BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

ISA - documentation

Matúš Tábi
xtabim01

November 20, 2023

Contents

1 Introduction **2**
1.1 Discovery 2
1.2 Offer 2
1.3 Request 2
1.4 Acknowledgment 2

2 Application design **3**
2.1 Brief description 3
2.2 Technologies used 3
2.3 Usage 3

3 Implementation **4**
3.1 Command line arguments 4
3.2 DHCP Packet parsing 4
3.3 Options field 4
3.4 Options overload 5
3.5 Handling DHCPACK 5

4 Resources **7**

Introduction

Welcome to the documentation for school project, the Monitoring of DHCP communication. The key function of this project is to display IP prefix utilization statistics from the point of view of the number of allocated IP addresses. DHCP is a network protocol used for automatically assigning IP addresses to devices connect to the network. DHCP is implemented with two UDP port numbers, where server listens on port 67 and client is listening on port 68. DHCP operation consist of four phases: discovery, offer, request and acknowledgment.

Discovery

The DHCP client broadcasts a DHCPDISCOVER message on the network subnet using the destination address 255.255.255.255 or the specific subnet broadcast address. A DHCP client may also request an IP address in the DHCPDISCOVER, which the server may take into account when selecting an address to offer.

Offer

When a DHCP server receives a DHCPDISCOVER message from a client, which is an IP address lease request, the DHCP server reserves an IP address for the client and makes a lease offer by sending a DHCPOFFER message to the client. This message contains the client's client id, the IP address that the server is offering, the subnet mask, the lease duration, and the IP address of the DHCP server making the offer.

Request

In response to the DHCP offer, the client replies with a DHCPREQUEST message, broadcast to the server, requesting the offered address. A client can receive DHCP offers from multiple servers, but it will accept only one DHCP offer. Before claiming an IP address, the client will broadcast an ARP request, in order to find if there is another host present in the network with the proposed IP address. If there is no reply, this address does not conflict with that of another host, so it is free to be used.

Acknowledgment

When the DHCP server receives the DHCPREQUEST message from the client, the configuration process enters its final phase. The acknowledgement phase involves sending a DHCPACK packet to the client. This packet includes the lease duration and any other configuration information that the client might have requested. At this point, the IP configuration process is completed.

Application design

Brief description

This project was created as a console application, which will be displaying IP prefix utilization statistics. When the prefix is filled with more that 50%, the application informs the administrator on the standard output and by logging through the syslog server. Program can listen on a given interface and will be displaying utilization statistics or can read communication from a pcap file. Each of these options must be specified in command line arguments, see *usage* below.

Technologies used

- Project was implemented in C++ language.
- libpcap library was used for capturing and parsing packets.
- ncurses library was used for creating console application, displaying and updating statistics.

Usage

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

-r <**filename**> : specifies file from which will be IP prefix utilization statistics created.

-i <**interface**> : specifies an interface on which will program listen and display IP prefix utilization statistics.

<**ip-prefix**> : the network range for which statistics will be created. Unlimited number of IP prefixes can be specified. IP prefix must be in correct form, e.g. 192.168.1.0/24 .

Note: Only one of these options can be specified. Program can display statistics either from a file or from the interface, not from both at the same time.

Implementation

Command line arguments

Program starts with command line arguments checking. Function

```
std::tuple<int, std::string> get_command_arguments(int argc, char **argv)
```

checks if user entered correct command line options with correct corresponding option argument. This will be stored into a tuple that consists of option argument and option argument identifier used in later processing.
Each IP prefix specified in command line arguments will be checked if it is in a correct form in

```
void get_prefixes(int argc, char **argv);
```

function. Correct form of IP prefix will be stored into a vector of IP prefixes, all other incorrect IP prefixes will be skipped and will not be included in final prefix utilization statistics.

DHCP Packet parsing

The key task of packet handling function that is a callback in *pcap_ loop* function is constructing a DHCP packet. DHCP information is stored in UDP payload. Program calculates correct offset for UDP payload in function.

```
#define ETHER_HEADER_OFFSET sizeof(struct ethhdr)
#define UDP_HEADER_OFFSET sizeof(struct udphdr)

size_t get_payload_offset(struct ip *ip_header) {
    return ETHER_HEADER_OFFSET + ip_header->ip_hl * 4 + UDP_HEADER_OFFSET;
}
```

