**POČÍTAČOVÉ A KOMUNIKAČNÉ SIETE**

**ak. rok 2021/22, zimný semester**

**Zadanie 1: Analyzátor sieťovej komunikácie**

**Marek Chrappa**

**Zadanie úlohy:**

Návrh a implementácia programového analyzátora Ethernet siete, ktorý má analyzovať komunikácie v sieti zaznamenané v .pcap súbore a poskytuje nasledujúce informácie o komunikáciách.

1) Výpis všetkých rámcov v hexadecimálnom tvare postupne tak, ako boli zaznamenané v súbore. Pre každý rámec uveďte:

a) Poradové číslo rámca v analyzovanom súbore.

b) Dĺžku rámca v bajtoch poskytnutú pcap API, ako aj dĺžku tohto rámca prenášaného po médiu.

c) Typ rámca – Ethernet II, IEEE 802.3 (IEEE 802.3 s LLC, IEEE 802.3 s LLC a SNAP, IEEE 802.3 – Raw).

d) Zdrojovú a cieľovú fyzickú (MAC) adresu uzlov, medzi ktorými je rámec prenášaný.

2) Pre rámce typu Ethernet II a IEEE 802.3 vypíšte vnorený protokol.

3) Analýzu cez vrstvy vykonajte pre rámce Ethernet II a protokoly rodiny TCP/IPv4: Na konci výpisu z bodu 1) uveďte pre IPv4 pakety:

a) Zoznam IP adries všetkých odosielajúcich uzlov,

b) IP adresu uzla, ktorý sumárne odoslal

4) V danom súbore analyzujte komunikácie pre zadané protokoly:

a) HTTP

b) HTTPS

c) TELNET

d) SSH

e) FTP riadiace

f) FTP dátové

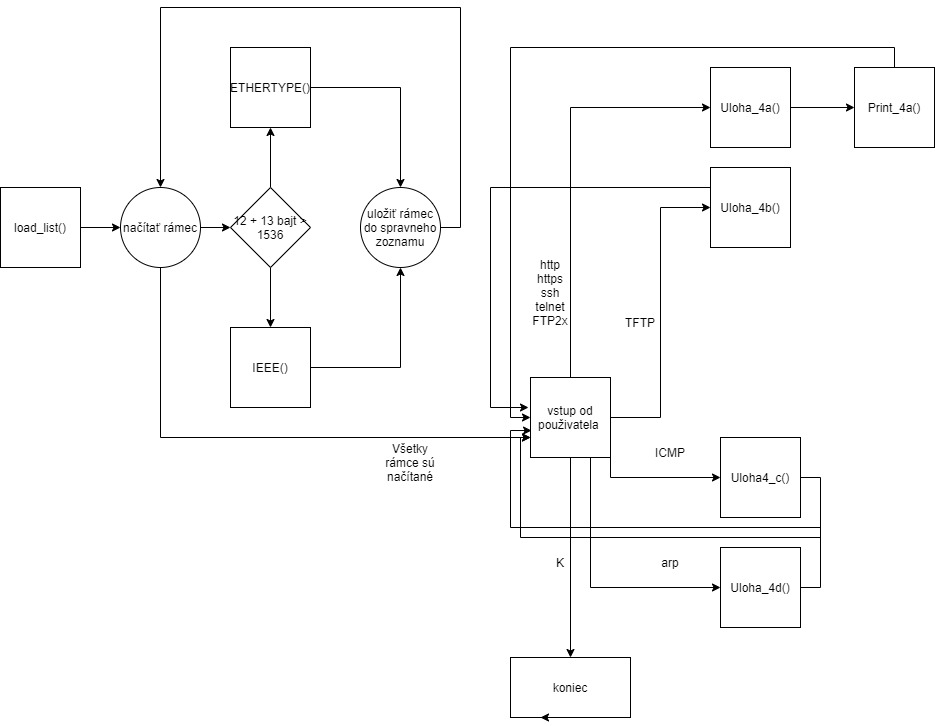
g) TFTP

h) ICMP

i) ARP dvojice

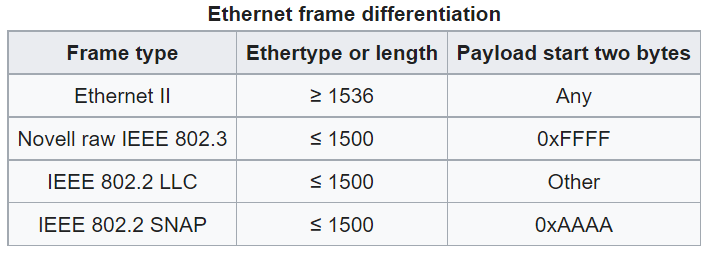
**Blokový návrh riešenia:**

Spôsob fungovania riešenia je že najskôr program načíta jeden rámec a potom na tom spraví analýzu a uloží si podstatné veci. Potom keď užívateľ bude volať jednotlivé procesy program jednoducho iba pošle informácie ktoré má už získané a pripravené.

