VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

DIAGNOSTIKA PROBLÉMŮ ZE ZACHYCENÉ SÍŤOVÉ KOMUNIKACE

DIAGNOSIS OF COMMUNICATION PROBLEMS IN CAPTURED NETWORK TRAFFIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETER MARKO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN KOŘENEK, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Marko Peter**
Program: Informační technologie
Název: **Diagnostika problémů ze zachycené síťové komunikace**
Diagnosis of Communication Problems in Captured Network Traffic
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s protokolem DHCP a TCP a možnými problémy, které mohou narušit uvedené protokoly.
2. Nejprve navrhněte a implementujte systém pro měření kvality zachycených dat. Zaměřte se na ztráty paketů způsobné nedostatečnou rychlostí záchytu.
3. Nastudujte a srovnejte existující možnosti diagnostiky problémů komunikace u uvedených protokolů.
4. Vytvořte sadu vzorků provozu, které budou demonstrovat vybrané problémy protokolů DHCP a TCP.
5. Vyberte a implementujte diagnostiky umožňující odhalit vybrané problémy.
6. Implementované metody ověřte nejprve na vytvořených datových vzorcích i na zachyceném síťovém provozu.
7. V závěru diskutujte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kořenek Jan, doc. Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 26. října 2018

Abstrakt

V tejto práci sa zaoberám systémom pre meranie kvality zachytených dát na sieti. Dôraz je kladený najmä na straty paketov spôsobené nedostatočnou rýchlosťou záchytu. Kvalitu záchytu zisťujem prostredníctvom protokolu TCP, ktorý implementuje poradové a potvrdzovacie čísla. Na základe týchto čísel je možné detegovať, že niektoré dáta boli správne prenesené, ale v záchyte ich nevidíme. Existujúce nástroje napríklad capTCP, alebo Wireshark, nie sú vhodné pre túto analýzu, lebo nedokážu komplexne analyzovať kvalitu zachytenej komunikácie, filtrovať komunikačné toky podľa nameraných metrík a nezapadajú do systému diagnostiky problémov komunikácie vyvíjaného v rámci projektu DISTANCE. Ďalej som sa zaoberal diagnostikou konfiguračných problémov protokolu DHCP.

Abstract

This thesis deals with a system for measuring the quality of captured data on the network. The focus is mainly on packet loss caused by insufficient capture rate. The quality of captured network traffic is evaluated on TCP protocol, which implements sequence and acknowledgment numbers. Based on these numbers, we can detect data, that has been correctly transmitted, but we do not see them in the capture. Existing tools, such as capTCP or Wireshark, are not suitable for this analysis because they cannot comprehensively analyze the quality of captured communication, filter communication flows according the metrics and do not fit into system DISTANCE. This thesis is also focused on configuration problems of DHCP protocol.

Kľúčové slová

kvalita zachytenej komunikácie, nezachytené segmenty, sekvenčné a potvrdzovacie čísla, diagnostika konfiguračných problémov DHCP, TCP, UDP, PCAP, Wireshark, RocksDB, systém DISTANCE

Keywords

quality of captured communication, not captured segments, sequence and acknowledgment numbers, DHCP configuration problems diagnosis, TCP, UDP, PCAP, Wireshark, RocksDB, system DISTANCE

Citácia

MARKO, Peter. *Diagnostika problémů ze zachycené síťové komunikace*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Kořenek, Ph.D.

Diagnostika problémů ze zachycené síťové komunikace

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Jana Kořenka, Ph.D., a že som uviedol všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Marko
14. mája 2019

Poďakovanie

Ďakujem vedúcemu práce Ing. Janovi Kořenkovi, Ph.D. za odborné vedenie mojej práce, za poskytnutie cenných rad, návrhov a pripomienok. Ďalej ďakujem Ing. Martinovi Holkovičovi a Doc. Ing. Ondřejovi Ryšavému, Ph.D. za poskytnutie odbornej pomoci a štúdijných materiálov.

Obsah

Kapitola 1

Úvod

Súčasnosť je poznamenaná dynamickými zmenami a rýchlym rozvojom komunikačných technológií, hlavne počítačových sietí. Množstvo pripojených zariadení na internete sa každým rokom zvyšuje a zvyšuje sa aj počet pripojených užívateľov. Zatiaľ čo v roku 2010 tvorili používatelia internetu 30% celosvetovej populácie, v roku 2017 to už bolo 48% [13]. To znamená, že tieto zmeny sa rôznymi spôsobmi dotýkajú mnohých z nás. Svet sa stáva stále viac prepojeným a je stále viac a viac závislý na komplexných telekomunikačných a sieťových systémoch. Internet nepoužívame len na vyhľadávanie, zálohovanie, zdieľanie či telefonovanie, ale aj na oveľa dôležitejšie systémy. Cez internet posielame peniaze, spravujeme svoje bankové účty, objednávame, či nakupujeme. Aby všetky tieto služby fungovali správne a pohotovo, je potrebné kvalitné pripojenie.

Internet je verejne prístupné médium a to znamená, že ktokoľvek je schopný komunikovať po tejto globálnej sieti. Táto dostupnosť a univerzálnosť nám však prináša aj riziká. Komunikácia na internete nemusí vždy prebiehať optimálne, môže dochádzať k útokom, konfiguračným problémom, či zahlteniu siete. Z narastajúcim počtom zariadení na internete rastie aj množstvo prenášaných dát. V takto veľkom množstve dát je čoraz dôležitejšia automatizovaná detekcia problémov na sieti, a preto je potrebné vytvoriť program, ktorý bude schopný analyzovať zachytenú komunikáciu a vyhľadávať problémy rôzneho druhu.

Na zaistenie správneho priebehu internetovej komunikácie je potrebné zistiť chyby, ktoré môžu pri komunikácii nastať a až potom pokračovať odstraňovaním chýb. Preto sa táto práca zaoberá analýzou sieťovej komunikácie. Práca je rozdelená na dve základné časti. V prvej sa zaoberám analýzou protokolu TCP a určujem kvalitu zachytených dat 2.2. V druhej časti sa venujem diagnostike protokolu DHCP, ktorý slúži na dynamické prideľovanie IP adries klientom na lokálnej sieti 2.4.

Táto práca zapadá do komplexnejšieho systému Distance, ktorého cieľom je vytvoriť efektívny nástroj pre diagnostiku problémov v počítačových sieťach založenú na pokročilej analýze sieťovej komunikácie. Výsledkom systému bude nástroj, ktorý bude zjednodušovať prácu administrátora, bude mu pomáhať vyhľadávať konfiguračné problémy na sieti prostredníctvom analýzy dôležitých a často používaných sieťových protokolov, ako napríklad: TCP, DHCP, DNS, POP3, IMAP, SMTP, ICMP, SIP, SAMBA a FTP.

Mojou úlohou bolo v rámci projektu Distance vyvinúť modul pre analýzu kvality zachytenej sieťovej komunikácie. Analýza sa zameriava na straty paketov spôsobené nedostatočnou rýchlosťou záchytu. V ďalšej časti vytváram popis diagnostiky protokolu DHCP pomocou tzv. diagnostického stromu. Diagnostika potom prebieha ako prechod diagnostickým stromom, pričom každý uzol zisťuje prítomnosť určitého problému.

Mnohé aplikácie potrebujú pre správne fungovanie spoľahlivý prenos informácií v sieti, ale pri prenose paketov môže vďaka rôznym vplyvom dôjsť k strate dát. Je preto potrebné použiť mechanizmus zabezpečujúci spoľahlivé doručenie dát medzi dvoma bodmi siete. Tento mechanizmus sa nazýva Transmission Control Protocol (TCP). Je to sada pravidiel, ktoré sú používané pri komunikácii medzi rôznymi zariadeniami. TCP protokol je jedným z najpoužívanejších protokolov pri komunikácii cez internet [7]. Je to protokol, ktorý zaručuje, že pakety dorazili, a zoraďuje pakety tak, aby boli v správnom poradí [14]. Vďaka tomuto systému je možné zaistiť spoľahlivý prenos dát. Protokol dokáže rozlišovať čísla portov, na základe ktorých určuje, ktorej aplikácii patrí paket. Vďaka tomu programy na sieti môžu vytvárať spojenia, prostredníctvom ktorých posielajú dáta. TCP nám svojou robustnosťou umožňuje zistiť informácie o stave siete a je preto dôležité venovať diagnostike protokolu TCP pozornosť.

Pri komunikácii prostredníctvom protokolu TCP môžu nastávať chyby, ako napríklad veľká stratovosť paketov, veľká latencia spojenia, alebo zahltenie siete. Protokol TCP nám svojou robustnosťou a vďaka tomu, že implementuje systém spoľahlivého prenosu, umožňuje vyhľadávať a analyzovať tieto problémy. Preto je potrebné vytvoriť program, ktorý nám uľahčí detekciu problémov na sieťovej vrstve tým, že bude analyzovať protokol TCP. Cieľom programu je odhaliť problémy, ako napríklad stratovosť paketov alebo odmietnuté spojenia a vytvoriť štatistiky o jednotlivých tokoch. Cieľom nástroja pre analýzu TCP je počítať generické štatistiky spoločné pre všetky toky, ako napríklad priepustnosť siete alebo počet retransmisií. Retransmisia je opakované odoslanie paketu, ktorý bol pri prenose stratený. Cieľom práce je tiež vizualizovať získané štatistiky formou grafov priebehu komunikácie. Práca sa tiež zameriava na vytváranie štatistík o jednotlivých tokoch, napr. čas pripájania k serveru, dĺžka odpovede servera alebo stav spojenia. Tiež sa zameriame na detekciu kvality zachytených dát a to prostredníctvom analýzy sekvenčných a potvrdzovacích čísel.

V rámci práce som sa zaoberal aj analýzou aplikačného protokolu DHCP(Dynamic Host Configuration Protocol). DHCP poskytuje rámec pre prenos konfiguračných informácií klientom v sieti TCP/IP. DHCP je založená na protokole Bootstrap Protocol (BOOTP) a pridáva možnosť automatického prideľovania opakovane použiteľných sieťových adries a ďalších konfiguračných možností. Účastníci DHCP môžu spolupracovať s účastníkmi BOOTP [8]. Pomocou analýzy DHCP sme schopní detegovať konfiguračné problémy či na strane klienta, alebo servera. Príkladom takého problému môžu byť duplikované IP adresy na lokálnej sieti, zahadzovanie ponúk od servera, alebo odmietnutie zo strany klienta.

Práca je rozdelená do viacerých logických celkov. V kapitole 2 sa zaoberám teoretickým rozborom analyzovaných protokolov TCP a UDP[1] na prenosovej vrstve a protokolom DHCP na aplikačnej vrstve. Taktiež je tu popis systému Distance, do ktorého táto práca zapadá. Kapitola 3 sa zaoberá detekciou kvality PCAP súboru zachyteného zo siete. V kapitolách 4 a 5 sa zaoberám implementáciou analýzy protokolov TCP a DHCP. V kapitole 6 je popis vytvorenia vzoriek sieťovej komunikácie a v nasledujúcej časti 7 zhodnotím dosiahnuté výsledky.

[1] UDP – User Datagram Protocol

Kapitola 2

Teoretický rozbor

2.1 ISO/OSI model

OSI, alebo Open System Interconnection [12] je model vytvorený Medzinárodnou organizáciou pre normalizáciu ISO, preto sa označuje ISO/OSI. Definuje architektúru a funkčné rozdelenie sieťovej komunikácie do siedmich vrstiev. Štandard fyzického rozhrania ako napríklad RS-232 zapadá do fyzickej vrstvy, zatiaľ čo ostatné vrstvy sa vzťahujú k rôznym ďalším protokolom. Množstvo dnešných protokolov má vytvorenú štruktúru práve na základe OSI modelu.

Tabuľka 2.1: model ISO/OSI

Vrstva	Názov PDU	Poznámka
Aplikačná	Dáta	Sieťový proces aplikácií
Prezentačná	Dáta	Reprezentácia dát a šifrovanie
Relačná	Dáta	Komunikácia medzi strojmi
Transportná	Segment	End-to-end spojenia a spoľahlivosť
Sieťová	Paket	Určovanie cesty a logické adresovanie (IP)
Linková	Rámec	Fyzické adresovanie (MAC a LLC)
Fyzická	Bit	Médium, signál, binárny prenos

Správy a dáta sú všeobecne odosielané po rámcoch, ktoré sú iba sekvenciou bajtov. Každý paket musí mať zdrojovú adresu a cieľovú adresu, aby systém vedel, kam ho poslať a aby príjemca vedel, odkiaľ pochádza. Pred vznikom paketu máme aplikačné dáta, ktoré sa postupne predávajú po vrstvách dole, v každej vrstve sú obalené metadátami, až vznikne paket. Potom je odoslaný. Keď paket prechádza smerom nadol, získava ďalšie hlavičkové informácie z každej vrstvy. To oznamuje ďalšej vrstve, čo má robiť s paketom. Po prijatí sú jednotlivé vrstvy paketu rozbaľované a odovzdávané príslušným vrstvám pre spracovanie, pričom každá časť informácie o hlavičke je odtrhnutá. Každá vrstva prečíta dáta z paketu, ako keby pochádzali zo zodpovedajúcej vrstvy na opačnom konci. Toto je známe ako komunikácia typu peer-to-peer, hoci skutočný paket je transportovaný prostredníctvom fyzického spojenia.

Model OSI [21] je užitočný pri poskytovaní univerzálneho rámca pre všetky komunikačné systémy. Neurčuje však skutočný protokol, ktorý sa má používať v každej vrstve. Predpokladá sa, že skupiny výrobcov v rôznych oblastiach priemyslu budú spolupracovať na definovanie štandardov softvéru a hardvéru zodpovedajúcich ich konkrétnemu sektoru.

Vrstvy ISO/OSI modelu

Fyzická Fyzická vrstva je najnižšou vrstvou OSI modelu. Zaoberá sa procesom samotného prenosu informácií prostredníctvom konkrétneho prenosového média. Existuje mnoho typov prenosových médií, napríklad ethernet, wi-fi, optický kábel, prenos prostredníctvom bluetooth a podobne. Technológia prenosu môže byť rôzna pre rôzne prenosové média, a preto existujú rôzne prenosové protokoly na fyzickej vrstve, ktoré definujú frekvencie, prípadne napäťové úrovne na metalickom kábli. Ako sa spomína v [12], cieľom týchto protokolov je optimálne využiť prenosové médium na prenos logických hodnôt, bitov. Protokol zaisťuje, že prijímacia stanica je schopná porozumieť odoslanej správe. Príkladom protokolov na tejto vrstve sú IEEE 802.11 (wi-fi), IEEE 802.3 (Ethernet), alebo token-ring.

Na linkové vrstvě prac

Linková Linková vrstva [12] vytvára spoj prostredníctvom konkrétnej dátovej linky. Táto vrstva vytvára rámce z informácií od vyššej vrstvy, ktoré doplňuje o MAC adresu a predáva fyzickej vrstve. Prenos dát prebieha medzi vzájomne prepojenými uzlami na jednej sieti. Na tejto vrstve dochádza k detekcii chýb pri prenose, prípadne k ich oprave. Používané protokoly sú napríklad frame relay, či Ethernet, alebo SDLC.

Sieťová Zaisťuje adresovanie a smerovanie dát v rámci jednej či viacerých sietí. Najdôležitejšou časťou komunikácie na tejto vrstve je proces prepínania paketov na smerovačoch. Vďaka tomu je možné posielať pakety na rôzne rozhrania na základe adresovania sieťovej vrstvy. Odosielanie paketov na rôzne rozhrania je možné kvôli unikátnosti IP adries na konkrétnej sieti. Najpoužívanejším protokolom na tejto vrstve je Internet Protocol verzie 4, ale v súčasnosti je snaha nahradiť ho modernejším protokolom IPv6 [12].

Transportná Transportná vrstva sa zaoberá prenosom informácií z aplikácie na zdrojovom počítači do aplikácie na cieľovom počítači. Vzniká tak logické spojenie dvoch procesov. Do transportnej vrstvy vstupuje súvislý tok dát, tu dochádza k rozdeľovaniu na jednotlivé pakety.

Posledné tri vrstvy sa v niektorých modeloch nerozčleňujú, napríklad TCP/IP model spojuje tri najvyššie vrstvy do jednej aplikačnej vrstvy. ISO/OSI model je ale členitejší.

Relačná Udržuje a spravuje relácie medzi dvomi komunikujúcimi aplikáciami. Relačná vrstva [11] zabezpečuje synchronizáciu jednotlivých dátových tokov medzi týmito aplikáciami. Služby zahrňujú vytvorenie, zrušenie vzťahu, obsluhu dialógov (pri polo-duplexnom spojení), synchronizáciu a pod.

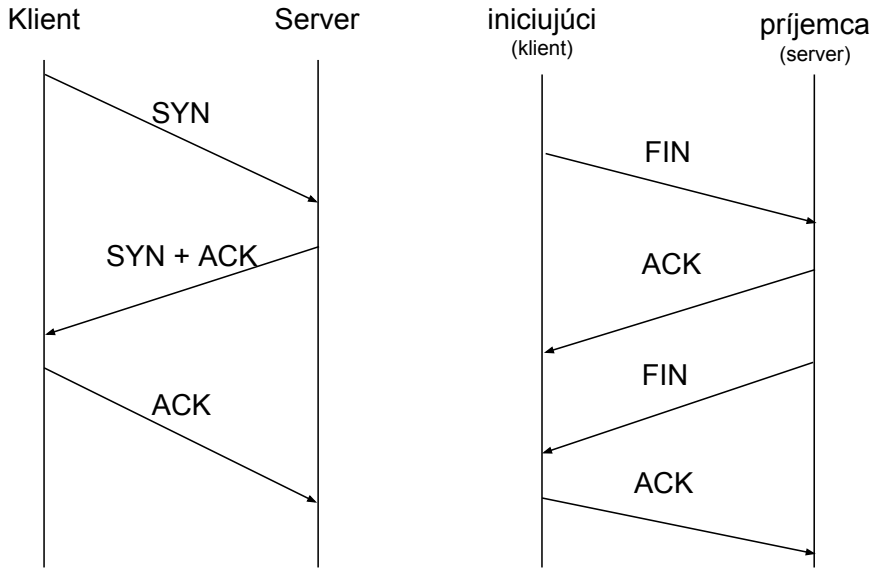
Prezentačná Dôvod existencie tejto vrstvy je ten, že nie vždy je formát prenosu dát vhodným formátom pre aplikáciu. Táto vrstva zaisťuje zobrazenie a prezentáciu dát medzi rôznymi aplikáciami. Používa sa tu štandard abstraktná syntaktická notácia ASN.1 pre prezentáciu informácii v hardvérových zariadeniach [11].

