s' %(printer, pdffile)

#os.startfile("output.pdf", "print")

os.system(cmd)

lpr = subprocess.Popen("/usr/bin/lpr", stdin=subprocess.PIPE)

#proc = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

#stdout, stderr=proc.communicate()

#exit\_code=proc.wait()

#lpr.stdin.write("This is a testing by kaitec")

**EXPERIMENT NO.12**

**Implementation of Sliding Window Protocol.**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of Sliding Window Protocol. | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**TITLE:** **Implementation of Sliding Window Protocol Using C++**

**OBJECTIVES:**

1. To study Sliding window Protocol working.

2. To express and apply the concepts in C++.

**PROBLEM STATEMENT:**

Implement The Sliding Window Protocol using C++.

**SOFTWARE REQUIRED**: Fedora 20.

**INPUT:** Input data as default values and user defined values.

**OUTPUT:** It will show the working of Sliding Window protocol..

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class sel\_repeat such that

S={s, r, i,t,o, DD, NDD, success, failure}

s=Sender side window.

r= Reciever side window.

i= input of the system.

t= timer

o=output of the system.

DD- deterministic data.

NDD- Non deterministic.

Success- Reciever gets all frames sent from sender.

Failure- If transmission Outs.

**THEORY:**

A **sliding window protocol** is a feature of packet-based data transmission **protocols**. **Sliding window protocols** are used where reliable in-order delivery of packets is required, such as in the Data Link Layer (OSI model) as well as in the Transmission Control **Protocol** (TCP).

The sliding window protocol is a useful protocol in network communications. Many important protocol such as TCP, SPX etc, carry the idea of Sliding window protocol. The sliding window protocol provides flow control in data communication over an unreliable connection. The key feature of the sliding window protocol is that it permits pipelined communication. In contrast, with a simple stop-and-wait protocol, the sender waits for an acknowledgment after transmitting every frame. As a result, there is at most a single outstanding frame on the channel at any given time, which may be far less than the channel's capacity. For maximum throughput, the amount of data in transit at any given time should be equal to (channel bandwidth) X (channel delay).

Originally, the TFTP protocol use a stop and wait protocol. This is a simple but not efficient protocol. In my project I tried to modify it to be a go back N sliding window protocol and made some statistics.

There are three types of Sliding Window protocols. The three differ among themselves in terms of efficiency, complexity, and buffer requirements. In all sliding window protocols, each outbound frame contains a sequence number, ranging from 0 up to some maximum. The maximum is usually 2n-1 so the sequence number fits nicely in an n-bit field. The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence number corresponding to frames it is permitted to send. These frames are said to fall within the sending window. Similarly there is a receiving window corresponding to the set of frames it is permitted to accept on receiver side.

**Sliding window protocol works this way:**

The sequence number within the sender’s window, represent frames sent out but as yet not acknowledged. Whenever a new packet arrives from the network layer, it is given the next highest sequence number, and the upper edge of the window is advanced by one. When an acknowledgment arrives, the lower edge of the window is advanced by one. So the window continuously maintain a list of unacknowledgement frames.

Since frame currently within the sender’s window may ultimately be lost or damaged in transit, the sender must keep all these frames in its memory for possible retransmission. Thus if the window size is n, the sender needs n buffers to hold all unacknowledgement frames. If the window ever grows to its maixmum size, the sender must forcibly shut off the network layer until another buffer becomes free.

The receiving data link layer’s window corresponds to the frames it may accept. Any frame falling outside the window is discarded without comment. Only the packets with sequence number fall in the receiver’s window size is accepted. And an acknowlegment will be generate accordingly. Other packets will be discarded. If the receiver’s window size is equal to 1, the receiver will only accept frames in order.

The most complicate sliding window protocol is Selective Repeat. Both sender and receiver maintains a window which size is greater than 1. Because the receiver has a window size greater than 1, that means the receiver can accept frames out of order as long as the sequence number is within the window. So certain method has to be applied to send ordered frame to upper layer.

