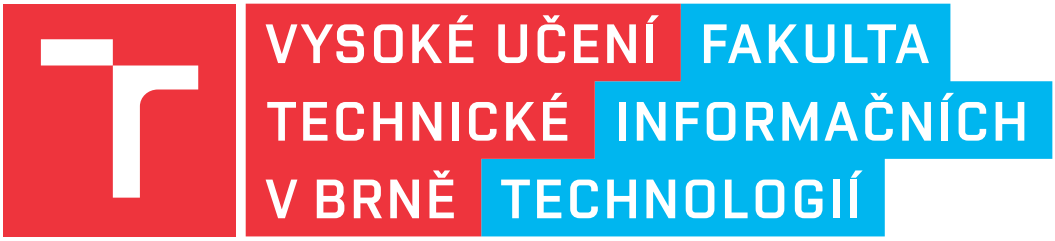

ISA 2024/2025

Network Traffic Statistics Application (isa-top)

November 18, 2024

Maksim Samusevich xsamus00

Contents

List of Figures

1 Introduction

This document describes the implementation of the ‘isa-top‘ application, which is designed to monitor real-time network traffic statistics. Program listens to a specified network interface, captures packets using the `libpcap` library [1], and calculates transmission speeds for each connection. It displays the results in the terminal using a `ncurses`-based interface [2].

2 Application Design

2.1 Packet Capturing Module

Using the `libpcap` library, this module is responsible for capturing network packets from a specified interface. It filters the relevant traffic based on protocol types and extracts important information such as source and destination IP addresses, ports and packet sizes.

2.2 Connection Management Module

This module tracking active connections by monitoring how much data is sent and received and counting the number of packets. It organizes and sorts connections to highlight the most active ones. It also handles traffic in both directions to ensure all connections are properly tracked.

To figure out the direction of a packet, the module checks if the source address matches the local device. If it does, the packet is outgoing (direction: 1). If not, it's incoming (direction: 0). If no local address is found, the module determines the direction based on the source (src) and destination (dst) addresses, assigning direction based on the packet flow.

2.3 User Interface Module

Using the `ncurses` library, the user interface module dynamically displays network statistics in the terminal. It presents the data in a structured tabular format, updating it at regular intervals to reflect changes in network traffic in real time.

2.4 Logging Module

For debugging and monitoring purposes, the application includes a logging module that records information about packet captures and connection statistics. When the debug mode is enabled via the `-d` flag, logs are written to `isa-top.log`.

3 Code Implementation

3.1 Main Functionality

The main function, `int main()`, is the starting point of the application. Its primary role is to initialize the environment, process user inputs, and manage the packet capturing loop. The function starts by parsing the command-line arguments with `parse_arguments()` [3] and also ensuring that essential parameters

such as the network interface are provided. After that, it retrieves the local IP addresses associated with the specified interface using `get_local_ips()` [4], which prepares the application to identify incoming and outgoing traffic.

To enable real-time packet capturing, `pcap_open_live()` [1] function used to bind the application to the chosen interface. For a user-friendly display, the `ncurses` library [2] is initialized, allowing for periodic updates of connection statistics (1sec by deafault). The main loop then enters a cycle of capturing packets via `pcap_next_ex()` [1] and processing them through `packet_handler()`.

The program's lifecycle is managed by a `SIGINT` handler [4], that ensure clean termination.

3.2 Command-Line Argument Parsing

The function `int parse_arguments()` is responsible for parsing and validating user-provided arguments. It ensures that the required network interface is specified and allows optional configuration of sorting mode (`-s`), update interval (`-t`), and logging (`-d`). If invalid inputs are detected, the function outputs usage instructions and terminates the program.

This function makes use of the `getopt()` library [4]. Valid arguments are stored in a `Settings` structure, which is later used to configure the application.

3.3 Packet Capture and Processing

The `packet_handler()` function is the core of the application's functionality. It processes each captured packet to extract important information such as source and destination IP addresses, port numbers, protocol type and packet size.

The function supports both IPv4 and IPv6 protocols, handling Ethernet and loopback link types. For each packet, it determines whether it is incoming or outgoing by comparing its IP addresses against the list of local IPs. This is very important for categorizing the packet as transmitted (`Tx`) or received (`Rx`). Based on the extracted data, the function invokes `update_connection()` to update or create a connection entry in the global `connections` array.

