Mohamed Ayoub – 900212213

Omar Saleh – 900205016

Yehia Ragab – 900204888

Ziad Amin – 900201190

Operating Systems – Spring 2024

Dr. Amr El-Kadi

Project: Network Monitor Report - NETSCOPE

Programming Language and GUI used

Rust is used throughout the monitor implementation in the Backend and the Frontend of the GUI. **“eframe”** and **“egui”** are used to build interactive and real-time visualizations. These libraries allow for the creation of windows, panels, and dynamic plots, providing a responsive and modern user interface. Here is a summary of some of the implementations that shows the usage of the rust and GUI tools:

Main Window

Purpose: Display the primary interface where users can view network statistics.

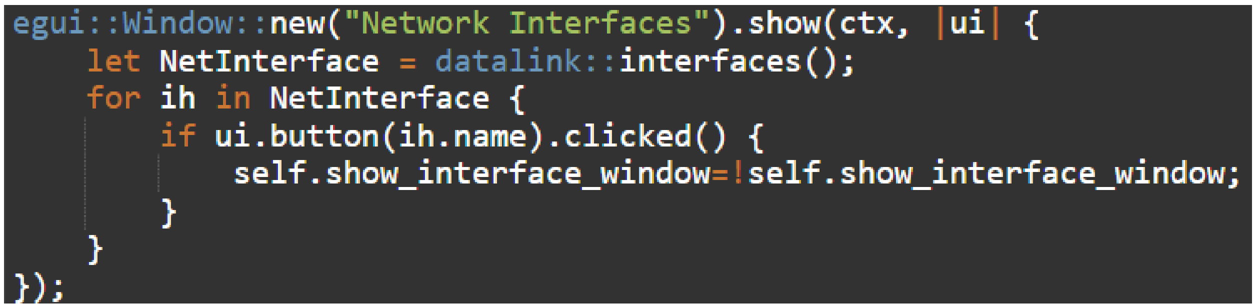
Implementation: eframe::run_native("Download Bits / Sec", native_options, Box::new(|_| Box::new(app)));

Description: This initializes the main application window where real-time data will be displayed.

Network Interfaces Window

Purpose: List and allow selection of network interfaces.

Implementation:



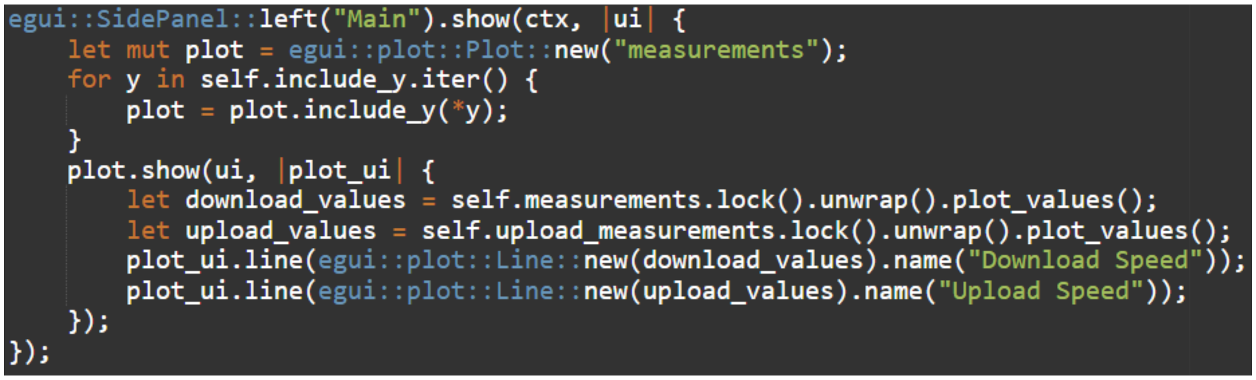
```
egui::Window::new("Network Interfaces").show(ctx, |ui| {
    let NetInterface = datalink::interfaces();
    for ih in NetInterface {
        if ui.button(ih.name).clicked() {
            self.show_interface_window=!self.show_interface_window;
        }
    }
});
```

Description: This window displays available network interfaces. Users can select an interface to view detailed statistics.

