GROUP ASSIGNMENT



NATIONAL SCHOOL OF BUSINESS MANAGEMENT

COMPUTER NETWORKS

Contents

[INTRODUCTION 3](#_Toc522049680)

[Networking fundamentals 3](#_Toc522049681)

[Overview 3](#_Toc522049682)

[Configuring the LAN and Switching Topology 3](#_Toc522049683)

[Subnetting 3](#_Toc522049684)

[Headquarters 4](#_Toc522049685)

[Remote branch 1/2 4](#_Toc522049686)

[WAN/links routers 4](#_Toc522049687)

[Creating virtual local area networks (VLANs) 5](#_Toc522049688)

[Inter -VLAN routing 6](#_Toc522049689)

[Assigning ports to each VLAN 7](#_Toc522049690)

[Assigning IP addresses 8](#_Toc522049691)

[VLAN 10 8](#_Toc522049692)

[VLAN 20 9](#_Toc522049693)

[VLAN 30 10](#_Toc522049694)

[VLAN 40 11](#_Toc522049695)

[Ping command outputs 12](#_Toc522049696)

[Ping in same VLAN 12](#_Toc522049697)

[Ping in different VLAN 12](#_Toc522049698)

[MAC-address table information 13](#_Toc522049699)

[Configuring WAN and Routing Architecture 13](#_Toc522049700)

[Routing tables 14](#_Toc522049701)

[Router0 14](#_Toc522049702)

[Router1 14](#_Toc522049703)

[Router2 15](#_Toc522049704)

[Connecting headquarters and other two branches via serial DTE cable 15](#_Toc522049705)

[Configuring OSPF as the routing protocol in all routers 16](#_Toc522049706)

[16](#_Toc522049707)

[Routing tables output 19](#_Toc522049708)

[Router 0 19](#_Toc522049709)

[Router 1 19](#_Toc522049710)

[Router 2 20](#_Toc522049711)

[Describe all routing table parameters 20](#_Toc522049712)

[Router’s interface IP address details and status(up/down) 21](#_Toc522049713)

[Router0 21](#_Toc522049714)

[Router1 21](#_Toc522049715)

[Router2 22](#_Toc522049716)

[Ping command outputs form each router to other two routers 22](#_Toc522049717)

[Ping from router0 to router2 22](#_Toc522049718)

[Ping from router0 to router1 23](#_Toc522049719)

[Ping from router2 to router0 23](#_Toc522049720)

[Ping from one location PC to other two locations’ few pcs and get the ping output 24](#_Toc522049721)

[Trace Route output 25](#_Toc522049722)

[**Interior gateway routing protocols** 26](#_Toc522049723)

[Operation of OSPF routing protocol 29](#_Toc522049724)

[Sample Proxy Server 29](#_Toc522049725)

[Simple example for how proxy servers work 30](#_Toc522049726)

[Java proxy server 31](#_Toc522049727)

[Part one 31](#_Toc522049728)

[HttpRequest.java 31](#_Toc522049729)

[HttpResponse.java 34](#_Toc522049730)

[ProxyCache.java 38](#_Toc522049731)

[Threads.java 42](#_Toc522049732)

[Part two 45](#_Toc522049733)

[Add better error handling 45](#_Toc522049734)

[Part 3 47](#_Toc522049735)

[Add caching 47](#_Toc522049736)

[Class diagram 48](#_Toc522049737)

[References 49](#_Toc522049738)

[Group members 49](#_Toc522049739)

# INTRODUCTION

## Networking fundamentals

A **computer network**, or **data network**, is a [digital](https://en.wikipedia.org/wiki/Digital_signal) [telecommunications network](https://en.wikipedia.org/wiki/Telecommunications_network) which allows [nodes](https://en.wikipedia.org/wiki/Node_(networking)) to share resources. In computer networks, [computing devices](https://en.wikipedia.org/wiki/Computing_device) [exchange data](https://en.wikipedia.org/wiki/Data_transmission) with each other using connections ([data links](https://en.wikipedia.org/wiki/Data_link)) between nodes. These data links are established over [cable media](https://en.wikipedia.org/wiki/Networking_cables) such as wires or optic cables, or [wireless media](https://en.wikipedia.org/wiki/Wireless_network) such as [Wi-Fi](https://en.wikipedia.org/wiki/WiFi).

Network computer devices that originate, [route](https://en.wikipedia.org/wiki/Routing) and terminate the data are called network nodes.[[1]](https://en.wikipedia.org/wiki/Computer_network#cite_note-1) Nodes can include [hosts](https://en.wikipedia.org/wiki/Host_(network)) such as [personal computers](https://en.wikipedia.org/wiki/Personal_computer), [phones](https://en.wikipedia.org/wiki/Phone), [servers](https://en.wikipedia.org/wiki/Server_(computing)) as well as [networking hardware](https://en.wikipedia.org/wiki/Networking_hardware). Two such devices can be said to be networked together when one device is able to exchange information with the other device, whether they have a direct connection to each other. In most cases, application-specific communications protocols are [layered](https://en.wikipedia.org/wiki/Communications_protocol#Layering) (i.e. carried as [payload](https://en.wikipedia.org/wiki/Payload_(computing))) over other more general communications protocols. This formidable collection of [information technology](https://en.wikipedia.org/wiki/Information_technology) requires skilled [network management](https://en.wikipedia.org/wiki/Network_management) to keep it all [running reliably](https://en.wikipedia.org/wiki/Uptime).

Computer networks support an enormous number of [applications](https://en.wikipedia.org/wiki/Application_software) and [services](https://en.wikipedia.org/wiki/Network_service) such as access to the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), [digital video](https://en.wikipedia.org/wiki/Digital_video), [digital audio](https://en.wikipedia.org/wiki/Digital_audio), shared use of [application and storage servers](https://en.wikipedia.org/wiki/File_server), printers, and [fax machines](https://en.wikipedia.org/wiki/Fax), and use of [email](https://en.wikipedia.org/wiki/Email) and [instant messaging](https://en.wikipedia.org/wiki/Instant_messaging) applications as well as many others. Computer networks differ in the [transmission medium](https://en.wikipedia.org/wiki/Transmission_medium) used to carry their signals, [communications protocols](https://en.wikipedia.org/wiki/Communications_protocol) to organize network traffic, the network's size, [topology](https://en.wikipedia.org/wiki/Network_topology), [traffic control](https://en.wikipedia.org/wiki/Network_traffic_control) mechanism and organizational intent. The best-known computer network is the Internet.Z

# Overview

A company consists of three branches dispersed across the island, the headquarters (HQ) of the company is in Colombo and two branch offices are situated at various remote locations in the Island. There are 80 nodes in HQ and nodes are spread equally over four functional departments namely Management, Sales, Finance, HR & Admin. Each branch has 12 nodes. we’ve been tasked to design a new network for the company by developing a Local Area Network (LAN) and a Wide Area Network (WAN) architecture that aligns with its data communication requirements.

