ISA PROJECT 2023/2024
DHCP MONITORING TOOL FOR PREFIX UTILIZATION

Jiří Štípek
xstipe02

November 20, 2023

Contents

1 Introduction to the issue

In today's dynamic and interconnected digital landscape, the expansion of network infrastructures and the proliferation of connected devices underscore the importance of efficient network management. The cornerstone of many networks is the Dynamic Host Configuration Protocol (DHCP), a fundamental component responsible for dynamically allocating IP addresses to devices.

As networks continue to grow in complexity and scale, administrators face the challenge of ensuring optimal utilization of address space within defined network prefixes. The concept of network prefixes, often associated with IPv4 and IPv6 addressing, refers to a contiguous block of IP addresses that share a common leading sequence of bits. Monitoring the utilization of these prefixes is critical for various reasons, including efficient resource allocation, troubleshooting, and capacity planning.

The primary objective of this project is to address the evolving needs of network administrators by developing a tool for gathering comprehensive statistics on the utilization of network prefixes. By focusing on DHCP environments, where IP address allocation is dynamic and often automated, the tool aims to provide administrators with actionable insights into the distribution and consumption of IP addresses within specific prefixes.

1.1 DHCP packet

The DHCP (Dynamic Host Configuration Protocol) packet is a fundamental component of network communication, particularly in IPv4 networks. There are 4 different type of comunication between client and server:

- DHCPDISCOVER: The client, upon joining the network or needing a new IP address, broadcasts a DHCPDISCOVER message to discover available DHCP servers. The message includes essential information such as the client's hardware (MAC) address and other configuration details.
- DHCPOFFER: DHCP servers on the network respond to the DHCPDISCOVER with a DHCPOFFER. The DHCPOFFER includes a proposed IP address (yiaddr), lease duration, and other configuration parameters. Servers may compete to offer the best configuration to the client.
- DHCPREQUEST: The client selects one of the offered configurations and broadcasts a DHCPREQUEST message. This message confirms the chosen DHCP server and requests the offered configuration.
- DHCPACK: The DHCP server, upon receiving the DHCPREQUEST, responds with a DHCPACK. The DHCPACK includes the final confirmation of the lease and any additional configuration information. The client is now configured with the provided IP address and related parameters.

2 Architecture

Each of these .cpp files have their own .h files included.

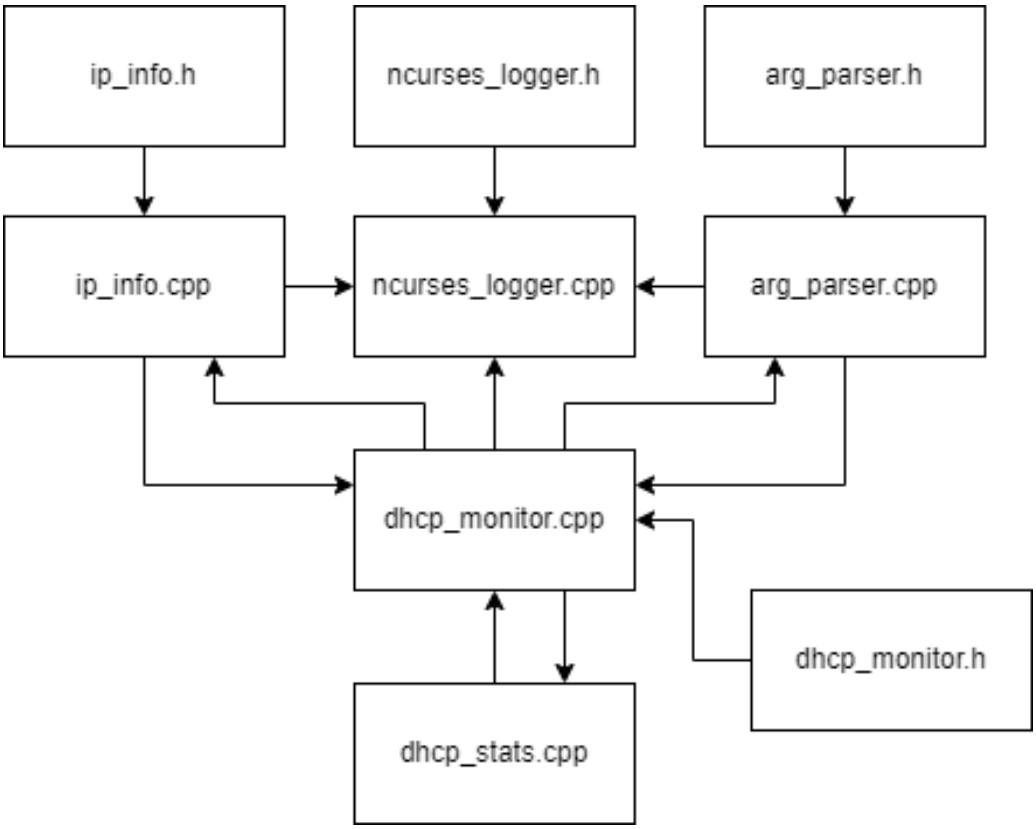

Figure 1: Architecture of the project

- DHCP stats - The main function initializes the DHCP monitor and starts the monitoring process.
- DHCP monitor - The DHCP monitor orchestrates the monitoring process by calling other necessary functions.
- Argument parser - The argument parser is responsible for parsing command-line arguments when the DHCP monitoring application is executed. It provides a mechanism for the user to configure certain aspects of the monitoring process.
- IP information - The IP info module is responsible for managing and providing information related to IP addresses obtained from DHCP messages during the monitoring process.
- Logger and ncurses - This module is responsible for ncurses library functions, syslog library functions and for exit function.

