**Error! Unknown document property name.**

**Error! Unknown document property name.**

High Level Design & Low Level Design

The purpose of this document is to provide with a template for documenting both HLD & LLD.

**Document Control :**

| **Project Revision History** | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 12/01/23 | 0.1 | 09-01-2023\_LINUX17NOVB1\_SPRINT-2\_F | Initial Draft on Introduction and Designing | | | |  | |
| 13/01/23 | 0.2 | 09-01-2023\_LINUX17NOVB1\_SPRINT-2\_F | Modification on Designing and System Architecture | | | |  | |
| 16/01/23 | 0.3 | 09-01-2023\_LINUX17NOVB1\_SPRINT-2\_F | Modification on System Design and Environment | | | |  | |

[**1. Introduction 3**](#_heading=h.3znysh7)

[1.1. Intended Audience 3](#_heading=h.2et92p0)

[1.2. Acronyms/Abbreviations 3](#_heading=h.tyjcwt)

[1.3. Project Purpose 3](#_heading=h.3dy6vkm)

[1.4. Key Project Objectives 3](#_heading=h.1t3h5sf)

[1.5. Project Scope and Limitation 3](#_heading=h.4d34og8)

[*1.5.1. In Scope 3*](#_heading=h.2s8eyo1)

[*1.5.2. Out of scope 3*](#_heading=h.17dp8vu)

[1.6. Functional Overview 3](#_heading=h.3rdcrjn)

[1.7. Assumptions, Dependencies & Constraints 3](#_heading=h.26in1rg)

[1.8. Risks 3](#_heading=h.lnxbz9)

[**2. Design Overview 3**](#_heading=h.35nkun2)

[2.1. Design Objectives 3](#_heading=h.1ksv4uv)

[*2.1.1. Recommended Architecture 3*](#_heading=h.44sinio)

[2.2. Architectural Strategies 3](#_heading=h.2jxsxqh)

[*2.2.1. Design Alternative 3*](#_heading=h.z337ya)

[*2.2.2. Reuse of Existing Common Services/Utilities 3*](#_heading=h.3j2qqm3)

[*2.2.3. Creation of New Common Services/Utilities 3*](#_heading=h.1y810tw)

[*2.2.4. User Interface Paradigms 3*](#_heading=h.4i7ojhp)

[*2.2.5. System Interface Paradigms 3*](#_heading=h.2xcytpi)

[*2.2.6. Error Detection / Exceptional Handling 3*](#_heading=h.1ci93xb)

[*2.2.7. Memory Management 3*](#_heading=h.3whwml4)

[*2.2.8. Performance 3*](#_heading=h.2bn6wsx)

[*2.2.9. Security 3*](#_heading=h.qsh70q)

[*2.2.10. Concurrency and Synchronization 3*](#_heading=h.3as4poj)

[*2.2.11. Housekeeping and Maintenance 3*](#_heading=h.1pxezwc)

[**3. System Architecture 3**](#_heading=h.49x2ik5)

[3.1. System Architecture Diagram. (Not Necessary) 3](#_heading=h.2p2csry)

[3.2. System Use-Cases 3](#_heading=h.147n2zr)

[3.3. Subsystem Architecture 3](#_heading=h.3o7alnk)

[3.4. System Interfaces 3](#_heading=h.23ckvvd)

[*3.4.1. Internal Interfaces 3*](#_heading=h.ihv636)

[*3.4.2. External Interfaces 3*](#_heading=h.32hioqz)

[**4. Detailed System Design 3**](#_heading=h.1hmsyys)

[4.1. Key Entities 3](#_heading=h.41mghml)

[4.2. Detailed-Level Database Design 3](#_heading=h.2grqrue)

[*4.2.1. Data Mapping Information 3*](#_heading=h.vx1227)

[*4.2.2. Data Conversion 3*](#_heading=h.3fwokq0)

[4.3. Archival and retention requirements 3](#_heading=h.1v1yuxt)

[4.4. Disaster and Failure Recovery 3](#_heading=h.4f1mdlm)

[4.5. Business Process workflow 3](#_heading=h.2u6wntf)

[4.6. Business Process Modeling and Management (as applicable) 3](#_heading=h.19c6y18)

