FAKULTA INFORMAČNÝCH TECHNOLÓGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Aplikace pro získání statistik o síťovém provozu

Dokumentácia k projektu

ISA 2024/25

November 16, 2024

Martin Valach (xvalac12)

Contents

1 Introduction

This is detailed technical documentation for a real-time network packet monitoring application. The goal of this application is to capture and display network traffic statistics for a entered network interface, categorizing them by source and destination IP addresses and ports, protocols and transmitted/received packets and their size.

2 Application Design

The program is implemented in C++ and uses multithreading to manage packet capturing and display functions simultaneously. The user provides a network interface name and optional parameters for sorting and refresh interval. The program sets up the capturing handler, applying filters *(tcp or udp or icmp)* to capture defined packets. Then separate thread runs `pcap_loop()` function to capture packets and calls `callback()`‘ funtion for each packet. The `callback()` function extracts key information from each packet and updates the ‘map‘ with communications. The main thread uses ncurses library to periodically print formatted communication statistics. The statistics are sorted based on either data size or packet count, as specified by the user. At the end, program cleans up resources from packet capturing, join sniffing thread back to main thread and ends ncurses window when the program is terminated (ctrl+c).

3 Implemented features

Application has these features implemented:

- working command line arguments
- working communication statistics print for maximum of 10 communications
- receiving and transmitting communication is print as bidirectional communication
- working sorting by size/packets
- received and transmitted packets are add together for sorting
- packets and packet size is print with correct unit
- default refresh rate is 1 second
- additionally, user can entered custom refresh rate
- IPv4 and IPv6 packets are supported
- TCP, UDP and ICMP protocols are supported

4 Program Description

4.1 Libraries Used

The program uses these libraries:

- **Standard C Libraries**: `stdio.h`, `unistd.h`
- **Standard C++ Libraries**: `iostream`, `string`, `map`, `vector`, `iomanip`, `sstream`, `algorithm`
- **Networking Libraries**: `pcap.h`, `netinet/ip.h`, `netinet/tcp.h`, `netinet/udp.h`, `netinet/ip_icmp.`
- **Terminal Handling**: `ncurses.h`, `thread`

4.2 Program Structure

The code is organized as follows:

- **Packet Struct**: A template for storing packet information used as a key in the map structure.
- **Map Data Structure**: A map to store packet statistics categorized by communication pairs.
- **Callback Function**: Processes each captured packet to extract relevant information and put them to map.
- **Helper Functions**: Convert protocol numbers from packet to protocol names, format bandwidth values to correct unit, and print statistics.
- **Main Function**: Handles user input, sets up the packet capture environment, initializes `ncurses` library and manages multithreading.

5 Implementation Details

5.1 Packet Struct

The `Packet` struct is used as a key in the `communication` map to track data and packet counts. In code looks like this:

```
struct Packet {
    string first_addr;
    string second_addr;
    int first_port;
    int second_port;
    int def_protocol;

    Packet(string& src_addr, string& dst_addr, int src_port, int dst_port, int protocol);
    bool operator<(const Packet& aux) const;
};
```

5.2 Callback Function

The `callback()` function is called whenever a packet is captured. It extracts protocol, source and destination information and updates the map storing communication data. This is prototype of function:

```
void callback(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);
```

5.3 Display with ncurses library

The `print_stats()` function formats and displays packet statistics in the terminal. This function sorts the statistics by data size or packet count based on user preference and prints them using `ncurses` library. This is the prototype:

```
void print_stats(bool sort_by_size);
```

6 Usage

6.1 Requirements

To use this application, you will need the following:

- g++ compiler
- libpcap library

6.2 Compiling the Program

For compilatiom were used the following commands:

```
g++ -Wall -c isa-top.cpp -o isa-top.o
g++ -Wall -o isa-top isa-top.o -lpcap -lncurses
```

6.3 Running the Program

The program has the following command-line options:

- `-i <interface>`: Specifies the network interface to sniff. (required)
- `-s <b|p>`: Sorts output by data size (b) or packet count (p).
- `-t <time>`: Sets the refresh interval in seconds. (default 1)

Example usage:

```
./isa-top -i enp4s0 -s b -t 1
```

7 Testing

The application was tested on a Linux environment with various network interfaces. Packets were sniffed and data were compared against known traffic to verify accuracy using wireshark program. In testing program successfully captured and displayed real-time traffic statistics for TCP, UDP, and ICMP protocols. Sorting by both data size and packet count was confirmed to work as expected.

References