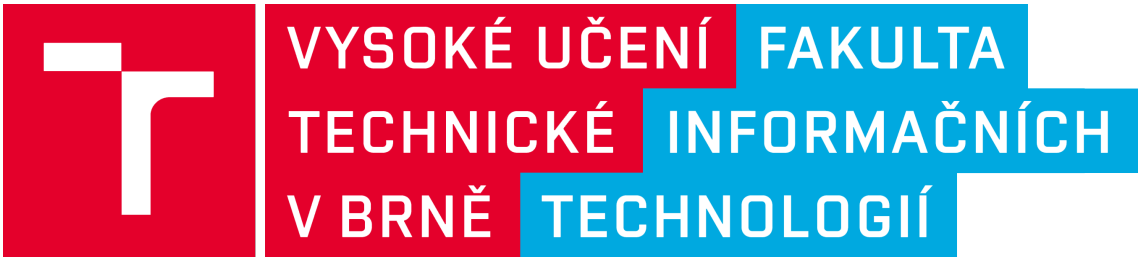

ISA - technická dokumentácia k projektu

Monitorování DHCP komunikace

2023/2024

Adrian Horvath (xhorva14)

November 20, 2023

Contents

1 Introduction

The dhcp-stats program is a tool designed for capturing and analyzing DHCP traffic on a network. The program's primary objective is to assess the percentage of allocated IP addresses relative to the total available addresses within a specified prefix.

1.1 DHCP Overview

In a typical network environment, DHCP operates as a client-server protocol. When a device, known as a DHCP client, connects to the network, it sends out a DHCPDISCOVER message to discover available DHCP servers. DHCP servers respond with a DHCPOFFER, providing the client with an IP address and other configuration parameters.

Upon receiving one or more offers, the client selects a server and sends a DHCPREQUEST message to request the offered configuration. The chosen server responds with a DHCPACK message, confirming the allocation of the IP address and finalizing the configuration.

1.2 dhcp-stats Objective

The primary objective of the dhcp-stats program is to assess the percentage of allocated IP addresses relative to the total available addresses within a specified prefix. This information is crucial for network administrators to monitor and manage IP address utilization efficiently.

The program achieves this objective by capturing DHCP packets and extracting relevant information such as IP addresses, message types (e.g., DHCPACK, DHCPRELEASE), and other options specified in the DHCP protocol.

1.3 DHCP Packet Analysis

The dhcp-stats program analyzes DHCP packets to determine the allocation status of IP addresses. It identifies key elements in DHCP packets, including the message type, allocated IP addresses, and other options. By examining these elements, the program categorizes packets into ACK (Acknowledgment) and RELEASE types, providing insights into IP address allocation and release activities on the network.

1.4 Usage

To use the dhcp-stats program, specify either a network interface using the `-i` option or a pcap file using the `-r` option. Additionally, include at least one IP address prefix for analysis. The program then captures and analyzes DHCP packets, presenting information about IP address allocation within the specified prefixes.
The program is working without timer or steps, if you need stepping (recommended for reading from file), you need to uncomment line 260 //getch() in dhcp-stats.cpp file. Program does not count broadcast ip and network ip into maximum number of addresses to be allocated and also ignores ACK packet with such ip (for example, in prefix 10.10.10.0/24 will be max available addresses for allocation 254, and will ignore broadcast (10.10.10.255) and network (10.10.10.0), but /23 this two ip addresses will be counted) In case program recieve prefix on input /31 or /32 maximum number of addresses is set to 0, and utilization is also 0%.

1.5 References

For further information on the DHCP protocol, refer to RFC 2131[1].

2 Usage

The program is executed from the command line with the following syntax:

```
./dhcp-stats [-r <filename>] [-i <interface-name>] <ip-prefix> [ <ip-prefix> [ ... ]
```

3 Command Line Options

- `-r, --filename <filename>`: Specify the name of the file containing captured DHCP packets.
- `-i, --interface <interface-name>`: Specify the network interface for live packet capture.
- `<ip-prefix> [ <ip-prefix> [ ... ] ]`: Specify one or more IP address prefixes to analyze.

4 Application Design

4.1 Overview

The program is structured to efficiently capture and analyze DHCP packets, providing valuable statistics related to IP address allocations and utilization within specified network prefixes. The source code is organized into several files, including dhcp-stats.cpp, dhcp-stats.h, and Makefile.

4.2 Program Components

4.2.1 NetworkPrefix Structure

The `NetworkPrefix` structure represents network prefixes and includes key information.

```
struct NetworkPrefix {
    char prefix[16];            ///< String representing the network prefix (e.g., "192.168.1.0/2
    uint32_t ip;                ///< IP address of the network prefix.
    int prefix_length;          ///< Length of the prefix (e.g., 24 for the example above).
    uint32_t allocated;         ///< Count of allocated IP addresses in the prefix.
    uint32_t maxAddress;        ///< Maximum address in the prefix.
    uint32_t mask;              ///< Subnet mask of the prefix.
    size_t MAX_PREFIXES;        ///< Maximum number of prefixes.
};
```