[4.7. Business Logic 3](#_heading=h.3tbugp1)

[4.8. Variables 3](#_heading=h.28h4qwu)

[4.9. Activity / Class Diagrams (as applicable) 3](#_heading=h.nmf14n)

[4.10. Data Migration 3](#_heading=h.37m2jsg)

[*4.10.1. Architectural Representation 3*](#_heading=h.1mrcu09)

[*4.10.2. Architectural Goals and Constraints 3*](#_heading=h.46r0co2)

[*4.10.3. Logical View 3*](#_heading=h.2lwamvv)

[*4.10.4. Architecturally Significant Design Packages 3*](#_heading=h.111kx3o)

[*4.10.5. Data model 3*](#_heading=h.3l18frh)

[*4.10.6. Deployment View 3*](#_heading=h.1egqt2p)

[**5. Environment Description 3**](#_heading=h.3ygebqi)

[5.1. Time Zone Support 3](#_heading=h.2dlolyb)

[5.2. Language Support 3](#_heading=h.sqyw64)

[5.3. User Desktop Requirements 3](#_heading=h.3cqmetx)

[5.4. Server-Side Requirements 3](#_heading=h.1rvwp1q)

[*5.4.1. Deployment Considerations 3*](#_heading=h.4bvk7pj)

[*5.4.2. Application Server Disk Space 3*](#_heading=h.2r0uhxc)

[*5.4.3. Database Server Disk Space 3*](#_heading=h.1664s55)

[*5.4.4. Integration Requirements 3*](#_heading=h.3q5sasy)

[*5.4.5. Jobs 3*](#_heading=h.25b2l0r)

[*5.4.6. Network 3*](#_heading=h.kgcv8k)

[*5.4.7. Others 3*](#_heading=h.34g0dwd)

[5.5. Configuration 3](#_heading=h.1jlao46)

[*5.5.1. Operating System 3*](#_heading=h.43ky6rz)

[*5.5.2. Database 3*](#_heading=h.2iq8gzs)

[*5.5.3. Network 3*](#_heading=h.xvir7l)

[*5.5.4. Desktop 3*](#_heading=h.3hv69ve)

[**6. References 3**](#_heading=h.1x0gk37)

[**7. Appendix 3**](#_heading=h.4h042r0)

# 

# Introduction

A port scanner is an application designed to probe a server or host for open ports. Such an application may be used by administrators to verify security policies of their networks and by attackers to identify network services running on a host and exploit vulnerabilities. A **portscaning** is a process that sends client requests to a range of server port addresses on a host, with the goal of finding an active port. The majority of uses of a port scan are not attacks, but rather simple probes to determine services available on a remote machine.

This project is using CLI to perform. A port scanner **determines the status of the ports** using different network scanning protocols, by sending a network packet to each port within the specified address range, one at a time. Based on the response received from the ports, it determines their status as listening or non-listening ports.

## Intended Audience

| Clients |  |
| --- | --- |
|  |  |

## Acronyms/Abbreviations

| PSIUN | PORT SCANNER IMPLEMENTATION USING NETSTAT |
| --- | --- |
| netstat | NETWORK STATISTICS |
| PS | PORT SCANNER |
| CLI | COMMAND LINE INTERFACE |

## Project Purpose

The main aim of the project is to develop a PSIUN to identify which ports are open on a computer. Port scanning is a network reconnaissance technique designed to identify which ports are open on acomputer. This can enable the scanner to identify the applications running on the system as certain programs listen on particular ports and react to traffic in certain ways.

## Key Project Objectives

The key objectives of this project are:

* Client-Server Connection: A connection will be established between the client and server, it is the initial step
* Client to display menu to perform below operations using netstat command with appropriate options.
* All outputs displayed should be properly formatted and indented at client
* Concurrent Server to Serve Multiple Clients: A concurrent Server has to be maintained in order to serve Multiple Client Connection. Multiple client requests maintenance to be done
* Server shall receive client request, execute netstat command with appropriate options, capture output, store in a datastructure and then send the data to the client.
* Server to close client socket after request serving and on client exit.
* Should include debug log messages with atleast 4 levels (FATA, INFO, WARNING, DEBUG).