3 Implementation

In this chapter, the implementation of the project is described. The code starts in the **main()** function, where the **DHCP_monitor()** function is called from the dhcp_monitor.cpp file.

3.1 Structures

There are 3 different structures in this project. Each of them is stored in different .h files.

```
struct arguments
{
    std::string filename;
    std::string interface;
    std::vector<std::string> IP_prefixes;
};
```

First of them is arguments structure, which stores arguments from command line. This structure is stored in arg_parser.h.

```
struct dhcp_packet
{
    u_int8_t op;
    u_int8_t htype;
    u_int8_t hlen;
    u_int8_t hops;
    u_int32_t xid;
    u_int16_t secs;
    u_int16_t flags;
    struct in_addr ciaddr;
    struct in_addr yiaddr;
    struct in_addr siaddr;
    struct in_addr giaddr;
    unsigned char chaddr[MAX_DHCP_CHADDR_LENGTH];
    char sname[MAX_DHCP_SNAME_LENGTH];
    char file[MAX_DHCP_FILE_LENGTH];
};
```

The second of them is dhcp_packet structure, which holds the various fields and information associated with a DHCP (Dynamic Host Configuration Protocol) packet.

```
struct IPInfo
{
    std::string ip_full_name;
    std::string ip_name;
    int prefix;
    unsigned int max_hosts;
    unsigned int allocated_addresses;
    double utilization;
};
```

Last of them is IPInfo, which is used for storing important information related to statistics.

3.2 Global variables

In this project, there are used 4 different global variables.

```
std::vector<IPInfo> IP_infos;
```

Purpose: Storing IP information

Explanation: This vector is used to store instances of the IPInfo structure, where each instance represents information about a specific IP prefix. It allows the program to keep track of various IP prefixes, their characteristics, and the corresponding statistics.

```
std::set<std::string> sent_IPs;
```

Purpose: Tracking sent IP addresses

Explanation: The set is used to keep track of IP addresses that the DHCP server has already sent. By using a set, the program can efficiently check whether a particular IP address has been sent before. This helps prevent redundant processing of the same IP address and ensures that each IP address is considered only once, contributing to the accuracy of the statistics.

```
pcap_t *handle;
```

Purpose: Packet capture handling

Explanation: The handle is used for packet capture operations. It is part of the PCAP library, which is commonly used for capturing and analyzing network packets. The handle allows the program to interact with the network interface or capture file, enabling the monitoring of DHCP traffic.

```
std::set<std::string> exceeded_prefixes;
```

Purpose: Tracking prefixes with high utilization

Explanation: This set is used to keep track of network prefixes that have exceeded 50 % of allocated addresses. By employing a set, the program ensures that each prefix is unique and avoids redundant logging or displaying of the same exceeded prefix. The set facilitates efficient lookups, preventing the program from processing the same prefix multiple times and contributing to the accuracy of utilization statistics.

3.3 Main loop

In the program there is an infinite while loop, that uses **pcap_loop** function, which is called **packet_caller** function. In this function it extracts headers from the packet data, including Ethernet, IP, UDP, and DHCP headers. Checks if the packet is an IPv4 UDP packet with a source port of 67 or a destination port of 68 (typical for DHCP). Iterates through the DHCP options to find the DHCPACK option (code 53) with a length greater than or equal to 1 and the DHCPACK value.

Extracts the leased IP address (yiaddr) from the DHCP packet. Checks if the leased IP address is not "0.0.0.0" (indicating DHCPINFORM).

Calls **check_IP_address** to ensure the IP address has not been processed before. If not, it calls **calculate_overlapping_prefix_utilization** to update statistics based on the leased IP address.

Calls **display_statistics** to display the updated statistics. The function breaks out of the loop if it encounters a DHCPINFORM packet to skip further processing for that packet. Finally, it calls **display_statistics** again to display empty statistics if no valid DHCPACK is found in the packet.

To close the program, the user must press **CTRL+C**.

4 User manual

In the project, there is a **print_help** function that will be called in case of an error.

print_help function:

```
./dhcp-stats [-r <filename>] [-i <interface-name>] <ip-prefix> [ <ip-prefix> [ ... ] ]
            -r <filename> - statistics from pcap file
            -i <interface> - interface
            <ip-prefix> - network scope for which statistics will be generated
```

Example with filename:

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

Output:

```
IP-Prefix Max-hosts Allocated addresses Utilization
192.168.1.0/24 254 0 0.00%
192.168.0.0/22 1022 1 0.10%
```

To close the program, the user must press CTRL+C. Both filename and interface work with ncurses library.

5 Citation

Matěj Grégr. *Monitorování DHCP komunikace*. [cit. 2023-11-14]. 2023. URL: https://www.vut.cz/studis/student.phtml?script_name=zadani_detail&apid=268266&zid=54265

C++ (Cpp) pcap_open_live Examples. [cit. 2023-11-14]. 2023. URL: https://cpp.hotexamples.com/examples/-/-/pcap_open_live/cpp-pcap_open_live-function-examples.html

NCURSES Programming HOWTO. [cit. 2023-11-14]. 2005. URL: https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/

Dynamic Host Configuration Protocol. [cit. 2023-11-14]. 1997. URL: https://datatracker.ietf.org/doc/html/rfc2131

CHECK_DHCP.C. [cit. 2023-11-14]. 2005. URL: https://cs.uwaterloo.ca/twiki/pub/CF/DhcpDebug/dhcp.c