Meno: Marko Stahovec
ID: 110897

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

ZADANIE 1 – ANALYZÁTOR SIEŤOVEJ KOMUNIKÁCIE

Meno a priezvisko: Marko Stahovec

Dátum vypracovania: 15.10.2021

Cvičiaci: Ing. Kristián Košťál, PhD.

Meno: Marko Stahovec
ID: 110897

OBSAH:

Meno: Marko Stahovec
ID: 110897

1. Zadanie

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.

Vo výpise jednotlivé bajty rámca usporiadajte po 16 alebo 32 v jednom riadku. Pre prehľadnosť výpisu je vhodné použiť neproporcionálny (monospace) font.

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.**

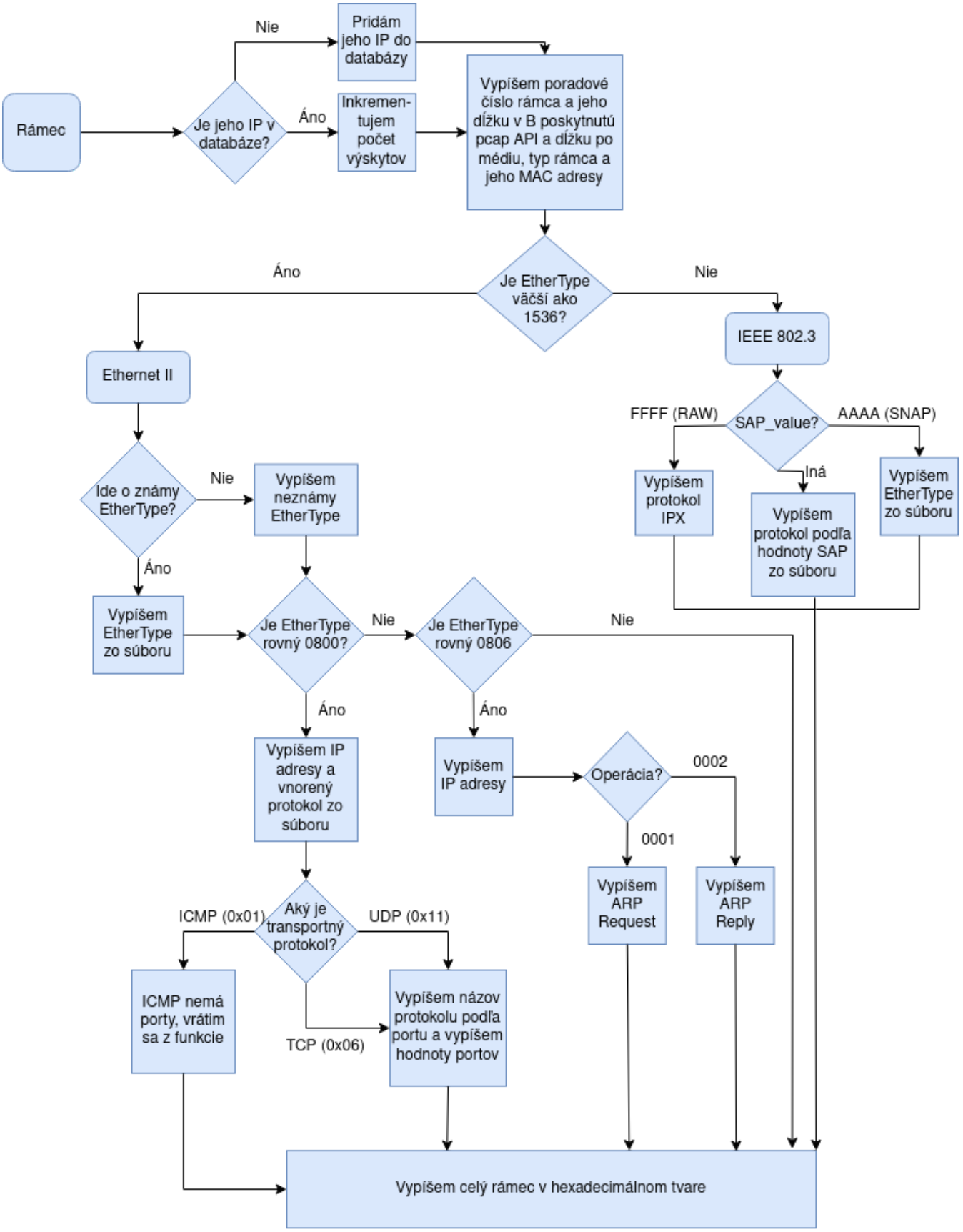
2. Analyzované vrstvy

Tento program analyzuje sieťovú komunikáciu na štyroch vrstvách (bližší popis v bode 3):

- **Linková vrstva** - uskutočňuje komunikáciu medzi dvoma fyzicky prepojenými zariadeniami, ktorej úlohou je preklad paketov do rámcov na rozposielanie po lokálnej sieti
- **Sieťová vrstva** - obaľuje segmenty do formy paketov, rozposiela ich po sieti a hľadá pre nich najvýhodnejšiu cestu k cieľu
- **Transportná vrstva** - riadi dátový tok, odosiela dáta rýchlosťou, ktorá zodpovedá rýchlosti pripojenia prijímacieho zariadenia a kontroluje, či boli údaje prijaté nesprávne a ak nie, požiada o ne znova.
- **Aplikačná vrstva** - poskytuje protokoly, ktoré umožňujú softvéru odosielať a prijímať informácie a prezentovať používateľom zmysluplné údaje.