## Project Scope and Limitation

Port scanner using netstat and client server connections.

### In Scope

PS work by attempting to connect to all or a set of predefined ports on a system that provides network or application access.

* It may help collecting all internet connection’s details for a particular IP address.
* User can choose any specific function for a particular detail of IP address.
* User can get routing table, active socket connection’s details for UDP and TCP, status of the port and pulling and viewing network statistics sorted by protocol.

Although a port scanner is designed to aid network administrators in strengthening security, if used unethically by hackers it can reveal open ports that could be exploited.

### Out of scope

* If the port is open, the scanner has to resent the packet multiple times leading to the delay.
* In this application user needs to choose one functionality at a time.

## Functional Overview

To create this project “netstat” command is used. Basically, the network statistics (netstat) command is a networking tool used for troubleshooting and configuration that can also serve as a monitoring tool for connections over the network. Both incoming and outgoing connections, routing tables, port listening, and usage statistics are common uses for this command.

## Assumptions, Dependencies & Constraints

With the use of c++ language , Internet and Vmware we can create a port scanner where a connection will be established between the client and server. Client to display a menu to perform different operations using netstat command.

## Risks

NA

# Design Overview

The project is designed by partitioning the program into functions. These functions are called based on the user choice inputs and his/her requirements. A specific CLI tool played a major role in this project i.e. netstat.

## Design Objectives

* *Client-Server Connection: A connection will be established between the client and server, it is the initial step by*
* *Create socket*
* *Bind*
* *Listen*
* *Client is sending request to the server to execute the following commands:*

*1. Print Protocol Statistics (netstat -s)*

*2. Print Routing Table Information (netstat -r)*

*3. Print Active TCP socket connections (netstat -tn)*

*4. Print Active UDP socket connections (netstat -un)*

*5. Print all Internet connections which are using applications like firefox/chrome (netstat -tanp | grep -i firefox)*

*6. print all the internet connections where the state is LISTEN/ ESTABLISHED for udp/tcp protocol (netstat -tanp | grep -i LISTEN)*

* Server shall receive client request, execute netstat command with appropriate options, capture output, store in a datastructure and then send the data to the client.
* Client receives the data from server.
* All outputs displayed should be properly formatted and indented at client

### Recommended Architecture

*The recommended system architecture is as follows.*

*· 1GB RAM*

*· 500MHz Processor*

*· 120GB HDD CPU*

*· Internet connection*

*· Linux operating system / CLI*

## Architectural Strategies

A port scanner is an application designed to probe a server or host for open ports. Such an application may be used by administrators to verify security policies of their networks and by attackers to identify network services running on a host and exploit vulnerabilities.

A **port scan** is a process that sends client requests to a range of server port addresses on a host, with the goal of finding an active port; this is not a nefarious process in and of itself. The majority of uses of a port scan are not attacks, but rather simple probes to determine services available on a remote machine.

### Design Alternative

NA

### Reuse of Existing Common Services/Utilities

The project is reusing the existing common services or utilities. We are using netstat commands repeatedly.

### Creation of New Common Services/Utilities

A multiple text file is being created that is storing results of the project.

### User Interface Paradigms

This system allows users to easily monitoring their networks using internet connection. Just by entering a choice, user can get details about their IP address. We have made a very simple and easy to understand interface. All the essential details about ports of every protocol can be viewed by pressing a key.

### System Interface Paradigms

Good design is good business. If the system has good interface and it satisfies user requirements, then the software can reach new heights. The port scanner offers basic ports tracking and scanning functionality. The user can scan available options and can also save their details on a GUI text file.

### Error Detection / Exceptional Handling

In this system some errors may occur if system requirements are not fulfilled. For example: If user wants to see all the internet connections which are using the browser in the system and the browser is not running on the host, then It will not provide any details about ports of browser. So, this can be handled by running the browser in background.