Aplikačná Definuje komunikáciu na úrovni užívateľských procesov. Každá aplikácia môže definovať svoj vlastný protokol, a preto tu existuje veľké množstvo rôznych protokolov. Aplikácia používa protokoly nižšej vrstvy podľa toho, aké parametre spojenia potrebuje.

Príkladom aplikačného protokolu je DHCP, FTP, HTTP, SMTP, DNS a pod. Táto vrstva tvorí tzv. aplikačné rozhranie, pričom každá aplikácia si definuje svoj vlastný formát dát.

2.2 TCP

Protokol TCP [14], alebo Transmission Control Protocol je najpoužívanjším protokolom na transportnej vrstve. Protokol TCP prenáša dáta z aplikácie na zdrojovom počítači do aplikácie na cieľovom počítači a vytvára tzv. logické spojenie procesov. Výhodou tohto protokolu je, že zaručuje doručenie dát, a to v poradí, v ktorom boli odoslané. Na nižších vrstvách ISO/OSI modelu môže dochádzať k chybám a stratám paketov. Protokol TCP IP používa systém sekvenčných a potvrdzovacích čísel, aby detegoval straty paketov a poslal ich znovu. Zároveň tieto čísla hovoria o poradí paketov. Pakety sa môžu po ceste medzi počítačmi preskupiť, napríklad niektoré pakety sú odoslané inou trasou ako iné. Protokol TCP využíva systém zásobníkov a ukladá pakety, ktoré prišli mimo poradia. Sprístupňuje len tie dáta, ktoré naväzujú na predchádzajúce pakety. Protokol TCP sa nehodí na real time aplikácie preto, že môže dochádzať k nerovnomernému spomaleniu prenosu, prípadne čakanie na znovuzaslanie paketu, ktorý sa stratil. Ďalšou nevýhodou je príliš veľká réžia, to znamená, že sa posiela veľké množstvo metadát určených na kontrolu prenosu. Podrobnú štruktúru hlavičky TCP nájdete v prílohe A.



Obr. 2.1: Proces vytvorenie a ukončenie spojenia medzi aplikáciami prostredníctvom protokolu TCP. Na vytvorenie sa používa príznak SYN (ľavá časť obrázku), na ukončenie sa používa príznak FIN (prvá časť obrázku).

Pre vytvorenie nového TCP spojenia je potrebná synchronizácia sekvenčných čísel na každej strane. To je zabezpečené výmenou niekoľkých segmentov, ktoré zahajujú spojenie a majú nastavený príznak SYN. Poradie zasielania inicializačných segmentov je možné vidieť na obrázku 2.1

Protokol TCP je definovaný v RFC[1] 793 [10]. Naviazanie TCP spojenia prebieha nasledovne. Najprv klient vykoná aktívne otvorenie komunikácie odoslaním segmentu s príznakom SYN na server. Klient nastaví sekvenčné číslo segmentu na náhodnú hodnotu A. Potom server odpovedá segmentom s nastavenými príznakmi SYN a ACK. Potvrdzovacie číslo je zvýšené na hodnotu A + 1. Server nastaví sekvenčné číslo tohoto segmentu na ďalšie náhodné číslo B. Nakoniec klient odošle segment s príznakom ACK na server. Sekvenčné číslo je nastavené na prijatú potvrdzovaciu hodnotu, tj. A + 1 a potvrdzovacie číslo je prijaté sekvenčné číslo zvýšené o 1, tj, B + 1. V tomto okamžiku je klientskej a serverovej aplikácii zaslané potvrdenie o dokončení naviazania spojenia.

Zatiaľ čo na vytvorenie spojenia sú potrebné tri segmenty, na jeho ukončenie sú potrebné štyri. To je spôsobené tým, že v TCP môže vzniknúť polovične uzavreté spojenie. Pretože v TCP sa dáta prenášajú v každom smere nezávisle, každý smer musí byť ukončený nezávisle. Vo chvíli, keď príjemca prijme FIN segment, musí oznámiť aplikácii, že protistrana ukončila svoju časť dátového toku. Spojenie môže byť ukončené ktorýmkoľvek z dvoch komunikujúcich koncových bodov a to tak, že koncový bod zašle protistrane segment obsahujúci príznak FIN. Tým sa komunikácia dostáva do stavu polovične uzavretého spojenia. Druhá stanica, ktorá stále neukončila spojenie, môže naďalej zasielať segmenty, ale v praxi sa to príliš nepoužíva. Keď príjemca dostane FIN segment, tak naspäť pošle ACK segment s potvrdzovacím číslom zvýšeným o jednotku. Príjemca ukončí svoju stranu spojenia zaslaním FIN segmentu. V tomto bode príjemca tiež informuje aplikáciu o ukončení spojenia.

Spojenia sú bežne iniciované klientom, pomocou prvého SYN segmentu zaslaného na server. Naproti tomu ukončenie spojenia môže byť aktívne zahájené aj klientom aj serverom. Často je to však klient, ktorý sa rozhodne ukončiť spojenie, pretože klientský proces je často riadený užívateľom, ktorý vyvolá ukončenie, viac obrázok 2.1.

V úvode 1 som spomínal, že spoľahlivý prenos prostredníctvom TCP garantuje doručenie toku dát odoslaných z jedného stroja na iný. Ako je ale možné, že protokol poskytuje spoľahlivý prenos dát prostredníctvom nespoľahlivého média? Odpoveď nie je taká jednoduchá. Protokol TCP používa systém pozitívnych potvrdení a opakovaného zasielania stratených paketov. Odosielateľ si ukladá informácie o každom segmente, ktorý odoslal a čaká na potvrdenie. Odosielateľ si tiež nastaví časovač, keď odosiela segment a pokiaľ časovač vyprší skôr, ako príde potvrdenie, tak odosielateľ prepošle segment znovu, lebo predpokladá, že sa stratil pri prenose, a preto nebol potvrdený [3]. Na detekciu straty paketov sa používajú sekvenčné a potvrdzovacie čísla v hlavičke TCP. Tieto polia sú kritickou súčasťou spoľahlivého prenosu dát TCP. TCP reprezentuje dáta ako neštrukturovaný, ale usporiadaný prúd bajtov.

Sekvenčné čísla identifikujú, ktoré bajty už boli odoslané a ktoré nie. Sekvenčné čísla slúžia na zoradenie prichádzajúcich segmentov do správneho poradia, na detekciu duplikovaných segmentov s rovnakým sekvenčným číslom a tiež vyhľadávanie stratených paketov. Potvrdzovacie čísla slúžia na potvrdenie, ktoré dáta dorazili na druhú stranu spojenia.

Nasledujúci odsek je prevzatý z knihy [14]. Predpokladajme, že proces na zariadení A chce odoslať sekvenciu dát do procesu na zariadení B prostredníctvom TCP spojenia. TCP na zariadení A implicitne očísluje každý bajt v dátovom toku. Predpokladajme, že dátový tok pozostáva zo súboru o veľkosti 500 000 bajtov, že maximálna dĺžka segmentu je 1000 bajtov a že prvý bajt dátového toku je očíslovaný 0. TCP konštruuje 500 segmentov z dátového toku. Prvému segmentu sa pridelí sekvenčné číslo 0, druhému segmentu sa

[1] RFC – Request For Comments, je to označenie dokumentov popisujúcich internetové protokoly a systémy.

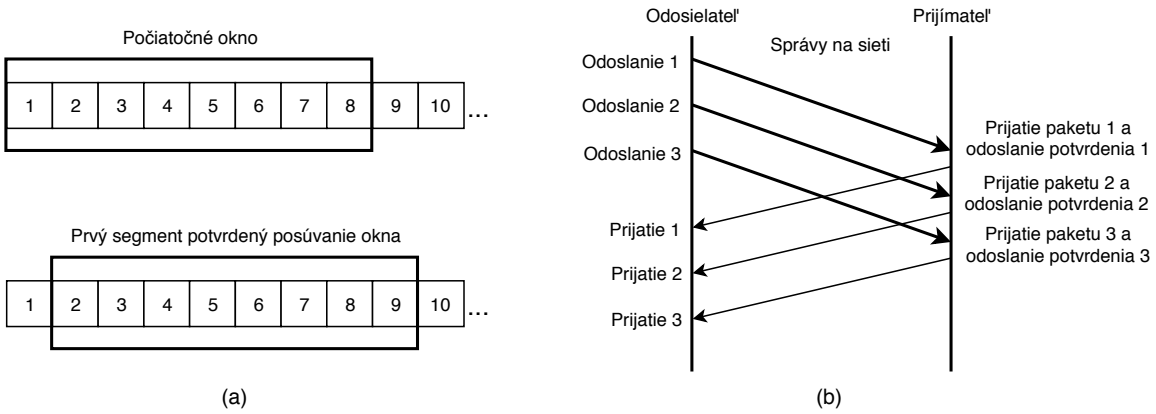
pridelí sekvenčné číslo 1 000, tretiemu segmentu sa pridelí sekvenčné číslo 2 000 atď. Každé sekvenčné číslo je vložené do poľa sekvenčného čísla v hlavičke príslušného TCP segmentu.

TCP je plne duplexný, takže zariadenie A môže prijímať dáta od B, zatiaľ čo vysiela dáta na B. Každý zo segmentov, ktoré prichádzajú od B, má sekvenčné číslo pre dáta plynúce z B do A. Potvrdzovacie číslo, ktoré A umiestňuje do svojho segmentu je sekvenčné číslo ďalšieho bajtu, ktorý A očakáva od B. V predchádzajúcom príklade som spomínal, že počiatočné sekvenčné číslo je 0. V skutočnosti obe strany TCP spojenia náhodne vyberú počiatočné sekvenčné číslo. Toto je z dôvodu, aby sa minimalizovala možnosť, že segment, ktorý je stále prítomný v sieti z predchádzajúceho už ukončeného spojenia medzi dvoma zariadeniami, je v neskoršom spojení medzi tými istými dvoma zariadeniami chybne platný segment [14].

Kontrola toku

Tok je ako umelý logický ekvivalent k hovoru, alebo spojeniu. Je to časť sieťovej komunikácie s konkrétnymi hodnotami zdrojovej a cieľovej IP adresy a tiež zdrojového a cieľového portu [15]. Medzi dvomi komunikujúcimi procesmi na sieti môže existovať viacero tokov preto, že môže dôjsť k ukončeniu komunikácie a následnému znovunaviazaniu.

Kontrola toku sa zaoberá frekvenciou zasielania segmentov po sieti. Cieľom je aby mal prijímateľ vždy dostatok miesta v zásobníku aby mohol segmenty prijať a následne spracovávať [9]. Na túto kontrolu sa používa mechanizmus posuvného okna.



Obr. 2.2: (a) Protokol posuvného okna s ôsmimi paketmi v okne. Po potvrdení prvého paketu dochádza k posunu okna, aby mohol byť odoslaný paket 9. (b) Príklad odoslania troch paketov pomocou protokolu posuvného okna. Princíp je taký, že odosielateľ môže poslať všetky pakety v okne bez toho, aby čakal na potvrdenie. Zdroj [2]

.

Pokiaľ by odosielateľ dát pri odoslaní každého segmentu musel čakať na prijatie potvrdenia, tak by to veľmi spomaľovalo komunikáciu a využitie plne duplexnej prenosovej linky by bolo minimálne. Preto sa používa koncept posuvného okna [2], ktorý umožňuje lepšie využiť prenosovú kapacitu a to tak, že umožní odosielateľovi posielať segmenty zatiaľ, čo čaká na potvrdenie. Základnou myšlienkou tohoto mechanizmu je, že odosielateľ môže zaslať len určitý počet paketov predtým, než dostane potvrdenie o prijatí. Protokol umiestni malé okno s pevnou veľkosťou na sekvenciu segmentov a prenesie všetky segmenty, ktoré sa nachádzajú vo vnútri okna. Akonáhle odosielateľ dostane potvrdenie pre prvý segment vo

vnútri okna, posúva okno a odošle nasledujúci segment, ako môžme vidieť na obrázku 2.2. Okno sa posúva, pokiaľ sú prijaté potvrdenia.

Protokol posuvného okna si vždy pamätá, ktoré pakety boli potvrdené a nastaví časovač pre každý nepotvrdený segment. V prípade straty segmentu časovač vyprší a odosielateľ prepošle paket znovu. Odosielateľ posunie okno za všetky potvrdené segmenty [2]. Na prijímacej strane spojenia existuje podobný mechanizmus na potvrdzovanie segmentov. Protokol teda rozdeľuje sekvenciu segmentov do troch množín : odoslané, prijaté a potvrdené.

V bežnej komunikácii sa používajú tzv. kumulatívne potvrdzovacie číslo, to znamená že komunikant môže jedným paketom potvrdiť prijatie celej skupiny segmentov. Celú skupinu je možné potvrdiť jedným potvrdzovacím segmentom. V tomto segmente bude určené potvrdzovacie číslo najvyššieho zachyteného bajtu v spojitom toku dát. To znamená, že keď dôjde k výpadku segmentu na linke, tak prijímateľ potvrdí len segmenty predtým než došlo k výpadku a to zaslaním potvrdenia segmentu s najvyšším sekvenčným číslom v spojitom toku dát. Segmenty, ktoré prišli mimo poradia sú ukladané do vyrovnávacej pamäti. Keď dorazia predchádzajúce chýbajúce segmenty, tak TCP usporiada segmenty tak, aby boli v správnom poradí a prijatie potvrdí zaslaním paketu, ktorý potvrdzuje prijatie najvyššieho segmentu vo vyrovnávacej pamäti [10].

2.3 UDP

Pre mnoho aplikácii nie je protokol TCP vhodný, hlavne kvôli blokovaniu čela fronty, to znamená, že pri strate jedného segmentu aplikácia nedostáva ďalšie segmenty z fronty dokiaľ nie je stratený segment znovu preposlaný. To spôsobuje veľké spomalenie a dočasný výpadok spojenia. Tento problém rieši protokol UDP, User Datagram Protocol, ktorý je definovaný v dokumente RFC 768 [18]. Je to ďalší známy protokol transportnej vrstvy, ktorý svojou jednoduchosťou doplňuje protokol TCP. Tento protokol na rozdiel od TCP nekontroluje straty segmentov. Prípadné kontroly straty a zmeny poradia segmenteov necháva na aplikácii.

Tabuľka 2.2: Formát hlavičky UDP

	0	1	2	3
0	Source Port		Destination Port	
4	Length		Checksum	

Protokol UDP je vhodnejší pre aplikácie, ktoré potrebujú získavať dáta v reálnom čase s malým spomalením, napr. streamované média, telefonovanie cez internet, alebo videohry. Pre tieto aplikácie je dôležitejšie, aby získali dáta včas, ako ich dostávať kompletné a v správnom poradí. Protokol UDP sa používa aj na zasielanie konfiguračných informácii po lokálnej sieti a to hlavne z dôvodu jeho jednoduchosti. Podobne ako u protokolu TCP aj protokol UDP obsahuje polia zdrojového a cieľového portu, ktoré slúžia na identifikáciu procesu bežiacom na konkrétnom zariadení. Pole length označuje počet bajtov v pakete, tzn. celková dĺžka aj hlavičky aj dát. Checksum je kontrolný súčet na kontrolu správneho prijatia segmentu, umožňuje detekciu prenosovej chyby napr. preklopenie jedného bitu správy.

2.4 DHCP

Protokol DHCP, Dynamic Host Configuration Protocol je definovaný v dokumente RFC 2131 [8]. DHCP patrí medzi aplikačné protokoly, slúži na prideľovanie konfiguračných informácií. Tento protokol vychádza z protokolu BOOTP, ktorý vyriešil problém s konfigurovaním systémov, ale nezaoberal sa problémom prideľovania adries. Tento problém rieši práve protokol DHCP, ktorého cieľom je automatické prideľovanie IPv4 adries klientom na lokálnej sieti. IP adresu definujeme, ako jedinečný číselný identifikátor, ktorý je priradený každému počítaču pracujúcemu v sieti založenej na protokole TCP / IP. Manuálna konfigurácia počítačov, ručné nastavenie statickej IP adresy a pridelenie iných konfiguračných parametrov TCP / IP nie je zložitá úloha. Avšak, manuálne konfigurovanie tisícok počítačov s jedinečnými IP adresami by bolo časovo náročné a ťažkopádne. Pri ručnom priradení IP adresy, sa zvyšuje riziko chyby. Môžu nastať nasledujúce problémy: duplicitné priradenie IP adries, konfigurácia nesprávnych sieťových masiek a nesprávne nastavené rôznych konfiguračných parametrov protokolu TCP / IP. Preto je dôležitý protokol DHCP. Je to služba zameraná na automatizáciu vyššie uvedených úloh, ktoré by inak musel robiť správca siete, a tým zjednodušuje správu adresovania IP v sieti založenej na TCP / IP. Jedným z hlavných nevýhod ručného priradenia IP adresy na stovky počítačov je, že by to mohlo viesť k tomu, že pridelené IP adresy nie sú jedinečné. Na komunikáciu na internete a na lokálnej sieti TCP / IP musia mať všetky zariadenie definovanú IP adresu. IP adresa má 32 bitov a slúži na jednoznačnú identifikáciu zariadenia na sieti.

Manuálna konfigurácia IP adresy sa používa iba za týchto okolností:

- Ak v sieti nie sú nakonfigurované DHCP servery a sieť má viacero segmentov.
- Pri konfigurácii počítača ako DHCP servera.
- Keď konfigurujete počítače ako dôležité sieťové servery, ako napríklad servery DNS[2].

DHCP klient je dnes dostupný prakticky v každom sieťovom zriadení a na každom operačnom systéme [19]. Veľkou výhodou DHCP je, že umožňuje mobilitu zariadení. Užívatelia často potrebujú pripojiť svoje zariadenie k rôznym Wi-Fi sieťam, tieto siete môžu byť veľkého rozsahu zo stovkami pripojených zariadení. Problém veľkých sietí je nedostatok IP adries a preto je potrebné klientom prenajímať adresy len na obmedzený čas a pokiaľ klient opustí sieť, tak nájom adresy nebude predĺžený, adresa je uvoľnená a neskôr bude pridelená inému zariadeniu na sieti. Aby klient zabránil exspirácii jeho IP adresy, čo by v podstate znamenalo odpojenie od internetu, klient pravidelne obnovuje svoju IP adresu predtým, než vyprší nájom.