****

**Návrh mechanizmu:**

1. Program hneď po zapnutí načíta .txt dokument v ktorom sa nachádza zoznam protokolov. Tento zoznam program spracuje a uloží do vlastných zoznamov na neskoršiu prácu.
2. Ďalej sa načíta .pcap súbor pomocou scapy.raw a pre jednotlivé rámce sa vykonajú nasledujúce body ktorými získame informácie z rámcov
3. Pre každý rámec sa vytvorí class RamecINF v ktorej budú dané informácie uložené a v neskorších bodoch využité
4. Program zistí dĺžku rámca poskytnutú cez API a z nej vypočíta jej dĺžku poskytnutú po médiu. Ďalšie je na poradí načítanie prvých dvanásť Bajtov rámca ktoré slúžia ako MAC zdrojová a cieľovú adresa. Tie sa uložia v classe v tvare stringu aby sa ušetrilo miesto a ďalej bolo možné s nimi pracovať.
5. Následne program načíta 12. a 13. bajt podľa hodnoty ktorých zhodnotí či sa jedná o typ rámca Ethernet II alebo o typ rámca z rodiny IEEE. Podľa toho o aký druh rámca sa jedná program pokračuje v získavaní informácií.



1. Ak sa jedná o rámec typu Ethernet II tak po načítaní 23 bajtu program prehľadáva svoj zoznam a zisťuje či má toto číslo ktoré hovorí o vnorenom protokole uložené. Ak sa jedná o vnorený protokol IPv4 program automaticky ukladá 26. až 33. rámec ktoré majú v sebe cieľovú a zdrojovú IP adresu ktoré sú potrebné pre ďalšie kroky. Zároveň program zisťuje či že z danej zdrojovej adresy boli odoslané packety a pripočíta k nej ďalšie. Program z hodnoty 12 a 13 bajtu ďalej zistí o aký druh Ethernetu II vnútorného protokolu ide a podľa toho pokračujú ďalšie akcie.
   1. Ak program zistí že názov vnútorného protokolu Ethernetu II je **ARP** zo špecifických bajtov rámca program zistí zdrojovú IP adresu a cieľovú IP adresu OP číslo z ktorého program určí o aký druh ARP správy sa jedná (REQUEST-REPLY).
   2. Ak program zistí že názov vnútorného protokolu Ethernetu II je **IPv4** zo špecifických bajtov rámca program zistí zdrojovú IP adresu a cieľovú IP adresu a číslo vnútorného protokolu vďaka ktorému program určí o aký protokol ide a bude s ním pracovať.   
        
      Program má v databáze viac možných vnútorných protokolov ale dôležité sú hlavne TCP a UDP pri ktorých program zo špecifických bajtov získa číslo zdrojového a cieľového portu ktoré sú potrebné pre ďalšie kroky.  
      Na koniec sa zdrojová IP adresa uloží do zoznamu ktorý sa využíva na úlohu č.3. Najskôr sa zoznam prehľadá či sa daná IP adresa už v ňom nenachádza. Ak sa nachádza tak sa jednoducho len zvýši číslo výskytu. Ak sa nenachádza tak sa pridá na koniec zoznamu.
2. V prípade že sa jedná o typ rámca z rodiny IEEE program načíta 14. a 15. rámec a zhodnocuje podľa veľkosti daných bajtov o aký druh IEEE rámca sa jedná.
   1. Ak sa jedná o **IEEE 802 LLC** tak sa prehľadá zoznam LLC protokolov a čísla týchto protokolov sa porovnajú s číslom na 14. bajte a vďaka tomu program vie určiť o aký vnútorný protokol ide.
   2. Ak sa jedná o **IEEE 802 RAW** tak ako vnútorný protokol sa zapíše IPX.
   3. Ak sa jedná o **IEEE 802 LLC + SNA**P tak sa prehľadá zoznam LLC protokolov a čísla týchto protokolov sa porovnajú s číslom na 14. bajte a vďaka tomu program vie určiť o aký vnútorný protokol ide a následne sa načíta 22. a 23. Bajt aby program podobným spôsobom zistili ethertype ktorý sa nachádza v hlavičke SNAP.