Real-time Data Plot

Purpose: Display real-time network data (e.g., download and upload speeds).

Implementation:



```
egui::SidePanel::left("Main").show(ctx, |ui| {
    let mut plot = egui::plot::Plot::new("measurements");
    for y in self.include_y.iter() {
        plot = plot.include_y(*y);
    }
    plot.show(ui, |plot_ui| {
        let download_values = self.measurements.lock().unwrap().plot_values();
        let upload_values = self.upload_measurements.lock().unwrap().plot_values();
        plot_ui.line(egui::plot::Line::new(download_values).name("Download Speed"));
        plot_ui.line(egui::plot::Line::new(upload_values).name("Upload Speed"));
    });
});
```

Description: This side panel displays real-time plots of download and upload speeds using egui::plot::Plot. The data is dynamically updated to reflect current network activity.

Technical Specifications & Features

- Data Collection
    - ➔ System Name
    - ➔ Number of cores/Disks
    - ➔ Kernel Version
    - ➔ Memory in bytes (Total, Available, Free)
    - ➔ Number of Processes
    - ➔ Memory Usage (In percentage)
    - ➔ CPU Usage (In percentage)
    - ➔ Bits per second (overall and per interface)
    - ➔ Distribution of protocols used (e.g., TCP, UDP, ICMP)
    - ➔ Downloaded and Uploaded packets per second
    - ➔ Downloaded and Uploaded speed per second
    - ➔ Domain Names and IP addresses
    - ➔ Processes Linkage to Domains
- Real-time Analysis
    - ➔ Bandwidth usage trends (Hosts Uploads and Downloads Rate)
    - ➔ Top traffic sources (Processes & IP addresses)
- Reporting
    - ➔ Users can generate historical reports on network traffic data for a defined period of time.
- Network Protocol Analysis
    - ➔ Support for a wide range of network protocols (IP, TCP, UDP).

- ➔ Control the and limit the transfer of protocols, downloads, and uploads.

- User Interface
  - ➔ Informative dashboards displaying real-time network activity
  - ➔ Well-structured menus for navigation
  - ➔ Interactive graphs and charts for data visualization
  - ➔ Sending Notification Emails for users for threshold Activity
  - ➔ Controlling thresholds for each IP

Technical Specifications & Features Explained

System Name

- **Function and Library**: uname crate.
- **Meaning and Purpose**: Retrieve the name of the operating system.
- **Expected Output**: String containing the system name.
- **Testing**: Compare with the actual system name of the server.

Number of cores/Disks

- **Function and Library**: sysinfo crate.
- **Meaning and Purpose**: Retrieve the number of CPU cores and disks.
- **Expected Output**: Integer values.
- **Testing**: Compare with actual CPU core count and disk count.

Kernel Version

- **Function and Library**: uname crate.
- **Meaning and Purpose**: Retrieve the kernel version.
- **Expected Output**: String representing the kernel version.
- **Testing**: Compare with the actual kernel version of the system.

Memory (Total, Available, Free)

- **Function and Library**: sysinfo crate.
- **Meaning and Purpose**: Retrieve total, available, and free memory.
- **Expected Output**: Integer values in bytes.

- **Testing**: Sum of available and free memory should equal total memory.

Number of Processes

- **Function and Library**: sysinfo crate.
- **Meaning and Purpose**: Retrieve the number of running processes.
- **Expected Output**: Integer value.
- **Testing**: Compare with the actual number of running processes.

Memory Usage (Percentage)

- **Function and Library**: sysinfo crate.
- **Meaning and Purpose**: Calculate memory usage percentage.
- **Expected Output**: Float value representing percentage.
- **Testing**: Verify the calculation against actual memory usage.

