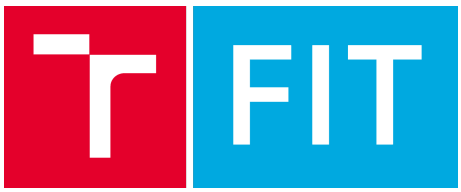
Monitoring of DHCP Communication

ISA - project

Author: Tomáš Husár (xhusar11)
Date: November 2023



Faculty of Information Technology
Email: xhusar11@stud.fit.vutbr.cz

Contents

1 Introduction

Dynamic Host Configuration Protocol (DHCP) is a client/server protocol that automatically provides an Internet Protocol (IP) host with its IP address and other related configuration such as the subnet mask and defalt gateway. RFCs 2131 and 2132 define DHCP as an Internet Engineering Task Force (IETF) standard based on Bootstrap Protocol (BOOTP)[2].

DHCP is built on a client-server model, where designated DHCP server hosts allocate network addresses and deliver configuration parameters to dynamically configured hosts[4]. Standard DHCP communication can be divided into 4 parts:

- **Discovery**: The client initiates the communication by broadcasting a DHCPDISCOVER message to locate available DHCP servers on the network.
- **Offer**: Upon receiving the DHCPDISCOVER message, DHCP server responds with a DHCPOFFER message. This message includes an available IP address (and other configuration parameters).
- **Request**: The client selects one of the offers and broadcasts a DHCPREQUEST message to request the offered configuration from a specific DHCP server. This message specifies which server the client has chosen and confirms the parameters it's requesting.
- **Acknowledgment**: The DHCP server that received the DHCPREQUEST responds with a DHCPACK message, providing the client with the network configuration details (if the offered configuration is no longer available, the server responds with a DHCPNAK message, and the process restarts).

The way these steps work and how the information is formatted is described in the RFC2131[4].

2 Implementation

The application is developed using the C programming language, using standard C libraries for essential functionality. For packet handling, it utilizes the libpcap library[5] and the ncurses library[3] for the user interface. Message logging is facilitated through the use of syslog.

Some implementation details of the packet sniffer have been inspired by Develop a Packet Sniffer with Libpcap article[6].

2.1 Overview

Structures:

- **Arguments** - cnotains interface, filename, array of ip prefixes and number of ip prefixes
- **IP_prefix** - contains octets, prefix, number of allocations, network ip, network mask, network start and network end
- **DHCP_header** - standard DHCP header structure from RFC2131[4]

Functions:

- **arg_parse()** - parses arguments from command line
- **init_IP_prefix_array()** - initializes the array of IP_prefixes
- **check_ip_range()** - checks the ip address validity
- **check_prefix_range()** - checks the ip prefix validity
- **packet_handler()** - callback function invoked by libpcap whenever a packet is captured
- **increment_allocated_addresses()** - increments the number of allocations for the ip prefix
- **check_prefix_allocation_threshold()** - checks if the number of allocations for the prefix exceeded the thresholds (50, 75, 90, 95%)
- **print_stats()** - prints the allocation statistics into the ncurses window
- **stop_program()** - handles proper memory management and program termination

2.2 Tools

- **standard C libraries** - essential functionalities
- **libpcap** - packet handling
- **ncurses** - user interface
- **syslog** - message logging

2.3 Implementation Details

The function **packet_handler()** performs the following steps:

- It parses the Ethernet header and verifies whether the frame is an IP packet.
- If confirmed as an IP packet, it sets the pointer to the IP header and proceeds.
- Next, it verifies whether the IP protocol is UDP and sets the pointer accordingly.
- Upon confirming the UDP protocol, the function examines if the UDP packet corresponds to DHCP (UDP ports 67 or 67) and sets the pointer to the DHCP header if applicable.
- Moving through the packet, the function iterates until it reaches the options section. Here, a straightforward finite state machine is employed.
- The FSM seeks out the DHCP MSG type and DHCP ACK within the options. If identified, the ACK_flag is set to 1.
- Moreover, when a DHCP ACK is detected, the function increments the allocation count for all subnets associated with the IP address found in the packet[1].