### Memory Management

NA

### Performance

NA

### Security

NA

### Concurrency and Synchronization

NA

### Housekeeping and Maintenance

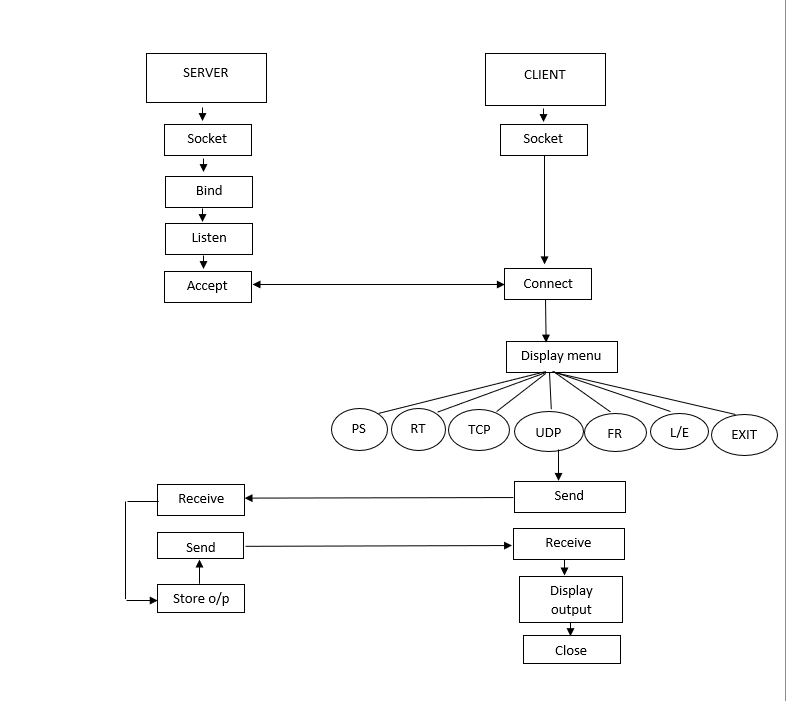
NA

# System Architecture

This project has one main function that is calling a menu function to perform different functions. In the menu functions if the user wants to get details of protocol statistics, then that particular function gets called. This menu function provides 7 functionalities; each functionality is being performed by calling the related function. If the user wants to exit from the scanner, he/she will press 7 to exit from the program.

After each function there is respond function that is being used to take user input to perform more operation or to exit from program at that stage. Functions are repeatedly calling to perform in this project. The major responsibility is carrying by netstat commands and those CLI commands are used in C++ program with system function. System function is executing the netstat commands.

## System Architecture Diagram. (Not Necessary)

**

## System Use-Cases

## 

## Subsystem Architecture

Inside every function, the netstat commands are used. A port scanner will send a TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) network packet and inquire the port about their status. The response can be as the following states:

· Open – An open port implies that when someone tries to connect to that port on the server, the server might respond in some way.

· Closed – As the name suggests, a closed port indicates that the server isn’t responding to any connections.

· Filtered – A filtered port indicates that a firewall or some antivirus/anti-malware program is blocking the port to avoid certain connections.

## System Interfaces

### Internal Interfaces

The internal interfaces comprise interfaces through which the system interacts with the clients, through which it provides them services i.e; netstat tool and internet connection.

### External Interfaces

The external interface comprises interfaces through which the users interact with the system i.e; Linux operating system and Internet connection.

# Detailed System Design

* *Client-Server Connection: A connection will be established between the client and server, it is the initial step by*
* *Create socket-Socket are generally employed in* ***client server applications****. The server creates a socket, attaches it to a network port addresses then waits for the client to contact it. The client creates a socket and then attempts to connect to the server socket.*
* *Bind-bind ()* ***assigns a socket to an address****. When a socket is created using socket (), it is only given a protocol family, but not assigned an address. This association with an address must be performed with the bind () system call before the socket can accept connections to other hosts.*
* *Listen-The listen () function applies only to stream sockets. It indicates a readiness to accept client connection requests, and creates a connection request queue of length backlog to queue incoming connection requests.*
* A menu function which has 6 functions we are calling that by the integer value which we are used in the switch case.