Meno: Marko Stahovec
ID: 110897

3. Blokový návrh riešenia

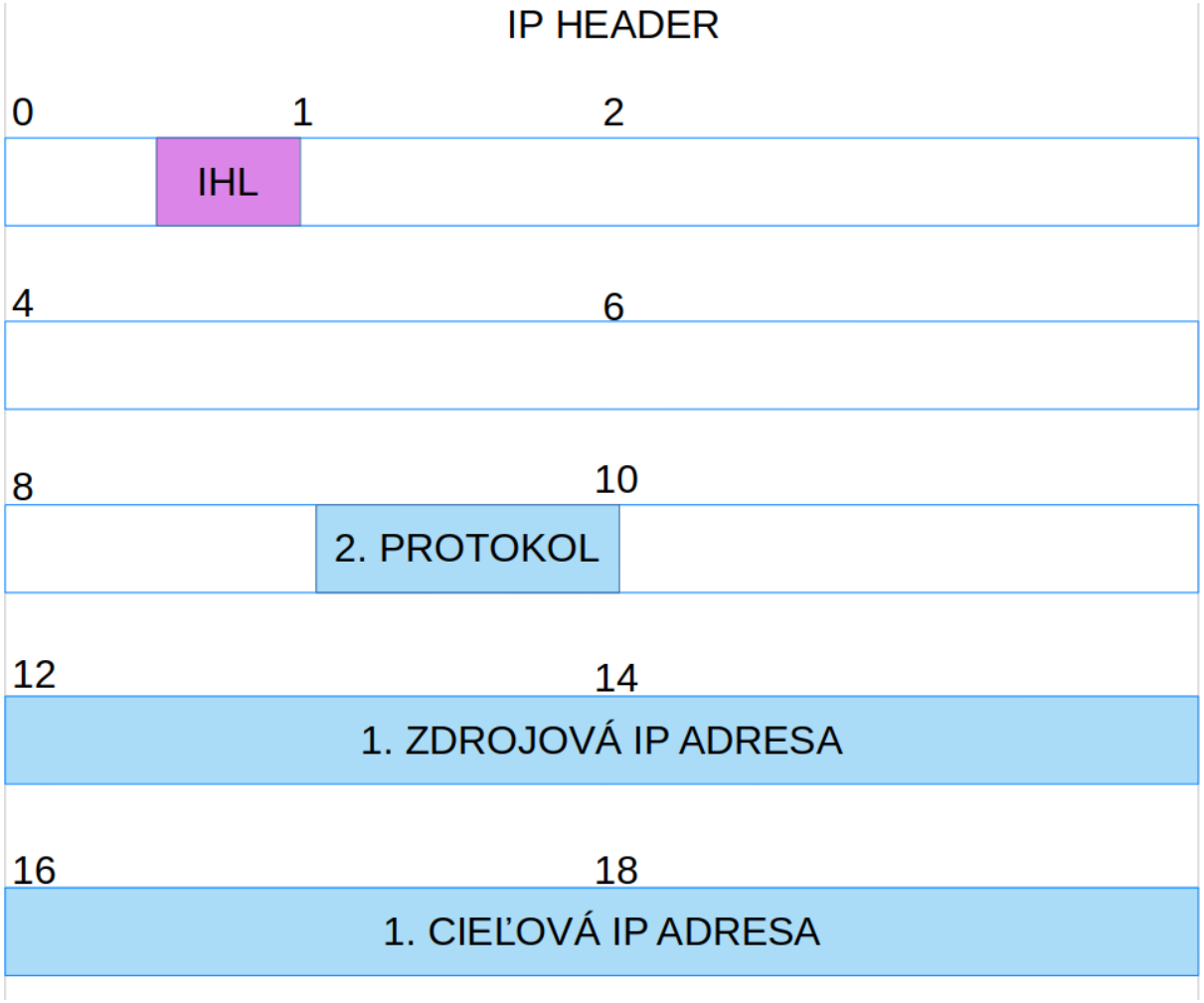


Po prejdení všetkých rámcov týmto spôsobom sa vypíše **štatistika IP adries** zo zadania, kde sa vypíše **zoznam všetkých IP adries**, ktoré odoslali aspoň jeden paket a takisto aj IP adresa, ktorá odoslala najviac paketov a zároveň aj počet všetkých paketov.
Riešenie komunikácií v danom diagrame **nie je zahrnuté**, no k tomu sa vrátim v časti 5 (**Mechanizmus analýzy komunikácií**).

4. Mechanizmus analýzy protokolov

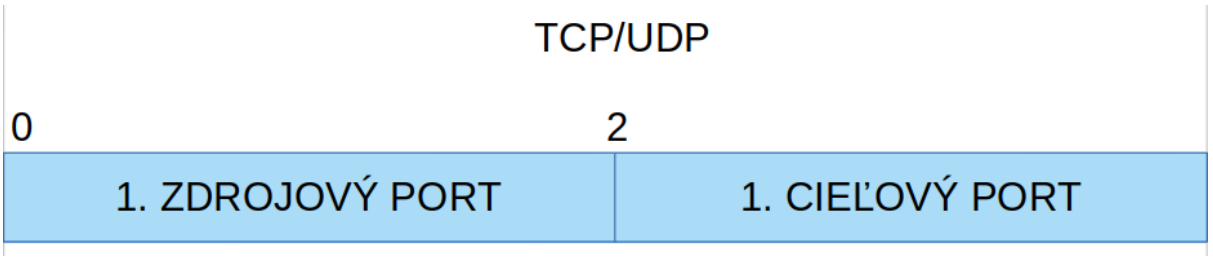
Všeobecné fungovanie programu pri vypisovaní údajov pre rámce je naznačené v časti 3, no v tejto časti bližšie popíšem **analýzu protokolov na jednotlivých vrstvách** v mojom riešení.
Analýza protokolov začína vo funkcii *start_program(data, selector)*, kde sa nachádza základný cyklus, ktorým sa **prechádzajú všetky rámce** a dekódujú sa do správneho tvaru pre analýzu. Následne sa zavolá funkcia *print_output(frame, fid)*, kde sa nachádzajú všetky funkcie na **výpis dát z rámcov**.