Obmedzovaním používania IP adries po uplynutí nájmu, a poskytnutím mechanizmu pre klientov, aby pokračovali v obnovovaní svojich nájmov, DHCP umožňuje spoľahlivé spätné získavanie nepoužívaných IP adries. Ak je zariadenie ponechané vypnuté dlhšiu dobu, musí sa po zapnutí znova prihlásiť na DHCP server. Ak nie je k dispozícii predchádzajúca adresa zariadenia, tak mu bude pridelená nová adresa. Toto zabraňuje väčšine konfliktov pri prideľovaní adries. Formát DHCP správy je možné vidieť v prílohe B.

[2]DNS – Domain Name System, je definovaný v RFC 1035 https://tools.ietf.org/html/rfc1035

Typy DHCP správ

Protokol DHCP [11] pracuje na princípe klient-server. Klientom je počítač, ktorý sa pripája do siete a žiada informácie o IP adrese a ďalšie konfiguračné parametre. Server je program, ktorý spravuje databázu dostupných IP adries a ďalších konfiguračných parametrov pre danú sieť, odpovedá na otázky klientov a posiela im požadované dáta.

DHCPDISCOVER Klient odošle túto správu pomocou broadcastu na port 67 aby zistil adresy dostupných DHCP serverov. Zdrojovú adresu vynuluje a vygeneruje náhodný identifikátor transakcie.

DHCPOFFER Server odošle túto správu klientovi, ako odpoveď na DHCPDISCOVER. Správa obsahuje tiež konfiguračné informácie ako napríklad ponúkanú IP adresu, alebo dobu prenájmu IP adresy. Ak sa server rozhodne adresu nepridelit – napríklad je adresa viazaná na pevnú hardvérovú adresu alebo nemá žiadne voľné IP adresy – zamietne žiadosť správou DHCPNAK [11].

DHCPREQUEST Správa od klienta na server [8]

- Požadovanie ponúkaných parametrov od jedného DHCP servera a implicitné odmietnutie ponúk od všetkých ostatných DHCP serverov.
- Potvrdenie správnosti IP adresy, ktorá bola alokovaná v minulosti napríklad pred reštartom systému.
- Predĺženie prenájmu konkrétnej IP adresy.

DHCPDECLINE Správu posiela klient na server, aby oznámil, že pridelená IP adresa sa už používa.

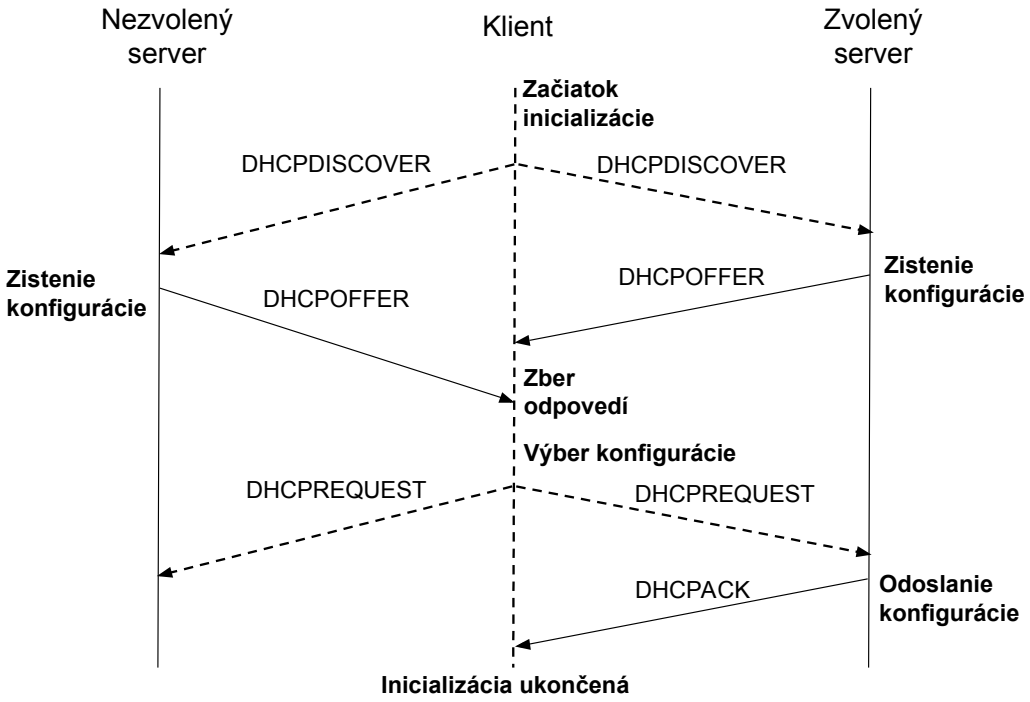
DHCPACK Server oznamuje klientovi konfiguračné parametre vrátane adresy siete a potvrdzuje správnosť konfigurácie. Touto správou server potvrdzuje prijatie voľby a pridelenie IP adresy na danú dobu [11].

DHCPNAK Server oznamuje klientovi, že jeho sieťová adresa je nesprávna napríklad, klient sa presunul na novú podsieť, alebo prenájom IP adresy vypršal [8].

DHCPRELEASE Klient pošle správu na server a oznamuje že vracia požičanú IP adresu a predčasne ukončuje jej prenájom [8].

DHCPINFORM

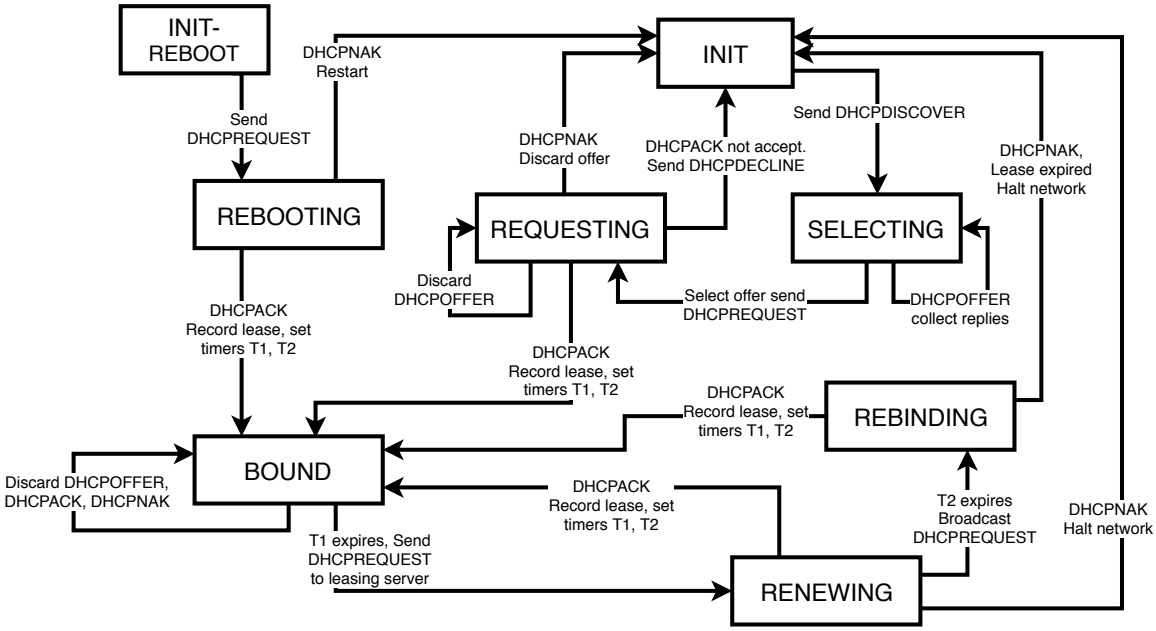
- Klient požaduje od servera konfiguračné informácie.
- Klient oznamuje serveru, že už má pridelenú a nakonfigurovanú IP adresu iným spôsobom, napríklad ručne.



Obr. 2.3: Priebeh pridelenia konfiguračných informácií jednému klientovi na sieti s dvomi DHCP servermi [8].

Interpretácia časových hodnôt [8] Klient si zapožičia sieťovú adresu na pevne stanovený čas (môže byť nekonečný). V celom protokole sú časy vyjadrené v sekundách. Maximálna hodnota je rezervovaná na reprezentovanie nekonečna. Keďže klienti a servery nemusia mať synchronizované hodiny, časy sú reprezentované ako relatívne, ktoré majú byť interpretované s ohľadom na miestne hodiny klienta. Rozsah reprezentácie času v sekundách na 32 bitoch je približne 100 rokov. Je to dostatočný rozsah pre potreby protokolu DHCP. Predpokladom pre fungovanie tohto algoritmu interpretácie trvania nájmu je, že klientské a serverové hodiny sú stabilné medzi sebou. Ak medzi týmito dvomi hodinami dochádza k posunu, server môže predpokladať, že nájom vypršal skôr, ako to zistil klient. Pre kompenzáciu server vracia klientovi kratšiu dobu nájmu, než si zapíše do lokálnej databázy.

Príklad komunikácie medzi DHCP klientom a servermi Na obrázku 2.3 môžeme vidieť časový diagram zobrazujúci komunikáciu počas alokácie IP adresy na lokálnej sieti. Ako vidíme klient komunikuje prostredníctvom broadcastu a server mu odpovedá pomocou unicastu. Je to preto, že klient na novej sieti nevie kto je DHCP server a aká je jeho IP adresa. Preto sa musí spýtať všetkých a ozve sa len DHCP server. Obecne môže na jednej sieti byť viacej DHCP serverov, aj keď sa to nestáva veľmi často. Preto nie je možné získať konfiguračné informácie v jednom kroku. Keď si klient vyberie z ponúk, ktoré dostal, tak odošle DHCPREQUEST obsahujúci IP adresu DHCP servera, ku ktorému sa chce klient pripojiť. Týmto spôsobom klient oznámi jednému serveru, že sa k nemu chce pripojiť a súčasne ďalším serverom oznámi, že sa k nim nechce pripojiť. Server potvrdzuje konfiguračné informácie pomocou DHCPACK paketu.



Obr. 2.4: Stavový diagram pre DHCP klientov [8] popisuje stav, v ktorom sa môže klient pri komunikácii nachádzať.

V diagrame 2.4 môžeme vidieť, akým spôsobom prebieha komunikácia medzi klientom a serverom a do akých stavov je možné sa dostať. Diagram je kreslený z pohľadu klienta, pokiaľ je na šípke napísané send znamená to, že klient posiela paket na server, inak server posiela odpoveď. Stav INIT je počiatočný stav komunikácie. V tomto stave sa klient nachádza, keď príde na novú sieť a potrebuje získať nové konfiguračné informácie. Existujú tri spôsoby inicializácie, ktorými môže klient získať IP adresu: dynamické pridelenie novej adresy, inicializácia so známou adresou a inicializácia s externe priradenou sieťovou adresou.

Dynamická inicializácia a priradenie sieťovej adresy prebieha nasledovne. Klient začína v stave INIT a tvorí správu DHCPDISCOVER, potom by mal čakať náhodný čas od jednej do desiatich sekúnd na zrušenie prípadnej synchronizácie používania DHCP pri spustení. Klient môže navrhnúť sieťovú adresu, alebo čas prenájmu. Potom klient generuje a zaznamenáva náhodný identifikátor transakcie, v ďalšom kroku zaznamená svoj miestny čas na neskoršie použitie pri výpočte exspirácie nájmu adresy. Potom pošle DHCPDISCOVER pomocou broadcastovej IP adresy 0xffffffff. Klient po nejaký čas prijíma správy typu DHCPOFFER, ktoré majú správny identifikátor a zapisuje si IP adresy serverov [8].

Ďalšia možnosť, ako môže klient získať konfiguračné informácie je inicializácia so známou sieťovou adresou. Klient začína v stave INIT-REBOOT a pošle DHCPREQUEST správu. Klient musí vkladať svoju známu sieťovú adresu ako požadovanú IP adresu. Klient generuje a zaznamenáva náhodný identifikátor transakcie, následne zaznamená svoj miestny čas na neskoršie použitie pre výpočet exspirácie nájmu adresy. Klient potom pošle DHCPDISCOVER pomocou broadcastu. Po prijatí správy DHCPACK so zhodným identifikátorom sa klient stáva inicializovaným a presúva sa do stavu BOUND. Klient zaznamenáva čas exspirácie ako súčet času, v ktorom je odoslaný DHCPREQUEST a doby prenájmu získaná z prijatej DHCPACK správy [8].

Inicializácia s externe priradenou sieťovou adresou [19] prebieha tak, že klient odošle správu DHCPINFORM, generuje a zaznamenáva náhodný identifikátor transakcie. Klient umiestni svoju sieťovú adresu do poľa "Client IP Address". Potom buď pomocou unicastu,

alebo broadcastu odošle správu typu DHCPINFORM na DHCP server. Po prijatí správy DHCPACK so zhodným identifikátorom, sa klient stáva inicializovaným a presúva sa do stavu BOUND. Ak klient nedostane DHCPACK v primeranom čase (60 sekúnd alebo 4 pokusy), potom by sa malo zobraziť hlásenie informujúceho užívateľa o probléme.

Zistiteľné problémy

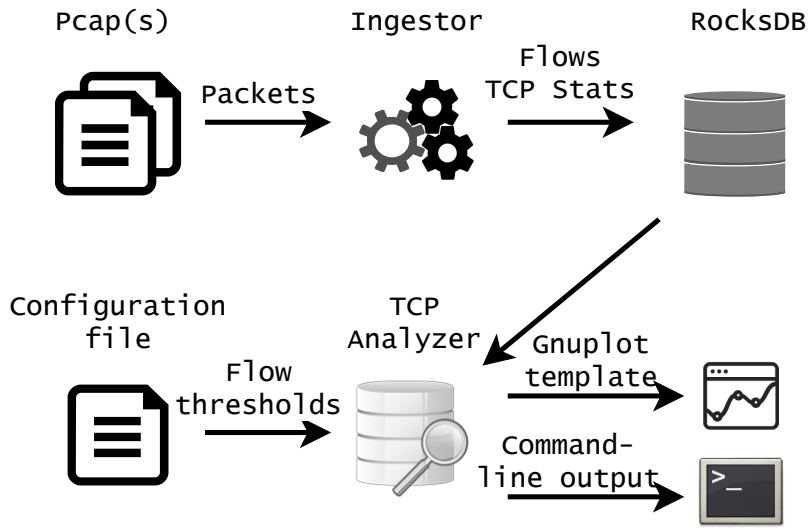
- **Requesty bez odpovede:** Kontrola že množstvo DHCP retransmisií nie je veľmi vysoké a že odpoveď bola delegovaná.
- **Duplikovaná IP adresa:** DHCP server môže v špecifických prípadoch prideliť rovnakú IP adresu viacerým klientom. Tento problém môže nastať napríklad, keď sa minú všetky adresy, ktoré server môže prideľovať. Tento problém je možné detektovať prostredníctvom ARP správy.
- **Server zahadzuje ponuky:** Detekcia, či server odpovedá prostredníctvom správy DHCPNAK. DHCP server pošle DHCPNAK klientovi, pokiaľ si je istý, že klient žiada adresu na lokálnej sieti, ktorá na tejto sieti neexistuje.
- **Klient posiela DHCPDECLINE:** Pokiaľ klient posiela správu DHCPDECLINE, tak oznamuje serveru, že klient zistil prostredníctvom iných prostriedkov, že ponúkaná IP adresa sa už používa. Môže to znamenať problém v konfigurácii DHCP.

2.5 Systém Distance

Distance – DIagnostika SíTě ze zAchycené komuNikaCE, je výskumný projekt pre diagnostiku počítačových sietí založený na pokročilej analýze sieťovej komunikácie a vytvorenie nástroju pre sieťových administrátorov hlavne menších a stredných firiem. Vytvorený analytický modul bude integrovaný do existujúcich produktov spoločne s vytvorením modulu pre pokročilé zachytávanie vybranej komunikácie na základe definovaných pravidiel a vizualizáciou diagnostických výsledkov z analýzy sieťovej komunikácie.

Architektúra

Diagnostický modul spracováva vstupné dáta v troch krokoch. V prvom kroku je načítaný vstupný PCAP súbor modulom Ingestor. Tento modul ukladá informácie o nájdených tokoch a ďalšie metadáta o jednotlivých paketoch do pracovnej databázy, ktorou je RocksDB. RocksDB [20] je výkonná databáza na ukladanie hodnôt vo formáte kľúč-hodnota. V druhom kroku sú spustené analyzátory protokolov, ktoré dekódujú pre vybrané toky jednotlivé pakety a vybrané položky ukladajú opäť do dátového úložiska. Posledným krokom je spustenie diagnostických procedúr, ktoré na základe definovaných pravidiel hľadajú problémy v komunikácii a identifikujú ich príčiny. Výstupom diagnostiky sú informácie, ktoré je možné opäť uložiť do databázy alebo exportovať vo zvolenom formáte, napríklad JSON. Tieto informácie budú použité vizualizačným modulom pre zobrazenie výsledkov diagnostiky administrátorovi.



Obr. 2.5: Architektúra Ingestoru a analýzy TCP

Databáza

Úložisko dát je realizované pomocou RocksDB databázy, ktorá poskytuje vysokú rýchlosť čítania a zápisu dát. RocksDB má tieto základné vlastnosti:

- Dáta uložené vo formáte kľúč hodnota – kľúčom či hodnotou môže byť ľubovoľné pole bajtov, každá informácia uložená v databáze musí mať unikátny kľúč. Dáta nie sú štruktúrované do tabuliek s viacerými stĺpcami, ako tomu je v databázach typu SQL.
- Vstavaná – databáza beží v rovnakom procese ako programy, ktoré ju používajú
- adaptovateľná – pomocou pokročilej konfigurácie je možné vyladiť databázu pre rôzne aplikačné scenáre.

Hoci RocksDB neposkytuje pokročilé dotazovanie obvyklé pre SQL databázy, je jej použitie vhodné vzhľadom na požiadavky na diagnostický modul a charakteru ukladaných dát. RocksDB nepožaduje definíciu schémy dát, na dáta je nazerané ako na dvojice kľúča hodnoty. RocksDB nevyžaduje ani určenie typu pre dáta, či už sa jedná o kľúče či hodnoty. Vďaka tomu je možné pridávať nový typ dát bez nutnosti meniť schému databázy. Na druhú stranu táto voľnosť môže znamenať komplikáciu pri komunikácii medzi jednotlivými komponentmi modulu. Preto bolo navrhnuté použitie tzv. *Protocol Buffers* ako mechanizmu pre ukladanie a čítanie dátových štruktúr, ktoré zabezpečia, že dáta budú interpretovaná rôznymi časťami modulu zhodným spôsobom.

