Zadanie č.1 'Analyzátor sieťovej komunikácie'*

Hlib Kokin ID: 117991
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xkokin@stuba.sk

20.10.2022

*Dokumentacia prveho zadania v predmete Počítačové a komunikačné siete, ak. rok 2022/23, vedenie: Ing. Miroslav Bahleda, PhD.

Obsah

1 Úvod

Analyzátor sieťovej komunikácie.

Mojou úlohou bolo implementovať analyzátor sieťovej komunikácie Ethernet, v ktorom by mala byť implementovaná tlač informácií o každom rámci zakódovaná v súbore pcap. Bolo potrebné vyvinúť každú vrstvu sieťovej komunikácie: fyzickú, lineárnu, sieťovú, transportnú a aplikačnú, byte po byte.

V rámci úlohy môžu byť v každom rámci súboru pcap 4 protokoly linkovej vrstvy: 802.3 Ethernet II, IEEE 802.3 LLC, Novell 802.3 RAW, IEEE 802.3 LLC&SNAP, na sieťovej vrstve sme museli vypracovať ARP, IPv6, IPv4, ECTP, LLDP, a v prípade IPv4 tiež zabezpečiť, aby protokoly transportnej vrstvy, ako sú TCP, UDP a ICMP, boli správne analyzované pre každý. Pre LLC bolo potrebné pripojiť SAP a pre LLC SNAP aj PID.

2 Mechanizmus analyzovania protokolov

Ramec analyzujem tak, že sa po mu posúvam ďalej a ďalej po bajtoch, pričom sa postupne presúvam z jednej vrstvy do ďalšej.

2.1 Linková vrstva

Na definovanie protokolu druhej vrstvy použijeme hodnoty dĺžky rámca v bajtoch 13-14. Ak je hodnota týchto bajtov v desiatkovej sústave rovná alebo väčšia ako 1536, potom je v rámci protokol Ethernet 2, ak je dĺžka menšia ako 1500, protokol môže byť jeden z troch: LLC, LLC&SNAP alebo RAW.

Ak sme zistili, že protokol určite nie je Ethernet 2, potom pomocou hodnôt bajtov 15-16 môžeme zistiť, ktorý protokol máme. Ak sú hodnoty v týchto bajtoch všetci 'A', potom je to LLC&SNAP, ak sú hodnoty 'F', potom RAW, všetky ostatné hodnoty sú protokol LLC.

2.2 Sieťová vrstva

Ak sme v predchádzajúcom kroku zistili, že protokol je Ethernet 2, tak pri spracovaní hodnoty EtherType zistíme, ktorý protokol vrstvy 3 sa používa ako ďalší. (IPv4, IPv6, ICMP, IGMP alebo PIM - spracované protokoly, podľa podmienok zadania). Inak, v prípade LLC spracovávame hodnotu na byte 15, podľa hodnôt, ktoré sme dostali, berieme zodpovedajúci protokol zo súboru s protokolmi(IPX, NETBIOS, STP, tieto protokoly zaberajú 3. aj 4. vrstvu súčasne, ako napríklad PID v SNAP). V prípade SNAP na byte 15 budeme mať protokol SNAP, potom spracovaním bajtov 21-22 určíme PID protokolu SNAP(AppleTalk, PVSTP+, CDP, DTP).

2.3 Transportná vrstva

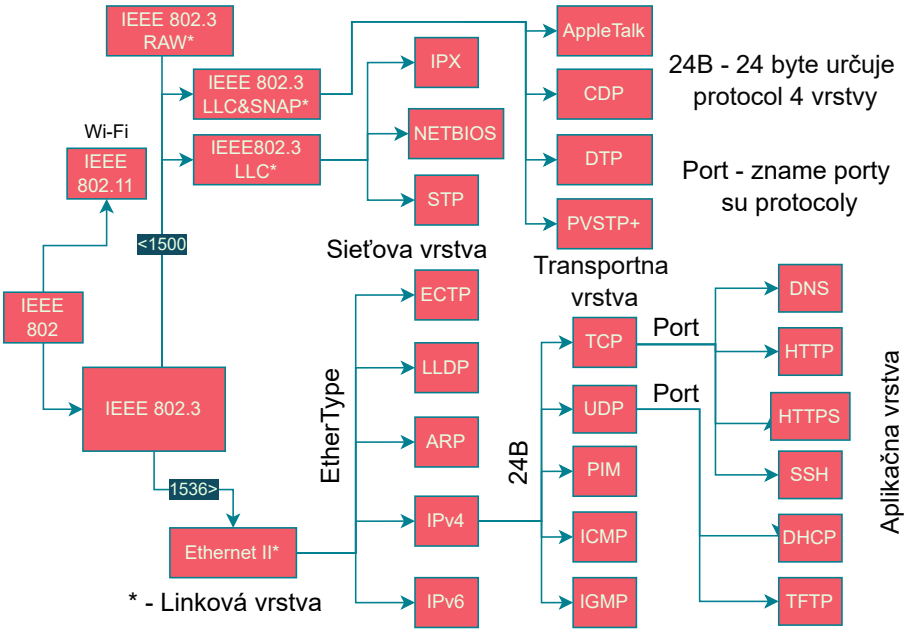
S protokolom IPv4, podľa hodnoty 24. bajtu, určíme protokol transportnej vrstvy (TCP, UDP, ICMP, IGMP, PIM)