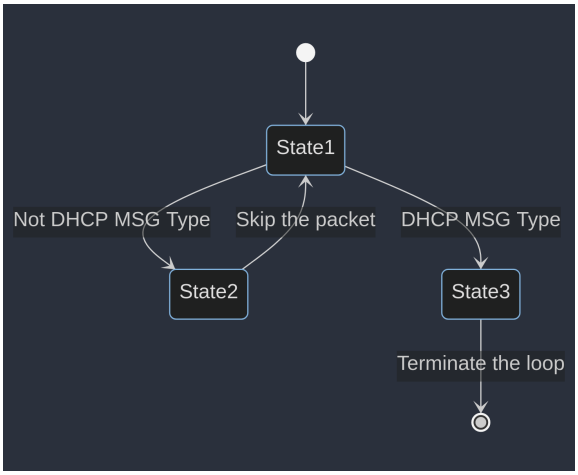


Figure 1: Finite state machine

3 Basic information about the program

The program obtains statistics of the network prefixes based on the number of allocated IP addresses from interface or pcap file. When the prefix allocation exceeds 50, 75, 90 and 95% the program informs the administrator by logging through the syslog server. After reading from a file, the program waits for a character input (or CTRL+C) to terminate itself. When obtaining packets from an interface, terminate the program with CTRL+C.

4 Usage

For proper function of the program, follow the neccesary steps during the setup and launch.

4.1 Dependencies (Ubuntu, Debian)

Program uses standard C, pcap, networking, packet handling, ncurses user interface and message logging libraries. Make sure to install necessary libraries with:

```
sudo apt-get install build-essential

sudo apt-get install libpcap-dev

sudo apt-get install libncurses5-dev libncursesw5-dev
```

4.2 Setup

Make the project before running the program with:

```
make
```

Clean the program with:

```
make clean
```

4.3 Usage Examples

Run the program with following command:

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

Where:

- -r <filename> - statistics will be created from pcap file
- -i <interface> - statistics will be created from traffic on interface
- <ip-prefix> - network range, for which the statistics will be generated

Generating statistics from file example:

```
./dhcp-stats -r dhcp.pcapng 192.168.1.0/26 172.16.32.0/24 192.168.0.0/22
```

Generating statistics from interface example:

```
sudo ./dhcp-stats -i eth0 192.168.1.0/26 172.16.32.0/24 192.168.0.0/22
```

4.4 Output Example

Example output from previous cases:

```
IP-Prefix Max-hosts Allocated addresses Utilization
192.168.1.0/26 62 50 80.65%
172.16.32.0/24 254 0 0.00%
192.168.0.0/22 1022 50 4.89%
```

References

[1] Sockets – Standard/safe way to check if IP address is in range/subnet. https://itecnote.com/tecnote/sockets-standard-safe-way-to-check-if-ip-address-is-in-range-subnet/.

[2] JasonGerend, dknappettmsft, khdownie, eross-msft, iangpgh, jamesmci. Dynamic Host Configuration Protocol (DHCP). https://learn.microsoft.com/en-us/windows-server/networking/technologies/dhcp/dhcp-top, 29/7/2021.

[3] Pradeep Padala. NCURSES Programming HOWTO. https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/, 20/6/2002.

[4] Ralph Dorms. Dynamic Host Configuration Protocol. https://datatracker.ietf.org/doc/html/rfc2131, March/1997.

[5] Tim Carstens. Programming with pcap. https://www.tcpdump.org/pcap.html, 2002.

[6] Vic Hargrave. Develop a Packet Sniffer with Libpcap. https://vichargrave.github.io/programming/develop-a-packet-sniffer-with-libpcap/libpcap-installation, 9/12/2012.