SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológii

Počítačové a komunikačné siete

Zadanie 1:

Analyzátor sieťovej komunikácie

Tomáš Socha

ID: 110896

Akademický rok 2021/2022

1) Zadanie úlohy

Navrhnite a implementujte programový analyzátor Ethernet siete, ktorý analyzuje komunikácie v sieti zaznamenané v .pcap súbore a poskytuje nasledujúce informácie o komunikáciách. Vypracované zadanie musí spĺňať nasledujúce body:

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**. Študent musí vedieť vysvetliť, aké informácie sú uvedené v jednotlivých rámcoch Ethernet II, t.j. vnáranie protokolov ako aj ozrejmiť dĺžky týchto rámcov.

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 (bez ohľadu na prijímateľa) najväčší počet paketov a koľko paketov odoslal (berte do úvahy iba IPv4 pakety).

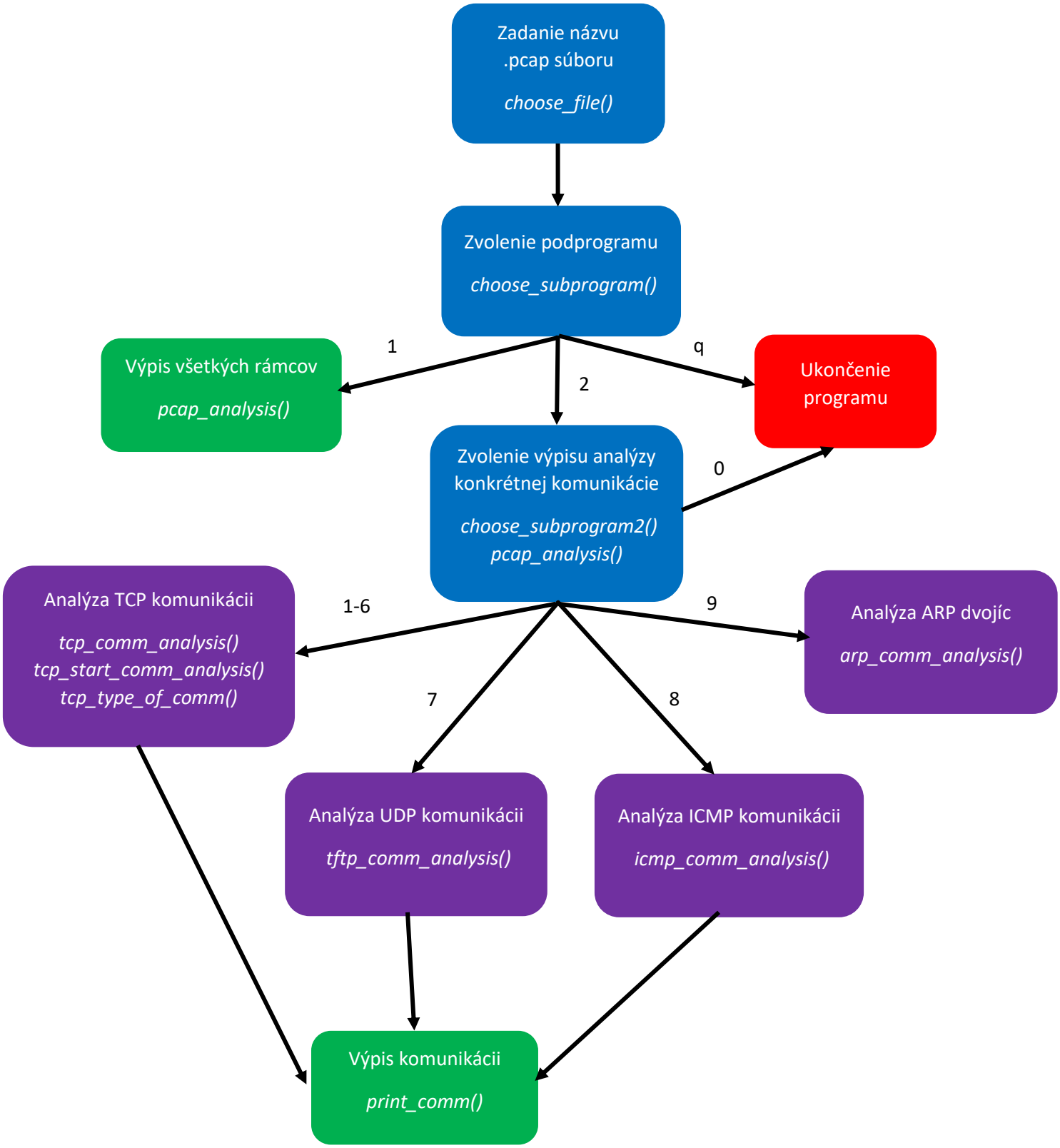
IP adresy a počet odoslaných / prijatých paketov sa musia zhodovať s IP adresami vo výpise Wireshark -> Statistics -> IPv4 Statistics -> Source and Destination Addresses.

