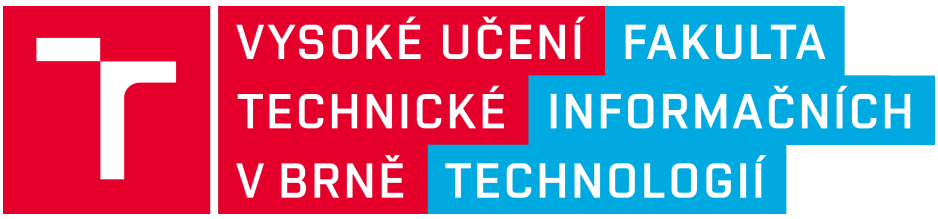

Síťové aplikace a správa sítí

Monitorování DHCP komunikace

20. listopadu 2023

Maksim Kalutski (xkalut00)

Obsah

1 Úvod

V dnešní době, kdy je konektivita a správa síťových prostředků klíčovou součástí každodenního života, je nezbytné monitorovat a analyzovat síťový provoz. Zvláště v rozsáhlých sítích, kde se vyskytuje mnoho zařízení, je důležité sledovat využití adresního prostoru a identifikovat případné problémy.

Tento program slouží k analýze síťových paketů na základě IP prefixů. Cílem je sledovat využití jednotlivých IP prefixů v síti a generovat statistiky, které pomáhají identifikovat případné problémy, jako je nadměrné využití adresního prostoru.

2 Návrh aplikace

Aplikace je navržena jako konzolová aplikace s možností sledování v reálném čase nebo analýzy existujících souborů. Program zpracovává pakety přenášené po síti a aktualizuje statistiky pro každý definovaný IP prefix. Program je omezen pouze na adresy IPv4. Tunelování není podporováno. Rozhraní pracuje s knihovnou ncurses. Pro ukončení programu musí uživatel stisknout klávesovou zkratku Ctrl+C pro signál. V případě, že překročí 50% alokací, program informuje uživatele o tomto faktu. Pokud utilization dosáhne 100%, přestane se vytížení prefixu počítat.

3 Teorie

Dynamic Host Configuration Protocol (DHCP) je síťový protokol používaný pro automatickou konfiguraci IP adres a dalších síťových parametrů v počítačových sítích. Jeho hlavním cílem je zjednodušit a automatizovat proces přidělování síťových konfigurací zařízením v síti, což zahrnuje přidělování IP adres, brány, DNS serverů a dalších informací potřebných pro správnou komunikaci na síti.

DHCP výrazně usnadňuje správu sítě tím, že eliminuje potřebu ruční konfigurace síťových parametrů na každém zařízení. Zároveň umožňuje efektivní využívání dostupných IP adres v síti a umožňuje jednoduchou adaptaci na změny v konfiguraci sítě.

4 Předpoklady

Před použitím programu musíte mít v operačním systému nainstalovány **g++**, **make**, **ncurses** a **libpcap**.

Pro **OS unixového typu**, můžete nainstalovat `g++` a `make` spuštěním následujícího příkazu:

```
$ sudo apt-get install build-essential
```

Pro instalaci `ncurses` do systému spusťte následující příkaz:

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

Pro instalaci `libpcap` do systému spusťte následující příkaz:

```
$ sudo apt install libpcap-dev
```

5 Sestavení programu

Pro sestavení programu spusťte následující příkaz:

```
$ make
```

6 Použití

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

6.1 Argumenty příkazového řádku

Argument	Popis
`-i`	Rozhraní, na kterém může program naslouchat.
`-r`	Statistika bude vytvořena z pcap souborů.
`ip-prefix`	Rozsah sítě pro které se bude generovat statistika.