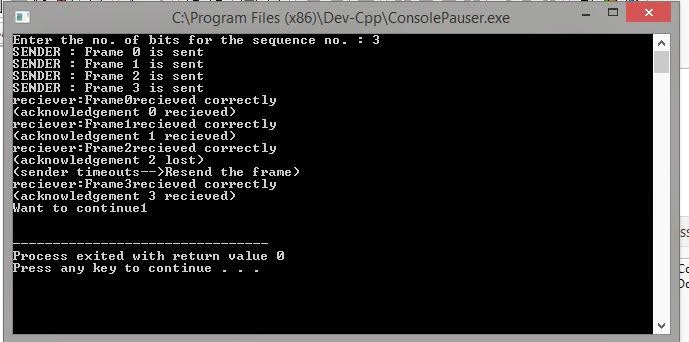
The most simply sliding window protocol is that the window size equal to 1. This is stop and wait protocol. Which is used in TFTP protocol. I enhanced my TFTP program by using a go back N protocol instead of stop and wait protocol. To provide a higher efficiency.

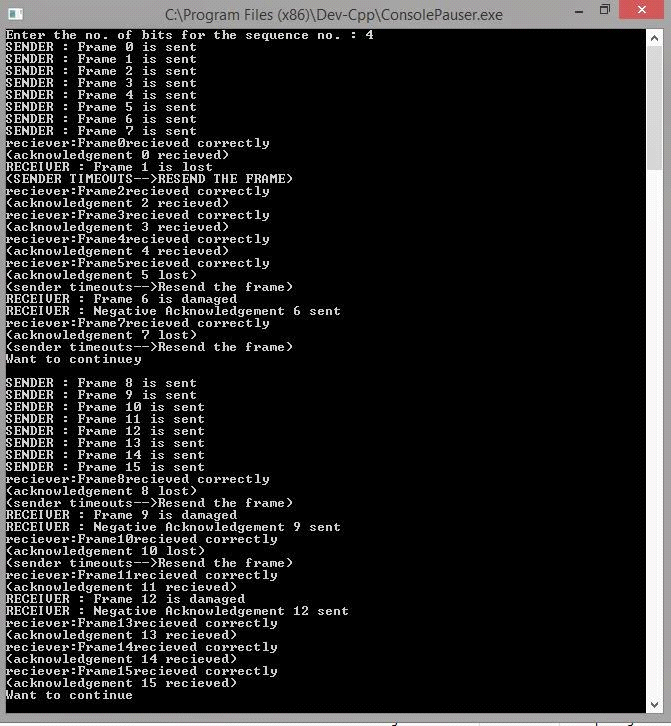
In go back n protocol, the sender’s window size is greater than 1. The receiver’s window size is 1. So the sender can send multiple frames before it receives an ackolegement. The receiver will discards all frames which sequence number is greater than the next one. If the expected frame lost during transmission, the sender will eventually get a time out event and retransmit this frame and flowing frame.

**Program:**

#include<iostream>  
using namespace std;  
#include<conio.h>  
  
#include<stdlib.h>  
  
#include<time.h>  
  
#include<math.h>  
  
  
  
#define TOT\_FRAMES 500  
  
#define FRAMES\_SEND 10  
  
  
  
class sel\_repeat  
  
{  
  
private:  
  
int fr\_send\_at\_instance;  
  
int arr[TOT\_FRAMES];  
  
int send[FRAMES\_SEND];  
  
int rcvd[FRAMES\_SEND];  
  
char rcvd\_ack[FRAMES\_SEND];  
  
int sw;  
  
int rw;       //tells expected frame  
  
public:  
  
void input();  
  
void sender(int);  
  
void receiver(int);  
  
};  
  
void sel\_repeat::input()  
  
{  
  
int n;     //no. of bits for the frame  
  
int m;    //no. of frames from n bits  
int i;  
cout<<"Enter the no. of bits for the sequence no. : ";  
  
cin>>n;  
  
m=pow(2,n);  
  
int t=0;  
  
fr\_send\_at\_instance=(m/2);  
  
for(i=0;i<TOT\_FRAMES;i++)  
  
{  
  
arr[i]=t;  
  
t=(t+1)%m;  
  
}  
  
for(i=0;i<fr\_send\_at\_instance;i++)  
  
{  
  
send[i]=arr[i];  
  
rcvd[i]=arr[i];  
  
rcvd\_ack[i]='n';  
  
}  
  
rw=sw=fr\_send\_at\_instance;  
  
sender(m);  
  
}  
  
void sel\_repeat::sender(int m)  
  