# Configuring the LAN and Switching Topology

A local area network (LAN) is a group of computers and associated devices that share a common communications line or wireless link to a server. Typically, a LAN encompasses computers and peripherals connected to a server within a distinct geographic area such as an office or a commercial establishment. Computers and other mobile devices use a LAN connection to share resources such as a printer or network storage.

## Subnetting

From the 3 class A, B, C IP blocks which have been given to us as following,

1. 192.168.15.0/24- headquarters
2. 172.16.10.0/24-branches
3. 10.1.1.0/28- WAN/links routers

### Headquarters

We have divided the headquarters ip block into 4 subnets for its Management, Sales, Finance, HR & Admin which are its functional departments. We have allocated 2 bits to the network side and retained 6 bits on the host side to accommodate the 20 pcs each department.

Subnet Mask- 255.255.255.192

|  |  |  |
| --- | --- | --- |
| Network ID | Department | Default gateway |
| 192.168.15.0/26 | Management | 192.168.15.1 |
| 192.168.15.64/26 | Sales | 192.168.15.65 |
| 192.168.15.128/26 | Finance | 192.168.15.129 |
| 192.168.15.192/26 | Admin & HR | 192.168.15.193 |

### Remote branch 1/2

We have divided the branch ip block into 2 subnets. We have allocated 1 bit to the network side and retained 7 bits on the host side to accommodate the 12 pcs each department.

|  |  |  |
| --- | --- | --- |
| Network id | Department | Default gateway |
| 172.16.10.0/25 | Branch 1 | 172.16.10.1 |
| 172.16.10.128/25 | Branch 2 | 172.16.10.129 |

### WAN/links routers

We have divided the ip block into 2 subnets. We have allocated 2 bit to the network side and retained 6 bits on the host side to accommodate the links and routers

# Creating virtual local area networks (VLANs)

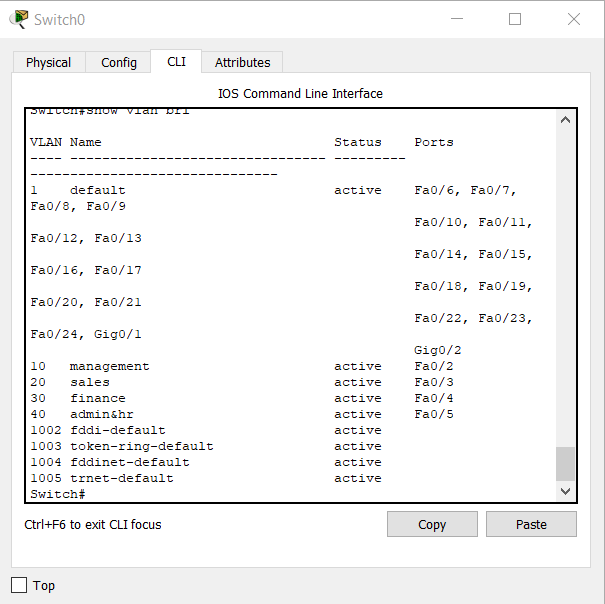
A VLAN is a subgroup of a LAN, in a VLAN computers or other devices behaves as if they are connected using the same wire together although they are on different segments of a LAN. A VLAN is created using software other than physically creating it.

Creating 3 VLANs in each switch for 3 departments:

A screenshot of a computer

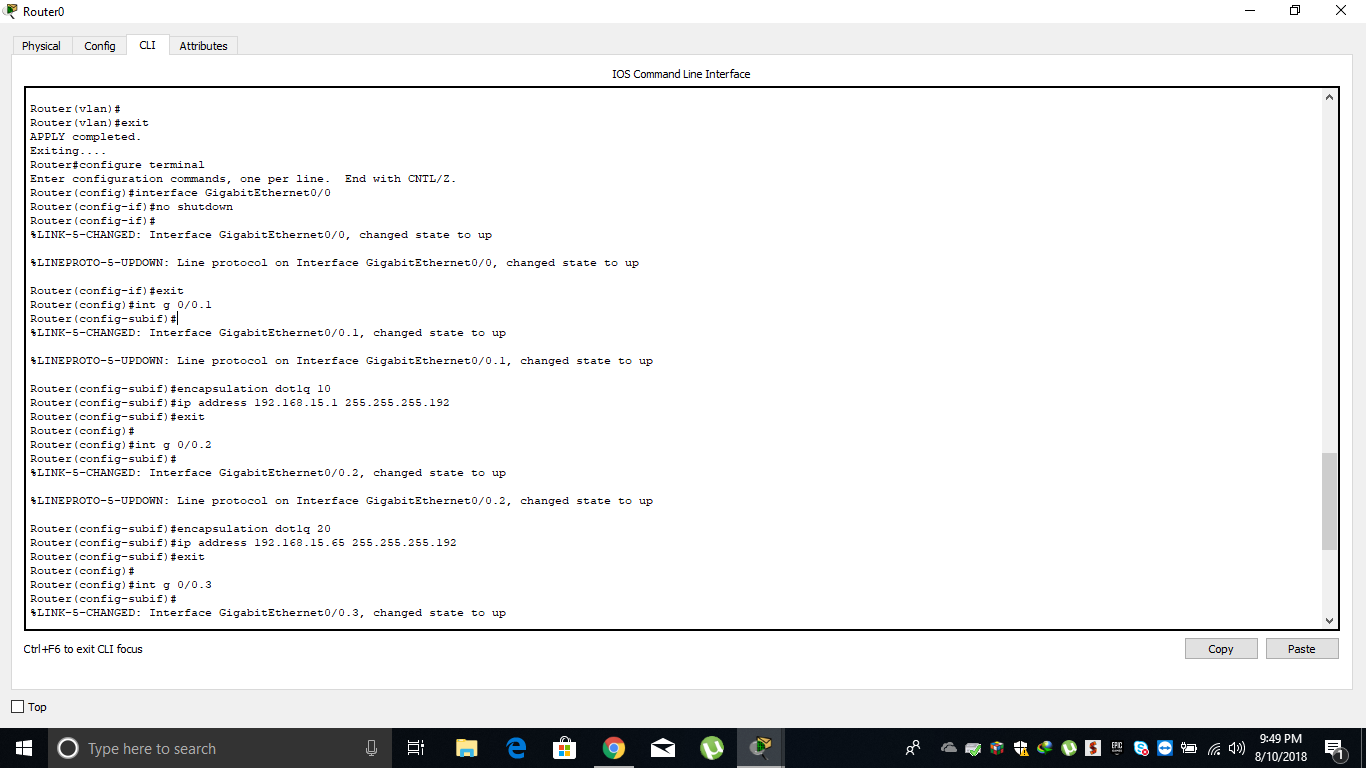
Description generated with very high confidence

VLANs created successfully in switch and ports assigned successfully.

