Practical Assessment Task

Phase 1 – Specification Document

Igor Karbowy

Table of Contents

1.1 — Problem Summary

PingPal is a Java-based application used to ping devices, scan ports, and retrieve most practically relevant information about network devices.

I have a particular affinity for networking and take an interest in learning the ins and outs of computer networks. However, what I have noticed is that even though many network scanning applications have been created in the past, most are command line interface (CLI) based and not beginner-friendly.

Therefore, *PingPal* was developed with the purpose of creating a graphical user interface (GUI) based and beginner-friendly network scanning application, such that the average person could work on basic networking projects without having to learn the complicated workings of most networking applications.

PingPal contains all the functionality needed by the average person to perform basic networking tasks, such as performing ping sweeps, pinging devices, scanning ports, and establishing messaging protocols between network devices.

PingPal is not designed to suit the needs of IT professionals and large-scale companies or to derive every piece of information about a given device in a network, rather it is to provide a simple solution to individuals working on home automation or DIY tech projects that require basic networking functionality.

Another use case of *PingPal* is for educational purposes in the classroom. The application can be used by teachers to teach the basics of networking to students. Therefore, the target market of *PingPal* is defined as anyone who requires an application that provides networking functionality but has limited knowledge of networking.

1.2 — Motivation and Research

Projects similar to *PingPal* include—

A. **Wireshark—***Wireshark* is a widely used open-source network protocol analyser that captures and inspects data packets in real-time. It allows users to delve deep into the specifics of traffic patterns, protocols, and potential network vulnerabilities. *Wireshark* is highly detailed and is often used by IT professionals for troubleshooting, analysis, and security purposes.[1]

   Both *Wireshark* and *PingPal* provide insights into network traffic and devices. They offer network discovery functionalities and allow users to retrieve information about network devices.

   However, *Wireshark* is a powerful, highly technical tool aimed at advanced users, offering in-depth analysis of packet-level traffic, while *PingPal* is designed with simplicity in mind. *Wireshark's* depth and complexity are often unnecessary for home projects or beginner-level educational purposes, which is where *PingPal* excels by offering basic networking tasks without overwhelming the user.[2]

B. **Nmap—***Nmap* is a popular network scanner which is primarily used to discover devices and services on a network. It scans for open ports, detects OS versions, and can perform extensive network exploration, making it a favourite for system administrators and cybersecurity experts.[3]

   Nmap and *PingPa*l both offer device pinging and port scanning capabilities, providing users with a way to assess the network structure and its connected devices.

   However, *Nmap* is a command-line interface (CLI) based application, which makes it less accessible to beginners unfamiliar with technical commands. *PingPal*, by contrast, simplifies the experience, aiming for ease of use through a graphical user

interface (GUI) and focusing on basic network functionality for individuals with limited networking knowledge.[4]

C. **Masscan—***Masscan* is known for being one of the fastest network scanners. It is highly specialised for speed and massive-scale network exploration, making it ideal for penetration testers and security researchers.[5]

Like *PingPal*, *Masscan* can scan network ports to identify open services on devices.

However, *Masscan* is widely used by professionals for its speed and ability to scan massive IP ranges, which is far beyond the scope of what *PingPal* aims to achieve. *Masscan* is for large-scale network scanning projects, which typically require an extensive background in network engineering or similar fields. In contrast, *PingPal* is intended for smaller-scale projects like home automation or DIY tasks. Moreover, *Masscan* is also CLI-based, while *PingPal* offers a user-friendly GUI to simplify network scanning tasks for non-professionals.[6]

Therefore, the motivation behind *PingPal* is to address the need for a simple, user-friendly network scanning tool that caters to individuals who are not networking experts. Unlike *Wireshark*, *Nmap*, and *Masscan*, which are powerful but often daunting tools for beginners due to their technical complexity and steep learning curves, *PingPal* aims to make basic networking tasks accessible to anyone.

PingPal's primary differentiation is found in its graphical user interface (GUI), which is designed to be intuitive and beginner-friendly, providing a gentler introduction to networking without requiring in-depth knowledge of network protocols or command-line instructions.

References:

[1] – Wireshark (2023). *Wireshark · About*. [online] www.wireshark.org. Retrieved January 18, 2025. Available at: https://www.wireshark.org/about.html.

[2] – Chandran, A. (2023). *Comprehensive List of Network Scanning Tools*. [online] Medium. Retrieved January 18, 2025. Available at: https://medium.com/@ajithchandranr/comprehensive-list-of-network-scanning-tools-d03da398c1d4.

[3] – Lyon, G. (2024). *Nmap*. [online] Nmap. Retrieved January 18, 2025. Available at: https://nmap.org/.

[4] – Chandran, A. (2023). *Comprehensive List of Network Scanning Tools*. [online] Medium. Retrieved January 18, 2025. Available at: https://medium.com/@ajithchandranr/comprehensive-list-of-network-scanning-tools-d03da398c1d4.