{  
  
for(int i=0;i<fr\_send\_at\_instance;i++)  
  
{  
  
if(rcvd\_ack[i]=='n')  
  
cout<<"SENDER : Frame "<<send[i]<<" is sent\n";  
  
}  
  
receiver(m);  
  
}  
  
void sel\_repeat::receiver(int m)  
  
{  
  
time\_t t;  
  
int f;  
int j;  
int f1;  
  
int a1;  
  
char ch;  
  
srand((unsigned)time(&t));  
  
for(int i=0;i<fr\_send\_at\_instance;i++)  
  
{  
  
if(rcvd\_ack[i]=='n')  
  
{  
  
f=rand()%10;  
  
//if f=5 frame is discarded for some reason  
  
//else frame is correctly recieved  
  
if(f!=5)  
  
{  
  
for(int j=0;j<fr\_send\_at\_instance;j++)  
  
if(rcvd[j]==send[i])  
  
{  
  
cout<<"reciever:Frame"<<rcvd[j]<<"recieved correctly\n";  
  
rcvd[j]=arr[rw];  
  
rw=(rw+1)%m;  
  
break;  
  
}  
int j;  
if(j==fr\_send\_at\_instance)  
  
cout<<"reciever:Duplicate frame"<<send[i]<<"discarded\n";  
  
a1=rand()%5;  
  
//if al==3 then ack is lost  
  
//else recieved  
  
if(a1==3)  
  
{  
  
cout<<"(acknowledgement "<<send[i]<<" lost)\n";  
  
cout<<"(sender timeouts-->Resend the frame)\n";  
  
rcvd\_ack[i]='n';  
  
}  
  
else  
  
{  
  
cout<<"(acknowledgement "<<send[i]<<" recieved)\n";  
  
rcvd\_ack[i]='p';  
  
}  
  
}  
  
else  
  
{int ld=rand()%2;  
  
//if =0 then frame damaged  
  
//else frame lost  
  
if(ld==0)  
  
{  
  
cout<<"RECEIVER : Frame "<<send[i]<<" is damaged\n";  
  
cout<<"RECEIVER : Negative Acknowledgement "<<send[i]<<" sent\n";  
  
}  
  
else  
  
{  
  
cout<<"RECEIVER : Frame "<<send[i]<<" is lost\n";  
  
cout<<"(SENDER TIMEOUTS-->RESEND THE FRAME)\n";  
  
}  
  
rcvd\_ack[i]='n';  
  
}  
  
}  
  
}  
  
for(int j=0;j<fr\_send\_at\_instance;j++)  
  
{  
  
if(rcvd\_ack[j]=='n')  
  
break;  
  
}  
  
int i=0;  
  
for(int k=j;k<fr\_send\_at\_instance;k++)  
  
{  
  
send[i]=send[k];  
  
if(rcvd\_ack[k]=='n')  
  
rcvd\_ack[i]='n';  
  
else  
  
rcvd\_ack[i]='p';  
  
i++;  
  
}  
  
if(i!=fr\_send\_at\_instance)  
  
{  
  
for(int k=i;k<fr\_send\_at\_instance;k++)  
  
{  
  
send[k]=arr[sw];  
  
sw=(sw+1)%m;  
  
rcvd\_ack[k]='n';  
  
}  
  
}  
  
cout<<"Want to continue";  
  
cin>>ch;  
  
cout<<"\n";  
  
if(ch=='y')  
  
sender(m);  
  
else  
  
exit(0);  
  
}  
  
int main()  
  
{  
  
sel\_repeat sr;  
  
sr.input();  
  
//ducslectures.blogspot.in  
  
}

**Output.......**



 **CONCLUSION:**

Hence, we have successfully studied working of Sliding Window protocol using c++ Language..

**OUTCOME**

**Upon completion Students will be able to:**

**ELO1:** Explain and apply the concept of Sliding Window Protocol**. **

**QUESTIONS:**

* What is sliding Window Protocol?
* What is Selective Repeat Sliding Window Protocol?
* How Sliding Window Protocol Works?
* How Selective Repeat Sliding Window Protocol works?
* What are the benefits Sliding Window Protocol?