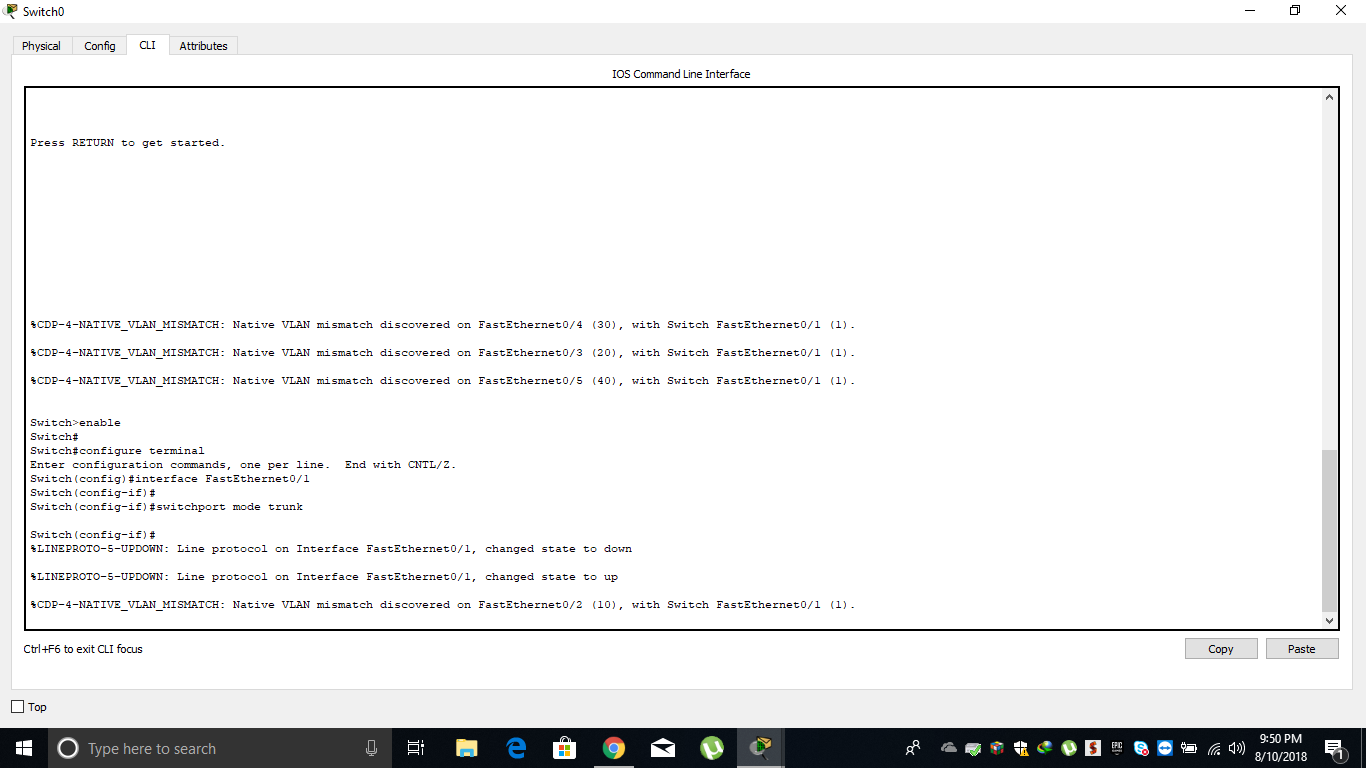


## Inter -VLAN routing

Inter-VLAN routing can be defined to forward traffic between different VLAN by implementing a router in the network. As we learnt previously, VLANs logically segment the switch into different subnets, when a router is connected to the switch, an administrator can configure the router to forward the traffic between the various VLANs configured on the switch. The user nodes in the VLANs forwards traffic to the router which then forwards the traffic to the destination network regardless of the VLAN configured on the switch.



### Assigning ports to each VLAN



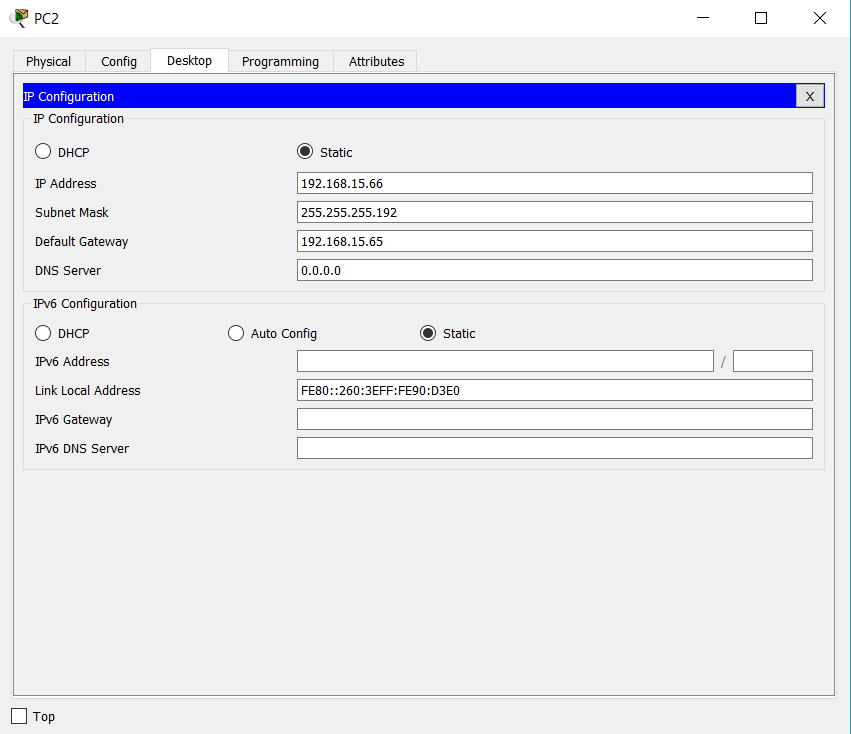
## Assigning IP addresses

### VLAN 10

A screenshot of a cell phone

Description generated with very high confidence

## VLAN 20



### VLAN 30

A screenshot of a social media post

Description generated with very high confidence

### VLAN 40

A screenshot of a social media post

Description generated with very high confidence

## Ping command outputs

### Ping in same VLAN

A screenshot of a computer

Description generated with very high confidence

### Ping in different VLAN

A screenshot of a cell phone

Description generated with very high confidence

### MAC-address table information

A screenshot of a cell phone

Description generated with very high confidence

A MAC address table is used to determine where to forward traffic on a LAN in Ethernet switches.