2.4 Aplikačná vrstva

Ak sme na predchádzajúcej vrstve mali protokoly TCP alebo UDP, potom prejdením na bajty 35-36 alebo 37-38 zistíme hodnoty portov, jedna z nich môže byť známa, bude to protokol aplikačnej vrstvy(HTTP, HTTPS, DNS, DHCP...)

2.5 Maximálny odosielateľ

Maximálny odosielateľ je určený počtom paketov, ktoré odoslal zo svojej IP adresy. V kóde sa určuje postupným zápisom IPv4 protokolov do zoznamu (v momente, keď ich nájdeme), ak protokol už má záznam v zozname, keď ho nájdeme, tak sa mu inkrementuje počítadlo odoslaných paketov. Potom, čo program prejde a zapíše všetky pakety, určíme odosielateľa s najvyšším počtom odoslaných paketov porovnaním odosielateľov v zozname, ktorý sme vytvorili počas procesu analýzy paketov.



Obr. 1: Diagram spracovávania protokolov na rôzných vrstvach

2.6 Diagram spracovávania

Na diagrame šípky označujú, ktoré zdroje majú protokoly pripojenia, v ktorých vrstvách sa nachádzajú a na aké hodnoty rámca odkazujeme, aby sme ich našli.

Začnime analyzovať diagram od ľavého okraja, vidíme, že dĺžkou 13-14 bajtov môžeme určiť protokol druhej vrstvy, ak je dĺžka striktne väčšia ako 1535, potom pracujeme s Ethernetom 2, inak ak dĺžka je menšia ako 1500 - s jedným z troch: RAB, Snap, LLC.

Na určenie protokolu tretej vrstvy Ethernet 2 analyzujeme ZterType, porovnávajúc jeho hodnotu s hodnotami protokolov, ktoré sú nám známe, určíme ten, ktorý máme v našom balíku. Ďalej rovnakým spôsobom spracujeme 24 bajtov paketu, zadefinujeme protokol transportnej vrstvy. Potom je možné podľa hodnoty jedného z portov určiť protokol poslednej vrstvy (iba ak aspoň jeden z portov má hodnotu známeho protokolu)

Vráťme sa k LLC a SNAP a RAV. v predchádzajúcej kapitole bolo popísané, ako určiť, ktorý z týchto protokolov máme (2.1), ako aj ako určiť protokoly ďalších vrstiev (2.2)

3 Štruktúry externých súborov

Program analyzátora používa dva typy externých súborov a vytvára súbor YAML.

3.1 Filtre

Niektoré z použitých externých súborov boli súbory na filtrovanie paketov, ako napríklad udpFilter, tcpFilter a arpFilter. V týchto súboroch boli napísané funkcie na filtrovanie o konkrétnom protokole, fungujú na princípe kontroly každého paketu na prítomnosť požadovaného protokolu, ak sa protokol nenájde, tak sa paket jednoducho preskočí. V opačnom prípade sa paket skontroluje na príznaky fragmentácie alebo typ správy a umiestni sa do existujúcej komunikácie (ak už boli pakety medzi určitými IP adresami odoslané predtým) alebo do novej.

Po dokončení zoznamu komunikácií v slučke vypíšeme všetky komunikácie, najprv úplné, potom neúplné (určené príznakmi v triede komunikácie) Funkcia so špecifickým filtrom sa volá, ak bol v možnostiach spustenia programu zadaný prepínač -p (protokol) a bol zadaný správny protokol.

Princípom TFTP filtrovania je prejsť každý paket a odfiltrovať tie, ktoré nemajú na jednom zo svojich portov číslo 69, ale ak už komunikácia bola spustená, tak tam nebude hodnota 69 (aj keď stále bude protokol TFTP), potom skontrolujeme hodnoty portov a IP adries z každej existujúcej komunikácie.

Princípom ARP filtrovania je vytvorenie komunikácie pri odoslaní paketu Request (ak sa pri prezeraní existujúcej komunikácie nenašla aktualna komunikácia), ak má komunikácia viac odpovedí ako požiadaviek (toto sa zistí okamžite, akonáhle existuje jedna viac odpovedí), potom posledný paket s odpoveďou odstránime do novej neúplnej komunikácie a predchádzajúci urobíme kompletnou. Ak je počet požiadaviek väčší alebo rovný počtu odpovedí a počet odpovedí je 2, potom posledný pár žiadosť – odpoveď vezmeme do novej dokončenej komunikácie a zvyšné pakety ponecháme v predchádzajúcej komunikácii a nastavime ho na dokončené.