**EXPERIMENT NO.13**

**Implementation of distance vector routing algorithm**

**Session Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Time**  **( min)** | **Content** | **Learning Aid / Methodology** | **Faculty Approach** | **Typical Student Activity** | **Skill / Competency Developed.** |
| 10 | Relevance and significance of Problem statement | Chalk & Talk , Presentation | Introduces, Explains | Listens, Participates, Discusses | Knowledge, intrapersonal |
| 15 | Explanation of Problem statement | Chalk & Talk , Presentation | Introduces, Facilitates, Explains | Listens,  Participates, | Knowledge, intrapersonal, Application |
| 15 | Concept of distance vector routing using bellman ford algorithms | Demonstration, Presentation | Explains, Facilitates, Monitors | Listens,  Participates,  Discusses | Knowledge, intrapersonal,  interpersonal  Application |
| 60 | Implementation of problem statement | N/A | Guides, Facilitates  Monitors | Participates, Discusses | Comprehension,  Hands on experiment |
| 10 | Assessment | N/A | Monitors | Participates, Discusses | Knowledge, Application |
| 10 | Conclusions | Keywords | Lists, Facilitates | Listens, Participates, Discusses | Knowledge, intrapersonal, Comprehension |

**OBJECTIVES:**

1. To perform shortest path between any nodes.

2. To execute and analyze shortest path using DVR (Distance Vector Routing).

**PROBLEM STATEMENT:**

Implementation of distance vector routing algorithm using C++

**SOFTWARE REQUIRED:** Latest version of 64 Bit Operating Systems Open Source Fedora-20, Windows 8 with Multicore CPU equivalent to Intel i5/7 4th generation onwards supporting Virtualization and Multi-Threading, 8 GB RAM, 500GB/1TB HDD, Latest versions of 64-Bit Programming languages such as Microsoft Visual Studio(ver. 12 or Higher) or equivalent open source, Eclipse 64-bit Platform.

**INPUT:** Input as cost matrix containing the distance between two nodes.

**OUTPUT:** shortest Path using DVR from any other nodes.

**MATHEMATICAL MODEL:**

Let S be the solution perspective of the class Weather Report such that

S={s, e, i, o, f, DD, NDD, success, failure}

s=initial state that is constructor of the class

e = the end state or destructor of the class.

i= input of the system.

o=output of the system.

DD-deterministic data it helps identifying the load store functions or assignment functions.

NDD- Non deterministic data of the system S to be solved.

Success-desired outcome generated.

Failure-Desired outcome not generated or forced exit due to system error.

For class Weather\_Report:

s=initial state or constructor of the class weather()

s={weather()}- sets the default values for all five variables to respective values given in assignment.

Input i=(I1,I2)

I1={day\_of\_moth,hightemp,lowtemp,amount\_rain,amount\_snow}

Variables declared-NDD

Day\_of\_month={1,-------,n} n≠Ø. n=30 or 31.

Hightemp={1,------,n} n≠Ø. n=999.

lowtemp={-999,------,n} n≠Ø. n=6.

amount\_rain={1,------,n} n≠Ø. n=10.

amount\_snow={1,------,n} n≠Ø. n=10.

I2={99,999,-999,0,0}

Default values set all integers –Deterministic Data because of function weather(),memory requirement based on DD.

Hence let weather():I1I2 be an ONTO mapping function shown by fig

Draw Association figure here of mapping values of I1 to I2:

f= {display ( ), override ( ), avg ( )}

display ( )={details of all variable values}

override( )={ default values get overwritten and stored by values specified in this function}

I

I1 I2

( Default values) ( New values)

avg( )={sum of values / no. of days}

n

Σ/n

i=1

Success- desired output is generated in tabular form as Weather Report with default as well as overwritten values along with average.

Failure- desired output is not generated in tabular form as weather Report.

**THEORY:**

Distance vector as the name suggests uses distance between remote networks to determine the best path to a remote network. The distance vector metric is typically the hop. It’s not a measure of distance as such, rather a count of the number of routers in between the router and the destination network. Since a router performs functions on packets that pass through it, and these processes take time, routers cause a certain amount of latency in the network. So it’s fair to assume that the more routers in between the source and the destination, the more latency you can expect. This is how a true distance vector protocol works.

Using hop count as your only metric however doesn’t take in to account the bandwidth of the paths in the routing table. If you are using a hop count only distance vector protocol such as RIP within your network, where all the interconnects are T1 lines then this won’t cause you too much of a problem.