Na analýzu protokolov slúži ako prvá funkcia *print_frame_type(frame)*, kde začína analýza na **linkovej úrovni**. Daná funkcia (ako aj iné) využíva ďalšiu funkciu *get_header_information(frame, start, end)*, ktorá vracia ukazovateľ na miesto v rámci, odkiaľ má čítať potrebnú informáciu. *print_frame_type(frame)* teda vypíše, či je daný rámec typu **Ethernet II**, **IEEE 802.3 LLC + SNAP**, **IEEE 802.3 RAW** alebo **IEEE 802.3 LLC**. Pre Ethernet II overuje správnosť EtherTypu na 12. a 13. bajte, pre SNAP a RAW overuje Service Access Point hodnotu na 14. a 15. hodnotu. Všetko, čo nespadá do prvých troch možností, je automaticky vyhodnotené ako IEEE 802.3 LLC.
Ďalšou zastávkou v mechanizme analýzy protokolov je funkcia *print_protocol(frame)*, ktorá analyzuje **všetky protokoly od sieťovej vrstvy až po aplikačnú**. Po overení, či je daný EtherType z rodiny Ethernet II, postupuje výpisom daného EtherTypu (čo je v podstate **výpis sieťového protokolu**) a rozhodnutím, či je hodnota EtherTypu konkrétne IPv4 alebo ARP, ktoré spracúva rôznym spôsobom.
Pri IPv4 vypíše nasledovné hodnoty:

Meno: Marko Stahovec
ID: 110897

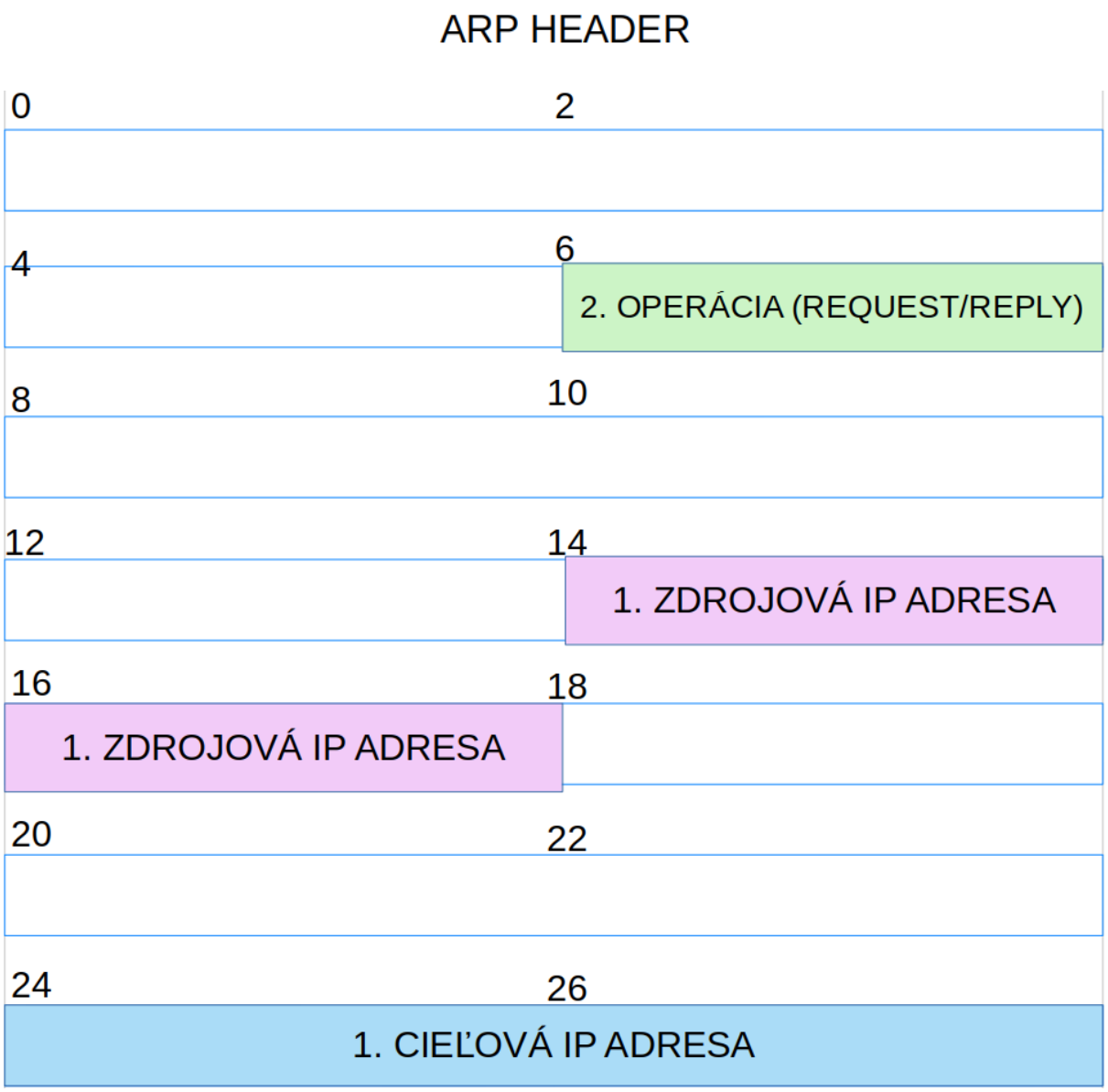


Následne program vykoná kontrolu, či je vnorený transportný protokol ICMP. Ak nie je, nasleduje výpis portov transportnej vrstvy, ktorý je rovnaký pre protokol **TCP** a **UDP**. Rozdiel je len v tom, ktorý súbor program načíta, či tcpports.txt alebo udpports.txt.

Pri správnom posune na transportnú vrstvu sa využíva aj funkcia *move_to_transport_protocol(frame)*, ktorá vráti ukazovateľ na miesto v rámci, kde **začína transportná vrstva**. Tento výpočet vykonáva z hodnoty IHL, ktorá sa nachádza na 5.-8. bite v IP hlavičke (vyznačené na predchádzajúcom obrázku). Z hodnôt týchto portov vypíše aj **vnorený protokol na aplikačnej vrstve**.



Ak je hodnota EtherTypu rovná 0806, protokol na sieťovej vrstve je **ARP**.