Použitá databáza RocksDB je rozdelená do štyroch častí:

pcaps Informácie o spracovaných PCAP súboroch, časové značky prvého a posledného paketu, meno súboru a pod.

packets Všetky spracované pakety uložené k jednotlivým tokom.

flows Podrobnosti o komunikačných tokoch napr. počet prenesených paketov a bajtov, časové značky, alebo číslo aplikácie.

statistics Metriky komunikačného toku napr. čas potrebný na vytvorenie TCP spojenia, alebo doba potrebná na prenos. Je tu tiež uložený priebeh komunikácie v čase.

Modul Ingestor

Modul Ingestor je nástroj, ktorý používam vo svojom programe na spracovanie vstupných dát. V tomto module prebieha prvá fáza spracovania zachytených paketov vo forme PCAP súborov. Ingestor je modul, ktorý načíta jeden alebo viac súborov určených na spracovanie a vykoná základnú analýzu a parsovanie. V tejto fáze dochádza k niekoľkým úkonom:

- Parsovanie hlavičiek paketov.
- Identifikácia protokolov obsiahnutých v paketoch.
- Identifikácia aplikačného protokolu prenášaného v obsahu paketu. Na identifikáciu protokolu vyššej, ako štvrtej vrstvy, je použitá existujúca open source knižnica *libprotoident*).
- Ukladanie nájdených pozícií začiatkov hlavičiek pre neskorší ľahký prístup k dátam.
- Zoskupenie paketov do komunikačných tokov a dohľadanie oboch smerov komunikácie medzi dvoma adresami.
- Uloženie získaných metadát do databázy RocksDB.

Pre vytvorenie parsera paketových hlavičiek, ktorý tvorí súčasť modulu Ingestor, bol použitý vysokoúrovňový popis v jazyku P4 [4]. Je to doménovo špecifický jazyk určený na spracovanie paketov. Využitie jazyka P4 umožňuje jednoduchú úpravu podporovanej množiny protokolov a celý systém je vďaka tomu flexibilný a je možné ho ľahko adaptovať na novovznikajúce komunikačné protokoly.

Oproti súčasným bežne používaným parserom paketových hlavičiek, generovaný parser podporuje neobmedzené rekurzívne zanorovanie pri spracovaní paketov. Táto funkcionalita umožňuje nástroju spracovávať pakety obsahujúce komplikovane zapuzdrené protokoly. Takéto pakety môžu obsahovať dokonca aj niekoľko vzájomne zapuzdrených sieťových protokolov naraz, čo sťažuje analýzu najmä pomocou bežných automatických nástrojov, ktoré končia pri prvej známej nájdenej vrstve. Okrem toho, že Ingestor spája komunikáciu všetkých zadaných PCAP súborov, pri analýze je potrebné správne rozdeľovať postupnosť paketov medzi dvoma adresami do správnych komunikačných tokov, tj. rešpektovať ukončené a znovu naviazané komunikačného toku. Za týmto účelom prebieha v module Ingestor sledovanie TCP príznakov (SYN, FIN a RST) a rozostupov medzi paketmi jedného spojenia (timeout), pretože oboje môže signalizovať prerušenie komunikácie. Rozdeľovanie komunikačných tokov je dôležité pre následnú ďalšiu analýzu a diagnostiku. Výstupom modulu Ingestor sú informácie uložené v pamäťových štruktúrach, ku ktorým sa dá následne pristupovať, napr. pri neskoršej analýze TCP popísanej v sekcii 2.5. Extrahované metadáta sa ukladajú do databázy RocksDB, ktorá je založená na princípe kľúč-hodnota. Vzhľadom na to, že sa s dátami v RocksDB pracuje nielen v module Ingestor (ktorý primárne dáta ukladá), ale aj v ďalších nadväzujúcich moduloch, bol vytvorený návrh a následne aj implementácia abstraktného rozhrania – knižnice pre prácu s dátami. Tento architektonický prvok systému umožňuje ukladať a načítať dáta v moduloch systému Distance nezávisle od konkrétnej databázovej vrstvy a verzie databázového systému.

Modul na výpočet TCP štatistík

Toto rozšírenie modulu Ingestor som vytvoril, aby bolo možné vytvárať výkonnostné štatistiky o spojení TCP. Modul, používa štruktúru `flow_cache`, ktorá obsahuje metadáta o parsovanom PCAP súbore. Štruktúra na uloženie komunikačných tokov `flow_cache` je inicializovaná modulom Ingestor, ktorý spracováva vstupné PCAP súbory. Z tejto štruktúry modul počíta následné TCP štatistiky:

Štatistiky o toku Sú to špecifické číselné informácie popisujúce charakter určitého spojenia, niektoré štatistiky sú spoločné a iné rôzne pre klienta a server. Sú to metriky vypočítané pre každý tok.

Vzorkované štatistiky Sledujú priebeh spojenia v čase a to tak, že sa celý tok vzorkuje s veľkosťou kroku napr. 100 milisekúnd a pre každú vzorku sa vypočítajú štatistiky napr.: počet paketov, počet bajtov, počet retransmisií a pod.

Pre výpočet štatistík sa používajú dva opačné navzájom odpovedajúce toky, vytvára sa tzv. biflow, kde na základe TCP príznakov je možné zistiť, či ide o klienta, alebo server.

Spoločné štatistiky pre klientský aj serverový tok sú také štatistiky, ktoré nezávisia od toho, či tok je od klienta, alebo od servera. Ku každému toku sa zisťuje počet paketov a bajtov, ktoré patria k tomuto komunikačnému toku. Ďalej sa tiež analyzuje počet retransmisií. To znamená, že sa zisťuje množstvo paketov a bajtov, ktoré museli byť znovu odoslané z dôvodu straty pri prenose. Na túto metriku nadväzuje ďalšia, ktorá určuje priemerne spomalenie pri retransmisii. Je to čas, ktorý bol potrebný na opakované zaslanie strateného segmentu. Modul vytvára aj štatistiky o počte nezachytených segmentov a nezachytených bajtov. Táto metrika sa používa na meranie kvality záchytu. Viac v sekcii 3.1.

Štatistika, ktorá sa generuje len pre klientský tok je čas pripojovania klienta na server. Počíta sa pri inicializácii TCP spojenia ako čas medzi počiatočným SYN segmentom od klienta a odpovedajúcim SYN + ACK segmentom od servera.

Pre každý serverový TCP tok, ktorý obsahuje inicializačné informácie sa počíta čas spojovania servera s klientom. Je to čas medzi inicializačným SYN + ACK segmentom zo servera a odpovedajúcim ACK segmentom od klienta. Ďalšou metrikou, ktorá sa vytvára, je čas potrebný na vytvorenie šifrovaného spojenia pomocou TLS. Čas medzi segmentom obsahujúcim client hello a segmentom s odpovedajúcim server hello. Určuje sa tiež celkový čas prenosu dát, ktorý je definovaný ako čas medzi prvým a posledným dátovým paketom odpovede zo servera. Posledná metrika, ktorú modul vytvára je obojsmerné spomalenie, alebo RTT. Je to čas od posledného paketu servera do odpovede klienta.

Analýzou biflow modul statistics získa požadované štatistiky, a potom sa výstupné dáta serializujú a exportujú do databázy RocksDB, kde sú uložené do samostatnej časti databázy tzv. `column family: statistics` ako páry kľúč, hodnota. Kľúčom je zdrojová a cieľová adresa toku a index toku, hodnotou sú samotné vypočítané štatistiky.

Kapitola 3

Analýza kvality zachytených dát

Cieľom nástroju pre analýzu kvality vstupných dát je, aby užívateľ vedel zistiť kvalitu záchytu a podľa toho posúdiť, či je vhodné pokračovať v ďalšej analýze. Je potrebné kvantifikovať kvalitu vstupných PCAP súborov prostredníctvom nejakej exaktnej metriky a pokiaľ sú vstupné dáta nekvalitné, tak upozorniť užívateľa o možných problémoch.

Vstupom analýzy sú súbory zachytávajúce komunikáciu na sieti. Na uloženie takých súborov sa používajú rôzne formáty, ale najčastejším je formát PCAP. Aby boli výsledky celkovej analýzy diagnostickým modulom vierohodné, je potrebné aby kvalita vstupných dát bola dostatočná. Pokiaľ v PCAP súbore chýbajú segmenty, alebo sú duplikované, nenadväzujú sekvenčné čísla, alebo časové značky nie sú v zostupnom poradí, potom to môže spôsobiť problémy pri analýze a výstupom analýzy budú nevalidné dáta. Preto je veľmi dôležité zistiť kvalitu záchytu hneď na vstupe v module Ingestor.

Analyzátor kvality zachytenej komunikácie je súčasťou existujúceho modulu Ingestor. Výstupom analýzy je počet poškodených tokov podľa jednotlivých služieb. Výstup sa zapisuje do databázy RocksDB, v ktorej je vyhradený priestor na ukladanie štatistík, tzv. `column family statistics`. Sieťové služby sú identifikované podľa čísel portov. Zobrazovanie zistených informácii z databázy zabezpečuje modul TCP analyzer, ktorý zisťuje, koľko tokov patriacich tejto službe prekročilo prahové hodnoty a koľko nie. Prahové hodnoty sú definované v konfiguračnom súbore. Výsledkom sú tiež vytvorené grafy časového priebehu kvality PCAP súborov. Viacej o analýze TCP v kapitole 4.

3.1 Detekcia nezachytených segmentov

Kvalitu záchytu je možné analyzovať rôznymi spôsobmi, ale najspoľahlivejším spôsobom je pravdepodobne detekcia segmentov, ktoré sa doručili, ale nestihli sa zaznamenať do záchytu. Spoľahlivosť tejto metriky na detekciu kvality záchytu spočíva v tom, že priamo vyjadruje, akú časť komunikácie sa podarilo zachytiť a ktoré pakety sa nestihli zaznamenať, čo nám ostatné metriky kvality záchytu neposkytujú. Wireshark[1] nezachytené pakety označuje výrazom *previous segment(s) not captured*. Tento problém v záchyte sa vyskytuje keď program, ktorý zaznamenáva komunikáciu na sieti nie je schopný dostatočne rýchlo spracovať prichádzajúce segmenty, ale samotná komunikujúca aplikácia je schopná tieto segmenty zachytiť, spracovať a potvrdiť odoslaním segmentu s odpovedajúcim potvrdzovacím číslom. Metrika nezachytených segmentov vypovedá o tom, že rýchlosť záchytu nebola

[1] Wireshark – program na zachytávanie a analýzu sieťovej komunikácie, www.wireshark.org

dostatočná a že program zaznamenavajúci komunikáciu bol zahltený a preto súbor PCAP ako celok môže byť poškodený.

Nezachytené pakety je možné detegovať prostredníctvom analýzy sekvenčných čísel v protokole TCP. Implementácia protokolu TCP a jeho robustnosť a spoľahlivosť nám umožňuje, aby sme dokázali vystopovať, ktoré pakety v skutočnosti pri komunikácii boli prítomné, ale v záchyte nie sú. Pri normálnej komunikácii za sebou idúce segmenty nadväzujú, sekvenčné číslo každého nasledujúceho segmentu je inkrementované o veľkosť aplikačných dát. To znamená, že zo súčasného paketu je možné veľmi jednoducho vypočítať sekvenčné číslo nasledujúceho paketu. Pokiaľ nikde v záchyte nie je segment s týmto sekvenčným číslom, tak to znamená, že segment nebol zachytený. Inak povedané, ku každému segmentu v záchyte, okrem segmentov na začiatku toku, musí existovať odpovedajúci segment, ktorého súčet sekvenčného čísla a veľkosti aplikačných dát je rovnaký ako sekvenčné číslo súčasného segmentu.

Nezachytený paket a retransmisia pri prenose pomocou protokolu TCP sú dve rozdielne veci. Pri prenose nastávajú straty paketov bežne, ale treba si uvedomiť, že strata paketu pri prenose je rozdiel od straty paketu pri zápise sieťovej komunikácie do súboru PCAP. Keď sa paket stratí pri prenose, tak stanica nedostane segment a preto ho nepotvrdí. Odosielateľ po istom čase zistí, že konkrétny paket, alebo skupina paketov nebola potvrdená a prepošle paket, prípadne viaceré pakety znovu a dochádza k tzv. retransmisii. To znamená, že rovnaký paket s rovnakým sekvenčným číslom sa prepošle znovu. Na rozdiel od straty pri prenose a následnej retransmisii, chyba zápisu paketu do PCAP súboru sa na sieti neprejaví, lebo v skutočnosti ku žiadnej strate na sieti nedošlo a komunikujúce strany dostali informácie, ktoré potrebovali a nedochádza k žiadnej retransmisii, chyba je len v záchyte. Pri analýze kvality PCAP súboru je teda možné rozlíšiť, či sa jedná o retransmisiu, alebo chybu záchytu a to tak, že pri retransmisii sa segment s konkrétnym sekvenčným číslom v záchyte nachádza, je len posunutý neskôr, prípadne sa v komunikácii nachádza viackrát. Pokiaľ však ide o chybu záchytu, tak sa v záchyte segment s odpovedajúcim sekvenčným číslom v záchyte nenachádza.

3.2 Implementácia

Implementácia je založená na dátovej štruktúre vytvorenej Ingestorom pri parsovaní súboru PCAP. Ide o dátovú štruktúru `flow_cache`, ktorá obsahuje všetky potrebné informácie o komunikačných tokoch. Algoritmus na detekciu nezachytených paketov nie je jednoduchý a to najmä z dôvodu, že je veľmi neefektívne pri každom jednom segmente prehľadávať celý PCAP súbor a zisťovať, či sa v ňom naozaj nenachádza segment s odpovedajúcim sekvenčným číslom. PCAP súbory môžu byť veľmi veľké, môžu mať niekoľko stoviek megabajtov, alebo aj gigabajty. Preto je potrebné vymyslieť algoritmus, ktorý by s minimálnou pamäťovou a výpočtovou zložitosťou vyhľadával medzery v postupnosti sekvenčných čísel.

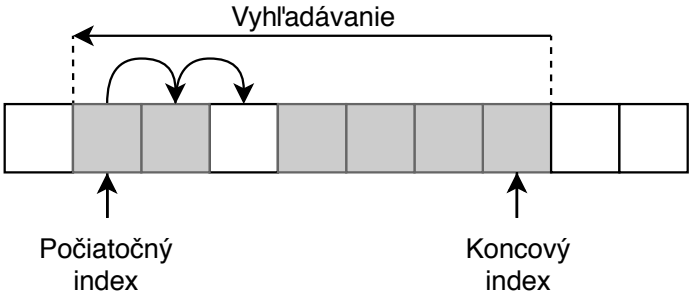
Kvôli týmto praktickým obmedzeniam som prehľadávanie PCAP súboru zúžil na lokálne okolie analyzovaného paketu. Preto som vytvoril štruktúru cyklického bufferu, ktorý bude ukladať očakávané relatívne sekvenčné čísla. Po inicializácii cyklický buffer obsahuje jedno sekvenčné číslo s nulovou hodnotou, čo reprezentuje hodnotu prvého relatívneho sekvenčného čísla v toku. Pri analýze nezachytených paketov dochádza k iterácii cez všetky toky v záchyte. Každý tok sa spracováva postupne od prvého paketu až do posledného.

Detekcia zachytenia predchádzajúceho segmentu prebieha nasledujúcim spôsobom. Pri analýze každého segmentu sa najprv vyhľadáva sekvenčné číslo analyzovaného paketu v cyklickom bufferi sekvenčných čísel. Pokiaľ sa nájde, tak to znamená, že predchádzajúci seg-

ment bol zachytený a nájdené nasledujúce sekvenčné číslo sa z bufferu odstráni. Pokiaľ sa sekvenčné číslo nenašlo v bufferi, tak to znamená, že predchádzajúci segment nebol zachytený. Do cyklického bufferu sa potom pridá ďalšie očakávané sekvenčné číslo, ktoré sa vypočíta ako súčet sekvenčného čísla analyzovaného paketu a veľkosti aplikačných dát. Je to sekvenčné číslo, ktoré patrí nasledujúcemu paketu.

Buffer sa používa z toho dôvodu, že segmenty v záchyte môžu byť poprehadzované a môže sa stať, že existuje viacero sekvenčných čísel čakajúcich na potvrdenie. Maximálna veľkosť bufferu je na začiatku definovaná a je mu staticky alokované príslušné množstvo pamäti. Aby sa pri vyhľadávaní nemuselo prechádzať celým poľom, sú definované dva indexy. Počiatočný a koncový index ohraničuje použitú časť poľa, ktorá sa dynamicky prispôsobuje počtu sekvenčných čísel, ktoré sú v ňom uložené. Táto funkcia je veľmi dôležitá, aby sa skrátil vyhľadávací čas. Pokiaľ sa buffer naplní a už nie je kde ukladať ďalšie očakávané sekvenčné čísla, tak sa začnú prepisovať najstaršie položky. Vyhľadávanie prebieha od vyšších indexov k nižším, je to preto, aby sa skorej našlo odpovedajúce sekvenčné číslo. Na vyšších pozíciach v pamäti sa nachádzajú novšie položky, ktoré obvykle odpovedajú hľadanému číslu, lebo najčastejšie sú pakety správne zoradené podľa sekvenčných čísel.

Maximálna veľkosť bufferu je obmedzená, ale je dôležité aby sme poznali, ktoré položky sú aktuálne využívané. Na to slúžia dva indexy, ktoré udržujú pozíciu prvej a poslednej dátovej položky. Pokiaľ položky pribúdajú, tak sa pozícia poslednej položky v poli posúva. Keď je buffer preplnený, tak sa začne posúvať aj index prvej dátovej položky a dochádza k prepisu starých dátových položiek novými.



Obr. 3.1: Vyhľadávanie v cyklickom bufferi sekvenčných čísel a zarovnanie a uvoľnenie nepoužitých položiek. Použité položky označené tmavou farbou sa zarovnávajú na pravú stranu vyrovnávacej pamäti.