CPU Usage (Percentage)

- **Function and Library**: sysinfo crate.
- **Meaning and Purpose**: Calculate CPU usage percentage.
- **Expected Output**: Float value representing percentage.
- **Testing**: Verify the calculation against actual CPU usage.

Bits per second (overall and per interface)

- **Function and Library**: pnet.
- **Purpose**: Measure network speed overall and per interface.

- **Expected Output**: Network speed in bits per second.
- **Testing**: Use network benchmarking tools like iperf for validation.

Distribution of protocols used (e.g., TCP, UDP, ICMP)

- **Function and Library**: pnet.
- **Purpose**: Analyze and distribute network protocols.
- **Expected Output**: Distribution of protocols.
- **Testing**: Validate against network traffic analysis using Wireshark.

Downloaded and Uploaded packets per second

- **Function and Library**: pnet.
- **Purpose**: Count downloaded and uploaded packets per second.
- **Expected Output**: Packets per second for download and upload.
- **Testing**: Compare with network statistics tools like nload.

Downloaded and Uploaded speed per second

- **Function and Library**: pnet.
- **Purpose**: Measure download and upload speed per second.
- **Expected Output**: Download and upload speed in bytes per second.
- **Testing**: Validate with network speed test tools.

Domain Names and IP addresses

- **Functionality**: Map domain names to IP addresses using DNS lookups.
- **Purpose**: Identify and verify network endpoints.
- **Expected Output**: List of domain names and corresponding IP addresses.
- **Testing**: Cross-check with nslookup or dig.

Processes Linkage to Domains

- **Functionality**: Link running processes to domain names.
- **Purpose**: Monitor and analyze the network activity of specific processes.
- **Expected Output**: Processes linked to domain names.
- **Testing**: Validate with network analysis tools.

Bandwidth usage trends (Hosts Uploads and Downloads Rate)

- **Functionality**: Analyze bandwidth usage trends over time.
- **Purpose**: Monitor and visualize network usage patterns.
- **Expected Output**: Trends in upload and download rates.
- **Testing**: Compare trends with historical data.

Top traffic sources (Processes & IP addresses)

- **Functionality**: Identify top traffic sources by process and IP address.
- **Purpose**: Detect high-usage processes and hosts.
- **Expected Output**: List of top traffic sources.
- **Testing**: Validate with network monitoring tools like ntop.

Generate historical reports on network traffic data for a defined period of time

- **Functionality**: Generate reports on network traffic data.
- **Purpose**: Provide insights and documentation on past network usage.
- **Expected Output**: CSV report of network traffic data.
- **Testing**: Compare report data with historical data.

Support for a wide range of network protocols (IP, TCP, UDP)

- **Functionality**: Analyze and support multiple network protocols.
- **Purpose**: Ensure comprehensive network monitoring and analysis.
- **Expected Output**: Protocol support analysis.
- **Testing**: Validate protocol support with analysis tools.

Control and limit the transfer of protocols, downloads, and uploads

- **Functionality**: Control and limit network transfers.
- **Purpose**: Manage and restrict network traffic as needed.
- **Expected Output**: Controlled network transfers.
- **Testing**: Test limits by simulating traffic.

Informative dashboards displaying real-time network activity

- **Functionality**: Display real-time network activity in dashboards.
- **Purpose**: Provide a clear and interactive view of network performance.

- **Expected Output**: Real-time dashboard display.
- **Testing**: Ensure real-time updates with accurate data.

Well-structured menus for navigation

- **Functionality**: Create intuitive and structured menus.
- **Purpose**: Facilitate easy navigation through the interface.
- **Expected Output**: User-friendly navigation menus.
- **Testing**: User testing for usability.

Interactive graphs and charts for data visualization

- **Functionality**: Visualize data with interactive charts.
- **Purpose**: Enhance understanding of network data through visual representation.
- **Expected Output**: Interactive charts displaying network data.
- **Testing**: Ensure charts are interactive and accurate.