7 Příklady Použití

```
$ sudo ./dhcp-stats -i lo 192.168.1.0/24 192.168.0.0/22 172.16.32.0/24
IP-Prefix Max-hosts Allocated addresses Utilization
192.168.0.0/22 1022 123 12.04%
192.168.1.0/24 254 123 48.43%
172.16.32.0/24 254 15 5.9%

$ ./dhcp-stats -r dhcp.pcapng 192.168.1.0/24 172.16.32.0/24
IP-Prefix Max-hosts Allocated addresses Utilization
192.168.1.0/24 254 50 19.69%
172.16.32.0/24 254 0 0.00%
```

8 Knihovny

Program používá následující knihovny:

- `<iostream>`: Poskytuje vstupně-výstupní streamy pro vstup a výstup.
- `<arpa/inet.h>`: Obsahuje funkce pro manipulaci s internetovými adresami a konverzi mezi textovými a binárními formáty.
- `<pcap/pcap.h>`: Poskytuje funkce pro zachytávání síťových paketů.
- `<cstdlib>`: Poskytuje obecné funkce pro manipulaci s pamětí a běžné funkce.
- `<netinet/ip.h>`: Obsahuje definici struktury pro IPv4 hlavičku.
- `<netinet/udp.h>`: Obsahuje definici struktury pro UDP hlavičku.
- `<netinet/ether.h>`: Obsahuje definice pro práci s ethernetovými rámci.
- `<map>`: Poskytuje implementaci asociativního kontejneru typu map.
- `<ncurses.h>`: Poskytuje funkce pro tvorbu textového uživatelského rozhraní v terminálu.
- `<syslog.h>`: Poskytuje funkce pro zápis do systémového logu.
- `<getopt.h>`: Poskytuje funkce pro zpracování příkazové řádky.
- `<cstring>`: Poskytuje funkce pro manipulaci s řetězci.
- `<vector>`: Poskytuje implementaci dynamického pole.
- `<stdexcept>`: Poskytuje vyjímky pro běhové chyby.
- `<algorithm>`: Poskytuje algoritmické funkce pro práci s kontejnery.
- `<signal.h>`: Poskytuje funkce pro zachytávání a zpracování signálů.
- `<unistd.h>`: Obsahuje funkce pro práci s POSIX systémovými voláními.

9 Popis implementace

9.1 Struktury (/include/structs.h)

- `PrefixStats`: Slouží k uchování statistik týkajících se IP adresového prefixu. Obsahuje samotný prefix, maximální počet hostitelů, přidělené adresy a využití.
- `Options`: Obsahuje programové volby, včetně síťového rozhraní, názvu souboru pro pcap soubory, seznamu IP adresových prefixů a mapy statistik prefixů.
- `UserData`: Slouží k předání dat do funkce `packetHandler` pro použití s pcap smyčkou.

9.2 Zpracování argumentů (/src/argument_parser.cpp)

Funkce `parseArguments()` slouží k parsování příkazové řádky. Podporuje zadání síťového rozhraní (`-i`) nebo názvu souboru (`-r`). Argumenty obsahující IP prefixy jsou uloženy v struktuře `Options`. Funkce `isValidPrefix()` ověřuje zda je prefix v rozsahu 1 až 32. Funkce `printUsage()` vypisuje návod k použití programu.

9.3 Zpracování paketů (/src/packet_process.cpp)

Funkce `processPacket()` analyzuje příchozí síťový paket. Kontroluje, zda se jedná o IP paket a následně porovnává zdrojovou IP adresu s definovanými prefixy. Funkce `updateStats()` aktualizuje statistiky pro daný IP adresový prefix na základě přidělených adres. Funkce `packetHandler()` je ovladač paketů pro použití s pcap smyčkou, volá funkci `processPacket`. Funkce `calculateMaxHosts()` spočítá maximální počet hostitelů pro daný IP adresový prefix.

9.4 Ncurses a aktualizace v reálném čase (/src/ncurses_utils.cpp)

Při sledování síťového rozhraní v reálném čase je použita knihovna `ncurses` pro vytvoření dynamického výstupu. Funkce `updateNcurses()` pravidelně aktualizuje obrazovku s aktuálními statistikami. Funkci `initNcurses()` a `endNcurses()` inicializuje a ukončuje knihovnu ncurses pro zobrazení statistik v terminálu.