DHCP packet contains options field, which is variable in length and which includes the message type, lease, domain name server IP address, and WINS IP address. These options are parsed and stored into a vector of *dhcp_ options* structs in function.

```
void set_options(const unsigned char *dhcp_options, const unsigned char *packet,
                 const struct pcap_pkthdr *header,
                 std::vector<struct dhcp_options> *options,
                 struct dhcp_header *dhcp);
```

Options field

DHCP uses the same message format as BOOTP, but DHCP uses the Option field to carry information for dynamic address allocation and to provide additional configuration information to clients. DHCP uses the option field in DHCP messages to carry control information and network configuration parameters, implementing dynamic address allocation and providing more network configuration information for clients.
In the *set_ options* function, parsing and processing DHCP options is in a cycle that lasts until the end of packet is met or until the end of the options (option 255).

```
#define DHCP_OPTIONS_END 255

while (dhcp_options < packet + header->len) {
    if (dhcp_options[0] == DHCP_OPTIONS_END) {
        break;
    }
```

Each option consist of option type, option length and data. Program then creates a struct that consists of these three values and is stored into a vector of options. Since option type and option length are 1 byte each, pointer, that is handling the cycle is moved by those 2 bytes + the length of data. DHCP options also have a padding option (option 0), which is used for alignment. This option is skipped and the handling pointer is moved by 1 byte.

```
#define DHCP_OPTIONS_PAD 0

if (dhcp_options[0] != DHCP_OPTIONS_PAD) {
        size_t code = dhcp_options[0];
        length = dhcp_options[1];
        struct dhcp_options temp = {
            .code = code,
            .len = length,
            .data = std::vector<char>(dhcp_options + 2, dhcp_options + 2 + length)
        };
        options->push_back(temp);
        dhcp_options += 2 + length;
    }
    else {
        dhcp_options += 1;
    }
```

Options overload

If DHCP options length is not enough for storing all options, option overload (option 52) might be present in options field. That indicates that the DHCP sname or file fields are being overloaded by using them to carry DHCP options. If the options overload is present, same mechanism is used for sname or file or both as for classic options field.

Handling DHCPACK

```
void ack_handle(struct dhcp_header *dhcp, std::vector<struct dhcp_options> *options)
```

Function *ack_handle* goes through all options in vector of options and searches for DHCP message type (option 53) and its value 5, which indicates DHCPACK message type.

```
void assign_address_to_prefix(struct in_addr ip_address)
```

Assigning IP address to prefix specified in command line arguments begins with checking, if the assigned IP address from a DHCP Acknowledgment was already assigned. For the IP prefix, that was not assigned, mask is calculated

```
uint32_t mask = (0xFFFFFFFFu << (32 - prefix));
```

and used on IP range and assigned IP. For IP address that matches the IP range statistics are updated except for network address and broadcast.

```
if ((ip_address.s_addr & netmask.s_addr) == (range.s_addr & netmask.s_addr)) {
        if (ip_address.s_addr != range.s_addr &&
        ip_address.s_addr != (range.s_addr | ~netmask.s_addr)) {
            ip_addresses[i].allocated_addresses++;
            ip_addresses[i].utilization =
            (float)((float)ip_addresses[i].allocated_addresses
            * 100.0f / (float)ip_addresses[i].max_hosts);
            mvprintw(ip_addresses[i].id, 35, "%ld", ip_addresses[i].allocated_addresses);
            mvprintw(ip_addresses[i].id, 55, "%.2f%%", ip_addresses[i].utilization);
            refresh();
        }
    }
```

Resources

[1] https://learn.microsoft.com/en-us/windows-server/networking/technologies/dhcp/dhcp-top
[2] https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol
[3] https://techhub.hpe.com/eginfolib/networking/docs/switches/5120si/cg/5998-8491_l3-ip-svcs_cg/content/436042653.htm
[4] https://techhub.hpe.com/eginfolib/networking/docs/switches/5120si/cg/5998-8491_l3-ip-svcs_cg/content/436042655.htm
[5] https://techhub.hpe.com/eginfolib/networking/docs/switches/5120si/cg/5998-8491_l3-ip-svcs_cg/content/436042656.htm
[6] https://efficientip.com/glossary/dhcp-option/
[7] http://www.tcpipguide.com/free/t_DHCPMessageFormat.htm
[8] http://www.tcpipguide.com/free/t_DHCPOptionsOptionFormatandOptionOverloading.htm
[9] https://www.tcpdump.org/pcap.html
[10] https://dev.to/tbhaxor/introduction-to-ncurses-part-1-1bk5