Keď sa nejaká položka uvoľní, tak dôjde k jej zaslepeniu a pri ďalšom vyhľadávaní k zarovnaniu ostatných namiesto vzniknutej diery a buffer sa zmenší zvýšením prvého indexu, ako to je zobrazené na obrázku 3.1. Zarovnávanie pri každom vyhľadávaní urýchľuje vyhľadávanie, pretože sa validné položky prehľadávajú skôr. Zarovnávanie umožní zmenšiť buffer, pri prehľadávaní sa potom iteruje cez menej položiek.

3.3 Spustenie analýzy

Analýzu kvality PCAP súborov je možné vytvoriť spustením programu Ingestor, ktorého vstupom sú vzorky sieťovej komunikácie a výstupom je databáza v zložke definovanej užívateľom. Výstup bude obsahovať štatistiky ku každému komunikačnému toku a to časové

metriky a priebeh komunikácie. Kvalita zachytených dát je len časť analýzy, ktorú Ingestor vytvára.

```
./ingestor -d output-database-folder PCAP-1 PCAP-2 ...
```

Ingestor má nasledujúce funkcie: Extrahuje metadáta odsadení hlavičiek z paketu. Vytvára agregované komunikačné toky, ktoré pozostávajú zo segmentov komunikácie určitej aplikácie. Ďalšou funkciou modulu Ingestor je identifikácia aplikačných protokolov v aplikáciach pracujúcich nad UDP alebo TCP prenosovými protokolmi. Potom modul separuje toky použitím časového limitu, alebo ukončením toku. To znamená, že Ingestor po určitom čase komunikačnej neaktivity v rámci toku zaradí tok medzi ukončené. Ďalšou možnosťou ukončenia je pomocou ukončovacieho TCP príznaku FIN. Ingestor tiež spravuje tunelované pakety a na záver vytvára a exportuje TCP štatistiky.

Kapitola 4

Diagnostika TCP komunikácie

Táto kapitola sa zaoberá popisom diagnostického modulu protokolu TCP. Zaoberám sa v nej požiadavkami na analýzu TCP, návrhom, implementáciou a použitím nástroja. Analýza TCP je problém, ktorého cieľom je vykonať diagnostiku na TCP vrstve a odhaliť čo najviac potenciálnych problémov. Tieto problémy sa dajú detegovať za pomoci údajov z TCP hlavičiek, ako sú napríklad zdrojovej a cieľovej IP adresy a porty, časové značky, príznaky TCP a veľkosť dát paketov. Príkladom problémov môžu byť napríklad: dlhá doba nadviazanie spojenia, veľký počet retransmisií, prerušené spojenia, alebo kolísavá priepustnosť. Diagnostický modul spracováva informácie z databázy a vytvára rôzne typy štatistík, ktoré sú v zrozumiteľnej podobe zobrazované užívateľovi.

4.1 Požiadavky

Cieľom diagnostiky je analyzovať databázu vytvorenú Ingestorom a vypísať jej obsah a zistené štatistiky užívateľovi. Dôležité je tiež, aby užívateľ mohol definovať, ktorá časť komunikácie ho zaujíma a ktoré služby chce analyzovať pomocou definovania filtrovacích pravidiel a konfiguračného súboru. Užívateľ definuje aplikačné porty, ktoré ho zaujímajú a z databázy sa vyberú len tie komunikačné toky, ktoré vyhovujú zadaným podmienkam.

Filtrovanie štatistík Nástroj by mal umožňovať filtrovanie štatistík podľa toho, do ktorej časti databázy patria. Ďalej bude možné definovať filter podľa IP adresy, alebo portu. Posledným typom konfigurácie je podľa hodnôt vytvorených štatistík. Táto možnosť bude určená len pre `column family` statistics. Je potrebné vytvoriť konfiguračný súbor, ktorý bude obsahovať filtračné informácie a prahové hodnoty filtrovaných metrík. Tento konfiguračný súbor sa pred samotnou analýzou lexikálne a syntakticky spracuje. Analyzátor TCP bude informácie z každého toku porovnávať s prahovými hodnotami a vypíše len tie toky, ktoré vyhovovali podmienke.

Agregované štatistiky Analytický nástroj by mal vytvárať agregované štatistiky a to tak, že bude agregovať štatistiky získané z jednotlivých tokov. Cieľom agregácie je, aby sme zistili kvalitu analyzovaných dát a počty nezachytených segmentov v špecifikovanej sieťovej komunikácii. Agregácia nebude prebiehať na celom PCAP súbore, ale len na tokoch definovaných pomocou filtru, ktorý som spomínal v sekcii 4.1. Výhodou tohoto riešenia je, že nemusíme analyzovať komunikačné toky, ktoré nás nezaujímajú. Agregované štatistiky

budú zobrazovať kvalitu komunikácie podľa jednotlivých sieťových služieb, ktoré pracujú nad protokolom TCP.

4.2 Popis diagnostiky

Diagnostika prebieha vo viacerých krokoch. Najprv sa zistí konfigurácia, ktorá definuje, ktoré toky a ktoré metriky nás zaujímajú. Konfigurácia sa zisťuje na základe lexikálnej a syntaktickej analýzy konfiguračného súboru, ktorý definoval užívateľ. Analýza prebieha pomocou nástrojov Flex a Bison. Flex je nástroj na vytvorenie lexikálneho analyzátoru. Lexikálny analizátor určuje, či slovo tzv. token patrí do definovaného jazyka [17]. Bison je nástroj, ktorý podľa definície gramatiky určitého jazyka vygeneruje syntaktický analyzátor. Analyzátor sa potom použije na zistenie, či určitý súbor spĺňa definovanú gramatiku [5]. Vstupný konfiguračný súbor obsahuje logické výrazy popisujúce metriky a toky, ktoré nás zaujímajú. Pomocou nástroju Bison sa vytvorí abstraktný syntaktický strom obsahujúci logický výraz, ktorý musia toky spĺňať. Toky, ktorý tento výraz nespĺňajú nie sú vypísané a ani analyzované.

V ďalšom kroku sa podľa argumentu z príkazového riadku zistí, kde sa nachádza databáza. Potom sa z databázy postupne vyberajú položky. Každá položka z databázy reprezentuje jeden tok, ktorý je najprv potrebné deserializovať a vytvoriť komplexné dátové štruktúry. Následne sa podľa adresy toku vyhľadáva odpovedajúci opačný tok a vytvára sa tzv. *biflow*.

Po tom, ako sú inicializované všetky potrebné štruktúry, je možné zistiť, či tento tok spĺňa konfiguráciu. Porovnanie s prahovými hodnotami z konfiguračného súboru prebieha ako rekurzívny prechod abstraktným syntaktickým stromom. Tok sa vypíše iba pokiaľ vyhovuje definovanému logickému výrazu.

Pokiaľ tok vyhovoval konfiguračnému výrazu, tak sa použije na vytvorenie časového priebehu sieťovej komunikácie. Vo výstupnej zložke sa vytvoria dočasné súbory, ktoré sa budú plniť hodnotami o časovom priebehu spojenia. Z týchto súborov sa neskôr prostredníctvom programu gnuplot[1] vytvoria grafy časového priebehu komunikácie. Príkladom môže byť počet retransmisií v čase, priepustnosť v čase, či veľkosť paketov. Analýza potom pokračuje načítaním ďalšej položky z databázy.

4.3 Použitie

Analýza sa spúšťa následovne:

```
./tcp_analyser --db /source/database/folder --out_dir /path/to/out/dir
```

Povinné argumenty sú `--db` a `--out_dir`. Argument `--db` určuje cestu k existujúcej databáze vytvorenej Ingestorom. Argument `--out_dir` určuje cestu k existujúcej zložke, kde budú vytvorené pomocné súbory, z ktorých sa neskôr budú pomocou nástroja gnuplot generovať grafy časového priebehu komunikácie.

Aby sme mohli spustiť analýzu TCP, potrebujeme pripravenú databázu, ktorá sa dá vytvoriť spustením modulu Ingestor, ako som spomínal v časti 3.3. Keď už máme vytvorenú databázu, tak ju môžeme použiť a vypísať jej obsah.

[1] Gnuplot je nástroj príkazového riadku, ktorý umožňuje vytvárať 2D, alebo 3D grafy na základe kolekcie dát, http://www.gnuplot.info/.

```
mkdir out_dir
./tcp_analyser --db /tmp/RocksDB --out_dir ./out_dir
cd out_dir
gnuplot gnuplot.plt
```

V prvom kroku vytvoríme prázdnu zložku, potom do nej vložíme dočasné súbory na vytvorenie grafov a na záver zavolaním programu gnuplot vytvoríme grafy vo formáte PDF.

Filtrácia

TCP analyzátor umožňuje filtrovať komunikačné toky podľa zdrojovej a cieľovej IP adresy a zdrojového a cieľového portu. Špecifikovanie filtru podľa adries sa vykonáva prostredníctvom parameterov na príkazovom riadku pri spustení analyzátoru. Je možné zadať viacero parametrov príkazového riadku na filtrovanie tokov podľa adresy, filter sa potom vytvorí tak, že musia platiť viaceré podmienky súčasne.

Tabuľka 4.1: Parametre príkazového riadku pre TCP analyzátor, ktoré slúžia na filtrovanie tokov podľa IP adries a portov. Za parametrom je možné špecifikovať viacero položiek, pri vyhodnotení potom stačí, aby platila jedna položka z definovaného zoznamu.

port	zdrojový, alebo cieľový port
srcport	zdrojový port
dstport	cieľový port
ip	zdrojová, alebo cieľová IP adresa
srcip	zdrojová IP adresa
dstip	cieľová IP adresa

Položky v zozname za parametrom sú spájané logickou operáciou *alebo*. Pokiaľ by sme chceli filtrovať toky len s portom 443, alebo 1220, tak by spustenie analýzy TCP vyzeralo následovne:

```
./tcp_analyser --db /tmp/RocksDB --out_dir ./out_dir --srcport 443 1220
```

Pri kombinácii viacerých parametrov na filtrovanie podľa adresy sa výrazy parametrov spájajú logickou operáciou *a súčasne*. Filtre podľa IP adries a portov je možné kombinovať. Napríklad pokiaľ by sme chceli definovať zdrojovú IP adresu 40.77.226.250 a súčasne port 80, tak by spustenie analyzátoru vyzeralo takto:

```
./tcp_analyser --db /tmp/RocksDB --out_dir ./out_dir
--srcip 40.77.226.250 --port 80
```

Konfiguračný súbor

Konfiguračný súbor určuje, ktoré toky budú spracované a ktoré nie. To znamená, že budú ovplyvnené aj štatistiky pre grafy a časové priebehy vo výstupnom adresári. Ak len jeden tok zodpovedá podmienkam z konfiguračného súboru, potom budú grafy vytvorené iba z tohto jedného toku. Jednotky prahových hodnôt v konfiguračnom súbore sú rovnaké ako vo výpise TCP analyzátoru.

Konfiguračný súbor umožňuje užívateľovi filtrovať pomocou vypočítaných metrík zo štatistických údajov. Cesta ku konfiguračnému súboru sa definuje pri spustení analyzátoru TCP. Pokiaľ nie je konfiguračný súbor zadaný, tak nedochádza k filtrovaniu štatistík

podľa jednotlivých metrík. Názvy tokenov metrík, napr. `client_connection_setup_time` sú rovnaké ako názvy metrík vo výstupe analyzátoru TCP v `column family statistics`.

Konfiguračný súbor pozostáva zo základných výrazov napr. `some_metric < 42.42` spojených prostredníctvom logických operátorov `AND, OR, NOT` a tiež je možné použiť zátvorky na vyznačenie priorít. Vo výrazoch sa nerozlišujú veľké a malé písmená. Je možné použiť podmienku `some_metric = N/A`, to znamená, že táto hodnota nie je známa. Skener dokáže pochopiť, rôzne varianty neznámeho operátora namiesto `N/A` môžete použiť napr. `n/a, null, NULL, none, ...` a tiež rôzne variácie s (malými/veľkými) písmenami. Operátor `>` a `>=` sú vyhodnotené rovnako a to ako `>=`, čiže väčšie alebo rovné. Dôvodom je, že hodnoty sú uložené s použitím dátového typu float. To isté platí pre operátory `<` a `<=`. Keď sa používa operátor `=`, potom sa rovnosť aproximuje pomocou prahovej hodnoty `FLOAT_CMP_APPROX` s hodnotou 0.000001. Riadkové komentáre v konfiguračnom súbore je možné vyznačiť prostredníctvom znaku `#`.

V konfiguračnom súbore sú dva typy metrík, buď sú metriky spoločné pre klientský a serverový tok, alebo sú rozdielne. Príkladom spoločných metrík sú počty stratených segmentov, alebo nezachytených segmentov. Tieto metriky sú vyhodnocované samostatne pre tok klienta a server, pretože oba toky obsahujú tieto údaje. Druhý prípad sú unikátne metriky pre tok klienta a server. Ide väčšinou o časové metriky. Vyhodnocujú sa spoločne pre klienta a server, pretože len klient alebo len server obsahuje potrebné údaje. Analýza prebieha pre biflow ako celok.

Spracujú sa len toky, ktoré sa zhodovali s výrazom z konfiguračného súboru a formát analyzovaného toku určujú metriky, ktoré vyhovovali pri vyhodnocovaní výrazu z konfiguračného súboru. Pre každý tok sa potom zobrazia len tie metriky, ktoré prekračovali konkrétnu prahovú hodnotu.

Príklad konfiguračného súboru

```
(
   client_connection_setup_time < 200.1  OR
   server_connection_setup_time >= 0.3
)
AND tls_setup_time = N/A
AND NOT line_loss_bytes_abs > 100    # in bytes
```

Príklad zobrazuje konfiguračný súbor analýzy TCP. Pár zátvoriek slúži na spojenie viacerých výrazov vo vnútri zátvoriek do jedného celku. Najprv sa vyhodnocuje obsah zátvorky a potom sa pokračuje vo vyhodnocovaní ostatných výrazov. Konfiguračný súbor na druhom riadku definuje, že program má zobrazovať toky, u ktorých bol čas pripojovania na klienta menší, ako 200 milisekúnd, alebo čas pripojovania na server je väčší, ako 0.3 milisekundy. Za koncom zátvorky je definované že hľadáme toky, ktoré nemajú určený čas na vytvorenie šifrovanej komunikácie. Táto metrika nie je určená pravdepodobne preto, že komunikácia nieje šifrovaná. Na poslednom riadku je použitý príklad negácie `NOT`, výraz určuje že stratovosť linky v rámci toku má byť menšia, ako 100 bajtov. Úplne na konci je príklad riadkového komentáru, všetko za znakom `#` bude pri vyhodnocovaní výrazu ignorované.

Kapitola 5

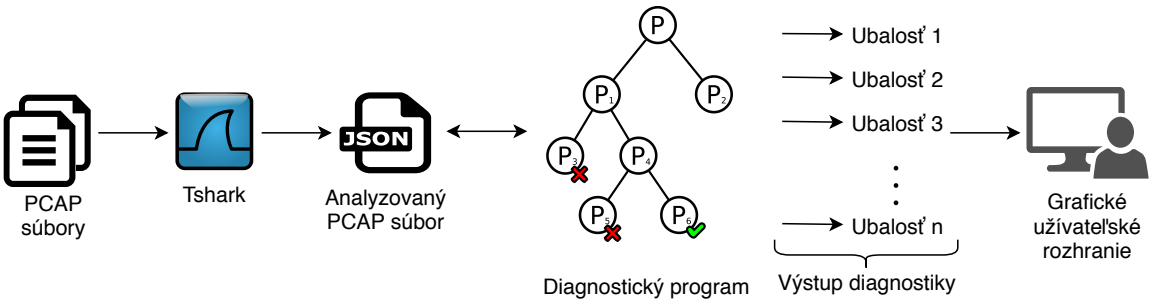
Diagnostika DHCP

Analýza DHCP sa zaoberá detekciou stavu spojenia prostredníctvom skúmania zachytenej komunikácie. Pre analýzu sa používa diagnostický modul implementovaný v systéme Distance. Cieľom je, aby užívateľ nemusel manuálne prehľadávať PCAP súbory vo Wiresharku a manuálne ich filtrovať, ale aby spustil diagnostiku často používaných protokolov a potom by podľa prehľadu udalostí videl chyby, ktoré v komunikácii nastali. Nástroj bude prípadne ponúkať aj riešenia problémov, ktoré nastali. Dôležité je uľahčiť základnú diagnostiku siete menej skúsenému používateľovi.

5.1 Diagnostický modul

Diagnostický modul je súčasťou systému Distance, je to jadro celej diagnostiky. Diagnostický modul na základe popisu rozhodovacieho stromu vytvorí program v jazyku Python, ktorý následne po spustení vykoná samotnú diagnostiku. Tento modul vo svojej práci používam na spracovanie diagnostiky protokolu DHCP. Cieľom diagnostiky je vytvoriť postupnosť významných udalostí, ktoré nastali pri komunikácii.

Priebeh diagnostiky je zobrazený na obrázku 5.1. Vstupom diagnostického modulu je PCAP súbor, ktorý je spracovaný externým nástrojom Tshark, ktorého výstupom je súbor formátu JSON obsahujúci základné diagnostické informácie získané programom Tshark. Diagnostický modul na základe prechodu diagnostickým stromom generuje protokol udalostí tiež vo formáte JSON, ktorý je určený pre vizualizáciu.



Obr. 5.1: Architektúra diagnostiky aplikačných protokolov v systéme Distance.

Definícia diagnostiky konkrétneho protokolu je rozdelená do troch súborov formátu YAML. YAML je užívateľský prívetivý serializačný formát určený pre všetky programovacie jazyky [16]:

events.yaml Obsahuje definíciu a popis udalostí, ktoré môžu pri komunikácii nastať. Pri definícii udalosti je potrebné definovať: meno, popis, závažnosť problému, popis odkazovaných polí paketu. Zoznam udalostí tvorí výstup analyzátoru.

facts.yaml Obsahuje pravidlá, ktoré určujú, či bola splnená určitá podmienka. Na tieto pravidlá sa potom odkazuje rozhodovací strom. Napríklad zistenie, či k určitému paketu existuje odpovedajúci paket. Cieľom je vyhľadať v paketoch z PCAP súboru špecifické fakty. Výsledkom dotazu je zoznam usporiadaných n-tíc paketov, ktoré spĺňali definované podmienky. Napríklad pokiaľ by sme vyhľadávali pokus o prihlásenie, tak výsledkom bude niekoľko dvojíc. Prvý paket bude obsahovať prihlasovacie parametre a druhý výsledok prihlasovania.

tree.yaml Obsahuje definíciu rozhodovacieho stromu, ktorá sa skladá z pravidiel. Každé pravidlo sa môže odkazovať na najviac dve ďalšie pravidlá. Na jedno pravidlo sa prejde v prípade úspechu a na druhé v prípade neúspechu podmienky zo súboru facts.yaml. Cieľom je previesť rozhodovací strom na podobu zrozumiteľnú pre počítač. Tento súbor definuje, do ktorého stavu sa prejde na základe výsledku vyhodnotenej podmienky. Prechodová hrana môže okrem definície nasledujúceho stavu obsahovať aj kód v jazyku Python, ktorý sa vykoná pri prechode. Najdôležitejšou súčasťou prechodu je generovanie diagnostického výstupu vo formáte JSON pomocou funkcie event.