9.5 Zpracování pcapng souboru (/src/pcap_process.cpp)

Funkce `processPcapngFile()` zpracovává pcapng soubor. Otevře soubor, inicializuje ncurses a spouští zpracování paketů pomocí `pcap_loop`. Následně aktualizuje statistiky a uzavře soubor.

9.6 Sledování síťového rozhraní v reálném čase (/src/live_capture.cpp)

Funkce `captureLivePackets()` otevírá živý odposlouchávací režim na zadaném síťovém rozhraní. Zpracovává pakety ve smyčce a aktualizuje statistiky v reálném čase. Reaguje na přerušení signálem `SIGINT`.

9.7 Výpis statistik (/src/print_options.cpp)

Funkce `displayPrefixStats()` vypisuje statistiky pro daný IP adresový prefix. Funkce `printOptions()` vypisuje celkové programové volby a seřazené statistiky prefixů IP adres.

9.8 Ovládání signálů CTRL-C (/src/signal_handler.cpp)

Funkce `handleSignal()` je určená pro ovládání signálů pro zpracování SIGINT (CTRL-C) a elegantní ukončení programu.

9.9 Hlavní funkce main (/src/dhcp-stats.cpp)

Hlavní funkce `main()` začíná parsováním argumentů a inicializací syslogu. Podle režimu (rozhraní nebo soubor) volí odpovídající funkci. Po dokončení zobrazí výsledné statistiky a případná varování syslogu.

10 Zajímavá část implementace

Následující část kódu představuje nejzajímavější část implementace. Tato část se nachází ve funkci `processPacket`, která analyzuje každý síťový paket a aktualizuje statistiky na základě poskytnutých IP prefixů.

```
void processPacket(const u_char *packet, const std::vector<std::string>
&prefixes, Options &options) {
    struct ether_header *eth_header = (struct ether_header *) packet;

    if (ntohs(eth_header->ether_type) == ETHERTYPE_IP) {
        struct ip *ip_header = (struct ip *) (packet + sizeof(struct
        ether_header));
        uint32_t src_ip = ntohl(ip_header->ip_src.s_addr);


        static std::vector<uint32_t> processedAddresses;
        if (std::find(processedAddresses.begin(), processedAddresses.end(),
        src_ip) != processedAddresses.end()) {
            return;
        }
        processedAddresses.emplace_back(src_ip);

        for (const auto &prefix: prefixes) {
            size_t pos = prefix.find('/');
            std::string ipPrefix = prefix.substr(0, pos);
            std::string mask = prefix.substr(pos + 1);
            uint32_t prefixMask = (0xFFFFFFFF << (32 - std::stoi(mask)))
            & 0xFFFFFFFF;
            uint32_t prefixStart = ntohl(inet_addr(ipPrefix.c_str())) &
            prefixMask;

            if ((src_ip & prefixMask) == prefixStart && src_ip != (prefixStart
            | 0) && src_ip != (prefixStart | 0xFFFFFFFF)) {
                updateStats(prefix, 1, options);

                for (const auto &otherPrefix: prefixes) {
                    if (otherPrefix != prefix) {

                        ...

                        if ((src_ip & otherPrefixMask) == otherPrefixStart &&
                        src_ip != (otherPrefixStart | 0) && src_ip !=
                        (otherPrefixStart | 0xFFFFFFFF)) {
                            updateStats(otherPrefix, 1, options);
                            break;
                        }
                    }
                }
```

```
                    break;
                }
            }
        }
    }
```

Tato funkce parsuje hlavičky Ethernetu a IP a extrahuje zdrojovou IP adresu. Následně iteruje přes poskytnuté IP prefixy a zjišťuje, zda zdrojová IP adresa odpovídá některé z nich. Nezahrnuje adresy broadcast nebo adresy yiaddr. Pokud je nalezena shoda, aktualizuje statistiky pomocí funkce `updateStats`.

Tato část je zajímavá, protože demonstruje hlavní funkcionalitu programu, tj. sledování síťového provozu, identifikaci paketů spojených s konkrétními IP prefixy a aktualizaci statistik podle potřeby.