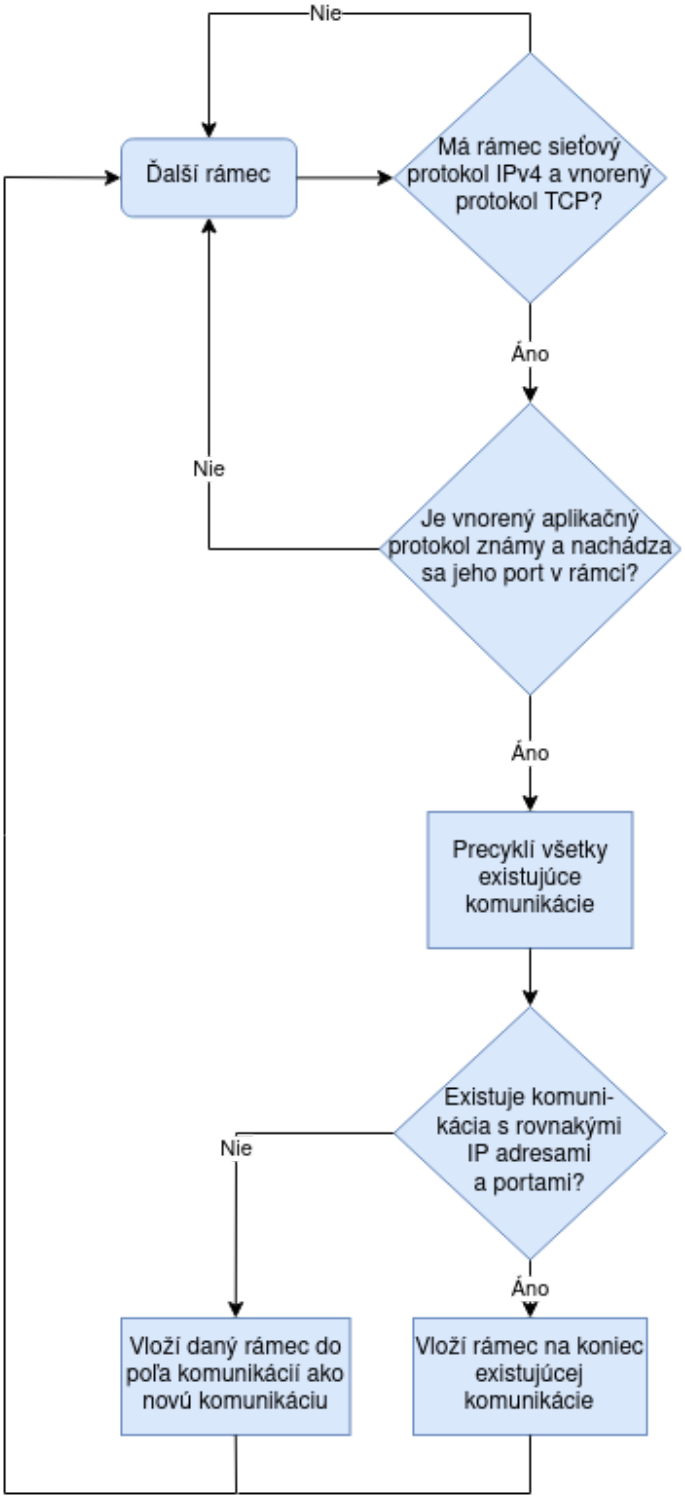
5. Mechanizmus analýzy komunikácií

Táto sekcia programu je delená na 4 časti (vo všetkých platí pravidlo, že ak je v komunikácii viac rámcov ako 20, tak sa vypíše len prvých 10 a posledných 10):

- **TCP komunikácie** (HTTP, HTTPS, Telnet, SSH, FTP riadiace a FTP dátové) - všetky zdieľajú rovnaký set funkcií
- **UDP komunikácie** - spracúva komunikácie pre TFTP protokol

- **ICMP komunikácie** - výpis ICMP komunikácií aj s ich správami
- **ARP komunikácie** - ARP kompletné a nekompletné komunikácie. Za nekompletné považujem tie, ktorým chýba korešpondujúci ARP-Request alebo ARP-Reply

a) Pre **TCP** komunikácie som vytvoril jednotný set funkcií, ktorý začína na *tcp_communication(data, selector)*. Proces selektovania rámcov a ich zaraďovania do komunikačných streamov je znázornený v nasledujúcom flow-charte.