Základom diagnostiky v systéme Distance je diagnostický strom, ktorý bude neskôr prepísaný do diagnostických pravidiel pre jednotlivé uzly. Najprv treba zistiť, v ktorých bodoch môže komunikácia začínať a podľa toho zvoliť počiatočný stav. Počiatočný stav môže byť implicitný, alebo explicitný. Implicitný znamená, že diagnostika je automaticky spustená v počiatočnom stave. Explicitný znamená, že diagnostika jedného protokolu môže odkazovať na diagnostiku iného protokolu v ktoromkoľvek jeho stave. Diagnostika protokolu môže byť ukončená ukončovacím uzlom, alebo prechodom na diagnostiku iného protokolu.

Na základe štúdia protokolu DHCP budem zostavovať zoznam možných chýb v protokole. Napríklad server neodpovedá, alebo požadovaná adresa je už použitá. Nájdené chyby komunikácie budú použité neskôr pri implementácii rozhodovacích pravidiel.

V ďalšom kroku bude potrebné na základe stavového diagramu DHCP vytvoriť diagnostický strom. To znamená, že bude potrebný prepis pravidiel, ktoré budú určovať, či daná podmienka je pravdivá, alebo nepravdivá. Na základe vyhodnotenia podmienky sa prejde do ďalšieho stavu. Pokiaľ sa neočakáva prechod po spracovaní vetvy, tak stav neobsahuje odkaz na ďalší stav a pokiaľ sa očakáva prechod do ďalšieho protokolu, tak odkaz na nasledujúci stav je nasledovný `meno_protokolu/meno_stavu`.

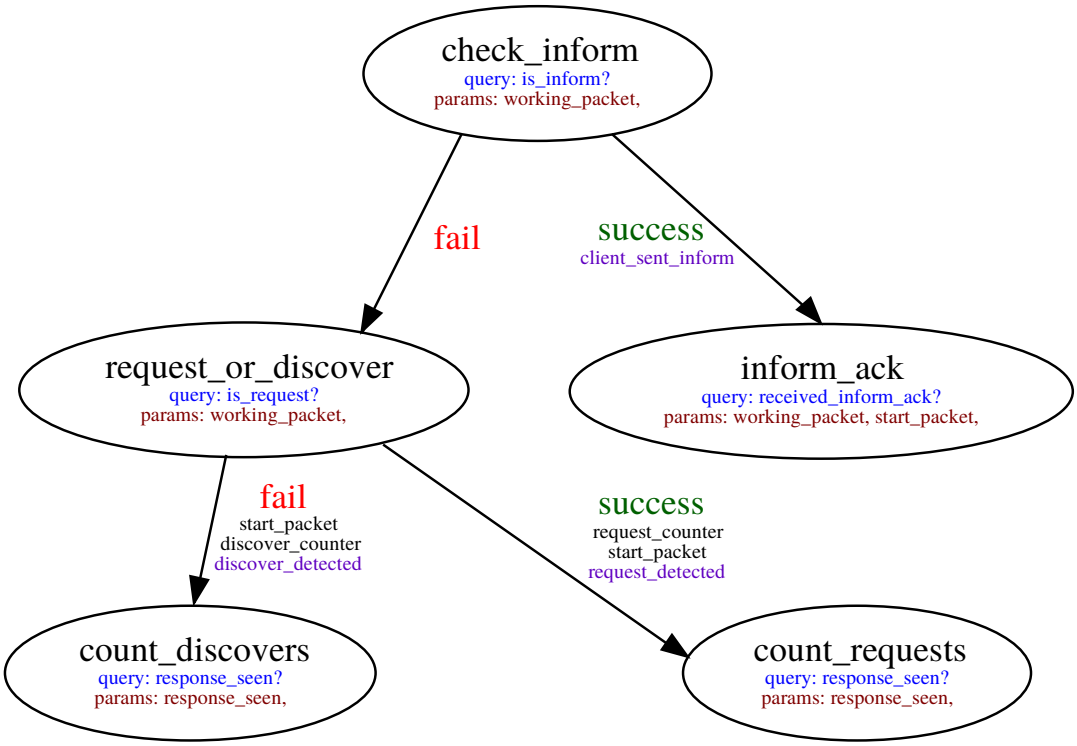
Prevod získaných informácií do rozhodovacích pravidiel Najprv treba vytvoriť rozhodovacie podmienky v súbore facts.yaml. Na začiatok vytvoriť podmienku na vyhľadanie počiatočného stavu, táto podmienka bude vyhľadávať pakety, ktoré sa môžu vyskytnúť na začiatku komunikácie. Až po vytvorení prechodovej podmienky môžme definovať uzol rozhodovacieho stromu. V uzle sa môže vykonať určitý kód a pokiaľ je definovaný nasledujúci stav, tak nastane prechod. Funkciou event, ktorá sa odkazuje do súboru events.yaml generujeme výstup programu.

5.2 Vizualizácia diagnostického stromu

Aby užívateľ mohol prehľadne vidieť, ako rozhodovací strom vyzerá a nemusel zložito vyhľadávať vo viacerých súboroch, tak som vytvoril skript v jazyku Python, ktorý umožňuje zobraziť štruktúru diagnostického stromu vo formáte PDF. Na implementáciu som použil knižnicu graphviz[1].

```
./graph_plotter.py [protocol-name] [pdf-file-name]
```

Na vygenerovanie grafickej reprezentácie stromu je potrebné zavolať skript graph_plotter, prvý parameter špecifikuje názov protokolu, ktorý chceme zobraziť a druhý parameter určuje cestu a názov vytvoreného PDF súboru obsahujúceho vizualizáciu.



Obr. 5.2: Časť vizualizácie diagnostického stromu DHCP. Každý stav obsahuje rozhodovaciu podmienku zakreslenú modrou farbou a parametre, na ktorých rozhodovacia podmienka závisí. Prechody obsahujú parametre, ktoré sa pri prechode zmenili zakreslené čiernou farbou. Fialovou farbou sú zobrazené udalosti, ktoré sa pri prechode generujú.

Na testovanie diagnostiky a jej funkcií je potrebné spustiť diagnostický skript napísaný v jazyku Python. Parametre skriptu sú protokol, ktorý má byť analyzovaný a vstupné PCAP súbory. Výsledkom diagnostiky sú vygenerované udalosti vo formáte JSON, ktoré sa vypíšu na štandardný výstup. Pokiaľ výstup presmerujeme do súboru, tak neskôr budeme môcť použiť jednoduchú webovú aplikáciu na prehľadné zobrazenie udalostí. Táto aplikácia zobrazuje výstup diagnostiky tak, aby bol ľahšie pochopiteľný pre užívateľa. Aplikácia graficky zobrazí, kde nastala chyba a po kliknutí vypíše podrobnosti o pakete.

[1]Graphwiz je knižnica umožňujúca vytváranie reprezentácie grafu v jazyku DOT a generovanie výsledného grafu vo formáte PDF https://pypi.org/project/graphviz/

Kapitola 6

Vytvorenie vzoriek sieťovej komunikácie

Vzorky sieťovej komunikácie som vytváral, aby som mohol overiť funkčnosť implementovaného algoritmu a zistiť, či pracuje podľa špecifikácie. Cieľom bolo vytvoriť sadu testovacích vzoriek, ktoré by demonštrovali rôzne komunikačné a konfiguračné problémy. V prvej časti je uvedená definícia rôznych sieťových problémov a sú tu podrobne rozpísané charakteristiky ako sa prejavujú tieto problémy v komunikácii prostredníctvom protokolu TCP. Základný pohľad na problém vytvárania vzoriek demonštrujúcich problémy na sieti je poskytnutý v nasledujúcich podkapitolách.

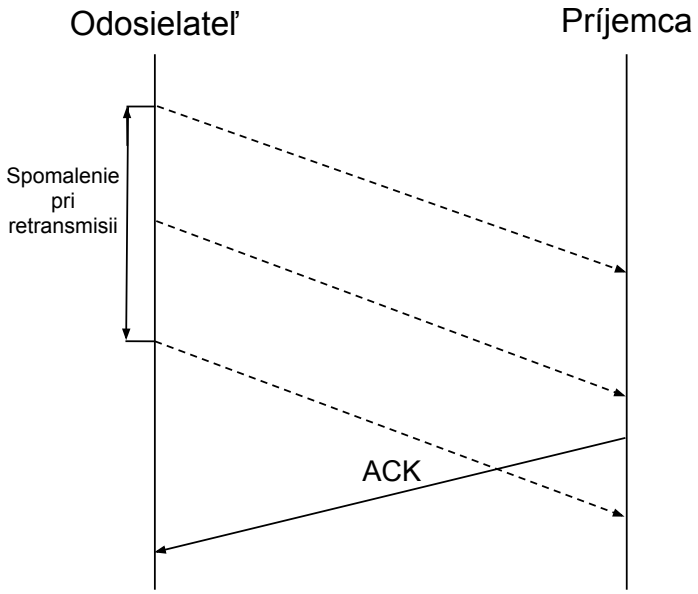
6.1 Vzorky TCP komunikácie

Demonštrujú problémy, ktoré sa prejavujú na transportnej vrstve. Vzorky boli vytvorené vo formáte PCAP. Každý súbor PCAP obsahuje väčšinou jeden hlavný problém, ktorý musí byť odhalený diagnostickým nástrojom. Boli vytvorené krátke súbory PCAP s malým množstvom paketov, aby sa v nich dalo jednoducho orientovať a aby bolo jasné, o aký problém ide. Vytvoril som vzorky sieťovej prevádzky pre nasledujúce problémy:

Nízka priepustnosť spojenia Sieťová priepustnosť je miera úspešného doručovania správ cez komunikačný kanál. Správy môžu byť doručené prostredníctvom fyzického alebo logického prepojenia, alebo môžu prechádzať určitým sieťovým uzlom. Priepustnosť sa zvyčajne meria v bitoch za sekundu (bit/s alebo bps) a niekedy aj v paketoch za sekundu (p/s alebo pps) [14]. TCP poskytuje mechanizmus riadenia preťaženia, takže zdroj reguluje svoju prenosovú rýchlosť, tak aby nepreťažoval sieť [10]. Z toho vyplýva, že ak je počet úspešne prenesených paketov nízky, tak to môže indikovať preťaženie niektorého sieťového prvku.

Vysoká stratovosť paketov Prejavuje sa veľkým počtom retransmisií, jedna z komunikujúcich strán posiela paket znova z dôvodu, že jej neprišiel potvrdzovací paket. V bežnej komunikácii pomocou TCP si odosielateľ nastaví časovač pre každý jeden paket, pokiaľ mu tento časovač vyprší skôr, než dorazí paket, ktorý ho potvrdzuje, tak odosielateľ usúdi, že sa paket stratil počas prenosu. Preto odošle nepotvrdený paket znovu a dochádza k retransmisii. Retransmisie sú bežnou súčasťou sieťovej komunikácie, napríklad rušenie na fyzickej vrstve môže spôsobiť zmenu niektorých bitov v pakete. Ak je táto chyba odhalená, tak

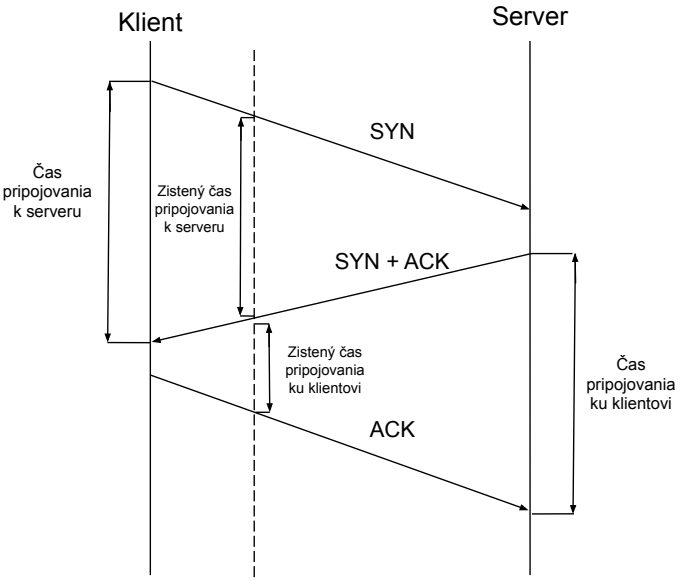
sa paket zahodí, časovač vyprší a odosielateľ odošle paket znova. Ďalším dôvodom vzniku retransmisie môže byť zahltenie siete, kedy sú pakety odoslané do siete rôznymi užívateľmi naraz vo veľkom objeme a sieť nedokáže tak veľký objem dát preniesť. Smerovače si pakety, ktoré čakajú na odoslanie ukladajú do fronty a keď sa sieť uvoľní, tak sú pakety z fronty odoslané a odstránené. Niekedy sa ale môže stať, že je sieť dlhodobo preťažená a pretože buffer smerovača má obmedzenú veľkosť, tak sa po zaplnení začnú novoprichádzajúce pakety zahadzovať.



Obr. 6.1: Čas potrebný na opätovné preposlanie TCP segmentu. K retransmisii môže dôjsť aj viackrát za sebou, lebo odosielateľ neobdržal potvrdenie. Prvé dva pakety boli pri prenose stratené a až potom poslal príjemca potvrdenie.

Veľké oneskorenie pri retransmisii (retransmission delay) Keď sa stratí paket a odošle sa znova, dochádza k retransmisii, ale k strate môže dôjsť znovu a paket sa musí preposlať znovu. Oneskorenie medzi retransmisiami je teda čas medzi prvým a posledným preposlaním konkrétneho paketu z dôvodu, že nebol potvrdený v dostatočnom časovom limite. Na obrázku 6.1 môžete vidieť znázornenie oneskorenia pri retransmisii, čiarkované šípky znázorňujú paket, ktorý je posielaný do doby, kým nie je zaznamenaný paket, ktorý ho potvrdzuje. Na obrázku je vidieť, že potvrdzovací paket môže byť odoslaný skôr, než je obdržaný posledný paket retransmisie.

Veľké obojsmerné oneskorenie (round trip time, RTT) Je to čas od odoslania paketu do obdržania zodpovedajúceho potvrdzovacieho paketu od druhého komunikanta. Túto hodnotu je prakticky možné spočítať v ktorejkoľvek časti komunikácie, ale naša implementácia momentálne počíta len jednu hodnotu RTT v rámci spojenia. V našom prípade sa RTT počíta ako čas od posledného odoslaného dátového paketu servera po potvrdzovací paket od klienta.



Obr. 6.2: Čas pripojovania klienta a servera pri zahájení spojenia.

Dlhý čas pripojenia medzi klientom a serverom Ako môžeme vidieť na obrázku 6.2, tak pripojenia k serveru (server connection setup time) je možné zistiť ako rozdiel medzi časom odoslania paketu, ktorý začína spojenie, tzn. obsahuje príznak SYN a odpovede zo strany servera, ktorá obsahuje príznaky SYN a ACK. Čas pripojenia ku klientovi (server connection setup time) je čas medzi odoslaním paketu s príznakmi SYN a ACK a prijatím potvrdzovacieho paketu od klienta. Prerušovaná čiara nám znázorňuje, že v skutočnosti máme len jeden záchyt zo siete a tak sú tieto metriky skreslené, pretože nepoznáme presný čas odoslania alebo prijatia paketu. Záchyt môže byť vytvorený niekde medzi klientom a serverom v ktoromkoľvek uzle siete.

Pomalá reakcia servera, (server response time) Je to čas medzi posledným paketom požiadavky od klienta do prvého paketu odpovede. Táto metrika môže indikovať zaneprázdnenosť servera, dôvodom môže byť že server musí spracovať príliš veľké množstvo požiadaviek naraz.

Požiadavka bez odpovede Táto chyba môže nastať, keď klient posiela pakety s príznakom SYN a nedostáva žiadnu odpoveď, Napríklad vďaka DROP pravidlu na firewallu.

Odmietnuté spojenie Je to taký prípad komunikácie, keď klient sa snaží pripojiť k serveru prostredníctvom odosielaných paketov SYN, ale server toto spojenie uzavrie prostredníctvom paketu s nastaveným príznakom RST.

Polovične otvorené spojenia, (half opened connection) Spojenie bolo nadviazané, ale jedna strana prestala úplne odpovedať. Pravdepodobne došlo k úplnému výpadku spojenia.

Počet nezachytených paketov Sú to pakety, ktoré sa v komunikácii nachádzali, ale neboli zaznamenané v PCAP súbore. Detekcia je možná prostredníctvom vyhľadávania chý-

bajúcich sekvenčných čísel v záchyte. Takéto poškodené PCAP súbory som vytvoril veľmi jednoducho a to odstránením niekoľkých segmentov zo správne zachytenej TCP komunikácie. podľa toho, ktoré segmenty som vymazal viem, kde má pri správnej funkcii programu dôjsť k nájdeniu nezachyteného segmentu.

Nástroj Scapy

Scapy [1] je program napísaný v jazyku Python, ktorý umožňuje užívateľovi odosielať, parsovať a vytvárať sieťové pakety. Tieto schopnosti umožňujú konštrukciu nástrojov, ktoré dokážu sondovať, alebo skenovať. Inými slovami, Scapy je výkonný manipulačný program pre manipuláciu s paketmi. Je schopný vytvárať alebo dekódovať pakety mnohých protokolov, odosielať ich do siete, zachytiť ich, odpovedať na požiadavky a odpovede a mnoho ďalšieho.