4) V danom súbore analyzujte komunikácie pre zadané protokoly:

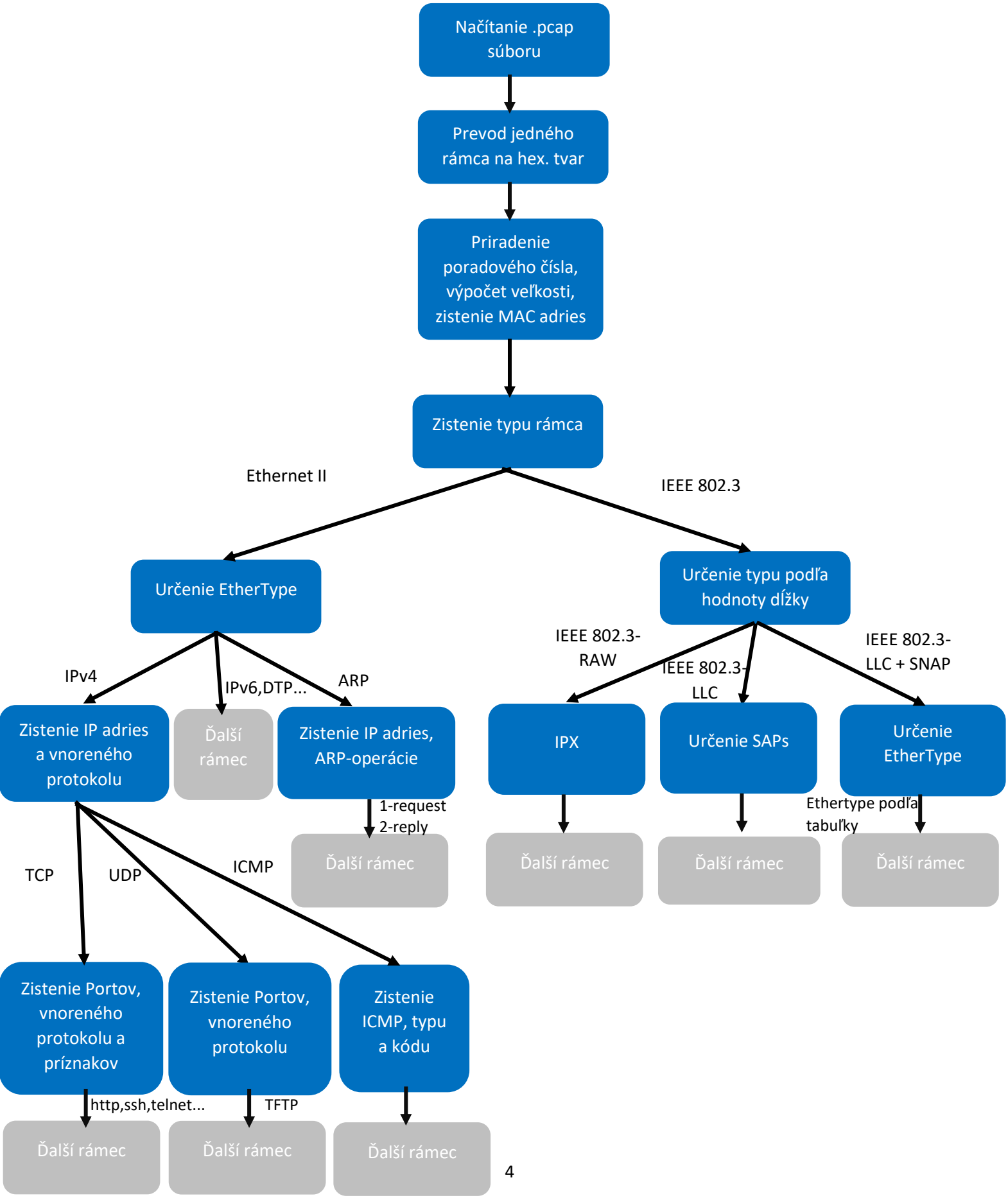
a) HTTP
b) HTTPS
c) TELNET
d) SSH
e) FTP riadiace
f) FTP dátové
g) TFTP, **uveďte všetky rámce komunikácie**, nielen prvý rámec na UDP port 69
h) ICMP, uveďte aj typ ICMP správy (pole Type v hlavičke ICMP), napr. Echo request, Echo reply, Time exceeded, a pod.
i) **Všetky** ARP dvojice (request – reply), uveďte aj IP adresu, ku ktorej sa hľadá MAC (fyzická) adresa a pri ARP-Reply uveďte konkrétny pár - IP adresa a nájdená MAC adresa. V prípade, že bolo poslaných viacero rámcov ARP-Request na rovnakú IP adresu, vypíšte všetky. Ak sú v súbore rámce ARP-Request bez korešpondujúceho ARP-Reply (alebo naopak ARP-Reply bez ARP-Request), vypíšte ich samostatne.