When a packet comes to this router, it will check the mac address table and based on the information it will switch the packet to the intended destination.

# Configuring WAN and Routing Architecture

A Wide Area Network (WAN) is a collection of computer and network resources that are connected through a network over a large geographical area, WANs are typically connected using the Internet or special connections provided by service providers.

## Routing tables

### Router0

A screenshot of a cell phone

Description generated with very high confidence

### Router1

A screenshot of a cell phone

Description generated with very high confidence

### Router2

A screenshot of a cell phone

Description generated with very high confidence

### Connecting headquarters and other two branches via serial DTE cable

A close up of a map

Description generated with high confidence

## Configuring OSPF as the routing protocol in all routers

### 

A screenshot of a social media post

Description generated with very high confidence

A screenshot of a social media post

Description generated with very high confidence

A screenshot of a computer

Description generated with very high confidence

A screenshot of a computer

Description generated with very high confidence

## Routing tables output

### Router 0

A screenshot of a social media post

Description generated with very high confidence

### Router 1

A screenshot of a social media post

Description generated with very high confidence

### Router 2

A screenshot of a social media post

Description generated with very high confidence

## Describe all routing table parameters

**Destination:** The IP address of the packet's destination

**Next hop:** The IP address to that the packet is forwarded

**Interface:** The outgoing network interface the device ought to use when forwarding the packet to consequent hop or destination

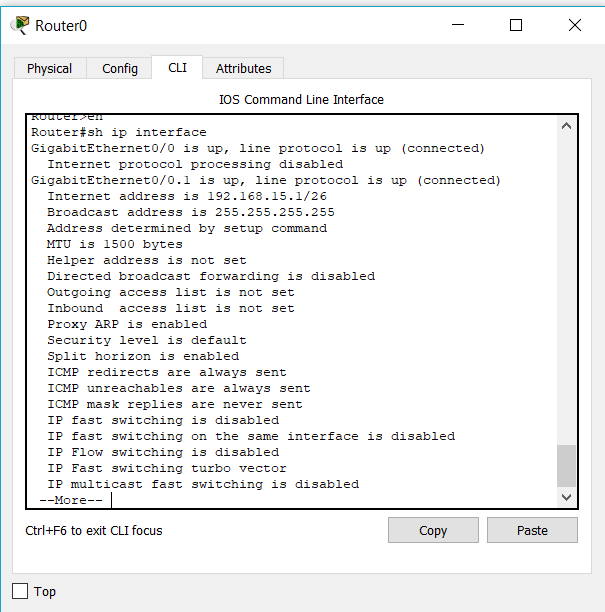
**Metric:** Assigns a cost to every accessible route so that the foremost cost-effective path can be chosen

**Routes:** Includes directly-attached subnets, indirect subnets that aren't hooked up to the device but may be accessed through one or additional hops, and default routes to use for certain forms of traffic or when info is lacking.

**Cost:** The link state cost of the trail to the destination. For all ways except sort a pair of external paths this describes the whole path's cost. For sort a pair of external paths, this field describes the cost of the portion of the trail internal to the AS. This cost is calculated because the total of the costs of the path's constituent links.

## Router’s interface IP address details and status(up/down)

### Router0



### Router1

A screenshot of a cell phone

Description generated with very high confidence

### Router2

A screenshot of a social media post

Description generated with very high confidence

## Ping command outputs form each router to other two routers

### Ping from router0 to router2

A screenshot of a social media post

Description generated with very high confidence

### Ping from router0 to router1

A screenshot of a social media post

Description generated with very high confidence

### Ping from router2 to router0

A screenshot of a cell phone

Description generated with very high confidence

## Ping from one location PC to other two locations’ few pcs and get the ping output

A screenshot of a computer

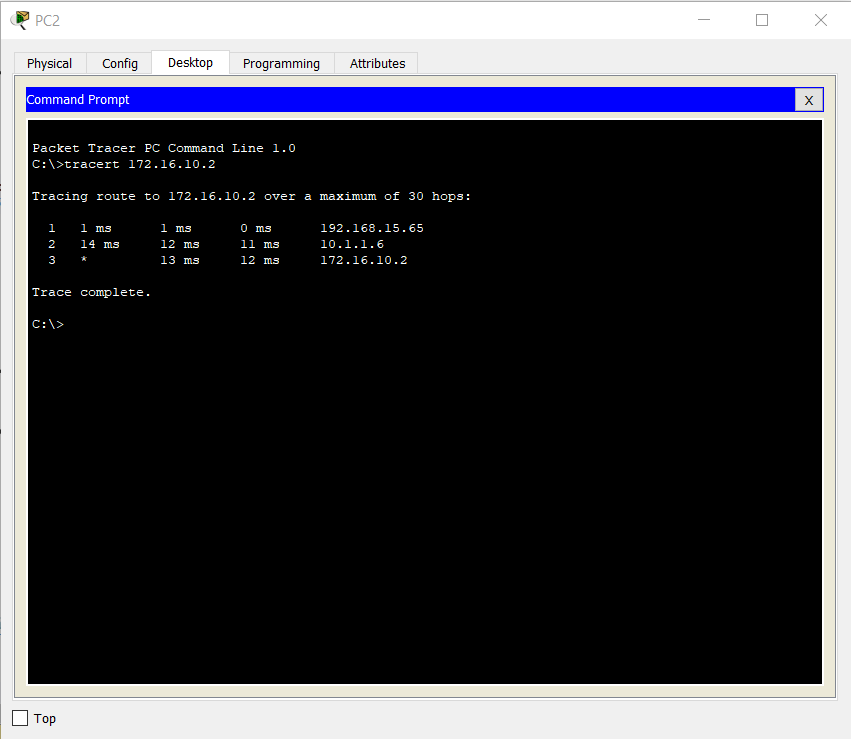
Description generated with very high confidence