*1. Print Protocol Statistics (netstat -s)*

*2. Print Routing Table Information (netstat -r)*

*3. Print Active TCP socket connections (netstat -tn)*

*4. Print Active UDP socket connections (netstat -un)*

*5. Print all Internet connections which are using applications like firefox/chrome (netstat -tanp | grep -i firefox)*

*6. print all the internet connections where the state is LISTEN/ ESTABLISHED for udp/tcp protocol (netstat -tanp | grep -i LISTEN)*

* *Client is sending request to the server to execute the following commands:*

* Server shall receive client requests, execute netstat command with appropriate options,

capture output, store in a data structure and then send the data to the client.

* Client receives the data from the server.
* All outputs displayed should be properly formatted and indented at client

## Key Entities

Key entities we used in our project are

* PS
* RT
* TCP
* UDP
* Firefox
* Listen
* Exit

## Detailed-Level Database Design

NA

### Data Mapping Information

NA

### Data Conversion

NA

## Archival and retention requirements

NA

## Disaster and Failure Recovery

NA

## Business Process workflow

NA

## Business Process Modeling and Management (as applicable)

NA

## Business Logic

*The Client constructor and Server constructer is starting the client and server program and calling other functions that will provide desired results.*

* *createSock()- The server creates a socket, attaches it to a network port addresses then waits for the client to contact it.*
* *bindClient()-When a socket is created using socket (), it is only given a protocol family, but not assigned an address. This association with an address must be performed with the bind () system call before the socket can accept connections to other hosts.*
* *listenClient()-The listen () function applies only to stream sockets. It indicates a readiness to accept client connection requests, and creates a connection request queue of length backlog to queue incoming connection requests.*
* *send\_recv()- messages can be transferred between server and client.*
* *closeSock()- Socket closed in the end of the program.*

## Variables

*The project has used few variable :*

* char cmd[] and char out[]
* int sockfd
* int client\_fd
* int newsockfd
* string temp
* char buffer[]
* char buffData[]
* int count
* string data
* string psData
* string rtData
* string tcpData
* string udpData
* string listenData
* string totalData
* string line

## Activity / Class Diagrams (as applicable)

## 

## Data Migration

NA

### Architectural Representation

NA

### Architectural Goals and Constraints

NA

### Logical View

NA

### Architecturally Significant Design Packages

NA

### Data model

**Legacy system data model**

**Proposed system data model**

**Interface data model**

NA

### Deployment View

In order to run this project, user needs an internet connection and command line interface. If there is no internet connection, user will not be unable to perform scanning.

# Environment Description

The Environment used is the Linux Operating System

## Time Zone Support

It will support time zone as per Indian standard time (IST) in (GMT +5:30) and IST standard.

## Language Support

**C++ programming language** was developed in 1980 by Bjarne Stroustrup at bell laboratories of AT&T (American Telephone & Telegraph), located in U.S.A.

## User Desktop Requirements

*Operating System: Linux or Windows with Linux Subsystem*

*Processor: Minimum 1GHz and More*

*Hard Drive: 30GB and More*

*Memory(RAM): Minimum 1GB and More*

## Server-Side Requirements

*Uninterrupted Internet connectivity*

### Deployment Considerations

*500Mhz Processor*

*120GB HDD CPU*

*4GB RAM*

*Network connectivity*

### Application Server Disk Space

*Disk space – Minimum 100GB*

### Database Server Disk Space

NA

### Integration Requirements

NA

### Jobs

*Copying command into a variable using string function - strcpy*

### Network

*Internet connectivity is required.*

### Others

NA

## Configuration

NA

### Operating System

*Linux Operating System*

*Processor: Minimum 1GHz and More*

*Hard Drive: 32GB and More*

*Memory(RAM): Minimum 1GB and More*

### Database

NA

### Network

*Only stable internet connectivity is required.*

### Desktop

vmware workstation Linux

# References

[*https://www.rapidfiretools.com/blog/2021/03/22/port-scanning/*](https://www.rapidfiretools.com/blog/2021/03/22/port-scanning/)

[*https://en.wikipedia.org/wiki/Port\_scanner*](https://en.wikipedia.org/wiki/Port_scanner)

[*https://www.redhat.com/sysadmin/netstat*](https://www.redhat.com/sysadmin/netstat)

# Appendix

**Change Log**

| **QMS Template Version Control (Maintained by QA)** | | | | | |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **Date** | **Version** | **Author** | | **Description** | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |