BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

ISA - Síťové aplikace a správa sítí
Aplikace pro získání statistik o síťovém provozu `isa-top`

October 25, 2024

Artur Sultanov (xsulta01)

Contents

1 Introduction

The isa-top application is a console-based tool designed to monitor and display real-time bandwidth usage between communicating IP addresses on a selected network interface.
isa-top uses libraries such as libpcap for packet capture [1, 2] and ncurses for a text-based user interface [3].

2 Overview of Related Tools and Concepts

isa-top was inspired by existing tool iftop [4, 5]. iftop is a widely used tool for monitoring bandwidth usage on Linux systems [4]. It captures live network data using libpcap [1] and presents it in a user-friendly manner using ncurses [3]. The isa-top program shares similar functionalities but introduces other logic for managing connections and providing statistical calculations.

3 Application Design

3.1 Design Goals

The main goals of isa-top include:

- **Network Packet Capture**: Uses libpcap [2] to capture packets at the network interface level, allowing analysis of all network traffic.
- **Reverse Connection Handling**: isa-top treats communication between two IP addresses as a single connection regardless of packet direction (Tx or Rx), making it easier to analyze bidirectional data exchange.
- **Text-based UI**: Uses ncurses [3] to display live statistics, providing real-time updates on the terminal without the need for a graphical interface.

3.2 Architecture Overview

At the core of isa-top is the packet capture mechanism, which relies on libpcap to access network data directly from a specified interface. This allows the program to listen to all incoming and outgoing packets in real time. The captured packets are passed to a handler that extracts necessary information, such as IP addresses, ports, and protocol details.

Data processing within isa-top involves parsing each packet to identify its characteristics and then using this data to track the connections between communicating IP addresses. Each connection is uniquely identified using a connection_key_t combination of source and destination IPs, ports, and the protocol in use. This information is stored in a data structure that keeps track of both transmitted (Tx) and received (Rx) data.

The interface is built using ncurses. Interface provides information about the source and destination addresses, protocol types, and current data transfer rates in bytes per second and packets per second.

There is a mechanisms for calculating bandwidth based on the bytes and packets observed over time. By maintaining counters for each connection and updating them with each new packet, isa-top computes average transfer rates for both directions of a connection.

4 Implementation Description

4.1 Packets processing pipeline

The packet processing pipeline in the isa-top program follows a series of steps to capture, analyze, and display network traffic data. Below is a detailed breakdown of each stage:

4.1.1 Packet Capture Initialization

Library Used: `libpcap`

Function: `pcap_open_live()` Opens a specified network interface (e.g., `eth0`) for live packet capture. The function takes parameters such as the interface name, snapshot length (amount of packet data to capture), and a timeout value.

Error Handling: If the interface cannot be opened (e.g., due to permissions or an invalid name), the program prints an error message and terminates.

Non-Blocking Mode: Sets the `pcap` session to non-blocking mode using `pcap_setnonblock()`, allowing the program to process packets without waiting for a timeout or new packets.

4.1.2 Packet Dispatch and Handling

Function: `pcap_dispatch()`

Continuously reads packets from the network and calls a callback function (`packet_handler()`) for each captured packet.

Callback Function: `packet_handler()` is invoked with each packet and extracts information from its headers.

4.1.3 Parsing Packet Headers in `packet_handler()`

Ethernet Header Parsing:
The program starts by casting the packet data to `ether_header` to determine the Ethernet type (IPv4, IPv6). It uses `ntohs()` to convert the `ether_type` field from network byte order to host byte order. Checks for `ETHERTYPE_IP` (IPv4) or `ETHERTYPE_IPV6` (IPv6) to identify the type of packet for further processing.

IPv4 Packet Parsing:
Extracts the IPv4 header using `ip`:

```
struct ip *ip4_hdr =/
(struct ip*)(packet + sizeof(struct ether_header));
```

Uses `inet_ntop()` to convert binary source and destination IP addresses to human-readable strings. Extracts the `ip_p` field to determine the protocol (e.g., TCP, UDP, ICMP).

TCP/UDP Header Parsing:

- **For TCP packets**:
  Extracts TCP headers using `tcphdr` and retrieves source and destination ports.

  ```
  struct tcphdr *tcp_hdr =/
  (struct tcphdr*)(packet + sizeof(struct ether_header) +/
  ip_header_length);
  uint16_t src_port = ntohs(tcp_hdr->source);
  uint16_t dst_port = ntohs(tcp_hdr->dest);
  ```

- **For UDP packets**:
  Uses `udphdr` for similar extraction of ports. The extracted IPs, ports, and protocol are stored in a `connection_key_t` structure.

ICMP Packet Handling:
For ICMP, the protocol is marked as `icmp`, and the port information is set to zero since ICMP does not use port numbers.

4.1.4 Connection Management and Data Update

Finding or Creating Connections: Uses `get_connection()` to find an existing connection with a matching `connection_key_t`. If a match is found, it returns the corresponding `connection_stats_t` pointer and determines if the packet is in the Tx (transmitting) or Rx (receiving) direction. If no match is found, a new connection is created and added to the list.

```
int dir = compare_keys(&current->key, key);
if (dir != 0) { /* Match found */ }
```

Updating Statistics: Uses `update_statistics()` to increment byte and packet counts:

- **Tx (Transmit) Direction**:
  Updates `tx_bytes` and `tx_packets`.
- **Rx (Receive) Direction**:
  Updates `rx_bytes` and `rx_packets`.

```
if (direction == 1) {
    conn->tx_bytes += packet_size;
    conn->tx_packets++;
} else if (direction == -1) {
    conn->rx_bytes += packet_size;
    conn->rx_packets++;
}
```

4.1.5 Rate Calculation and Display

Uses `time_t` to calculate the time interval between updates. Computes the average bytes per second (b/s) and packets per second (p/s) based on the time elapsed since the last update:

$$\text{tx\_rate} = \frac{\text{tx\_bytes}}{\text{interval}}$$

$$\text{rx\_rate} = \frac{\text{rx\_bytes}}{\text{interval}}$$

Uses `qsort()` to sort connections based on user-defined criteria (`-s b` for bytes, `-s p` for packets). Updates the UI using `ncurses` functions like `mvprintw()` to print connection data to the terminal, including IPs, ports, and calculated rates.

4.2 Display Update and User Interaction

The user interface (UI) of the `isa-top` program is built using the `ncurses` library, which provides a flexible framework for creating text-based user interfaces within terminal applications. This allows `isa-top` to present real-time data in a well-organized manner directly in the console. Below is a detailed description of the UI design and implementation using `ncurses`.

4.2.1 Data Display Management

Function `mvprintw()` moves the cursor to a specific position on the screen and prints text. Used for displaying headers like source IP, destination IP, protocol, Rx/Tx rates, and connection details.

Example of displaying a connection:

```
// Print headers with formatted width
mvprintw(0, 0, "%-30s\s%-30s\s%-7s\s%5s\s%15s",
         "Src\sIP:port", "Dst\sIP:port", "Proto", "Rx", "Tx");
```

Function `refresh()` updates the terminal display with the latest content. Called after all data is printed to ensure that the user sees the most recent statistics. Uses a loop with a time interval (default is 1 second, configurable with `-t`) to update the displayed statistics. The program captures packets, updates statistics, and then calls the display function to refresh the output at each interval.

4.2.2 Graceful Exit with `Ctrl+C`

Uses `getch()` to check for user input without blocking the execution (thanks to `nodelay()`).

Captures the `SIGINT` signal using `signal()` and a handler function (`handle_sigint()`), allowing users to terminate the program with `Ctrl+C`.

```
if (signal(SIGINT, handle_sigint) == SIG_ERR) {
    endwin();
    fprintf(stderr, "Error\ssetting\sup\ssignal\shandler.");
}
```

This ensures that the `ncurses` environment is properly closed before exiting, restoring the terminal to its original state.

Summary of program processing logic

1. **Capture**: Uses `libpcap` to capture raw packets on a network interface.
2. **Parse**: Extracts Ethernet, IP, and transport layer headers to identify source/destination IPs, ports, and protocol.
3. **Track**: Uses `connection_key_t` to find or create a connection, identifying the direction as Tx or Rx.
4. **Update**: Adjusts statistics for bytes and packets, storing cumulative totals.
5. **Calculate Rates**: Computes real-time bandwidth usage for display.
6. **Display**: Updates the `ncurses` interface to present a sorted view of the top connections.

5 User Guide

5.1 Installation Instructions

Prerequisites: Install `libpcap` and `ncurses`:

```
sudo apt install libpcap-dev ncurses-dev  # On Debian/Ubuntu
sudo dnf install libpcap-devel ncurses-devel  # On Fedora
```

Building the Application:
```
make
```

5.2 Command-Line Options

- `-i <interface>`: Specify the network interface to monitor.
- `-s b|p`: Sort by bytes (`b`) or packets (`p`).
- `-t <interval>`: Set the update interval in seconds (default: 1 second).

5.3 Sample Usage

```
./isa-top -i eth0 -s b -t 2
```

This command monitors `eth0`, sorts connections by bytes, and updates every 2 seconds.

6 Testing and Results

6.1 Testing Environment

Test Tools:

- Simulated traffic using `iperf3`.
- Simulated traffic using `ping -4` (my PC does not have IPv6).
- Using `iftop` for reference output.

6.2 Test Results

Scenario 1: Monitored traffic on localhost using `iperf3`.
Set up:

- Setup the `iftop`: `sudo iftop -n -N -p -B -i lo`
- Setup the `isa-top`: `sudo ./isa-top -i lo -s b -t 1`
- Setup the `iperf3 server`: `iperf3 -s`
- Setup the `iperf3 client`: `iperf3 -c 127.0.0.1 -u -b 10M -t 0`

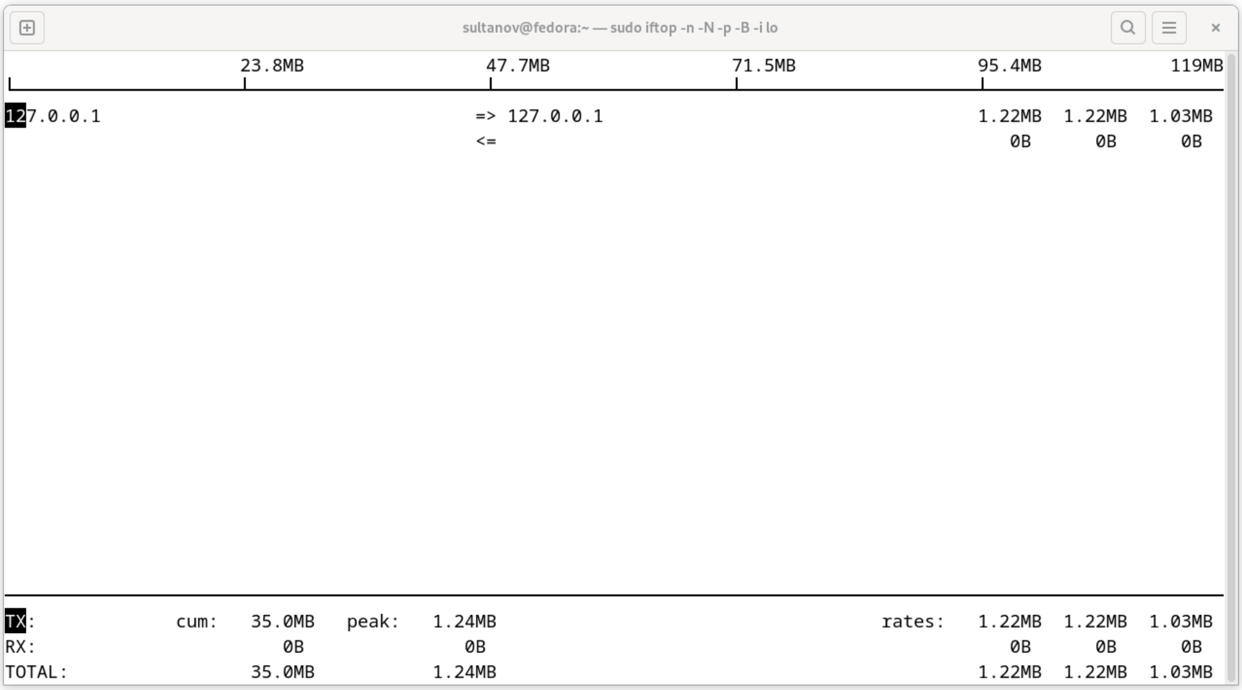

Figure 1: Test 1: Iftop results.

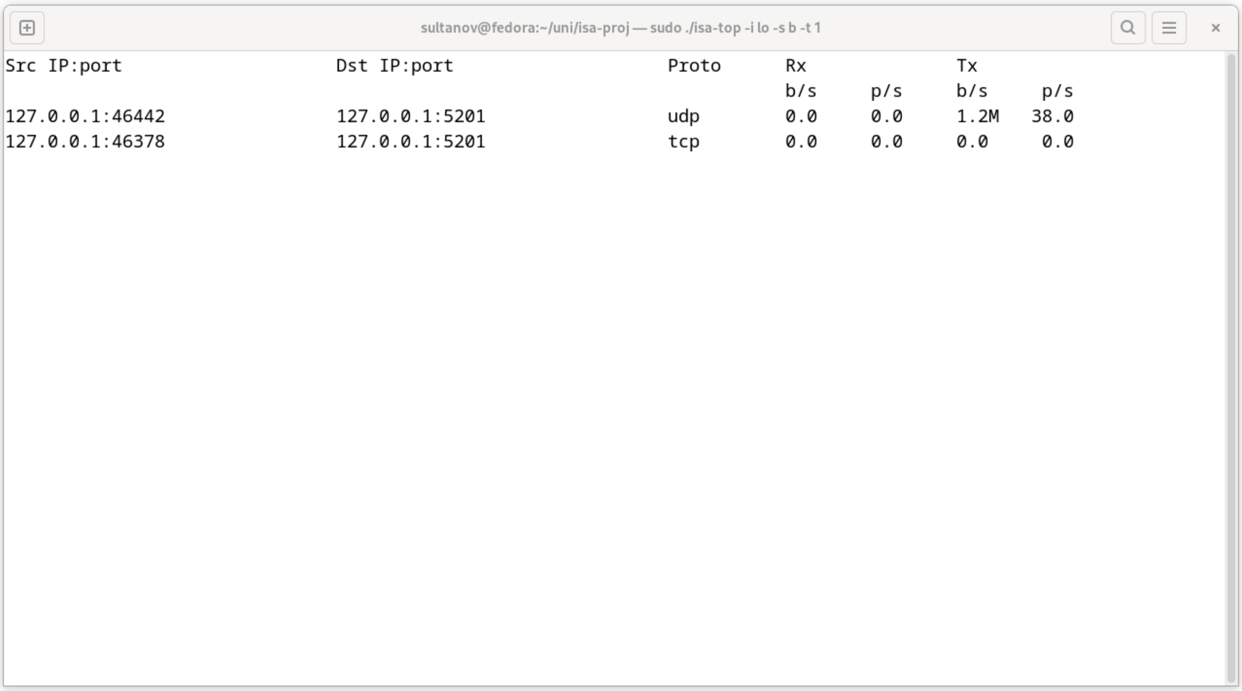

Figure 2: Test 1: Isa-top results.

Result: Accurate display of Rx and Tx rates.

Scenario 2: Monitored traffic on wlan interface.

Set up:

- Setup the `iftop`: `sudo ./isa-top -i wlp10s0f3u4 -s b -t 1`
- Setup the `isa-top`: `sudo iftop -n -N -p -B -i wlp10s0f3u4`
- Setup the `ping IPv4`: `ping -4 google.com`

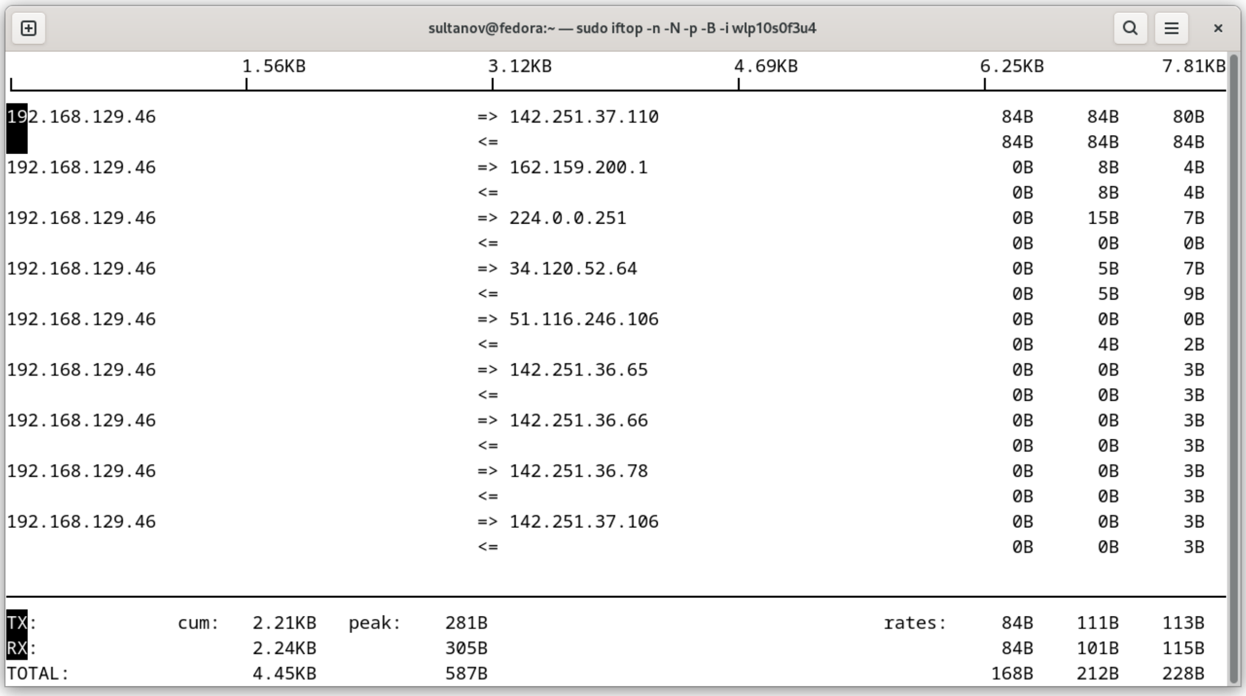

Figure 3: Test 1: Iftop results.

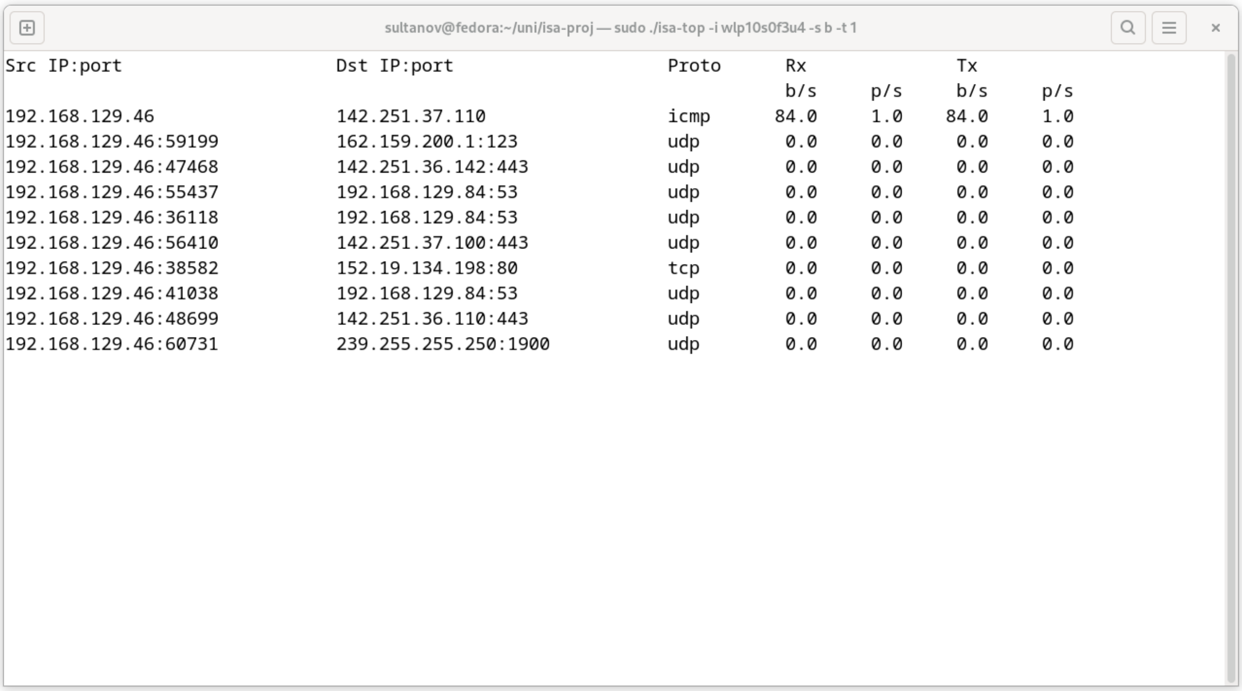

Figure 4: Test 1: Isa-top results.

Result: Accurate display of Rx and Tx rates.

7 Other Implementation Notes

The `iftop` tool counts data transmitted at the IP layer, focusing on IP packets that include protocols like TCP, UDP, and ICMP, while excluding the size of the Ethernet frame header from its bandwidth calculations. This approach effects on the payload sizes.

So then, `isa-top` tool, inspired by `iftop`, is designed to explicitly subtract the Ethernet header size from packet payload when calculating the payload length. This is achieved through the following line in the packet processing logic:

```
size_t payload_len = header->len - sizeof(struct ether_header);
```

This subtraction ensures that the payload length accurately reflects the size of the data transmitted at the IP layer.

Conclusion

The `isa-top` program offers a robust and efficient solution for real-time network bandwidth monitoring on Linux systems. By leveraging the power of `libpcap` for packet capture and `ncurses` for a terminal-based user interface, `isa-top` provides a detailed view of data exchanges between IP addresses.

8 References

References

[1] libpcap documentation. Accessed: 2024-10-25. [Online]. Available: https://libpcap.readthedocs.io/

[2] tcpdump page. Accessed: 2024-10-25. [Online]. Available: https://www.tcpdump.org/

[3] T. E. Dickey. ncurses library documentation. Accessed: 2024-10-25. [Online]. Available: https://invisible-island.net/ncurses/

[4] A. Kili. iftop - a real time linux network bandwidth monitoring tool. Accessed: 2024-10-25. [Online]. Available: https://www.tecmint.com/iftop-linux-network-bandwidth-monitoring-tool/

[5] RedHat. Linux interface analytics on-demand with iftop. Accessed: 2024-10-25. [Online]. Available: https://www.redhat.com/en/blog/linux-interface-iftop