Na koniec sa uložia Bajty celého rámca do classy odkiaľ môže byť volaný pre neskoršie použitie a taktiež podľa rôznych parametrov (napr. či sa jedná o http,https rámce a pod.) sa rámce rozdelia do zoznamov pre ďalšie úlohy.

**Mechanizmus úloh**

Po načítaní vstupu na konzole od používateľa program vykoná jednu z nasledujúcich úloh

**Úloha 1 a 2:** Program jednoducho vypíše informácie zo zoznamu v ktorom sú uložené všetky rámce.

**Úloha 3:** Počaj analyzovania rámcov sa zároveň v špeciálnom zozname ukladali všetky použité IPv4 adresy takže v úlohe 3 sa len vypíšu + sa dodatočne vypíše aj tá IPv4 zdrojová adresa z ktorej bolo poslaných najviac rámcov.

**Úloha 4:** abcdef : Vždy sa pracuje so zoznamom v ktorom sú len špecifické rámce (http,https,ssh,telnet,ftp-data,ftpcontrol) a jednoduchým for cyklom kde sa kontrolujú IP adresy a zdrojové a cieľové porty sa roztriedia podľa komunikácií. Keď je rozdelenie hotové tak sa špeciálne pozerá na prvé 3 rámce a posledné 4 rámce a presnejšie či ide o správne začatú a správne ukončenú konverzáciu. Výstup úlohy má byť jedna kompletná a jedna nekompletná komunikácia, a tie sa vypíšu ak existujú.

**Úloha 4g:** V zozname sú uložené prvé TFTP rámce, čiže vždy ten ktorý má UDP port 69, potom sa prehľadávajú všetky rámce a kontrolujú sa porty a IP adresy a podľa toho program vie presne určiť ktoré rámce patria do ktorej TFTP komunikácie.

**Úloha 4h:** Podobne ako v predchádzajúcich úlohách kontroluje sa len špeciálny zoznam, tentoraz s ICMP komunikáciami. Program ICMP rámce roztriedi do hotových komunikácii a vypíše ich.

**Úloha 4i:** Všetky ARP rámce sa podelia na komunikácie tak aby bol vždy aspoň jeden Request a jeden reply, ak existujú rámce ktoré nemajú ARP dvojicu tak sa vypíšu na koniec.

**Príklad štruktúry externého súboru pre učenie protokolov a portov**

#ethertypes

0x0800 IPv4

0x0806 ARP

0x0842 Wake-on-LAN

0x22F0 AVTP

#IEEEs

0xffff IEEE 802.3 RAW

0xaaaa IEEE 802.3 LLC + SNAP

0 IEEE 802.3 LLC

#LSAPs

0x00 Null LSAP

0x02 Individual LLC Sublayer Mgt

0x03 Group LLC Sublayer Mgt

0x04 SNA Path Control (individual)

#ARPs

1 Request

2 Reply

#IPv4s

1 ICMP

2 IGMP

6 TCP

#ICMPs

0 Echo Reply

3 Destination Unreachable

4 Source Quench

5 Redirect

8 Echo

#TCPs

7 ECHO

19 CHARGEN

20 FTP-DATA

21 FTP-CONTROL

22 SSH

#UDPs

7 ECHO

19 CHARGEN

37 TIME

53 DOMAIN

#END

\*UDP a TCP porty sú v decimálnej sústave, zbytok je v hexadecimálnej sústave

**Používateľské rozhranie:**

Program funguje ako konzolová aplikácia ktorej výsledok je vypisovaný do nového txt. Dokumentu z dôvodu že konzolová aplikácia má maximum riadkov ktoré vie vypísať a v prípade že sa toho číslo prekoná, staré riadky sa začnú prepisovať.

Používateľ ovláda program pomocou jednoduchých príkazov ktoré vpisuje do konzole

Príkazy:

a – Výpis úlohy 1, 2 a 3

b – Výpis úlohy 4a

c – Výpis úlohy 4b

d – Výpis úlohy 4c

e – Výpis úlohy 4d

f – Výpis úlohy 4e

g – Výpis úlohy 4f

h – Výpis úlohy 4g

i – Výpis úlohy 4h

j – Výpis úlohy 4i

K - koniec

**Voľba Implementačného prostredia**

Zvolený programovací jazyk je python s knižnicou scapy.all hlavne kvôli jednoduchšej práci s pamäťou a celkovej organizácii projektu ktorá mi prišla lepšia a príjemnejšia ako v inom implementačnom prostredí alebo s inou knižnicou.