However consider that your network has two routes to a remote network. One of those routes has a hop count of 2, but is a T1 link, and the other is an ISDN line with a hop count of 1.The ISDN route would be chosen, by the router, over the T1 route since it has a lower hop count, even though it’s the slower of the two routes.

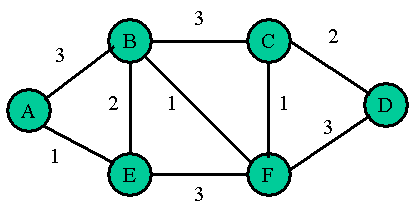
Fortunately not all distance vector routing protocols use hop count as their only metric. Cisco’s [propriety](http://www.internetworktraining.com/article-propriety_systems.aspx) [IGRP](http://www.internetworktraining.com/article-igrp.aspx) is considered distance vector even though it uses bandwidth and delay as it’s default metric.

Distance vector protocols all have one thing in common. That is that they all send out their complete routing table at periodic intervals. They do this in the form of a broadcast made to all directly connected routers. The problem associated with this method is the increased bandwidth usage even though the network topology may have changed.

There are two distance vector algorithms used by the distance routing protocols supported by Cisco routers. They are Bellman-Ford algorithm, which is used by RIP v1, RIP v2 and IGRP, and the Diffusing Update Algorithm (DUAL) used by EIGRP.

**How DVR works?**

The purpose of this programming assignment is to learn how distributed dynamic routing protocols like distance vector accomplish packet routing in practice.  In this assignment, you will be asked to build a distance vector routing protocol that implements the distributed Bellman-Ford algorithm.



**Figure 1 Network Topology**

Figure 1 shows the topology of the network that you will be building, with nodes and link costs.  Each router will be a separate process that communicates with other routers by exchanging UDP datagrams through the UDP socket interface.  Each router implementation must be written in C++.

Each of your routers will be identical, running the same routing code.  You should be able to start your routers in any order.  The router processes should then begin exchanging routing updates in the form of UDP-based distance vectors (DV) between neighbors.  As each DV arrives, each router should execute again its shortest path algorithm and construct a routing/forwarding table.  After many exchanges of routing updates, the routing tables should eventually converge to a stable state.  You should be able to verify by inspection that the stable routing tables will route packets along the lowest cost paths.

In addition, your network of routing processes should also route actual data packets using the forwarding tables constructed at each router.  These packets should follow the actual shortest path through the network as calculated by the distance vector algorithm.

**(a) Building distance vector routing tables**

You will be implementing distance vector (DV) routing using UDP processes.  This is similar to how RIP is implemented, but differs in that you will be implementing *RIP-like* distance vector routing with different link costs.  You will not be implementing an exact replica of RIP, since RIP assumes uniform link costs of one for all links.  Your DV routing implementation will first need to exchange routing updates in the form of distance vectors between router processes that are neighbors.  For simplicity and easier testing, let each DV router periodically advertise its distance vector to each of its neighbors every 5 seconds.  For faster convergence, you can choose to speed up the DV advertisements to once per second, though you should document this in your README.

The distance vector advertised by router *i* should contain a list of destination nodes currently known by router *i*as well as the lowest cost paths to those currently known destination nodes.  In the beginning, each router's first DV will list only the immediate neighbors of that router in the DV's list.  As DVs are exchanged, each router's DV will grow.  The resulting DV, after all destination nodes in the network topology have been discovered, will look like the following.  You should include in the DV additional header information such as the ID of the node that generated this DV, the length of the DV, etc.   