[1] `https://datatracker.ietf.org/doc/html/rfc2131`

4.2.2 dhcp_packet Structure

The `dhcp_packet` structure represents DHCP packets, capturing essential fields and options.

```
struct dhcp_packet {
    uint8_t op;             ///< Message opcode: 1 for request, 2 for reply.
    uint8_t htype;          ///< Hardware address type.
    uint8_t hlen;           ///< Hardware address length.
    uint8_t hops;           ///< Number of hops.
    uint32_t xid;           ///< Transaction ID.
    uint16_t secs;          ///< Seconds elapsed.
    uint16_t flags;         ///< Flags.
    uint32_t ciaddr;        ///< Client IP address.
    uint32_t yiaddr;        ///< Your client IP address.
    uint32_t siaddr;        ///< Server IP address.
    uint32_t giaddr;        ///< Relay agent IP address.
    uint8_t chaddr[16];     ///< Client hardware address.
    uint8_t sname[64];      ///< Server host name.
    uint8_t file[128];      ///< Boot filename.
    uint32_t magic_cookie;  ///< DHCP magic cookie.

    // Options follow here
    uint8_t options[576];   ///< Array to store all options.
};
```

4.2.3 IpHash Structure

The `IpHash` structure defines a hash function for IP addresses, crucial for tracking allocated IP addresses.

```
struct IpHash {
    std::size_t operator()(const uint32_t& ip) const {
        // Use a simple hash function for demonstration
        return std::hash<uint32_t>{}(ip);
    }
};
```

4.2.4 Command Line Arguments Structure

The `CommandLineArgs` structure holds command line arguments passed to the program.

```
struct CommandLineArgs {
    char *interface;        ///< Network interface name.
    char *filename;         ///< PCAP file name.
    char **ipPrefixes;      ///< Array of IP prefixes.
    int ipPrefixCount;      ///< Number of IP prefixes.
};
```

4.3 Functions

Explore various functions such as `initializePrefixes`, `openLiveCapture`, `openFile`, `packet_handler`, and more, each serving specific roles in the program.

5 Implementation Description

The DHCP Stats program is designed to analyze DHCP packets within a network, providing insights into the allocation of IP addresses within a specified address prefix. The implementation is structured around key functionalities, design decisions, and interesting aspects that contribute to the program's functionality.

5.1 Command Line Argument Processing

The program starts by setting up a signal handler for SIGINT and then proceeds to parse command line arguments using the `getopt` function. The `argumentCheck` function is responsible for validating the command line arguments, ensuring that either the `-r` or `-i` option is used but not both. Additionally, it checks for the presence of at least one IP prefix. The `isValidPrefix` function is then invoked to validate the format of each provided IP prefix.

5.2 Network Prefix Initialization

Upon successful command line argument processing, the program proceeds to initialize network prefixes using the `initializePrefixes` function. This function sets up the `NetworkPrefix` structure for each provided IP prefix, extracting relevant information such as the network address, maximum number of allocated IPs, mask, and more.

5.3 Packet Capture and Processing

The main function then determines whether to capture packets from a live network interface or read them from a file. Depending on the source, either `pcap_open_live` or `pcap_open_offline` is used to obtain a handler. The `setPacketFilter` function is responsible for setting up a packet filter to capture only DHCP packets on ports 67 and 68.

The program uses the `pcap_loop` function to capture packets in a loop. For each captured packet, the `packet_handler` function is invoked. This function extracts relevant DHCP data from the packet and checks for the appropriate ports (according RFC 2131 there can't be same port for source and dest. port in one packet). The `printOptions` function is then called to analyze the DHCP options and determine if the packet is an acknowledgment (ACK) or release (RELEASE).

5.4 IP Address Allocation and Prefix Updates

The `modifyDataForMatchingPrefix` function manages IP address allocation and updates network prefixes. It checks if the allocated IP is already present in the hash table. If it is an ACK packet, the IP is added to the hash table. If it is a RELEASE packet, the IP is removed from the hash table, and the corresponding network prefix structures are updated.

5.5 User Interface

The program utilizes the NCurses library for a user-friendly interface. The `printPrefixes` function updates and displays information about the network prefixes, allocated IP addresses, and usage statistics in real-time.

6 Basic Program Information

6.1 Prerequisites

Ensure the presence of a C++ compiler, pcap library, and ncurses library before attempting to build and run the program.

7 Building and Running

7.1 Building the Program

To build the program, use the provided Makefile:

make

7.2 Running the Program

Execute the program with appropriate command line options:

```
./dhcp-stats -r DHCPfile.cap 192.168.0.0/24
```

8 Usage Instructions

Detailed instructions on using the DHCP Stats program, including live capture and file analysis.

Live Capture:
Analyzing DHCP packets on the eth0 interface within the 192.168.1.0/24 address prefix:

```
./dhcp-stats -i eth0 192.168.1.0/24
```

File Analysis:
Analyzing DHCP packets from the DHCPcopy.cap file within the 10.0.0.0/16 address prefix:

```
./dhcp-stats -r DHCPcopy.cap 10.0.0.0/16
```

9 References

References

[1] https://datatracker.ietf.org/doc/html/rfc2131

[2] https://liw.fi/manpages/

[3] https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/