A screenshot of a cell phone

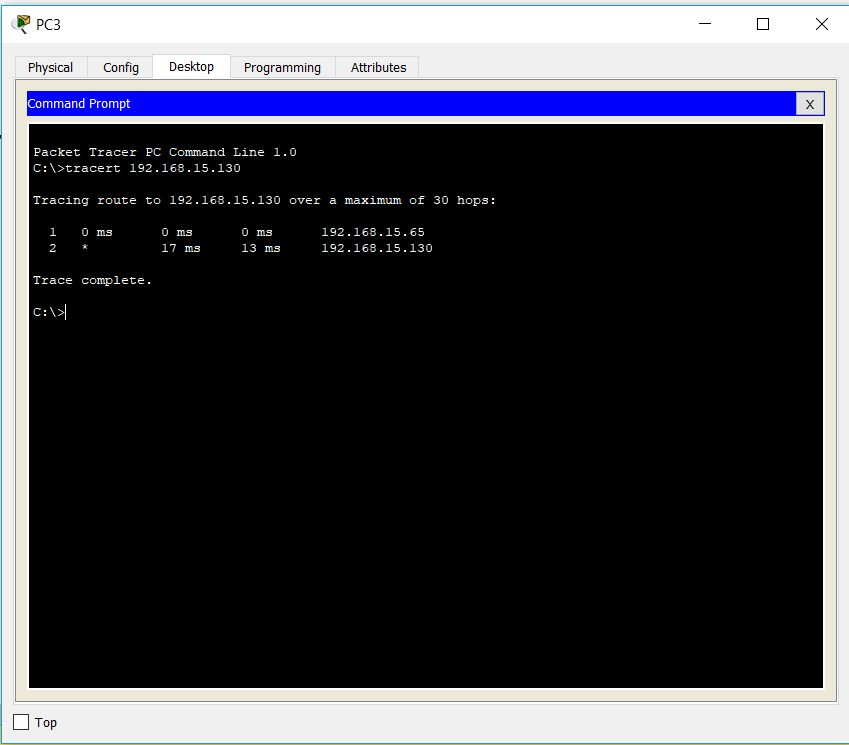
Description generated with very high confidence

### Trace Route output

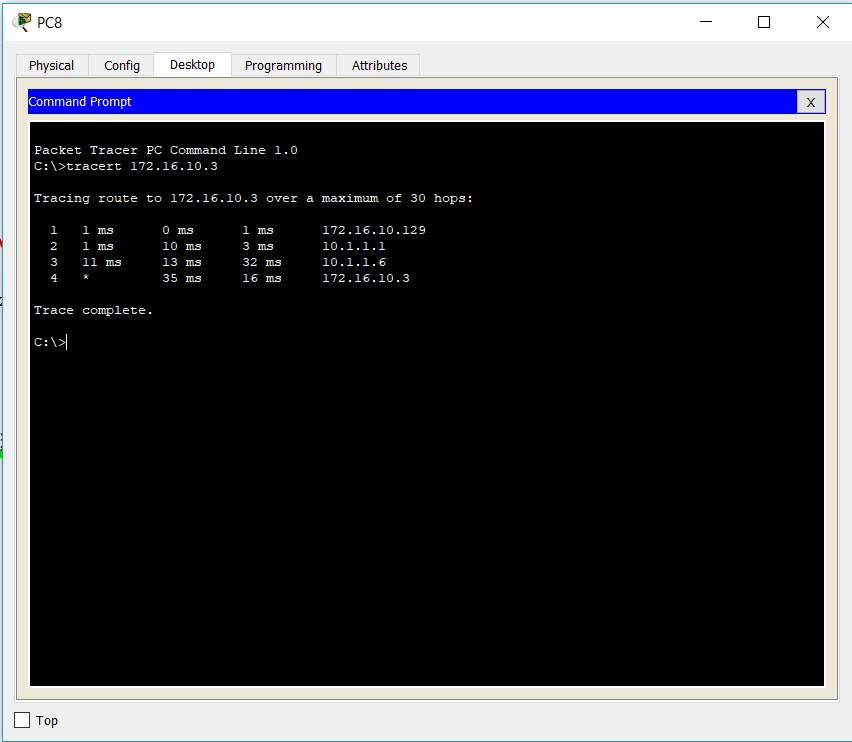
#### Pc2 to pc10



#### Pc3 to pc 4



#### Pc8 to pc11



## **Interior gateway routing protocols**

Routing is one of the most important parts of a network which enables large networks to communicate with each other. This is facilitated by a device called router. A router will pass routing information to its neighboring routers and so forth thus materializing communication between networks.

There are basically two types of routing.

* Static routing
* Dynamic routing

In static routing the network administrator or person responsible will have to manually create and maintain the routing tables in routers. If it’s a large network this task would be tedious where the network administrator will have to go to individual router and create the routing table.

A way out of this is dynamic routing which a functionality is built into the routers which allows them to share routing information on their own automatically and create routing tables for themselves. Routing protocols aid routers in this respect.

There are two major routing protocols

* Interior gateway protocol
* Exterior gateway protocol

Interior gateway protocol deals with exchange of routing information within routers of a single autonomous system where as exterior gateway protocol deals with exchange of routing information between different autonomous systems. (Kozierok, 2005)

Interior gateway protocol can be divided into two parts as

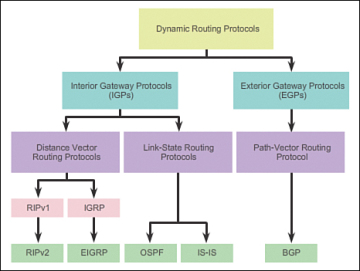
* Distance vector routing protocol
* Link state routing protocol

Distance vector routing protocol determines the routes based on the distances between routers. This distance is calculated based on the number of hops between routers. Routers using distance vector routing protocol will only have knowledge about routers connected to it and the routers which it can reach via the connected routers hence they do not have knowledge about the network topology as a whole. (Kozierok, 2005)

Link state routing protocol is different in that it selects routes based on the shortest path between networks. Routers using this routing protocol will have a map of the current network topology and based on advanced algorithms the routers will determine the shortest path for a particular route. (Kozierok, 2005)

There are several popular routing protocols belonging to these two main internal gateway protocols. Those are

* RIP v1, RIP v2, IGRP and EIGRP for distance vector routing protocol.
* OSPF and IS-IS for link state routing protocol



**RIP version 1**

RIP (Routing information protocol) is a distance vector routing protocol and it is one of the oldest routing protocols that exist today. It uses the Bellman-Ford algorithm which is named after its inventors to determine the routes. It is a classful protocol which broadcasts information through the UDP protocol on address 255.255.255.255. To prevent packets from looping in the network RIP uses a technique called counting to infinity which limits the maximum hop count to 15 hops per packet. RIP protocol transmits network information updates every 30 seconds amongst its neighbors hence causing heaving bandwidth consumption.