3.4 Connection Management

Managing active connections is a really important aspect of our application. The `update_connection()` function is responsible for adding new connections or updating statistics for existing ones.

To ensure consistent and direction-agnostic connection tracking, the function uses a `ConnectionKey` structure. The `find_connection()` function searches for an existing connection matching the key [3]. If no match is found, than a new entry is created, provided the maximum number of connections has not been exceeded. And once a connection is identified or created, `update_connection()` updates its statistics, including bits and packet counts for both transmission and reception.

3.5 Displaying Statistics

The `display_statistics()` function provides real-time feedback to the user by rendering the connection data in a tabular format. Using the `ncurses` library [2], it clears and redraws the terminal interface with updated statistics. The function sorts connections based on the selected mode (bits or packets) using

`qsort()` [4]. It then formats and displays the top 10 most active connections, including their source and destination, protocol type, and bandwidth metrics.

To ensure accuracy, counters for each connection are reset after being displayed, allowing for the next cycle of updates to begin afresh.

3.6 Utility Functions

Several utility functions enhance the application's modularity and maintainability:

`get_local_ips(char *interface)`: Retrieves all local IP addresses for the specified interface, supporting both IPv4 and IPv6. These addresses are used to determine the directionality of traffic [4].

`is_local_ip(char *ip)`: Checks if a given IP address matches any of the local addresses [3].

`format_bandwidth(double bytes, double interval, char *output, char *unit_str)`: Converts raw byte counts into human-readable bandwidth values in bits per second, adjusting units dynamically (e.g., `k`, `M`, `G`).

3.7 Logging

For debugging and monitoring purposes, the application supports logging of packet details and connection statistics. The `-d` option implemented in the command line enables logging [1]. The log file is created in the current directory with the name `isa-top.log`.

4 Usage

4.1 Compilation

To compile the application, run the following command:

```
make
```

This will generate the executable file `isa-top`.

4.2 Running the Application

The application can be run with the following syntax:

```
./isa-top -i <interface> [-s b|p] [-t interval] [-d]
```

Options:

- `-i <interface>` — Specifies the network interface to monitor (required).
- `-s b|p` — Sorting mode:
  - `b` — Sort by transferred bytes (default).
  - `p` — Sort by transferred packets.
- `-t interval` — Update interval in seconds (optional, default is 1 second).
- `-d` — Enable logging.

4.3 Example Commands

- Monitor traffic on the `eth0` (for Linux) or on the `en0` (for macOS) interface with default settings:

```
./isa-top -i eth0
```

- Monitor the `eth0` interface and sort results by packets:

```
./isa-top -i eth0 -s p
```

5 Testing

To validate the `isa-top` application, several scenarios were created and tested on a macOS Sonoma environment. Each test scenario demonstrates the application's ability to monitor various types of network traffic, including TCP, UDP, and high-bandwidth traffic from a speed test website.

5.1 Testing Scenarios

5.1.1 TCP Traffic Test Using `iperf`

This test demonstrates the `isa-top` application's ability to monitor TCP traffic on the loopback interface using `iperf` [5]. It validates real-time capture of TCP connections, including source and destination IPs, ports, and bandwidth usage.

1. Start an `iperf` server in one terminal:

```
iperf -s
```

2. In another terminal, run the `iperf` client to generate TCP traffic:

```
iperf -c 127.0.0.1
```

3. Start the `isa-top` application to monitor the loopback interface:

```
./isa-top -i lo0 -d
```

Results: The `isa-top` application accurately displayed all active TCP connections, including bandwidth usage and connection details.