Sending Notification Emails for users for threshold Activity

- **Functionality**: Send emails when thresholds are reached.
- **Purpose**: Alert users to significant network events.
- **Expected Output**: Email notifications.
- **Testing**: Simulate threshold events and verify email delivery.

Controlling thresholds for each IP

- **Functionality**: Set and control thresholds for each IP.
- **Purpose**: Manage and monitor network usage per IP.
- **Expected Output**: Controlled thresholds for IP addresses.
- **Testing**: Set various thresholds and observe behavior.

Code Analysis

The implementation of the network monitor demonstrates various functionalities related to network monitoring, data collection, real-time analysis, and visualization using Rust.

1. Network Monitoring:
    - Libraries: pnet, eframe, rand, tracing.
    - Functionality:
        - Capture network packets.
        - Analyze packet data (e.g., protocols, traffic statistics).
        - Display real-time graphs and charts using eframe and egui.
2. Traffic Analysis:
    - Functions:
        - handle_packet(), handle_ethernet_frame(), update_traffic_stats().
        - Process packets, update traffic statistics, and handle different protocols (e.g., TCP, UDP, ICMP).
    - Data Structures:
        - TrafficStats, TrafficData: Store upload and download statistics.
        - HashMap<String, TrafficStats>: Map IP addresses to traffic statistics.
3. Real-time Visualization:
    - Libraries: eframe, egui, gtk, cairo.
    - Functionality:
        - Display real-time network activity.
        - Draw graphs and charts representing download/upload speeds.

4. Throttling and Control:
    - Functions: throttle(), set_transfer_limits().
    - Purpose: Control and limit network transfer rates.
    - Testing: Simulate high traffic and observe throttling behavior.

Therefore, each function and module is geared towards capturing, analyzing, and visualizing network data in real-time. By integrating libraries like pnet for packet capture, eframe for UI, and tracing for logging, the system can efficiently manage and display network activities, ensuring robust monitoring and user-friendly interaction.

<u>Features Not Included</u>

Advanced Security Monitoring (Intrusion Detection/Prevention)

Description: Features like intrusion detection systems (IDS) or intrusion prevention systems (IPS) monitor network traffic for suspicious activity and potential threats.

Reason for Exclusion: Implementing advanced security monitoring requires specialized algorithms and significant processing power, which may exceed the scope of our basic network monitoring tool. Additionally, these features would require continuous updates to stay effective against new threats, necessitating a more dedicated security-focused solution.

User Authentication and Access Control

Description: Implementing user authentication and role-based access control (RBAC) would allow only authorized users to access the network monitor and manage its settings. Having accounts for each user will make it easier to store the data and user’s information.

Reason for Exclusion: While important for enterprise environments, user authentication adds a layer of complexity that might not be necessary for a standalone network monitoring tool. These features are more relevant in a multi-user or enterprise setting.

Historical Data Storage and Analytics

Description: This feature would store network traffic data over long periods and provide advanced analytics for historical trends and anomalies.