**RIP version 2**

Router information protocol version 2 is also a distance vector routing protocol same as its predecessor but with several improvements built into to it. It is a classless protocol which supports classless inter domain routing thus enabling RIP v2 to use variable length subnet masking. It differs from RIP v1 in that it multicasts information to its neighbors on address 224.0.0.9 instead of broadcasting. Because of this feature It supports triggered updates. (Thomas, 2015)

**IGRP**

Interior gateway routing protocol (IGRP) is also a distance vector routing protocol which is a proprietary protocol developed by Cisco. Compared to RIP protocol IGRP can be used in larger networks. IGRP is a classful routing protocol similar to RIP. When selecting routes IGRP uses multiple metrics such as bandwidth, load and reliability among others and uses an algorithm to select the best route possible. IGRP broadcasts update information to its neighbors every 90 seconds which is an improvement over the RIP protocol.

**EIGRP**

Enhanced interior gateway routing protocol (EIGRP) is also a distance vector routing protocol but with improvements to overcome the shortcomings of IGRP. It is a classless protocol which supports variable length subnet masking (VLSM). It also uses a more advanced algorithm called diffusing update algorithm (DUA) to determine the best and efficient route and also to reduce packet looping. In this protocol, if only there is a change in one of the routers in the network the neighboring routers will be notified of it instead of sending out all the information.

**OSPF**

Open shortest path first (OSPF) is a link state routing protocol which uses an algorithm called Dijkstra’s algorithm or shortest path first algorithm to determine the best possible path to a destination. Compared to other protocols, routers using this will converge quickly and also will be less prone to errors. Routers in a network using this protocol will have a map of the current topology of the network stored in them which they will use in determining the shortest path to a destination.

**IS-IS**

Intermediate system to intermediate system (IS-IS) is also a link state routing protocol which is more or less similar to OSPF protocol but compared to OSPF, it is more scalable hence can be used in larger networks and it converges faster than OSPF. IS-IS uses the same shortest path first algorithm to determine the best path to a destination and shares the map of the current network topology with all the converged routers.

**Advantages of using OSPF over other protocols**

* Relatively the converging time is faster
* Highly scalable
* Supports variable length subnet masking (VLSM)
* Uses multicasting instead of broadcasting thus reduces bandwidth consumption
* It is an open standard not a proprietary standard
* OSPF only sends updates on routing tables instead of the entire table

### Operation of OSPF routing protocol

1. Each router using link state protocol will identify its directly connected networks known as neighbors.
2. Then routers will exchange hello packets with their neighbors.
3. After each router receives hello packets from its directly connected neighbors, it will create its own link state packet (LSP) which includes the neighbors network ID, link type and bandwidth.
4. After the LSP is created, it will be flooded to all neighbors who then will save it in a link state database and forward it to their neighbors until all networks have knowledge about each other. (LSPs are sent mainly on two occasions (1) When a router starts up (2) when there is a change in topology).
5. Once all routers have information about every other router, using their link state database they create a map of the network topology.
6. Using this network topology, they calculate the shortest path to each destination by considering the costs of each route. The route with least cost will be selected.
7. Routers use link state advertisements to share this topology with other routers in the network.
8. Once the shortest paths for each route is determined, routers will create routing tables based on these paths.

# Sample Proxy Server

A proxy server is a dedicated computer or a software system running on a computer that acts as an intermediary between an endpoint device, such as a computer, and another server from which a user or client is requesting a service. The proxy server may exist in the same machine as a firewall server or it may be on a separate server, which forwards requests through the [firewall](https://searchsecurity.techtarget.com/definition/firewall).

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](https://whatis.techtarget.com/definition/troubleshooting).

## Simple example for how proxy servers work

When a proxy server receives a request for an Internet resource (such as a Web page), it looks in its local [cache](https://searchstorage.techtarget.com/definition/cache) of previously 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](https://searchunifiedcommunications.techtarget.com/definition/Internet-Protocol) 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.

Proxy servers are used for both legal and illegal purposes. In the enterprise, a proxy server is used to facilitate security, administrative control or [caching](https://whatis.techtarget.com/definition/server) services, among other purposes. In a personal computing context, proxy servers are used to enable user privacy and [anonymous surfing](https://searchsecurity.techtarget.com/definition/anonymous-Web-surfing). Proxy servers can also be used for the opposite purpose: To monitor traffic and undermine user privacy.

To the user, the proxy server is invisible; all Internet requests and returned responses appear to be directly with the addressed Internet server. (The proxy is not actually invisible; its IP address has to be specified as a [configuration](https://searchexchange.techtarget.com/definition/configuration) option to the browser or other protocol program.)

A close up of a device

Description generated with high confidence

# Java proxy server

## Part one

#### Proxy functionality

1. The proxy listens for requests from clients

2. When there is a request, the proxy spawns a new thread for handling the request and creates and HttpRequest-object which contains the request.

3. The new thread sends the request to the server and reads the server's reply intoan HttpResponse-object.

4. The thread sends the response back to the requesting client.

Fill in code is **bolded** below.