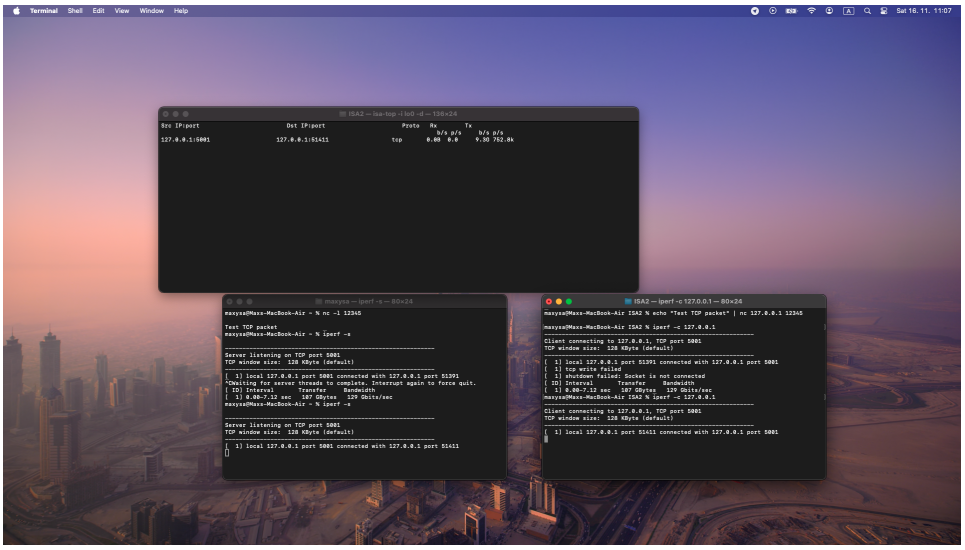

Figure 1: TCP Traffic Captured by `isa-top`

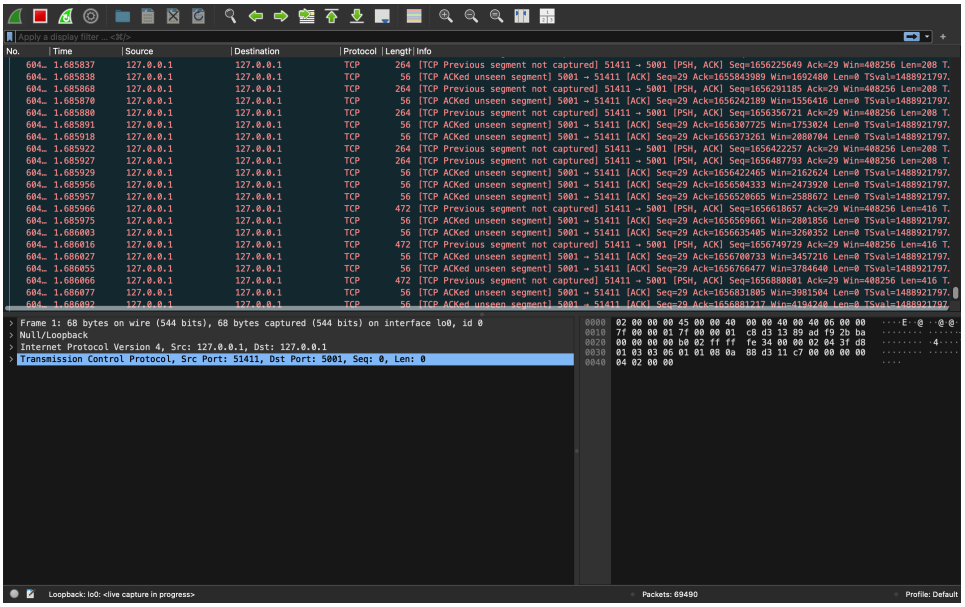

Figure 2: Wireshark Capturing TCP Traffic

5.1.2 UDP Traffic Testing Using `iperf`

This test evaluates the `isa-top` application's ability to monitor UDP traffic on the loopback interface. `iperf` was used to simulate high-bandwidth UDP traffic [5].

1. Start the `isa-top` application on the loopback interface:

```
./isa-top -i lo0 -d
```

2. Launch the `iperf` server in a second terminal:

```
iperf -s -u
```

3. In the third terminal, send UDP traffic to the server:

```
iperf -u -c 127.0.0.1 -b 10M
```

Results: The `isa-top` application successfully captured the UDP traffic, displaying source and destination IPs, ports, protocol type, and bandwidth statistics.