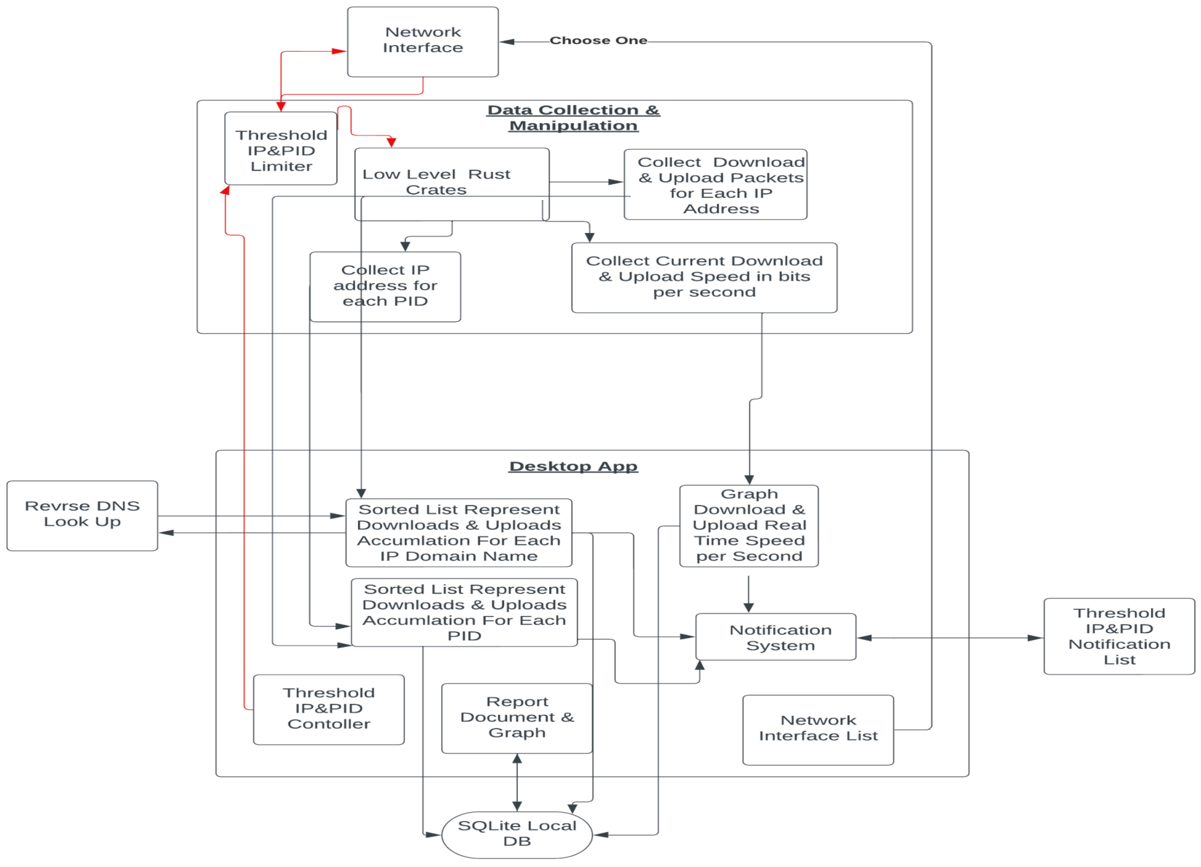
Reason for Exclusion: Storing large volumes of historical data requires robust database management and can significantly increase storage and processing requirements. Also, there was not enough time to connect the network monitor to a database to store information. One feature we aimed to implement was a reporting button on the GUI. This button would flag specific data, including the average download and upload speeds calculated by a function, and generate a list of active hosts at the precise moment the user clicked it. Upon a subsequent click, the data collected between the two clicks would be extracted for detailed analysis. However, this feature was not included due to its complexity, the need for extensive backend support, and potential performance impacts on real-time data processing. Additionally, ensuring the accuracy and reliability of the data over multiple clicks posed significant challenges that required more development time than was available in our project timeline.

Mobile App Interface

Description: A mobile application interface would allow users to monitor network activity on their smartphones or tablets.Reason for Exclusion:

Developing a mobile application would have increased the complexity of the Network monitor and it could be considered as redundant as well because it is better to be implemented in a linux operating system.