3.2 Protokoly

Ďalším externým súborom je súbor so zaznamenanými protokolmi a portami. S týmto súborom interagujeme pomocou funkcie getType, do ktorej odovzdávame určité bajty z paketu (v závislosti od protokolu, ktorý chceme nájsť a vrstvy, na ktorej sa nachádza), ako aj typu vyhľadávania - číslo z 1 až 6, kde

1. EtherType;
2. Ip protokoly (2);
3. LCC SAPy;
4. UDP&TCP porty;
5. Hľadanie protokolu na filter;
6. SNAP PIDy (2);

Keď funkcia dostane číslo skupiny protokolov, v ktorých potrebuje nájsť podobnosť s balíkom, funkcia sa presunie nadol v zozname, kým nenájde požadovanú kapitolu (názvy kapitol v súbore začínajú #), a potom vyhľadá protokol podľa hodnôt balíka, ak ho nenájde, potom funkcia vráti prázdny reťazec, čo znamená, že protokol je neznámy.

Obrázky ukazuju, ako je orámovaný záznam určitých protokolov v externom súbore, riadok začína znázornením čísla v hexadecimalnom tvare, nasleduje tabulátor a potom názov samotného protokolu, riadok končí znakom konca riadku.

```
#IP Protocol numbers
01  ICMP
02  IGMP
06  TCP
11  UDP
67  PIM

#SNAP PID
2000    CDP
2004    DTP
809B    AppleTalk
010B    PVSTP+
```

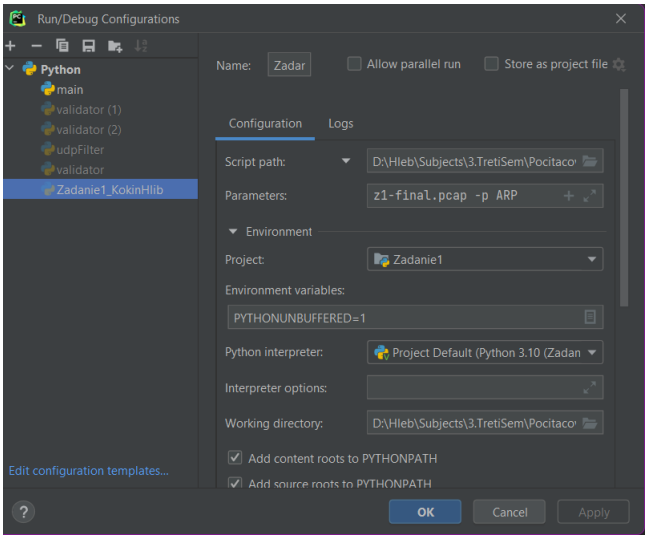
Obr. 2: Príklad zaznamenávania protokolov transportnej vrstvy do externého súboru s protokolmi

3.3 Výstupný súbor yaml

Výstupný súbor vytvoríme vo formáte yaml podľa šablóny uvedenej v úlohách a validátore. Súbor sa vypíše postupne vytlačením každej kategórie a hodnoty v nej (niektoré kategórie sa líšia v závislosti od protokolu). V kapitole o mechanizme analýzy protokolov(2) bol postupne popísaný algoritmus na určovanie protokolov, v tom istom algoritme napíšeme kategóriu a aký protokol máme definovaný v súbore yaml.

4 Používateľské rozhranie

Riešenie nemá grafické rozhranie ako také, všetky výstupné dáta sa zapisujú do súboru yaml a chybové hlásenia sa vypisujú do konzoly. Možnosti spustenia sa nastavujú buď spustením z konzoly a nastavením prepínača s protokolom, alebo v IDE ($Run- > EditConfigurations- > Zadanie1_{K}okinHlib- > Parameters$(meno pcap suboru, prepinač -p, protokol))



Obr. 3: Príklad pridania prepinača -p ku programu

5 Voľba implementačného prostredia

Ako jazyk na implementáciu som si vybral Python a napísal som kód, rozbehnul a testoval v IDE "PyCharm". Dôvodom výberu Pythonu bola jeho pohodlnosť a jednoduchosť pri písaní kódu v porovnaní s C a C++ a tiež potrebné knižnice bolo jednoduchšie zahrnúť do Pythonu ako do posledných dvoch jazykov.

6 Zhodnotenie a možnosti rozšírenia

Implementácia úlohy má veľký potenciál na rozšírenie pridaním nových filtrov pre protokoly, ako aj rozšírením štandardnej funkcie na tlač všetkých paketov v súbore pcap. Keďže tlačová funkcia súboru yaml je napísaná tak, že v každom pakete spracovávame určité bajty, potom keď sa objavia nové protokoly, nebudeme musieť celé riešenie úplne prerábať, bude stačiť pridať pár podmienky spracovania, nový protokol do súbor zu protokolmy a napíšte novú funkciu na filtrovanie nového protokolu, ktorý zodpovedá špecifikám jeho komunikácie a budeme mať pokročilé riešenie s minimom námahy.