Tento nástroj je veľmi komplexný a ponúka mnoho zaujímavých funkcií, ale používal som ho iba na jednu špecifickú vec a tou je generovanie PCAP vzoriek obsahujúcich špecifické problémy. Vytvoril som skript v jazyku Python a špecifikoval som, ako budú vyzerať jednotlivé pakety a to tak, že som v každom pakete vyplnil položky v hlavičke. Spustením skriptu sa vygeneroval PCAP, ktorého správne vytvorenie som overil v programe Wireshark.

Simulácia problémového spojenia

PCAP vzorky obsahujúce konkrétny problém možno vytvoriť aj pomocou vytvorenia skutočného spojenia. Napríklad pomocou vytvorenia webového servera, na ktorom je možné definovať jeho správanie v konkrétnych prípadoch. Potom spustíme zachytávanie sieťovej komunikácie pomocou nástroja Wireshark. Klienta necháme posielať nejaké dotazy a server bude naprogramovaný tak, aby simuloval sieťový problém, napríklad bude odpovede posielať neskoro. Tento spôsob vytvárania demonštračných PCAP vzoriek som osobne neskúšal. Takéto PCAP vzorky mi vytvoril Petr Velan.

6.2 Vzorky DHCP komunikácie

Na realistickú simuláciu komunikačných problémov protokolu DHCP som použil program VirtualBox, pomocou ktorého som vytvoril virtuálnu sieť. Oracle VirtualBox [22] je multiplatformová virtualizačná aplikácia, ktorá umožňuje naraz spúšťať viacero operačných systémov vo viacerých virtuálnych počítačoch. Je možné spustiť veľké množstvo virtuálnych strojov, ktoré je ale limitované priestorom na disku a operačnou pamäťou.

Vnútorná sieť [6] je úplne izolovaná sieť, a preto je veľmi tichá, to znamená že neobsahuje rušivú komunikáciu ďalších sieťových aplikácii, ktoré nás nezaujímajú. To je dobré pre testovanie, keď potrebujete samostatnú, sieť. VirtualBox umožňuje vytvoriť sofistikované vnútorné siete s virtuálnymi strojmi, ktoré poskytujú vlastné služby pre vnútornú sieť. (napr. Active Directory, DHCP, atď.). Hostiteľský počítač nie je členom vnútornej siete. Tento režim umožňuje, aby vnútorná sieť neobsahovala komunikáciu s reálnou sieťou, na ktorej je hostiteľský počítač, inak by mohol DHCP klient na vnútornej sieti interagovať so serverom na vonkajšej sieti.

Vo virtuálnej sieti som vytvoril serverový počítač, na ktorom bežal operačný systém ubuntu 16.04, nakonfiguroval som na ňom jednoduchý DHCP server a spustil som zachytávanie komunikácie prostredníctvom programu Wireshark. Na klientských virtuálnych

počítačoch bol nainštalovaný minimálny operačný systém, ktorý obsahoval implementáciu DHCP klienta. Klientské počítače sa po spustení začali pripájať na DHCP server.

Problémy som simuloval tak, že som menil nastavenia servera. Nastavenia DHCP servera boli upravované prostredníctvom konfiguračného súboru dhcpd.conf. V tomto systémovom konfiguračnom súbore je možné meniť masku siete, adresu smerovača, doménové meno a rozsah prideľovaných IP adries. Vzorové PCAP súbory som vytvoril práve záchytom komunikácie pri rôznych stavoch konfigurácie. Napríklad som obmedzil počet IP adries špecifikovaním malého rozsahu adries, ktoré server prideľuje. Tiež som menil čas prenájmu adresy, keď bol čas prenájmu malý tak bolo množstvo komunikácie väčšie, lebo klienti museli často obnovovať adresu. Rôznymi nastaveniami konfigurácie sa mi podarilo vytvoriť PCAP súbory s problémami ako duplikované IP adresy, zahadzovanie dotazov serverom a veľké množstvo strát. Každá vzorka komunikácie sa zameriava na jeden hlavný problém, ktorý demonštruje. Vytvorené vzorky obsahujú nasledujúce problémy:

- **Požiadavky bez odpovede:** Kontrola, že množstvo DHCP retransmisií nie je veľmi vysoké a že odpoveď bola delegovaná.
- **Duplikovaná IP adresa:** DHCP server môže v špecifických prípadoch prideliť rovnakú IP adresu viacerým klientom. Tento problém môže nastať napríklad keď sa minú všetky adresy, ktoré server môže prideľovať. Tento problém je možné detegovať prostredníctvom ARP správy.
- **Server zahadzuje ponuky:** Deteguje, či server odpovedá prostredníctvom správy DHCPNAK. DHCP server pošle DHCPNAK klientovi, pokiaľ si je istý, že klient žiada adresu na lokálnej sieti, ktorá na tejto sieti neexistuje.
- **Klient posiela DHCPDECLINE:** Pokiaľ klient posiela správu DHCPDECLINE, tak oznamuje serveru, že klient zistil prostredníctvom iných prostriedkov, že ponúkaná IP adresa sa už používa. Môže to znamenať problém v konfigurácii DHCP.

Kapitola 7

Výsledky

Táto kapitola popisuje dosiahnuté výsledky celej práce, ktorá je zložená z návrhu a implementácie systému na meranie zachytenej sieťovej komunikácie. Kapitola je tiež zameraná na testovanie, ladenie programu a tiež výsledky testovania.

Overenie analýzy TCP na vytvorených PCAP súboroch

Väčšinu testovania a vyhodnotenia TCP analyzátoru som vykonával na dátovej sade vytvorených PCAP súborov, podrobnosti o jednotlivých súboroch sú rozpísané v časti 6.1. Na PCAP súboroch som ručne vypočítal hodnoty, ktoré by mal analyzátor namerať a podľa toho som vytvoril referenčné súbory na vyhodnotenie správneho fungovania programu. Výsledky som tiež porovnával so štatistikami, ktoré vytvára Wireshark, napríklad počet retransmisií alebo oneskorenie pri retransmisii.

Testovaciu sadu vytvorených PCAP vzoriek som neskôr použil, ako základ automatizovaného testu TCP analýzy. Jedná sa o shell skript, ktorý spúšťa Ingestor a zisťuje, či výsledky uložené v databázy zodpovedajú referenčnému výstupu. Referenčný výstup bol dôkladne skontrolovaný a štatistiky boli ručne spočítané. Tento test je súčasťou komplexnej testovacej sady systému Distance. Výhodou automatizovaného testovania je, že testy stačí napísať raz a potom je ich možné spúšťať automaticky pri každej zmene git repozitáru. V rámci Distance sa na automatizované spúšťanie testov používa nástroj Jenkins. Vďaka tomuto riešeniu je možné zistiť, ktorý commit spôsobil chybu a kde konkrétne chyba nastala.

Tabuľka 7.1: Testovacia sada TCP komunikácie.

Počet PCAP súborov	24
Počet tokov	66
Počet paketov	65910
Počet megabajtov	49,59
Priemerná veľkosť paketu v bajtoch	752,3933
Priemerná veľkosť toku v bajtoch	751367
Priemerná veľkosť toku v paketoch	998,6

Tabuľka 7.1 zobrazuje popis testovacej sady PCAP súborov, ktorá bola použitá ako referenčná sada súborov, podľa ktorých prebiehalo testovanie. Testovacia sada bola priebežne dopĺňaná o novovytvorené súbory a referenčné výstupy analýzy boli tiež priebežne aktualizované, a to najčastejšie pri doplnení novej funkcionality, alebo pri odhalení chyby v referenčnom výstupe.

Tabuľka 7.2: Výsledné štatistiky získané z testovacích PCAP súborov analýzou TCP komunikácie. Väčšina metrík v tabuľke je uvedená v percentách, v milisekundách sú len metriky spomalenie retransmisiou a RTT. RTT, alebo Round Trip Time je obojstranné spomalenie linky, je to čas od zaslania dotazu, po prijatie očakávanej odpovede.

PCAP	Nezachytené [%]		Stratovosť linky [%]		Spomalenie [ms] retransmisiou	RTT [ms]
	segmenty	bajty	segemntov	bajtov		
corrupted	0,42	0,77	4,97	9,29	23,32	0,29
large_loss	0	0	15,79	46,31	3,5	4
long_rtt	0	0	0	0	N/A	997
syn_rexmt	0	0	11,19	30,77	2,5	1,23
dead_link	0,074	0,248	0,112	0,258	1,3	0,423
dead_router	0,2	0,02	5,51	10,5	175,8	48,38
drop_client	3.88	7,67	4,58	9,01	43,07	0,44

V tabuľke 7.2 vidíme percentuálne počty nezachytených segmentov a nezachytených bajtov. Vo väčšine prípadov je počet nezachytených segmentov menší, ako je to u nezachytených bajtov. Dôvodom tohoto rozdielu je, že jedna medzera v za sebou idúcich sekvenčných číslach môže znamenať viacero nezachytených paketov, ale analyzátor nevie zistiť, koľko paketov v tejto medzere mohlo byť, preto je každá takáto medzera v sekvenčných číslach považovaná za jeden nezachytený paket. Počet nezachytených bajtov je možné zistiť presne, ako rozdiel medzi súčasným a očakávaným sekvenčným číslom. Preto, napríklad pokiaľ máme veľkosť segmentu 1000 bajtov a nedôjde k zachyteniu dvoch segmentov za sebou, bude detegovaný len jeden nezachytený segment, ale 2000 nezachytených bajtov.

Tabuľka 7.3: Analýza kvality zachytených dát podľa jednotlivých aplikačných portov.

Port	Služba	Počet tokov	nezachytené [%]	
			segmenty	bajty
21	FTP	4	0	0
80	HTTP	48	1,47	1,56
443	HTTPS	54	0,22	0,04

Výstupom TCP analyzátoru sú agregované štatistiky kvality záchytu podľa aplikačných portov, ako je to vidieť v tabuľke 7.3. Tieto štatistiky nám zobrazujú s akou kvalitou sa podarilo zachytiť pakety patriace určitému aplikačnému protokolu. Získané údaje je možné využiť pri ďalšej analýze. Podľa získaných údajov vieme podľa prahovej hodnoty stratených paketov určiť, či sa oplatí analyzovať aplikačný protokol. Pokiaľ je počet nezachyených paketov významný, tak by neskoršia analýza mohla poskytnúť nevalidné výsledky.

Overenie na reálnej zachytenej komunikácii Správnosť výpočtu TCP štatistík som overoval aj na skutočnej komunikácii, ktorú som zachytil zo siete pomocou nástroju Wireshark. Výhodou tohto spôsobu overenia je, že zistíme, ako program reaguje v skutočných podmienkach, na PCAP súboroch, ktoré sú väčšie a komplexnejšie, než tie čo sme vytvorili ručne. Jedna z nevýhod je časová náročnosť takéhoto overovania. Každá metrika musí byť najprv ručne spočítaná a až potom je možné výsledky porovnať. Nevieme dopredu, aký problém sa vo vzorke nachádza a preto ho treba analyzovať manuálne. Ale keď už spočítame

jednotlivé metriky zo súboru PCAP, tak sa dá tento PCAP zaradiť do testovacej kolekcie a už ho nie je potrebné znovu prepočítavať.

Tabuľka 7.4: Rýchlosť spracovania dát, vytvorenia a zápisu štatistík do databázy modulom Ingestor.

Veľkosť záchytu [MB]	Čas spracovania [s]	
	reálny	systémový
0,429	1,129	0,136
1,017	1.424	0,205
1,946	1.506	0,233
5,133	2.221	0,255
9,569	3.617	0,464
55,834	15.509	1,249

Tabuľka 7.4 popisuje rýchlosť, ktorou je program schopný spracovať PCAP súbor určitej veľkosti. Na spustenie programu som použil virtuálny počítač, ku ktorému som pristupoval prostredníctvom programu VirtualBox. Rozdiel medzi reálnym a systémovým časom je pravdepodobne spôsobený práve virtualizáciou, ktorej dôsledkom je výrazné spomalenie procesov bežiacich vo vnútri virtuálneho stroja. Z tabuľky vidíme, že trend rastu časovej náročnosti je približne lineárny a závisí od veľkosti vstupného PCAP súboru.

Jednotkové testy Jednotkové testovanie sa týka testov, ktoré overujú funkčnosť určitej časti kódu, zvyčajne na úrovni funkcie. V objektovo orientovanom prostredí je to zvyčajne na úrovni triedy a minimálne jednotky zahŕňajú konštruktory a deštruktory. Jednotkové testy spadajú do kategórie White box testing, čiže máme prístup k implementácii a vidíme vnútornú štruktúru programu. Tieto typy testov obvykle vytvárajú vývojári pri práci s kódom, aby zabezpečili, že konkrétna funkcia pracuje podľa očakávania. Jedna funkcia môže mať niekoľko testov, zachytávajúcich extrémne prípady alebo špecifické vetvy v kóde. Jednotkové testy nemôžu overiť funkčnosť softvéru, ale skôr sa používajú na zabezpečenie toho, aby stavebné bloky softvéru fungovali nezávisle na sebe.

Jednotkové testovanie je proces vývoja softvéru, ktorý zahŕňa synchronizovanú aplikáciu širokého spektra stratégií prevencie a detekcie defektov s cieľom znížiť riziká, čas a náklady potrebné na vývoj softvéru[1]. Vykonáva ich vývojár alebo softvérový inžinier počas konštrukčnej fázy životného cyklu vývojového softvéru. Jednotkové testovanie je zamerané na odstránenie chýb v konštrukcii predtým, než je kód propagovaný na ďalšie testovanie, táto stratégia má zvýšiť kvalitu výsledného softvéru, ako aj účinnosť celkového procesu vývoja. Na vytváranie jednotkových testov som použil nástroj Google Test[2]. Je to knižnica pre programovací jazyk C ++. Pomocou tejto testovacej knižnice je možné testovať aj programy napísané v jazyku C. Pri vytváraní testov som najprv vytvoril testovací program v C ++. Na začiatku programu som použil include pre zahrnutie závislostí testovaného súboru. Potom som pre každú funkciu vytvoril niekoľko testovacích prípadov, pričom každý prípad obsahuje mnoho volaní makra Assert. Pomocou makra Assert je možné zadať podmienku, ktorá musí platiť, aby mohol byť test vyhodnotený ako úspešný. Pre vykonanie komplexného testovania bolo nutné vytvoriť a naplniť štruktúru flow cache, potom sa flow

[1] https://en.wikipedia.org/wiki/Unit_testing
[2] https://github.com/google/googletest

cache používala pri volaní jednotlivých funkcií a nakoniec sa vyhodnocovalo, či sú výsledky správne.

Overenie analýzy DHCP na vytvorených PCAP súboroch

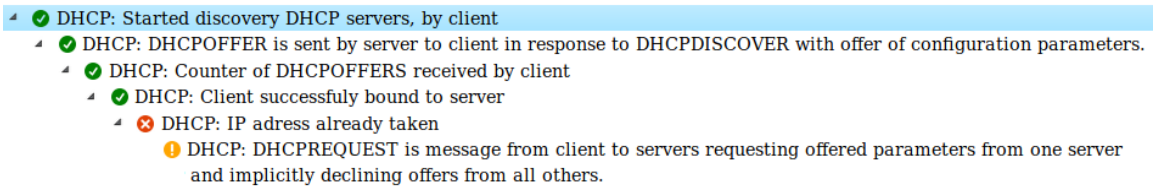
Testovanie analýzy protokolu DHCP som vykonal na PCAP súboroch vygenerovaných prostredníctvom virtuálnej siete v programe VirtualBox, viac v časti 6.2. PCAP súbory boli vytvorené tak, aby obsahovali konfiguračné problémy. Podľa toho, na aký problém sa PCAP zameriaval, som vedel, čo má analyzátor namerať. Výsledky som tiež ručne skontroloval podľa špecifikácie protokolu DHCP z RFC 2131 [8] a podľa stavového diagramu protokolu som zisťoval, či komunikácia prebehla správne.

Tabuľka 7.5: Sada PCAP súborov obsahujúca konfiguračné problémy protokolu DHCP, ktorá je určená na testovanie diagnostiky.

Počet PCAP súborov	12
Počet tokov	2086
Počet paketov	6366
Počet megabajtov	1,1
Priemerná veľkosť paketu v bajtoch	173
Priemerná veľkosť toku v bajtoch	527
Priemerná veľkosť toku v paketoch	3,05

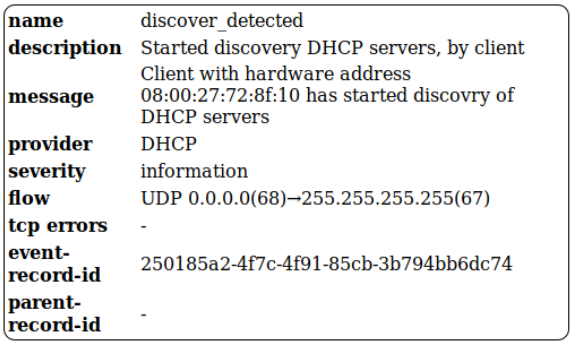
V tabuľke 7.5 je možné vidieť vlastnosti testovacej sady diagnostiky konfiguračného protokolu DHCP. Niektoré PCAP súbory obsahujú okrem protokolu DHCP aj komunikáciu prostredníctvom protokolu ARP. Je to preto, že práve vďaka protokolu ARP je možné nájsť duplikovaný výskyt IP adresy. Rovnako, ako u vzoriek komunikácie prostredníctvom protokolu TCP, aj vzorky DHCP komunikácie sú súčasťou automatizovaného testovania. Zaujímavosťou v tabuľke je, že tok obsahuje priemerne len tri pakety. Dôvod je pravdepodobne ten, že pri inicializácii DHCP komunikácie sa posielajú broadcastové pakety, ktoré sa zaradia do samostatného toku. Tok ako ho poznáme medzi dvomi IP adresami, vznikne až v momente, keď je klientovi pridelená IP adresa.

Analýza pri prechode diagnostickým stromom vygeneruje postupnosť udalostí, ktoré môžu byť použité na diagnostické účely. Obrázok 7.1 zobrazuje výsledok diagnostiky DHCP, analýza zistila že IP adresa už bola pridelená niekomu inému. Formát JSON, ktorý je výstupom diagnostiky bol prevedený do formátu zrozumiteľného človeku prostredníctvom webovej aplikácie, ktorú vytvoril Martin Holkovič. Aplikácia zobrazuje udalosti vygenerované prechodom rozhodovacím stromom. Po kliknutí na konkrétnu udalosť sa zobrazia podrobnosti o paketoch, na základe ktorých sa udalosť vygenerovala.