V prípadoch komunikácií so spojením vypíšte iba jednu kompletnú komunikáciu a aj prvú nekompletnú komunikáciu.
Ak počet rámcov komunikácie niektorého z protokolov z bodu 4 je väčší ako 20, vypíšte iba 10 prvých a 10 posledných rámcov tejto komunikácie.

2) Blokový návrh fungovania riešenia



Blokový návrh spracovania jedného rámca



3) Mechanizmus analyzovania protokolov na jednotlivých vrstvách

Po spustení programu sa ešte pred samotnou analýzou pomocou funkcie load_protocols() načítajú potrebné hodnoty a názvy jednotlivých protokolov, portov a správ. Tieto hodnoty sa uložia do príslušných slovníkov, z ktorých sa budú počas analýzy vyberať a porovnávať.

a) Analýza jednotlivých protokolov

Na celom mechanizme analyzovania rámcov sa podieľa viacero funkcii. Všetko sa to začína funkciou *pcap_analysis()* ktorá prejde všetky rámce z načítaného súboru. Argumentom tejto funkcie je premenná filter, v ktorej sa voliteľne nachádza názov protokolu/portu, ktoré je potrebné uložiť do listu, pretože budú potrebné na ďalšiu **analýzu**. Ak táto premena filter nič neobsahuje, tak netreba ukladať žiadne **rámce**, a po zbehnutí analýzy konkrétneho rámca sa vypíše na obrazovku. Taktiež sa príslušnému rámcu priradí **poradové číslo**.

Funkcia obsahuje vnorenú funkciu *frame analysis(),* ktorá zoberie konkrétny rámec a analyzuje ho. Ako prvé sa zistí **veľkosť** rámca poskytnutá pcap API na základe veľkosti hex. poľa reprezentujúci daný rámec. Následne program vypočíta **dĺžka** rámca prenášaná po médiu. Táto veľkosť je minimálne 64 bajtov, a ak je veľkosť rámca väčšia ako 60 B tak sa len k tomuto číslo pripočítaju ešte ďalšie 4B.

Ďalej program zistí cieľovú a zdrojovú **MAC adresa**. Cieľová MAC adresa sa nachádza na prvých 6 bytoch, Zdrojová na ďalších 6, čiže na miestach reprezentujúcich 7-12 bajt.

Ďalej sa podľa 13-14 bajtu kde sa nachádza dĺžka alebo EtherType, určí typ rámca. Program zoberie toto číslo, a pozrie sa na jeho decimálnu hodnotu. Môžu nastať 2 prípady:

Číslo >= 1536 - ide o *Ethernet II*

Číslo <= 1500 - ide o *IEEE 802.3*

i. IEEE 802.3

Ako prvé si rozpíšeme postup spracovania rámca typu **IEEE 802.3**. Podľa konkrétnej hodnoty tohto čísla sa určí typ IEEE.

AAAA => *IEEE 802.3 LLC + SNAP*, následné určenie ethertypu pozretím na 21-22 b

FFFF => *Novell 802.3 RAW* , následne vypísanie vnútorného protokolu *IPX*

Iné => *IEEE 802.3 LLC*, následne určenie SAPu

Týmto sa analýza IEEE 802.3 končí.

ii. Ethernet II

Pri **Ethernete** sa na základe hodnoty 13-14 bajtu určí **EtherType** a podľa neho sa rozvíja následná analýza:

0800 = > *IPv4* , zavolanie funkcie *ipv4_analysis()* na analýzu ďalších vrstiev

0806 = > *ARP* ,zistenie ARP operácie podľa 22. bajtu a ukončenie analýzy

Iné => koniec analýzy

Funkcia *ipv4_analysis()* najprv zistí cieľovú a zdrojovú **IP adresu**, kde zdrojová sa nachádza na 27-30 bajte a cieľová na 31-34 bajte.

Následné sa zistí vnorený protokol podľa hodnoty na 24 bajte. Ak ide o **TCP** alebo **UDP** protokol, tak program identifikuje vnorený port, ktorého pozícia sa mení vzhľadom na veľkosť IP hlavičky. Zdrojový port sa nachádza na prvých 2 bajtoch TCP/UDP hlavičky a cieľový nasleduje hneď za ním.

V prípade, že ide o ICMP protokol, program zistí typ a kód icmp správy. Typ je možné nájsť v prvom bajte ICMP hlavičky a k nemu prislúchajúci kód na druhom bajte.

b) Analýza komunikácií

Program dokáže analyzovať všetky komunikácie podľa bodu 4 zo zadania. Program si pri každej analýze najskôr vyfiltruje príslušný port/protokol, ktorého komunikáciu ide analyzovať.

i. TCP komunikácia

Pri TCP komunikácii sa rozlišujú porty HTTP, HTTPS, TELNET, SSH, FTP- riadiace a FTP-dátové. Postup analýzy je pri všetkých portoch rovnaký. Po vyfiltrovaní rámcov obsahujúcich príslušný port sa zavolá funkcia *tcp_comm_analysis()* ktorá nájde prvú kompletnú a nekompletnú komunikáciu ak sa v danom súbore nachádza a zavolaním funkcie *print_comm()* ich vypíše.

Funkcia najskôr identifikuje prvú komunikáciu zavolaním ďalšej funkcie -*tcp_start_comm_analysis(),* ktorá na základe rovnakých dvojíc portov a IP adries a identifikovaním začiatku komunikácie pomocou príznakov vytvorí list obsahujúci jednu komunikáciu. Komunikácia je považovaná za správne otvorenú ak prvé tri rámce obsahujú jednotlivo príznaky SIN, SIN ACK a ACK.

Po nájdení prvej komunikácie funkcia *tcp_comm_analysis()* ďalej zavolá funkciu *tcp_type_of_comm()* ktorá identifikuje, či ide o kompletnú alebo nekompletnú na základe správneho ukončenia. To môže nastať po presne stanovenej postupnosti:

- Posledné 4 rámce obsahujú príznaky FIN ACK, ACK, FIN ACK, ACK alebo FIN, ACK, FIN, ACK
- Posledné 3 rámce obsahujú príznaky FIN, FIN ACK, ACK
- Posledné rámce ukončené RST alebo RST ACK

Následne funkcia obsahuje while cyklus, ktorý prechádza všetky ostatné komunikácie, a ak bola už nájdená kompletná tak hľadá nekompletnú a naopak.

ii. UDP komunikácia

Pri UDP komunikácii je spracovávaný len TFTP protokol. Jednotlivé komunikácie sú roztriedené vo funkcii *pcap_analysis()*. Identifikuje sa prvý rámec komunikácie, ktorého cieľový port má hodnotu 69 decimálne, uloží sa dvojica zdrojovej a cieľovej IP adresy a zdrojový port a na ďalšom rámci aj zmenený cieľový port. Následne sa ukladajú všetky rámce ktoré obsahujú rovnaké dvojice IP adries a portov. Pri nájdení iných IP adries alebo portov, alebo znovu nájdení portu 69 ide už o ďalšiu komunikáciu. Každá komunikácia sa nachádza v jednom liste a z funkcie sa posielajú všetky funkcie naraz ako listy v liste. Ako posledný krok sa zavolá funkcia tftp_comm_analysis(), ktorá zabezpečí výpis všetkých nájdených komunikácii, a ich očíslovania.

