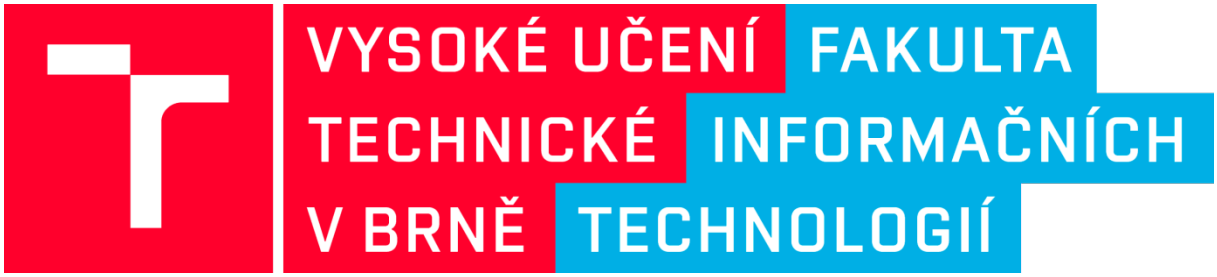
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technológií



Síťové aplikace a správa sítí

2016/2017

Zadání – DHCP server

Tibor Dudlák
xdudla00@vutbr.cz
20.10.2016

Obsah

Úvod

Hlavnou úlohou tohto projektu bolo implementovať IPv4 DHCP server, ktorý bude reagovať na požiadavky od *klientov* na danej sieti. Server ako výsledný program sa musí volať dserver a musí podporovať správy DISCOVER, OFFER, REQUEST, ACK, NACK a RELEASE, ktoré sú definované v RFC2131.

Implementácia tohto zjednodušeného DHCP servera musí byť v jazyku C/C++ pomocou knižnice BSD sockets a podporovať aspoň parametre *-p* ktorý bude reprezentovať sieť, ktorú dhcp server podporuje a *-e* reprezentuje zoznam IP adries oddelených čiarkou, ktoré server nesmie priradiť. Rozšírením, teda nepovinný, je parameter *-s*, ktorým je používateľ shchopný nastaviť statické mapovanie adries na IP adresy v danej sieti.

Program má byť preložiteľný a spustiteľný v prostredí systému Unixového typu (testovaný bude na virtuálnom počítači Linux/Ubuntu). Ukončiť sa má po obdržaní signálu SIGINT.

1. DHCP Server

Dynamic Host Configuration Protocol (DHCP) server poskytuje konfiguračné parametre *klientom* sieti[2]. Má dve hlavné funkcie a to spostredkovávať pre *klienta* špecifické nastavenia siete pre korektnú komunikáciu v nej a uchovávanie korektnej alokácie sieťových adries ku *klientom.*

DHCP je postavený na modeli *klient-server*, kde si *klienti* závislí od DHCP servera uňho alokujú sieťové adresy a ten im poskytne konfiguračné parametre. V tomto dokumente bude zviazaný termín *server* s počítačom v sieti, ktorý takéto konfiguračné parametre spolu s alokáciou poskytuje a termín *klient* bude spätý s počítačom závislým na týchto konfiguračných parametroch od DHCP servera.

DHCP *server* štandardne beží na sieťovom porte 67, na ktorom počúva a následne spracováva požiadavky od *klientov*, teda správy DHCPDISCOVER, DHCPREQUEST a DHCPRELEASE. Na tieto požiadavky pošle na broadcast alebo unicast klientom jednu z odpovedí DHCPOFFER, DHCPDECLINE, DHCPACK alebo DDCPNAK, podľa konfigurácie *servera* a požiadavkov na konfiguračné parametre od *klientov*. *Klienti* čakajú odpovede na sieťovom porte 68.

2. Návrh programu

Výsledný program je navrhnutý pre prostredia operačných systémov Unixového typu. Po spustení *server* skontroluje parametre programu a pri chybných skončí s chybovou hláškou a návratovým kódom 1. Pokiaľ chyba nenastala vytvorí socket za pomoci ktorého načúva na porte 67 a následne spracováva požiadavky iteratívne. K spracovaným požiadavkam, ktoré boli úspešne vyriešené *server* vypíše na stdout jeden riadok. Riadok obsahuje informácie o *klientovi*, presnejšie MAC adresu sieťovej karty, ďalej IP adresu ktorá mu bola pridelená a nakoniec informácie o tom, kedy začína a končí rezervácia adresy. *Server* je možné ukončiť signálom SIGINT po ktorom následne zavriem vytvorený socket a program skončí s návratovým kódom 0.