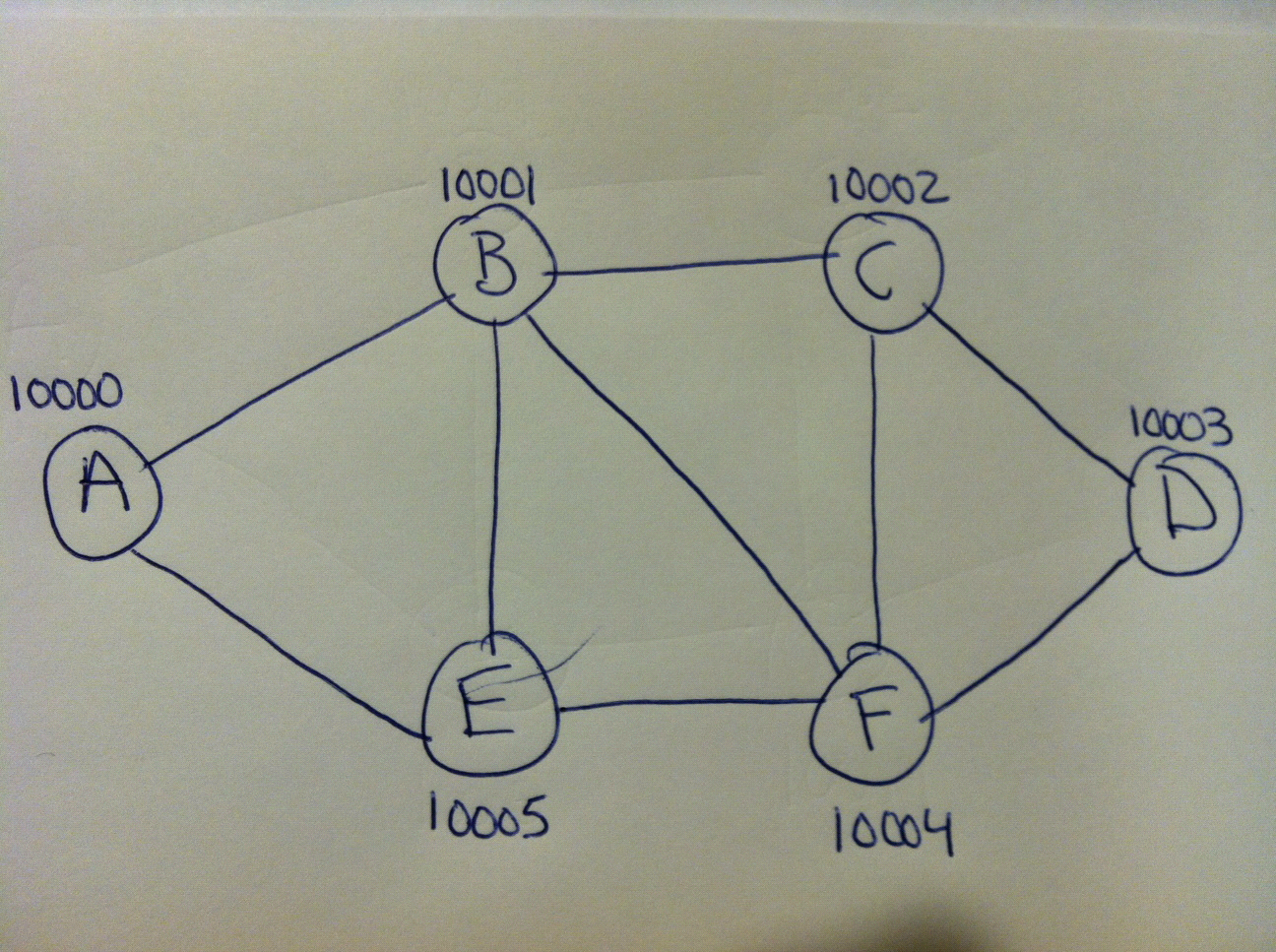
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Destination | A | B | C | D | E | F |
| Cost | -- | -- | -- | -- | -- | -- |

As each DV arrives at a router interface, your UDP router process should recalculate the routing table and DV based on the distributed Bellman-Ford Algorithm.  That is, if you find from the just-arrived DV that a neighbor is advertising a shorter distance to a destination than you currently list in your routing table/DV, then you should update your routing/forwarding table accordingly.  For example, suppose a router R currently lists in its routing table that the shortest path to destination S costs 20 through neighbor X.  Suppose next that R receives a DV from neighbor Y advertising a distance cost of 10 to S.  Also, suppose that the link cost between R and Y is 3.  Then, router R should compute a distance cost of 13 to S through Y, which is less than its stored cost of 20 to S through X.  Therefore, router R should update its routing table to reflect the new shortest path of 13 to S through Y.  In case there are two routes with the same distance cost through different neighbors, then choose the neighbor with the lowest ID, i.e. A<B<C<D<E<F