[5] – Graham, R.D. (2020). *robertdavidgraham/masscan*. [online] GitHub. Retrieved January 18, 2025. Available at: https://github.com/robertdavidgraham/masscan.

[6] – Chandran, A. (2023). *Comprehensive List of Network Scanning Tools*. [online] Medium. Retrieved January 18, 2025. Available at: https://medium.com/@ajithchandranr/comprehensive-list-of-network-scanning-tools-d03da398c1d4.

1.3 — Specification of Program Functions

Program functions of *PingPal* include—

A. **Subnet scanning—**

   a. Scan local network for devices. Allow users to scan their local network (subnet) and list all connected devices to the GUI. The user can enter parameters that specify the test conditions—this includes how long the program should wait before considering the target address device offline. The subnet scanning function is multithreaded to improve execution speed.

B. **Pinging devices—**

   a. Send Internet Control Message Protocol (ICMP) echo requests. Allow users to ping devices to check connectivity. Display results to GUI.

   b. Test latency and packet loss. Run tests that ping devices that show ping results—this includes the round-time of the 3-way handshake and packet loss percentage. The user can enter parameters that specify the test conditions—this includes whether the test is continuous or run for a specific amount of pings, and the interval at which the packets should be sent. Display results to GUI.

C. **Port scanning—**

   a. Scan a specific IP for open ports from a range selected by the user. Enable users to scan a device's open ports within a range or for specific known ports—for example, HTTP: 80, FTP: 21. The user can select the specific port or range of ports to scan—the range includes all, 65535 ports. If a port is found to be open, and it is a popular port, its name should be displayed.

Display results to GUI. The port scanning function is multithreaded to improve execution speed.

D. **Simple TCP messaging functionality—**

a. Listen for connections from devices on a network to enable message exchange. Open a port specified by the user and wait a finite amount of time for a device to connect to the port. Once a connection is established, handle message exchange between the connected devices. Display the messages to the GUI.

b. Connect to a specified port to enable message exchange. Connect to a device on the network on a port specified by the user. Once a connection is established, handle message exchange between the connected devices. Display the messages to the GUI.

E. **Exporting information to and importing information from JSON files—**

a. Save results to a file. Enable users to export the results of scans—for example, ping results, and open ports—to a JSON file for future reference. Enable users to export TCP message chats to a text file for future reference.

b. Read results from a file. Allow users to input a JSON file that displays its contents to the GUI. The type of test that this JSON file is storing the results of is automatically detected, and the user is redirected to a new tab of the corresponding test (see 1.4).

F. **Error handling—**

a. Handle different errors that may occur at different parts of the application. The application makes use of different data validation techniques—for example, check digit, format check, length check, lookup table, presence check, range

check, type check, and logic check—to prevent the application from crashing due to user input error.

1.4 — Specification of Interface

The graphical user interface (GUI) of *PingPal* consists of—

A. **Home Page—**

a. This is the main window of the application. It is where the users access all the key functionalities.

b. The main window displays the name of the application—*PingPal*—in the title bar.

c. The home page contains a menu with all the key functionalities. The *Networking* menu contains the list of networking functionalities provided by the application (see 1.3). Clicking on one of the items in the menu switches to the card in the application that contains the functionality and logic of that program function.

B. **Subnet Scanning Card—**

a. The title of this card is *Subnet Scan*.

b. One of the input fields is the network/subnet input field. The network/subnet field, allows users to input the range—for example, 192.168.0.0/24—to scan for online devices. This field has a hint—for example, “e.g. 192.168.0.0/24”.

c. Another input field is the timeout field. The timeout field allows the user to select how long the program waits before that IP address is considered offline. This field is a spinner and has a set minimum and maximum timeout value.

d. There is a button, labelled *Start Subnet Scan*, which starts the subnet scan. Once pressed, this button should change to an *End Subnet Scan* button which, if pressed, stops the scan prematurely.

e. The results section of this card displays a table of discovered devices. The one column of the table displays the IP addresses of devices which are online.

f. There is a progress bar which indicates how much of the scan has been completed.

g. There is a button, labelled *Export Results*, which exports the results of the subnet scan to a JSON file.

C. **Device Pinging Card—**

a. The title of this card is *Device Ping*.

b. One of the input fields is the input for the target device's IP address. It is the input field for the user to enter the IP address to ping—for example, 192.168.0.1. This field has a hint—for example, "e.g. 192.168.0.1".

c. Another input field in this card is the ping interval. It is the input field for the interval between pings (entered in milliseconds). This field has a set minimum and a maximum value.

d. Another input field in this card is the number of pings. It is the input field for the user to specify how many pings should be sent. This field has a set minimum and a maximum value.

e. The last input field is a checkbox that, if ticked, allows continuous pinging. If the checkbox is ticked, the value in the number of pings field is disregarded.

f. There is a button, labelled *Start Device Ping*, which starts the subnet scan. Once pressed, this button should change to an *End Device Ping* button which, if pressed, stops the test prematurely.

g. The results section of this card displays real-time ping results in a table. This includes the IP address of the device being pinged, the round trip time (ms), whether the device is reachable, and real-time packet loss percentage.

h. Once the scan is completed or has been interrupted, two more tables show information which summarises the test. The first table shows the minimum round trip time, maximum round trip time, and the average round trip time. The second table shows the amount of pings sent, the amount of successful pings, the among of unsuccessful pings, and the final packet loss percentage.

i. There is a button, labelled *Export Results*, which exports the results of the ping test to a JSON file.