11 Testování

Program byl testován na **Windows Subsystem for Linux 2 (WSL2)** se systémem **Ubuntu 22.04.1 LTS** a na **Merlinu**. Pro testování byly použity soubory pcap z adresáře `tests` a síťové rozhraní `lo`. Program byl testován na různých scénářích, včetně přímého zachycení sítě i rozboru souborů pcap. Abych otestoval, zda program správně zachycuje DHCP provoz z rozhraní `lo`, spustil jsem ve druhém terminálu příkaz `tcpreplay` s různými soubory pcap. Výsledky testů jsou uvedeny níže.

11.1 Sekce 1: Monitorování DHCP provozu na specifikovaném rozhraní

11.1.1 Test 1: Monitorování DHCP provozu na rozhraní lo

Vstup:

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

Očekávaný výstup:

```
IP-Prefix Max-hosts Allocated addresses Utilization
192.168.1.0/26 62 20 32.26%
192.168.0.0/22 1022 22 2.15%
172.16.32.0/24 254 0 0.00%
```

Skutečný výstup:

```
IP-Prefix Max-hosts Allocated addresses Utilization
192.168.1.0/26 62 20 32.26%
192.168.0.0/22 1022 22 2.15%
172.16.32.0/24 254 0 0.00%
```

11.1.2 Test 2: Monitorování DHCP provozu na rozhraní lo s překročením 50% využití

Vstup:

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

Očekávaný výstup:

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

Skutečný výstup:

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

11.2 Sekce 2: Monitorování DHCP provozu z pcap souboru

11.2.1 Test 3: Monitorování DHCP provozu ze souboru dhcp-ack-second.pcapng

Vstup:

```
$ ./dhcp-stats -r tests/dhcp-ack-second.pcapng 192.168.1.0/24 192.168.0.0/22 172
```

Očekávaný výstup:

```
IP-Prefix Max-hosts Allocated addresses Utilization
192.168.1.0/24 254 20 7.87%
192.168.0.0/22 1022 20 1.96%
172.16.32.0/24 254 0 0.00%
```

Skutečný výstup:

```
IP-Prefix Max-hosts Allocated addresses Utilization
192.168.1.0/24 254 20 7.87%
192.168.0.0/22 1022 20 1.96%
172.16.32.0/24 254 0 0.00%
```

11.2.2 Test 4: Monitorování DHCP provozu ze souboru dhcp-ack-random.pcapng s překročením 50% využití

Vstup:

```
$ ./dhcp-stats -r tests/dhcp-ack-random.pcapng 192.168.1.0/26 172.16.32.0/24 192
```

Očekávaný výstup:

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

Skutečný výstup:

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

12 Známé chyby

V případě, že uživatel zadá prefix `/31` nebo `/32`, bude mu vrácena hodnota využití `-nan%`.

13 Závěr

Jinak program úspěšně monitoruje provoz DHCP, analyzuje přidělené IP adresy a vypočítává využití zadaných síťových prefixů. Poskytuje aktualizace v reálném čase pomocí ncurses a zaznamenává zprávy prostřednictvím mechanismu syslog, pokud využití překročí 50 %. Program byl testován na různých scénářích, včetně přímého zachycení sítě i rozboru souborů pcap, a prokázal přesné a spolehlivé výsledky.

14 Literatura

[1] **RFC 2131: "Dynamic Host Configuration Protocol.":**
https://datatracker.ietf.org/doc/html/rfc2131

[2] **Dynamic Host Configuration Protocol (DHCP) and Bootstrap Protocol (BOOTP) Parameters:**
https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml

[3] **Beej's Guide to Network Programming:**
http://beej.us/guide/bgnet/

[4] **PCAP(3PCAP) MAN PAGE:**
https://www.tcpdump.org/manpages/pcap.3pcap.html

[5] **syslog(3) — Linux manual page:**
https://man7.org/linux/man-pages/man3/syslog.3.html

[6] **Ncurses Man Page:**
https://invisible-island.net/ncurses/man/ncurses.3x.html

[7] **Ncurses Documentation:**
https://invisible-island.net/ncurses/ncurses-intro.html