Figure 3: UDP Traffic Visualization in `isa-top`

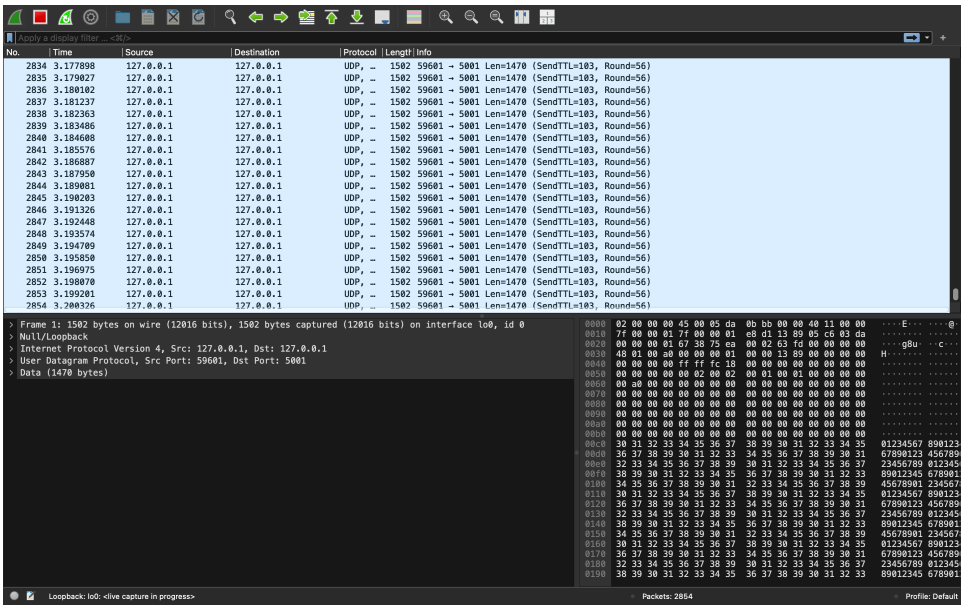

Figure 4: UDP Traffic Captured in Wireshark

5.1.3 Testing with Speed Test Website

This test demonstrates the `isa-top` application's ability to monitor high-bandwidth traffic during a speed test using services such as speedtest.net [6].

1. Start the `isa-top` application on the `en0` interface:

```
./isa-top -i en0 -d
```

2. Open a browser and navigate to a speed test website (e.g., speedtest.net).

3. Start the speed test and observe the connections displayed by `isa-top`.

Results: During the test, `isa-top` displayed high Rx (received) rates during the download phase and high Tx (transmitted) rates during the upload phase, providing real-time statistics for all active connections.

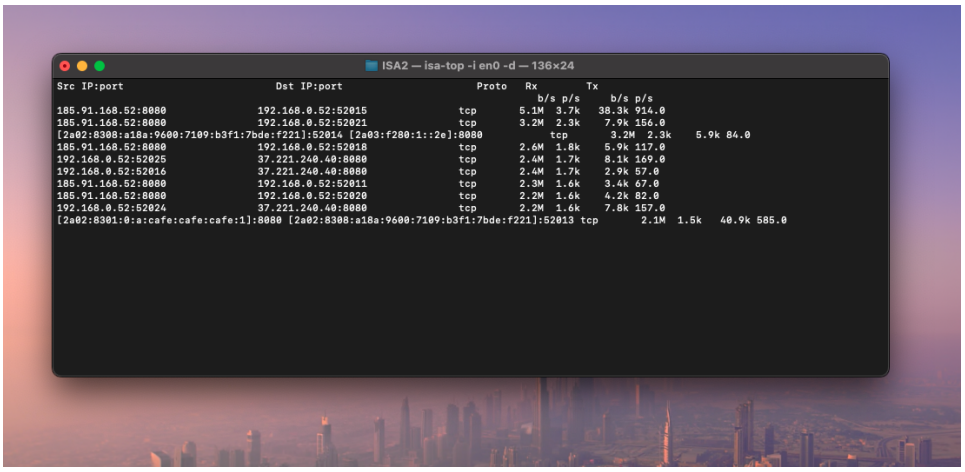

Figure 5: `isa-top` Monitoring High-Bandwidth Traffic During a Speed Test

6 References

References

[1] *Libpcap Documentation*: https://www.tcpdump.org/manpages/pcap.3pcap.html

[2] *Ncurses Documentation*: https://invisible-island.net/ncurses/man/ncurses.3x.html

[3] *Beej's Guide to Network Programming*: https://beej.us/guide/bgnet/

[4] *Linux Man Pages*: https://man7.org/linux/man-pages/

[5] *Iperf Documentation*: https://iperf.fr/

[6] *Speedtest.net*: https://www.speedtest.net