3. Implementácia servera

Ako implementačný jazyk som si zvolil jazyk C++. Dôvodom bola jednoduchšia práca s dynamicky alokovanými štruktúrami, ktoré som pri implementácii používal. Komunikácia *servera* s *klientmi* bola zabezpečená funkciami z knižnice BSD sockets. Na vytváranie odpovedí *klientovi* som použil štruktúru struct_dhcp_packet, ktorú som napĺňal požiadavkami *klientov*, následne poupravil, teda naplnil základnými konfiguračnými parametrami a poslal späť *klientovi*.

3.1. Spúšťanie programu dserver

Príklady spustenia programu sú upresnené v súbore README. Parametre programu sa môžu ľubovoľne kombinovať no každý voliteľný parameter smie byť zadaný maximálne jeden krát a povinný parameter -p práve jeden krát. O to sa postarajú funkcie v súbore arg_parser.cpp. Pokiaľ táto podmienka nieje splnená skončí program s chybovou hláškou a návratovým kódom 1. Formát argumentov každého parametru je taktiež podstatný pre správne načítanie nastavení DHCP *servera* a ich nesprávny zápis vedie k chybovému ukončeniu programu. Parametre sú následovné:

-p <netwoek_addres/cird> # určuje rozsah prideľovaných adries v CIDR zápise

-e <ip_addresses> # čiarkou oddelené ip adresy z rozsahu, ktoré *server* neprideľuje

-s <static_file> # súbor obsahujúci statické priradenie adries z rozsahu k MAC adresám

Parameter *-s* obsahuje ako argument cestu k súboru v ktorom je vo formáte “MAC_ADRESA IP_ADRESA” zapísané statické mapovanie IP adries pre MAC adresy oddelené medzerou a na každý riadok pre jedno zariadenie. Pri nedodržaní tohto formátu skončí program taktiež s chybou.

3.2. Spracovávanie požiadavkov

Po načítaní správy do štruktúry paketu, ktoré je v súbore req_handler.hpp funkcia `handle_request()` zhodnotí aký typ správy teda požiadavky došiel na socket. Následne podľa toho vyplní základné konfiguračné údaje teda IP adresu, masku siete, identifikátor *servera* a lease time vo funkcii `set_resp()`, po ktorom rezervácia vyprší v prípade že *server* odpovedá na správu DHCPREQUEST alebo DHCPOFFER. Po obdržaní správy DHCPRELEASE *server* vymaže záznam z databázy funkciami `return_ip_to_scope()` a `delete_record()`.

3.3. Ukladanie a mazanie záznamov

Záznamy si *server* ukladá do vektora štruktúr record_struct ktorá obsahuje potrebné položky pre správnu funcionalitu *servera*. Mazanie záznamov je možné vďaka funkcii `delete_record()`, ktorá využíva funkciu `record_position()` pre nájdenie záznamov podľa vstupných parametrov.

3.4. Statické mapovanie zo súboru

Statické mapovanie pre *server* zo súboru bolo vďaka dobrému návrhu dátových štruktúr implemenovať jednoduchšie. Pred začatím počúvania na porte 67 *server* vo funkcii `handle_request()` skontroluje validitu vstupného súboru a nahraje do vektora záznamov položky z flagom permanent a okamžite pridá adresy mapované staticky do vektora exlude_list. Po obdržaní správy release flag permanent zaistí, že záznamy ostanú vo vektore.

3.5. Výstup programu

Výstup programu bol špecifikovný nasledovne: <mac_adresa> <ip_addresa> <cas_prideleni_ip_adresy> <cas_vyprseni_prirazeni_adresy> a každý tento riadok popisoval zmeny ktoré nastali v rozsahu adries, ktoré *server* prideľuje. Napríklad:

c8:0a:a9:cd:7d:81 192.168.0.101 2016-09-29_13:45 2016-09-29_15:45

pričom formát ktorý bol uvedený ako príklad je aj dodržaný. *Server* vypisuje tieto informácie pomocou funkcie `printrecord()` ktorá používa funkcie `inet_ntoa()` na prevod ip adresy na človeku čítateľný formát a funkcie `strftime()`, ktorá prevedie time stampy do formátu času a dátumu uvedeného vyššie.

4. Záver

Implementovaný DHCP *server* reaguje na základné požiadavky od *klientov* na danej sieti. Podporuje správy DISCOVER, OFFER, REQUEST, ACK, NACK a RELEASE, ktoré sú definované v RFC2131[1]. Využíva funkcie z knižnice BSD sockets a podporuje všetky parametre ktoré boli zadané vrátane rozšírenia statického mapovania adries. Program je možné ukončiť zaslaním signálu SIGINT. Vyvíjaný bol v jazyku C++ na operačnom systéme Linux Fedora 24 a otestovaný na poskytnutom virtuálnom stroji isa2015 s operačným systémom Linux Ubuntu 16.04 LTS.

Literatúra

[1] https://tools.ietf.org/pdf/rfc2131.pdf

[2] https://tools.ietf.org/html/rfc1533