iii. ICMP komunikácia

Pri ICMP komunikácii len zgrupujem všetky rámce obsahujúce rovnaké dvojice IP adries a po prejdení všetkých rámcov a ich zatriedení program vypíše rámce podľa jednotlivých komunikácii.

iv. ARP komunikácia

Na analýzu ARP komunikácie sa používa funkcia *arp_comm_analysis()* ktorá zoberie list už vyfiltrovaných rámcov, ktorý sa skladá z ďalších dvoch listov. V jednom sú uložené ARP- request a druhý obsahuje ARP-reply rámce. Postupne vyberá jednotlivé ARP-reply a hľadá ku ním ARP-requests na základe rovnakých dvojíc IP adries, ktoré ale musia mať vymenené poradie.

Po nájdení každej dvojice, prípadne viac requestov spojených s jedným reply, ich vypíše a vyberie z listu. Akonáhle nájde všetky páry pozrie sa či zostali nejaké rámce bez dvojice a ak hej tak vypíše aj tie samostatne pod označením, že im chýba dvojička.

4) Štruktúra externých súborov

Navrhnutý analyzátor používa 2 externé súbory a to protokoly.txt a ICMP.txt z ktorých si načíta hodnoty a názvy protokolov, portov a ICMP správ.

Oba súbory obsahujú 2 stĺpce, ktoré sú od seba oddelené medzerou. Prvý stĺpec obsahuje hodnoty v hexadecimálnom tvare a druhý názov daného protokolu/portu/ICMP správy

Súbor protokoly.txt je rozdelený na viacej častí pomocou riadku, ktorý sa začína znakom „#“. Ide o identifikátor, ktorý určuje po ňom nasledujúce hodnoty, teda ku ktorému protokolu patria dané porty, či ide o ethertypes, LSAPs alebo IP protokoly.

Súbor ICMP.txt je rozdelený na 2 časti, kde sa v prvej nachádzajú typy ICMP správy, a v druhej sú podľa tejto správy zoskupené kódy.

ICMP.txt

```
#ICMP types
0x00 Echo reply
0x03 Destination unreachable
0x04 Source quench
0x05 Redirect
0x08 Echo request
0x09 Router advertisement
0x0A Router selection
0x0B Time exceeded
0x0C Parameter problem
0x0D Timestamp
0x0E Timestamp reply
0x0F Information request
0x10 Information reply
0x11 Address mask request
0x12 Address mask reply
0x1E Traceroute
#ICMP codes
#03
0x00 Net unreachable
0x01 Host unreachable
0x02 Protocol unreachable
0x03 Port unreachable
0x04 Fragmentation needed and Don't Fragment was set
0x05 Source route failed
0x06 Destination network unknown
0x07 Destination host unknown
0x08 Source host isolated
```

protokoly.txt

```
#Ethertypes
0x0800 IPv4
0x0806 ARP
0x86DD IPv6
0x2004 DTP
0x809B Appletalk
#LSAPs
0x06 IP
0x0E PROWAY-NM
0x4E MMS
0x5E ISI-IP
0x7E X.25-PLP
0x8E PROWAY-ASLM
0xAA SNAP
0xF4 LAN
0xFE ISO
0xFF DSAP-GLOBAL
0x42 STP
0xAA SNAP
0xE0 IPX
#IP Protocol
0x01 ICMP
0x02 IGMP
0x06 TCP
0x09 IGRP
0x11 UDP
0x26 IDPR-CMTP
0x28 IPv6
```

5) Používateľske rozhranie

Navrhnutý program využíva používateľské prostredie ktoré vyžaduje interakciu s používateľom. Po spustení programu sa vypíše hlavička obsahujúca názov programu a meno autora. Taktiež sa vypíše prvá požiadavka pre ďalšiu prácu programu, ktorou je zadanie správneho názvu .pcap súboru ktorý si používateľ vybral na analýzu.

Následne program zobrazí ponuku, ktorá ponúka výpis celého súboru alebo analýzy komunikácie. V prípade zvolenia analýzy je potrebné vybrať jednu možnosť z ďalšej ponuky, čiže komunikáciu ktorého protokolu chceme sledovať.

Program je možné ukončiť stlačením tlačidla „q“ po zobrazení ponuky pre výpis alebo stlačením „0“ po zobrazení ponuky pre výber konkrétneho protokolu na analýzu.