### HttpRequest.java

*/\*\**

*\* HttpRequest - HTTP request container and parser*

*\**

*\* $Id: HttpRequest.java,v 1.2 2003/11/26 18:11:53 kangasha Exp $*

*\**

*\*/*

*import java.io.\*;*

*import java.net.\*;*

*import java.util.\*;*

*public class HttpRequest {*

*/\*\* Help variables \*/*

*final static String CRLF = "\r\n";*

*final static int HTTP\_PORT = 80;*

*/\*\* Store the request parameters \*/*

*String method;*

*String URI;*

*String version;*

*String headers = "";*

*/\*\* Server and port \*/*

*private String host;*

*private int port;*

*/\*\* Create HttpRequest by reading it from the client socket \*/*

*public HttpRequest(BufferedReader from) {*

*String firstLine = "";*

*try {*

*firstLine = from.readLine();*

*} catch (IOException e) {*

*System.out.println("Error reading request line: " + e);*

*}*

*String[] tmp = firstLine.split(" ");*

***/\* Format GET = GET http://www.google.pt HTTP/1.0 \*/***

*method =* ***tmp[0]; /\* method GET \*/***

*URI =* ***tmp[1]; /\* URI \*/***

*version* ***= tmp[2]; /\* HTTP version \*/***

*System.out.println("URI is: " + URI);*

*try {*

*String line = from.readLine();*

*while (line.length() != 0) {*

*headers += line + CRLF;*

*/\* We need to find host header to know which server to*

*\* contact in case the request URI is not complete. \*/*

*if (line.startsWith("Host:")) {*

*tmp = line.split(" ");*

*if (tmp[1].indexOf(':') > 0) {*

*String[] tmp2 = tmp[1].split(":");*

*host = tmp2[0];*

*port = Integer.parseInt(tmp2[1]);*

*} else {*

*host = tmp[1];*

*port = HTTP\_PORT;*

*}*

*}*

*line = from.readLine();*

*}*

*} catch (IOException e) {*

*System.out.println("Error reading from socket: " + e);*

*return;*

*}*

*System.out.println("Host to contact is: " + host + " at port " + port);*

*}*

*/\*\* Return host for which this request is intended \*/*

*public String getHost() {*

*return host;*

*}*

*/\*\* Return port for server \*/*

*public int getPort() {*

*return port;*

*}*

*/\*\**

*\* Convert request into a string for easy re-sending.*

*\*/*

*public String toString() {*

*String req = "";*

*req = method + " " + URI + " " + version + CRLF;*

*req += headers;*

*/\* This proxy does not support persistent connections \*/*

*req += "Connection: close" + CRLF;*

*req += CRLF;*

*return req;*

*}*

*}*

### HttpResponse.java

*/\*\**

*\* HttpResponse - Handle HTTP replies*

*\**

*\* $Id: HttpResponse.java,v 1.2 2003/11/26 18:12:42 kangasha Exp $*

*\**

*\*/*

*import java.io.\*;*

*import java.net.\*;*

*import java.util.\*;*

*public class HttpResponse {*

*final static String CRLF = "\r\n";*

*/\*\* How big is the buffer used for reading the object \*/*

*final static int BUF\_SIZE = 8192;*

*/\*\* Maximum size of objects that this proxy can handle. For the*

*\* moment set to 100 KB. You can adjust this as needed. \*/*

*final static int MAX\_OBJECT\_SIZE = 100000;*

*/\*\* Reply status and headers \*/*

*String version;*

*int status;*

*String statusLine = "";*

*String headers = "";*

*/\* Body of reply \*/*

*byte[] body = new byte[MAX\_OBJECT\_SIZE];*

*/\*\* Read response from server. \*/*

*public HttpResponse(DataInputStream fromServer) {*

*/\* Length of the object \*/*

*int length = -1;*

*boolean gotStatusLine = false;*

*/\* First read status line and response headers \*/*

*try {*

*String line =****fromServer.readLine();***

*while (line.length() != 0) {*

*if (!gotStatusLine) {*

*statusLine = line;*

*gotStatusLine = true;*

*} else {*

*headers += line + CRLF;*

*}*

*/\* Get length of content as indicated by*

*\* Content-Length header. Unfortunately this is not*

*\* present in every response. Some servers return the*

*\* header "Content-Length", others return*

*\* "Content-length". You need to check for both*

*\* here. \*/*

*if (line.startsWith****("Content-Length:")*** *||*

*line.startsWith****("Content-length:"))*** *{*

*String[] tmp = line.split(" ");*

*length = Integer.parseInt(tmp[1]);*

*}*

*line = fromServer.readLine();*

*}*

*} catch (IOException e) {*

*System.out.println("Error reading headers from server: " + e);*

*}*

*try {*

*int bytesRead = 0;*

*byte buf[] = new byte[BUF\_SIZE];*

*boolean loop = false;*

*/\* If we didn't get Content-Length header, just loop until*

*\* the connection is closed. \*/*

*if (length == -1) {*

*loop = true;*

*}*

*/\* Read the body in chunks of BUF\_SIZE and copy the chunk*

*\* into body. Usually replies come back in smaller chunks*

*\* than BUF\_SIZE. The while-loop ends when either we have*

*\* read Content-Length bytes or when the connection is*

*\* closed (when there is no Connection-Length in the*

*\* response. \*/*

*while (bytesRead < length || loop) {*

*/\* Read it in as binary data \*/*

*int res = fromServer.read(buf, 0, BUF\_SIZE); /\* Read binary date until BUFF\_SIZE to buf (array in bytes) \*/*

*if (res == -1) {*

*break;*

*}*

*/\* Copy the bytes into body. Make sure we don't exceed*

*\* the maximum object size. \*/*

*for (int i = 0;*

*i < res && (i + bytesRead) < MAX\_OBJECT\_SIZE;*

*i++) {*

***body[bytesRead + i] = buf[i]; /\* copy bytes read to buf to body \*/***

*}*

*bytesRead += res;*

*}*

*} catch (IOException e) {*

*System.out.println("Error reading response body: " + e);*

*return;*

*}*

*}*

*/\*\**

*\* Convert response into a string for easy re-sending. Only*

*\* converts the response headers, body is not converted to a*

*\* string.*

*\*/*

*public String toString() {*

*String res = "";*

*res = statusLine + CRLF;*

*res += headers;*

*res += CRLF;*

*return res;*

*}*

*}*

### ProxyCache.java

*/\**

*\* ProxyCache.java - Simple caching proxy*

*\**

*\* $Id: ProxyCache.java,v 1.3 2004/02/16 15:22:00 kangasha Exp $*

*\*/*

*import java.net.\*;*

*import java.io.\*;*

*import java.util.\*;*

*import java.lang.\*;*

*public class ProxyCache {*

*/\*\* Port for the proxy \*/*

*private static int port;*

*/\*\* Socket for client connections \*/*

*private static ServerSocket socket;*

*/\*\* Create the ProxyCache object and the socket \*/*

*private static Map<String, String> cache = new Hashtable<String, String>();*

*public static void caching(HttpRequest req, HttpResponse res) throws IOException{*

*File cachedSites;*

*DataOutputStream output;*

*cachedSites = new File("cachedSites.txt");*

*FileOutputStream fileOutputStream = new FileOutputStream(cachedSites);*

*ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);*

*objectOutputStream.writeObject(cache);*

*objectOutputStream.close();*

*fileOutputStream.close();*

*cache.put(req.URI, cachedSites.getAbsolutePath());*

*System.out.println("Caching from: "+req.URI+" para "+cachedSites.getAbsolutePath());*