Obr. 7.1: Stránka vygenerovaná vizualizačným nástrojom. Nástroj na vizualizáciu zobrazuje udalosti vygenerované prechodom diagnostickým stromom protokolu DHCP.

Nástroj na zobrazenie diagnostických udalostí zrozumiteľnou formou je určený hlavne na ladenie výstupnej časti diagnostického nástroja, nebude sa používať vo výslednej vizualizácii. Použitie tohoto nástroja výrazne urýchlilo prácu a to hlavne vďaka prehľadnému zobrazeniu udalostí a prípadných chýb.



Obr. 7.2: Po kliknutí na určitú udalosť sa riadok zafarbí namodro a zobrazia sa detaily. Informácie zodpovedajú poliam zo zdrojového JSON súboru.

Kapitola 8

Záver

Cieľom tejto práce bolo zoznámiť sa s protokolmi TCP a DHCP, so systémom Distance a s programom Wireshark, prípadne s inými nástrojmi na analýzu kvality PCAP súborov. Štúdium malo vyústiť k návrhu modulu na meranie kvality zachytených dát na základe strát paketov spôsobených nedostatočnou rýchlosťou záchytu. Zaoberal som sa aj diagnostikou protokolu DHCP, zrovnal som existujúce možnosti diagnostiky tohoto protokolu a potom som sa zameral na návrh a implementáciu diagnostiky protokolu DHCP prostredníctvom diagnostického modulu Distance.

Na začiatku prvej časti je úvod k sieťam a základný popis vrstvového modelu sieťovej komunikácie. Prvá časť tiež obsahuje popis protokolov TCP, DHCP, a tiež protokolu UDP, nad ktorým beží aplikačný protokol DHCP. Popis vychádza z detailného štúdia protokolov z dokumentov RFC. V tejto časti sa zameriavam najmä na popis toho, ako protokoly fungujú a presný popis hlavičiek protokolov je zmienený v prílohách. Pri protokole TCP sa najviac zameriavam na popis sekvenčných a potvrdzovacích čísel, ktoré sú základom pre analýzu kvality PCAP súborov prostredníctvom detekcie nezachytených segmentov.

Ďalším krokom bolo zoznámiť sa s existujúcimi nástrojmi diagnostiky uvedených protokolov a to ako z teoretického, tak aj praktického hľadiska. Zameral som sa preto na programy capTCP a Wireshark, ktoré dokážu vytvárať podobnú diagnostiku protokolu TCP, ako potrebujem. Po preštudovaní, existujúcich možností diagnostiky som však zistil, že bude potrebná vlastná implementácia, a to hlavne z dôvodu, že capTCP nedokáže filtrovať toky podľa vytvorených štatistík a nedokáže vyhodnocovať kvalitu zachytených dát. Wireshark síce dokáže zistiť nezachytené pakety, ale nevytvára časové štatistiky o komunikačnom toku z pohľadu klienta a servera. Súčasne nezapadá do systému vyvíjaného v rámci projektu Distance, ktorý využíva špecifickú dátovú štruktúru a databázu RocksDB.

Na základe predchádzajúcej analýzy som vytvoril model identifikácie kvality PCAP súboru, ktorý je založený na vyhľadávaní nezachytených paketov prostredníctvom analýzy sekvenčných čísel. Analýza protokolu TCP je implementovaná prostredníctvom jazyka C. Samotná realizácia detekcie prebieha ako sekvenčný prechod cez TCP tok a zápis očakávaných sekvenčných čísel do vyrovnávajúcej pamäti, keď je neskôr očakávané sekvenčné číslo nájdené, tak sa z pamäti odstráni, ak sa nenájde, tak to znamená stratu segmentu spôsobenú nízkou rýchlosťou záchytu. Potom prebieha export získaných štatistík do databázy. Výpis z databázy a agregácia štatistík podľa aplikačných portov je realizovaný prostredníctvom programu TCP analyzátor, ktorý umožňuje pokročilé filtrovanie a vykreslenie štatistík vo forme grafov.

Okrem analýzy protokolu TCP som sa zameral aj na diagnostiku protokolu DHCP a detekciu možných konfiguračných problémov, ktoré môžu pri procese prideľovania IP adresy

nastať. Preštudoval som stavový diagram protokolu DHCP a na základe stavov, do ktorých sa môže komunikácia prostredníctvom tohoto protokolu dostať, som definoval diagnostický strom. Strom som neskôr transformoval do popisu v serializačnom formáte YAML. Vytvoril som popis udalostí, chýb a popis ich závažností, ktoré pri komunikácii nastávajú. Analýza protokolu DHCP potom prebieha prostredníctvom diagnostického modulu, ktorý je súčasťou systému Distance a slúži na transformáciu diagnostických pravidiel zo serializačného formátu YAML do jazyka Python.

Aby som zistil, či diagnostika protokolov TCP a DHCP funguje správne, tak som vytvoril sadu vzoriek komunikácie, ktoré demonštrujú určité problémy. Vzorka komunikácie je zvyčajene veľmi krátky PCAP súbor, ktorý demonštruje jeden problém na sieti. Vzorky som vytvoril buď pomocou knižnice scapy, alebo pomocou virtuálnej siete v rámci programu VirtualBox. Ku každému vzorovému záchytu som vytvoril referenčný výstup diagnostiky, ktorý očakávame. Vytvorené PCAP súbory a vzorové výstupy som potom zapojil do testovacieho systému Jenkins, ktorý je súčasťou Distance repozitáru.

Práca otvára veľa možností ďalšieho výskumu. Analýzu protokolu TCP je možné rozšíriť o ďalšie výkonnostné štatistiky o toku, je tiež možné veľmi jednoducho definovať ďalšie štatistiky časových závislostí, z ktorých sa vytvárajú grafy priebehu komunikácie. V prípade potreby sa dá rozšíriť aj diagnostika protokolu DHCP. Tvorbou sady podmienok je možné model identifikácie konfiguračných problémov protokolu DHCP rozšíriť o ďalšie udalosti. Je tiež možné využiť diagnostický modul systému Distance na realizáciu diagnostiky mnohých iných aplikačných protokolov. Nad diagnostikou bude potrebné v budúcnosti vytvoriť grafické užívateľské rozhranie, aby mohol sieťový administrátor jednoduchšie spravovať sieť s mnohými sieťovými zariadeniami.

Literatúra

[1] Bondi, P.: *Scapy Documentation*. Read the Docs, [Online; navštívené 01.04.2019]. URL https://scapy.readthedocs.io/en/latest/introduction.html

[2] Comer, D. E.: *Internetworking with TCP/IP*. Prentice Hall, 1995, ISBN 0-13-216987-8, volume 1.

[3] Comer, D. E.: *Internetworking With TCP/IP*. West Lafayette: Prentice Hall, Štvrté vydanie, 2000, ISBN 0-13-018380-6.

[4] Consortium, T. P. L.: P4 Language Specification. [Online; navštívené 07.05.2019]. URL https://p4.org/specs/

[5] Corbett, R.; Stallman, R.: *bison*. Linux man page, [Online; navštívené 11.04.2019]. URL https://linux.die.net/man/1/bison

[6] Coter, S.: *Oracle VM VirtualBox: Networking options and how-to manage them*. Oracle Corporation, 2017, [Online; navštívené 02.04.2019]. URL https://blogs.oracle.com/scoter/networking-in-virtualbox-v2

[7] DongJin Lee, N. B., Brian E. Carpenter: *Media Streaming Observations: Trends in UDP to TCP Ratio*. The University of Auckland, 2010, [Online; navštívené 08.04.2019]. URL https://www.cs.auckland.ac.nz/~brian/media-UDP-TCP.pdf

[8] Droms, R.: *Dynamic Host Configuration Protocol*. RFC Editor, Marec 1997, rfc : 2131. URL http://tools.ietf.org/html/rfc2131

[9] Halsall, F.: *Computer Networking and the Internet*. Pearson, piate vydanie, 2005, ISBN 0-321-26358-8.

[10] Information Sciences Institute, U. o. S. C.: *Transmission Control Protocol*. RFC Editor, September 1981, rfc : 793. URL https://tools.ietf.org/html/rfc793

[11] Ing. Petr Matoušek, M., Ph.D.: *Síťové služby a jejich architektura*. VUTIUM, 2014, ISBN 978-80-214-3766-1.

[12] ISO/IEC: *7498-1:1994 Information technology Open Systems Interconnection Basic Reference Model: The Basic Model*. International Organization for Standardization/International Electrotechnical Commission, [Online; navštívené 28.02.2019].

URL https://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip

[13] ITU-T: *ICT Facts and Figures 2017*. International Telecommunication Union, Geneva, Switzerland, [Online; navštívené 26.02.2019].
URL https://www.itu.int/en/ITU-D/Statistics/Pages/facts/default.aspx

[14] Kurose, J. F.; Ross, K. W.: *Computer Networking, A Top-Down Approach Featuring the Internet (6th edition)*. Addison-Wesley, 2012, ISBN 0-321-17644-8.

[15] N. Brownlee, G. R., C. Mills: *Traffic Flow Measurement: Architecture*. RFC Editor, Október 1999, rfc : 2722.
URL https://tools.ietf.org/html/rfc2722

[16] Oren Ben-Kiki, I. d. N., Clark Evans: YAML Ain't Markup Language. YAML Specification Index, [Online; navštívené 07.05.2019].
URL https://yaml.org/spec/1.2/spec.html

[17] Paxson, V.: *flex*. Linux man page, [Online; navštívené 11.04.2019].
URL http://dinosaur.compilertools.net/flex/manpage.html

[18] Postel, J.: *Transmission Control Protocol*. RFC Editor, August 1980, rfc : 768.
URL https://www.ietf.org/rfc/rfc768.txt

[19] Ralph Droms, T. L.: *The DHCP Handbook*. Sams Publishing, 2003, ISBN 978-0-672-32327-0.

[20] Sanjay Ghemawat, J. D.: A persistent key-value store for fast storage environments. RocksDB documentation, [Online; navštívené 07.05.2019].
URL https://rocksdb.org/

[21] Steve Mackay, J. P., Edwin Wright: *Practical Data Communications for Instrumentation and Control*. Newnes, 2003, ISBN 0750657979.

[22] VirtualBox, O. V.: *User Manual*. Oracle Corporation, 2019, [Online; navštívené 02.04.2019].
URL http://download.virtualbox.org/virtualbox/UserManual.pdf

Príloha A

Štruktúra TCP hlavičky

Tabuľka A.1: Formát hlavičky TCP

	0		1		2	3
0	Source Port				Destination Port	
4	Sequence Number					
8	Acknowledgment Number					
12	Data Offset	Reserved		Flags	Window Size	
16	Checksum				Urgent Pointer	
20	Options					

Nasledujúce vysvetlenie jednotlivých položiek TCP hlavičky bolo čiastočne prevzaté z RFC 793 [10].

- **Source Port:** 16 bitov
  Číslo určujúce zdrojový port odosielateľa. Port identifikuje špecifický proces, alebo sieťovú aplikáciu bežiacu na konkrétnom počítači. Pokiaľ sú dáta odosielané zo serverovej sieťovej aplikácie, tak je pravdepodobné že číslo portu patrí medzi známe čísla portov. To znamená že ku konkrétnej aplikácii je priradené konkrétne číslo portu a žiadna iná aplikácia ho nemôže použiť. Pokiaľ sú dáta odosielané z klientského procesu, tak bude zvolené dočasné číslo portu.

- **Destination Port:** 16 bitov
  Číslo určujúce port prijímateľa. Podobne, ako u zdrojového portu. Pokiaľ sú dáta odosielané zo serverového procesu, tak bude zvolené číslo portu odpovedať zdrojovému portu klienta, ktorý sa predtým dotazoval. Pokiaľ sú dáta odoslané z klientského procesu, tak pravdepodobne bude číslo portu zvolené podľa aplikácie, ktorú klient používa.

- **Sequence Number:** 32 bitov
  Poradové číslo prvého dátového oktetu v súčasnom pakete. Pokiaľ je prítomný príznak SYN, tak je inicializované číslo na počiatočnú hodnotu a prvý dátový oktet má počiatočnú hodnotu + 1.

- **Acknowledgment Number:** 32 bitov
  Keď je nastavený príznak ACK, tak toto pole obsahuje hodnotu nasledujúceho sek-

venčného čísla, ktorú odosielateľ očakáva, ako potvrdenie o správnom doručení dát. Po vytvorení spojenia sa toto číslo vždy posiela.

- **Data Offset:** 4 bity
  Počet 32 bitových slov v hlavičke TCP, ktorý indikuje kde začínajú dáta. DĺžkaTCP hlavičky je vždy násobkom 32 bitov.

- **Reserved:** 6 bitov
  Pre budúce použitie, nastavuje sa na nulu.

- **Flags:** 6 bitov
  Pole príznakov v základnej verzii TCP implementácie obsahuje 6 bitov. Nasleduje popis jednotlivých príznakových bitov z lava do prava.

  - URG: Ukazteľ naliehavých dát je platný
  - ACK: Potvrdzovacie číslo je platné
  - PSH: Prijímač má okamžite preniesť dáta do hornej vrstvy
  - RST: Reset spojenia
  - SYN: Vytvorenie spojenia a synchronizácia sekvenčných čísel
  - FIN: Ukončenie spojenia a znamená, že už nebudú nasledovať žiadne dáta

- **Window Size:** 16 bitov
  Je počet bajtov, ktoré môže počítač prijať, bez toho aby poslal potvrdenie.

- **Checksum:** 16 bitov
  Kontrolný súčet TCP / IP sa používa na zistenie poškodenia dát. Ak je bit preklopený, alebo nastal nejaký iný problém, potom je veľmi pravdepodobné, že prijímač si kvôli nesúladu kontrolného súčtu všimne, že paket je chybný.

- **Urgent Pointer:** 16 bitov
  Ukazteľ na naliehavé dáta, interpretuje sa iba pokiaľ je nastavený príznak URG. Určuje umiestnenie posledného bajtu naliehavých dát. Ak existujú naliehavé údaje TCP musí informovať entitu hornej vrstvy prijímateľa a odovzdať ukazovateľ na koniec naliehavých dát.

- **Options:** násobok 8 bitov
  Voliteľné položky TCP hlavičky.

Príloha B

Formát DHCP správy

Tabuľka B.1: Formát DHCP správy

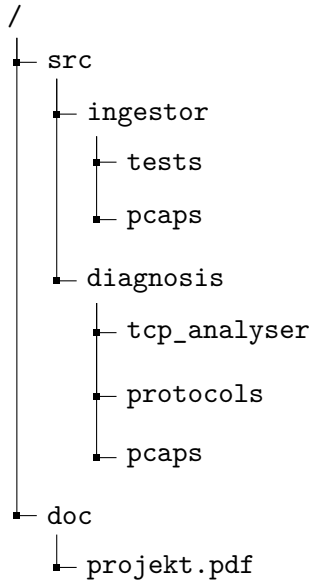
	0	1	2	3
0	OpCode	Hardware type	Hardware length	Hops
4	Transaction ID			
8	Seconds Elapsed		Flags	
12	Client IP Address			
16	Your IP Address			
20	Server IP Address			
24	Gateway IP Address			
28+	Client Hardware Address (16 B)			
	Server Host Name (64 B)			
	Boot File (128 B)			
	Options (variable length)			

- **OpCode:** Definuje operáciu, ktorá sa má vykonať. Typ správy môže byť BOOTREQUEST - 1, alebo BOOTREPLY - 2.
- **Hardware Type:** Typ MAC adresy.
- **Hardware length:** Dĺžka MAC adresy napríklad 6 pre ethernet.
- **Hops:** Klient nastaví počiatočnú hodnotu na 0. Môže byť použitý prenosovým agentom, ktorí umožňuje, aby viacej sieti spravoval jeden DHCP server.
- **Transaction ID:** Náhodné číslo zvolené klientom, ktoré sa používa na priradenie správ a príslušných odpovedí medzi klientom a serverom.
- **Seconds Elapsed:** Vypĺňa klient. Označuje počet sekúnd, uplynulých od začiatku získavania IP adresy, alebo začiatku jej obnovy.
- **Flags:** Používa sa len broadcast flag. Nastavením tohto príznaku klient požaduje aby server zaslal odpovede ako broadcast.

- **Client IP Address:** Používa sa iba, keď je klient v stave BOUND, RENEW, alebo REBINDING a môže odpovedať na ARP dotazy. To znamená že už má pridelenú IP adresu.
- **Your IP Address:** IP adresa klienta. Používa sa pri získaní novej IP adresy.
- **Server IP Address:** IP adresa serveru vracia server v správach typu DHCPOFFER, DHCPACK.
- **Gateway IP Address:** Adresa prenosového agenta.
- **Client Hardware address:** MAC adresa klienta, aby mu bolo možné doručiť správy, keď nemá pridelenú IP adresu.
- **Server Host Name:** Voliteľná položka, plne kvalifikované doménové meno serveru ukončené nulovým bajtom.
- **Boot File:** Cesta k inicializačnému súboru.
- **Options:** Voliteľné parametre.

Príloha C

Obsah CD



Adresár /src obsahuje implementáciu celého projektu. Nachádza sa tu tiež README, ktoré obsahuje popis adresárovej štruktúry, zoznam inštalačných závislosti a popisuje postup pri inštalácii.

Adresár /src/ingestor obsahuje moduly na vytvorenie programu Ingestor, jedným z modulov je aj modul statistics ktorý slúži na analýzu kvality zachytených dát a tiež na výpočet výkonnostných štatistík. Ďalej je tu adresár tests, ktorý obsahuje testovaciu sadu. Testy sa dajú spustiť príkazom `make check`. Sú tu tiež vzorové PCAP súbory na testovanie.

Adresár /src/diagnosis obsahuje adresár tcp_analyser s implementáciou analyzátoru TCP, ktorý slúži na výpis obsahu databáze a vytváranie grafov. Zložka protocols obsahuje definíciu diagnostiky konfiguračných problémov protokolu DHCP. Adresár pcaps obsahujé vzorové PCAP súbory určené na testovanie.

Adresár /doc obsahuje dokumentáciu. Súbor bp_xmarko15.pdf obsahuje vygenerovaný text bakalárskej práce vo formáte PDF. Ďalej sa tu nachádzajú zdrojové súbory tohoto dokumentu. Súbory obsahujú značky jazyka LaTeX.