Ponuka pre typ výpisu

```
Vyber jednu z nasledujúcich možností:
-------------------------------------
zobrazenie základného výpisu         -       stlač 1
analýza komunikácie pre konkrétny protokol - stlač 2
ukoncenie programu             -             stlač q
Stlač klavesu:
```

Výpis jedného rámca

```
rámec 1047
dĺžka rámca poskytnutá pcap API: 54 B
dĺžka rámca prenášaného po médiu: 64 B
Ethernet II
Zdrojová MAC adresa:00 14 38 06 E0 93
Cieľová MAC adresa:00 02 CF AB A2 4C
IPv4
zdrojová IP adresa:192.168.1.33
cieľová IP adresa:74.125.43.97
TCP
HTTPS
zdrojový port: 4278
cieľový port: 443
00 02 CF AB A2 4C 00 14  38 06 E0 93 08 00 45 00
00 28 2C 8A 40 00 80 06  96 9E C0 A8 01 21 4A 7D
2B 61 10 B6 01 BB 1C 1B  CE CC 1F 19 59 E9 50 10
FF FF 01 D2 00 00
```

Ponuka pre výber protokolu

```
Vyber jednu z možností
----------------------
HTTP   - '1'
HTTPS  - '2'
TELNET - '3'
SSH    - '4'
FTP riadiace -'5'
FTP dátové -  '6'
TFTP - '7'
ICMP - '8'
ARP  - '9'
ukončenie programu - '0'
Zadaj hodnotu:
```

6) Implementačné prostredie

Ako implementačné prostredie som si zvolil jazyk Python vo verzii 3.9 hlavne kvôli jednoduchej práci s reťazcami znakov a poľami.

Použité knižnice:

- scapy.all - importovaná len funkcia rdpcap() ktorá slúži na načítanie dát z .pcap súboru
- Sys – použitá funkcia exit() na ukončenie programu
- Os - na vyčistenie konzoly

Použité vlastné triedy:

- ARPFrame – trieda predstavujúca jeden ARP rámec. . Obsahuje atribúty, kde sú uložené všetky zistené informácie o danom rámci. Taktiež je jej súčasťou metóda výpis() ktorá vypíše všetky zistené informácie o príslušnom rámci.
- Frame – trieda odvodená od triedy ARPFrame. Predstavujúca jeden rámec ipv4. Obsahuje atribúty navyše, ktoré sa nenachádzajú pri ARP Ethertype. a prekonáva metódu výpis()
  Obe triedy taktiež obsahujú funkciu vypis_subor() na vypísanie do externého súboru

Použité vlastné funkcie :

def print_hexdump(array):

- výpis rámca v hexadecimálnom tvare

def print_hexdump_file(array):

- výpis rámca v hexadecimálnom tvare do externého súboru

def arp_comm_analysis(list_of_frames):

- Analýza a výpis ARP komunikácii

def icmp_comm_analysis(list_of_frames):

- Analýza a výpis ICMP komunikácii

def tftp_comm_analysis(list_of_frames):

- Výpis TFTP komunikácii

def tcp_type_of_comm(list_of_communication):

- Rozhodne, či ide o kompletnú alebo nekompletnú komunikáciu

def tcp_comm_analysis(list_of_frames):

- Hľadá jednu kompletnú a jednu nekompletnú TCP komunikáciu

def print_comm(list_of_communication):

- Vypíše jednu komunikáciu

def tcp_start_comm_analysis(list_of_frames):

- Nájde tcp komunikáciu

def pcap_analysis(file_reader, filter):

- Prejdenie všetkých rámcov v pcap súbore, zavolanie funkcie na ich analýzu a uloženie rámcov potrebných pre ďalšie spracovanie

def frame_analysis(array, frame):

- Určenie typu rámca a ethertype/SAP, v prípade ipv4 zavolanie funkcie na analýzu ipv4

def ipv4_analysis(array, frame):

- Analýza vnorených vrstiev ipv4

def choose_subprogram(file_reader):

- Výber typu konkrétneho výpisu(podľa úlohy 1 alebo 4)

def choose_subprogram_2(file_reader):

- Výber analýzy komunikácie konkrétneho protokolu vybraného používateľom

def load_protocols():

- načítanie protokolov z externého textového súboru

def choose_file():

- načítanie názvu súboru od používateľa