Architecture


Network Interface
Choose One
Data Collection & Manipulation
Threshold IP&PID Limiter
Low Level Rust Crates
Collect Download & Upload Packets for Each IP Address
Collect IP address for each PID
Collect Current Download & Upload Speed in bits per second
Desktop App
Revrse DNS Look Up
Sorted List Represent Downloads & Uploads Accumlation For Each IP Domain Name
Graph Download & Upload Real Time Speed per Second
Sorted List Represent Downloads & Uploads Accumlation For Each PID
Notification System
Threshold IP&PID Notification List
Threshold IP&PID Contoller
Report Document & Graph
Network Interface List
SQLite Local DB

Architecture Breakthrough:

The user starts by choosing a network interface from the list provided on the first page on GUI. Then the user is given a page to either insert a threshold or not to the device. Then the data collection unit collects PIDs, IPs, download, and upload packets. Then a graph is generated representing the upload and download data speed per second. Moreover, a table is generated beside the graph representing the hosts that affected a certain peak. Then a new page is generated representing the total number of downloaded and uploaded packets for each IP and PID. The IP is then represented as a domain if we were able to detect a domain name. Then a notification system sends the user an email if a certain threshold is passed. Furthermore, a local DB is used to accumulate data gatherings and analytics.

What makes our tool Unique?

Real-Time Visualization with eframe and egui

Uniqueness: The use of eframe and egui provides a modern, interactive, and highly responsive GUI that updates in real-time, making it visually appealing and user-friendly compared to traditional command-line tools or static GUIs.

Customizable Graphs and Charts

Uniqueness: Users can visualize network data in real-time with interactive plots, allowing for easy monitoring and quick identification of network issues. The customization options add flexibility to the data presentation.

Comprehensive Network Interface Analysis

Uniqueness: Unlike many network tools that only provide basic interface statistics, this tool offers a detailed view and allows users to select specific interfaces for a more targeted analysis.

Reverse DNS Lookup

Uniqueness: By resolving IP addresses to domain names, users can better understand the entities communicating over the network. This adds a layer of human-readable context to the raw IP addresses, making network analysis more intuitive and actionable. Reverse DNS lookups can help identify potentially malicious domains associated with suspicious IP addresses, aiding in security monitoring and threat detection.Unlike many network monitoring tools that provide only

IP-based data, integrating reverse DNS lookups directly into the tool enhances its analytical capabilities without requiring additional steps or external tools.

Data Extraction for PID, Upload, and Download

Uniqueness: This feature provides detailed, second-by-second data, enabling precise tracking of network activity at the process level. Users can pinpoint exact moments when specific processes were active and measure their network impact.

<u>Pros and cons of NETSCOPE</u>

Advantages	Disadvantages
Fast Real-Time Networking	Reporting Feature Not Fully Implemented
User-Friendly GUI	Consumes a Lot of Memory
Updating Real-Time Graphs	
Good Accuracy of Upload and Download Speeds	

1. Fast Real-Time Networking: The Network Monitor excels in providing fast, real-time networking data. This feature ensures that users can view and analyze current network statistics instantaneously, which is crucial for timely decision-making and troubleshooting in dynamic network environments.

2. User-Friendly GUI: The graphical user interface (GUI) is designed to be exceptionally user-friendly. Its intuitive layout and clear visual elements make it easy for users of all technical levels to navigate and utilize the monitor effectively, enhancing the overall user experience.

3. Updating Real-Time Graphs: One of the key features is the real-time updating graphs. These graphs continuously refresh to display the latest network performance data, allowing users to monitor trends and detect issues as they arise without any delay.

4. Good Accuracy of Upload and Download Speeds:The Network Monitor provides highly accurate measurements of upload and download speeds. These measurements have been thoroughly tested and validated against popular online speed tests, ensuring that users can rely on the data for precise network performance assessment.

Disadvantages:

1. Reporting Feature Not Fully Implemented: A notable drawback is that the reporting feature, intended to provide detailed historical analysis of network performance, is not fully implemented. This limitation means users cannot generate comprehensive reports over specific time intervals, which restricts the tool's utility for in-depth performance reviews and long-term monitoring.

2. Consumes a Lot of Memory: The Network Monitor is also known for its high memory consumption. This can lead to inefficiencies, particularly on systems with limited resources, as the tool's substantial memory usage may slow down other applications and overall system performance. This drawback affects the monitor's efficiency and suitability for use on less powerful devices.