*}*

*public static byte[] uncaching(String name) throws IOException{*

*File file1cached;*

*FileInputStream fileInput;*

*String hashfile;*

*byte[] bytescached;*

*if((hashfile = cache.get(name))!=null){*

*file1cached = new File(hashfile);*

*fileInput = new FileInputStream(file1cached);*

*bytescached = new byte[(int)file1cached.length()];*

*fileInput.read(bytescached);*

*System.out.println("Caching: Hit on "+name+" returning cache to user");*

*return bytescached;*

*}*

*else {*

*System.out.println("Caching: No hit on "+name);*

*return bytescached = new byte[0];*

*}*

*}*

*public static void init(int p) {*

*port = p;*

*try {*

*socket = new* ***ServerSocket(port);***

*} catch (IOException e) {*

*System.out.println("Error creating socket: " + e);*

*System.exit(-1);*

*}*

*}*

*/\*\* Read command line arguments and start proxy \*/*

*public static void main(String args[]) {*

*int myPort = 0;*

*File cachedir = new File("cache/");*

*if (!cachedir.exists()){cachedir.mkdir();}*

*try {*

*myPort = Integer.parseInt(args[0]);*

*} catch (ArrayIndexOutOfBoundsException e) {*

*System.out.println("Need port number as argument");*

*System.exit(-1);*

*} catch (NumberFormatException e) {*

*System.out.println("Please give port number as integer.");*

*System.exit(-1);*

*}*

*init(myPort);*

*/\*\* Main loop. Listen for incoming connections and spawn a new*

*\* thread for handling them \*/*

*Socket client = null;*

*while (true) {*

*try {*

*client* ***= socket.accept();*** */\* Accepts new customers \*/*

***(new Thread(new Threads(client))).start(); /\* Create threads for each new client \*/***

*} catch (IOException e) {*

*System.out.println("Error reading request from client: " + e);*

*/\* Definitely cannot continue processing this request,*

*\* so skip to next iteration of while loop. \*/*

*continue;*

*}*

*}*

*}*

*}*

### Threads.java

*import java.net.\*;*

*import java.io.\*;*

*import java.util.\*;*

*import java.lang.\*;*

*public class Threads implements Runnable{*

*private final Socket client;*

*public Threads(Socket client) {*

*this.client = client;*

*}*

*public void run() {*

*Socket server = null;*

*HttpRequest request = null;*

*HttpResponse response = null;*

*/\* Process request. If there are any exceptions, then simply*

*\* return and end this request. This unfortunately means the*

*\* client will hang for a while, until it timeouts. \*/*

*/\* Read request \*/*

*try {*

*BufferedReader fromClient = new BufferedReader(new InputStreamReader(client.getInputStream()));*

*request = new HttpRequest(fromClient);*

*} catch (IOException e) {*

*System.out.println("Error reading request from client: " + e);*

*return;*

*}*

*/\* Send request to server \*/*

*try {*

*/\* Open socket and write request to socket \*/*

*server = new Socket(request.getHost(), request.getPort()); /\* Create socket \*/*

*DataOutputStream toServer = new DataOutputStream(server.getOutputStream()); /\* Create outputstream for server on socket \*/*

*toServer.writeBytes(request.toString()); /\* Create outputstream for the serviceScribe request to the outputstreamidor on the socket \*/*

*} catch (UnknownHostException e) {*

*System.out.println("Unknown host: " + request.getHost());*

*System.out.println(e);*

*return;*

*} catch (IOException e) {*

*return;*

*}*

*/\* Read response and forward it to client \*/*

*try {*

*byte[] cache = ProxyCache.uncaching(request.URI);*

*if(cache.length==0) {*

*DataInputStream fromServer = new DataInputStream(server.getInputStream()); /\* Create server inputstream \*/*

*response = new HttpResponse(fromServer); /\* Create object with server response \*/*

*DataOutputStream toClient = new DataOutputStream(client.getOutputStream());*

*toClient.writeBytes(response.toString()); /\* Write headers \*/*

*toClient.write(response.body); /\* Write body \*/*

*/\* Write response to client. First headers, then body \*/*

*ProxyCache.caching(request, response); /\* Save to cache \*/*

*client.close();*

*server.close();*

*/\* Insert object into the cache \*/*

*/\* Fill in (optional exercise only) \*/*

*}*

*else{*

*DataOutputStream toClient = new DataOutputStream(client.getOutputStream());*

*toClient.write(cache);*

*client.close();*

*server.close();*

*}*

*} catch (IOException e) {*

*System.out.println("Error writing response to client: " + e);*

*}*

*}*

*}*

## Part two

### Add better error handling

Task 2: Add better error handling

The simple proxy which you have completed in task 1 does no error handling. This can be a

problem especially when the client requests an object which is not available, since the "404 Not

found" response usually has no response body and the proxy assumes there is a body and tries

to read it. You need to modify code to so that it can HTTP responses with error codes.

As you can see in the ProxyCache.java above, the error handling is done by using exceptions. Additional class apart from the given classes name Threads.java has also fulfilled with error handling.

A screenshot of a computer

Description generated with high confidence

A close up of a logo

Description generated with high confidence

**ProxyCache.java**

*catch (ArrayIndexOutOfBoundsException e) {*

*System.out.println("Need port number as argument");*

*System.exit(-1);*

*} catch (NumberFormatException e) {*

*System.out.println("Please give port number as integer.");*

*System.exit(-1);*

*}*

**Threads.java**

*} catch (UnknownHostException e) {*

*System.out.println("Unknown host: " + request.getHost());*

*System.out.println(e);*

*return;*

*} catch (IOException e) {*

*return;*

*}*

## Part 3

### Add caching

Task 3: Add caching

Add the simple caching functionality to the proxy you completed in task 2 above. You do not

need to implement any replacement or validation policies. Your implementation will need to be

able to write responses to the disk (i.e., the cache) and fetch them from disk when you get a

cache hit. For this you need to implement some internal data structure in the proxy to keep track

of which objects are cached and where they are on disk. You can keep this data structure in

main memory; there is no need to make it persist across shutdowns.

Here is the screenshot for cached packets in the text file.

A screenshot of a computer screen

Description generated with very high confidence

*public static void caching(HttpRequest req, HttpResponse res) throws IOException{*

*File cachedSites;*

*DataOutputStream output;*

*cachedSites = new File("cachedSites.txt");*

*FileOutputStream fileOutputStream = new FileOutputStream(cachedSites);*

*ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);*

*objectOutputStream.writeObject(cache);*

*objectOutputStream.close();*

*fileOutputStream.close();*

*cache.put(req.URI, cachedSites.getAbsolutePath());*

*System.out.println("Caching from: "+req.URI+" para "+cachedSites.getAbsolutePath());*

*}*

### Class diagram

A screenshot of a cell phone

Description generated with very high confidence

# References

1. <https://github.com/Insatisfeito/ProxyServer/blob/master/ProxyCache.java>
2. <https://stackoverflow.com/questions/29091648/cache-webpage-using-java-proxy-server>
3. [www.youtube.com](http://www.youtube.com)
4. <http://study-ccna.com/>

# Group members

1. Pamodi Rathnayke-BSC-UCD-CSC-16.2-016/10009210
2. Achini Rathnayke-BSC-UGC-MIS-16.2-042/10009227
3. Helitha Thalagalaarachchi-BSC-UCD-CSC-16.2-025/10009190
4. Kavindu Fonseka-BSC-UCD-CSC-004/10009215
5. Hashan Jayalath- BSC-UCD-CSC-16.2-009/10009207
6. Buddi Perera-BSC-PLY-COM-16.2-199/10009004