D. **Port Scanning Card—**

a. The title of this card is *Port Scan*.

b. One of the input fields is the input for the target device's IP address. It is the input field for the user to enter the IP address of the device whose ports are being scanned—for example, 192.168.0.1. This field has a hint—for example, "e.g. 192.168.0.1".

c. There are two spinners which work hand in hand to allow the user to select a range of ports to scan. These fields have a set minimum and a maximum value, indicating the minimum and maximum port numbers possible.

d. There is a button, labelled *Start Port Scan*, which starts the port scan. Once pressed, this button should change to an *End Port Scan* button which, if pressed, stops the test prematurely.

e. There is a progress bar which indicates how much of the scan has been completed.

f. The results section of this card displays a table of open ports. The columns of this table include the port number and the port protocol/service associated with that port number. The results should be dynamically updated in real-time as the scan progresses.

g. There is a button, labelled *Export Results*, which exports the results of the port scan to a JSON file.

E. **TCP Message Listen Card—**

a. The title of this card is *TCP Message Listen*.

b. The input field in this card is a spinner which allows the user to select the port to listen on for a TCP connection. This field has a minimum and a maximum value, indicating the minimum and maximum port numbers possible.

c. There is a button, labelled *Start TCP Listen*, which starts the server. Once pressed, this button should change to an *End TCP Listen* button which, if pressed, stops the connection.

d. There is a text pane which displays all the system messages regarding the connection status and messages exchanged between connected devices. The different types of messages should be colour-coded such that the user will be able to differentiate them.

e. Another input field is the input for the message the user wants to send. It is the input field for the user to enter the message to send to the device on the other side of the TCP connection.

f. There is a button, labelled *Send*, which sends the message written in the message input field. Once pressed, this button sends the message to the device on the other side of the TCP connection, displays it in the text pane, and finally, clears the message text field.

g. There is a button, labelled *Export Results*, which exports the messages exchanged (the text data from the text pane) to a text file.

F. **TCP Message Connect Card—**

a. The title of this card is TCP Message Connect.

b. One of the input fields is the input for the server device's IP address. It is the input field for the user to enter the IP address to connect to—for example, 192.268.0.1. This field has a hint—for example, "e.g. 192.168.0.1".

c. Another input field in this card is a spinner which allows the user to select the port to listen on for a TCP connection. This field has a minimum and a maximum value, indicating the minimum and maximum port numbers possible.

d. There is a button, labelled *Start TCP Connect*, which starts trying to establish a connection to the server. Once pressed, this button should change to an *End TCP Connect* button which, if pressed, stops the connection.

e. There is a text pane which displays all the system messages regarding the connection status and messages exchanged between connected devices. The different types of messages should be colour-coded such that the user will be able to differentiate them.

f. Another input field is the input for the message the user wants to send. It is the input field for the user to enter the message to send to the device on the other side of the TCP connection.

g. There is a button, labelled *Send*, which sends the message written in the message input field. Once pressed, this button sends the message to the device on the other side of the TCP connection, displays it in the text pane, and finally, clears the message text field.

h. There is a button, labelled *Export Results*, which exports the messages exchanged (the text data from the text pane) to a text file.

G. **Import Results Pop-Up—**

a. The title of this window is *Import Results*.

b. This window itself is a file chooser input field which allows the user to select the file location of the results JSON file. There is a button, labelled *Open*, which imports the results from the JSON file.

c. Once the results are identified, the user is redirected to the card of the corresponding test type and the results from the file are displayed to the GUI.

1.6 — Specification of Permanent Data Storage

Permanent data storage of *PingPal* includes—

A. **File storage for exported results—**

   a. *PingPal* allows users to export the results of subnet scans, ping tests, and port scans to JSON files. The format for each result file will follow a structured, readable format, so the user can review or import the file easily.

   b. *PingPal* allows users to export the messages sent between two devices during a TCP message session to text files. The format of each text will just be the raw text displayed on the message panel.

   c. The files are exported to a folder of the user's choosing.

B. **File storage regarding importing results—**

   a. *PingPal* allows users to import the results of previous subnet scans, ping tests, and port scans from JSON files. The format for each result follows a structured, readable format, designed by the program.

   b. The files are imported from a folder of the user's choosing.

C. **File storage of read-only files—**

   a. *PingPal* stores a comma-separated values file, titled *port_list.csv*, within the program data. This file stores the relationship between port number and protocol for each port. This data is final and cannot be modified by the user, thus the file is never written to—only read.

   b. PingPal stores an image file, titled *icon.png*, within the program data. This is the icon for the application. The icon does not change, thus the file is never written to—only read.