For example, the initial routing/forwarding table at node A should look like: 

|  |  |  |  |
| --- | --- | --- | --- |
| Destination | Cost | Outgoing UDP port | Destination UDP port |
| B | 3 | 10000 (Node A) | 10001 (Node B) |
| E | 1 | 10000 (Node A) | 10005 (Node E) |

The recommended UDP port #'s to use are shown in the figure below:



**Figure: Network Addresses**

As shown in Figure 2, each router should have a *distinct* UDP port. If you choose to test your routing implementation by placing at least one routing processes on a separate host, then you will only have to change the IP address of the remote router(s), not the port #.  [Note: in an actual routing implementation, each routing interface will have a distinct IP address, rather than a distinct UDP port.]  If you find that these port #'s are occupied on your machine(s), then you're free to modify the initialization file to use different distinct UDP port #'s above 5000.  Please document this in the README.

For this assignment, each routing process should print out to an output file the routing table, but only when there has been a *change* in the routing table.  If an arriving DV advertisement causes a change in the routing table, then you should print out the timestamp, the routing table before the change, the DV that caused the change (including the neighbor it came from), and the new routing table.  If an arriving DV does not cause a change in the routing table, then you do not need to print anything (though while you're debugging, you may wish to print out the routing table for each arriving DV).  In the code that you hand in you should only print a routing table for those DV's that cause a change in the routing table.

**Initialization**

At startup, the neighboring nodes should find out about their immediate neighbors by reading the neighborhood topology information file. The format of each line of this file is a four-tuple:

<source router, destination router, source UDP port, link cost>

For example, the link between A and B contains the line <A,B,10000,3>.  When router A starts up, it should read all lines of the initialization file where node A is listed as the source router.  Also, router A should look at its neighbors' entries to find the UDP port where a DV packet should be sent.  For example, for A to send to B, node A needs the UDP port # corresponding to B's interface with A, namely UDP port # 10001.  This information is contained in the initialization file under the entries where A is the destination node, i.e. in the line <B,A,10001,3>.

Here’s the file:

A,B,10001,3

A,E,10005,1

B,A,10000,3

B,C,10002,3

B,E,10005,2

B,F,10004,1

C,B,10001,3

C,D,10003,2

C,F,10004,1

D,C,10002,2

D,F,10004,3

E,A,10000,1

E,B,10001,2

E,F,10004,3

F,B,10001,1

F,C,10002,1

F,D,10003,3

F,E,10005,3

Your router should discover all nodes that are not immediate neighbors by exchanging DVs rather than reading directly from the initialization file (otherwise, that would defeat the whole purpose of this routing assignment!).  For example, at startup, router A should only read information from the initialization file about neighboring routers B and E, i.e. their link costs and UDP send & receive port #'s.  To find out about the existence of (and shortest paths to) nodes C, D, and F, router A must obtain this information by exchanging DVs.  Your router is not allowed to obtain this information from the topology file.  After many such DV exchanges, each router process will converge to the same list of all reachable nodes, though the distances to these nodes will differ depending on the router.

**Command line**

* You should create a file called routedDV.cpp, and create an executable called routedDV.  When starting a router, simply pass in its name in the command line, e.g. "routedDV A" to start the router labeled "A".  [In UNIX systems, RIP is typically implemented in a process called*routed*, namely a *route*r *d*aemon process.]

**Output files**

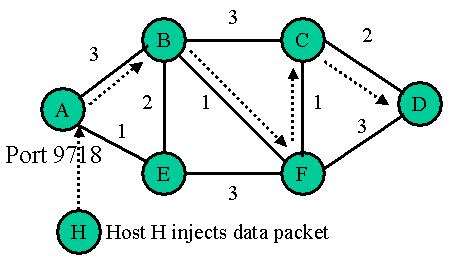
* Each routed\_DV process should generate an output files containing print outs of the timestamped progression of routing tables for that router, as well as information about any data packets that have been routed through that router.  Router A should produce an output file called routing\_outputA.txt.  Router B should produce one output file called routing\_outputB.txt, etc.

**(b) Routing data**

After all routing tables have converged, which should occur after 30-60 seconds after the last router was started, then you should use your network of routers to route actual data packets from node A to node D.  Your packet header should contain a type field that distinguishes packets of type "data" from packets of type "control" (containing the DV routing updates).  Hence, data packets and control packets will arrive on the same UDP port numbers and socket buffers, and routers should look at the type field in each packet's header to determine whether the packet is data or a DV upate .  Your data packet can be any length, though for simplicity you could set it small at about 100 bytes to avoid fragmentation during the multi-hop routing .  The data payload should contain a unique text phrase of your choosing.  Please document your packet format, header fields, and payload phrase in your README.

Each router should forward the data packet towards the destination via the one neighbor that is along the shortest path to the destination, as determined by each node's routing table.  Also, for each data packet that arrives at a router, you should print to the router's output file the timestamp, the ID of the source node of the data packet, the destination router, the UDP port in which the packet arrived, and the UDP source port along which the packet was forwarded.  For the final destination D, the arrival of the data packet should trigger printing out of all the information above as well as the text phrase in the data payload.

The recommended way to inject a data packet into the network is to start up another router, call it host H, and have it send a data packet to router A with ultimate destination D.  This is shown in Figure 3 below.  For example, you could call "routedDV H", and then include a conditional if() clause in the code of routedDV.cpp such that if ID=H, then the code would send one data packet to UDP port 10000 of router A with ultimate destination D and then exits immediately.  But before H exits, H should print out to its output file the data packet that it sent to destination D through router A, i.e. print out the header fields of the data packet such as the UDP destination port # on router A, the destination ID=D, and the text phrase in the data payload.  The role of host H is simply to send one data packet and then exit, so H will not participate in DV routing.  That is, if ID=H, then there is no need to execute the rest of the code in routed\_DV.c corresponding to normal DV routing for routers A-F.



**Figure 3 Host H injects a data packet into node A that is routed to destination D**

**CONCLUSION:**

Hence, we have successfully executed Distance Vector Routing Algorithm using Bellman ford algorithm.

**OUTCOME**

**Upon completion Students will be able to:**

**ELO1:** Able to Execute DVR using Bellman Ford Algorithm ****

**QUESTIONS:**

* What is distance vector routing protocol?
* Explain Bellman ford algorithm?
* Differentiate between bellman ford and dijikstra’s algorithm.?
* Draw the graphs and show the flow of shortest distance between source to any other nodes?
* What is static routing?
* What is dynamic routing?
* Difference between static and dynamic routing?
* What is count to infinity problem?

**PRACTICE ASSIGNMENTS:**

* Implementation of Dijikstra’s Algorithm?
* Implementation of flooding algorithm?
* Implementation of any other dynamic routing protocol?

**References:**

* **Lab Manual developed by the College Teachers approved by the BoS Digital Content:(Down Loadable PDF for students only)**
* **Steven W. Smith, “The Scientist and Engineer's Guide to Digital Signal**
* **processing”,California Technical Publishing, 2nd Edition, PDF ISBN 0-9660176-6-8 (Colleges are expected to purchase Paperback Edition ISBN 0-9660176-4-1.)**
* **Laboratory Manual developed by the College Teacher and get it approved by the BoS.**
* [**http://www.tcpipguide.com/free/t\_DHCPMessageFormat.htm**](http://www.tcpipguide.com/free/t_DHCPMessageFormat.htm)
* [**http://technet.microsoft.com/en-us/library/cc781008%28v=ws.10%29.aspx**](http://technet.microsoft.com/en-us/library/cc781008%28v=ws.10%29.aspx)
* [**http://www.itrinegy.com/index.php/solutions/by-task/wannetwork-simulation**](http://www.itrinegy.com/index.php/solutions/by-task/wannetwork-simulation)
* [**http://compnetworking.about.com/cs/lanvlanwan/g/bldef\_wan.htm**](http://compnetworking.about.com/cs/lanvlanwan/g/bldef_wan.htm)
* [**http://www.howtogeek.com/168769/4-easy-ways-to-remotely-print-over-the-network-or-internet/**](http://www.howtogeek.com/168769/4-easy-ways-to-remotely-print-over-the-network-or-internet/)
* [**http://windows.microsoft.com/en-in/windows-vista/print-from-a-remote-computer-to-your-printer**](http://windows.microsoft.com/en-in/windows-vista/print-from-a-remote-computer-to-your-printer)