Po spracovaní všetkých rámcov nasleduje **spracovanie komunikácií** ako celkov.

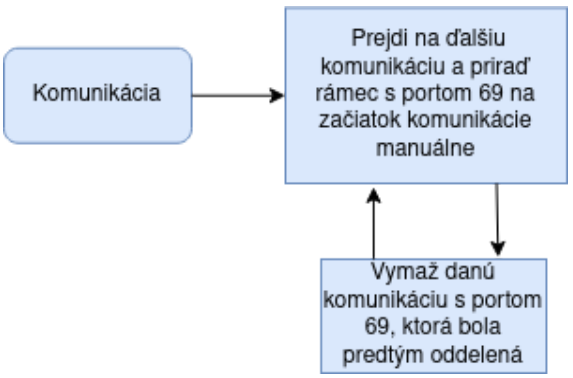
Princíp hľadania jednej kompletnej a jednej nekompletnej komunikácie spočíva v **porovnávaní flagov** v TCP hlavičke. Základom je korektné otvorenie 3-way handshakeom **>SYN [<ACK] <SYN,ACK >ACK**. Pre vyhodnotenie kompletnej komunikácie je potrebné, aby mala aj korektné ukončenie. To kontrolujem pre 4 prípady, ktoré majú nasledujúce kombinácie flagov (flagy v hranatých zátvorkách sú nepovinné):

- >**FIN**,[PSH,ACK] <**ACK** <**FIN**,[PSH,ACK] >**ACK**
- >**FIN**,[PSH,ACK] <**FIN**,[PSH,ACK] >**ACK** <**ACK**
- >**FIN**,[PSH,ACK] <**FIN**,[PSH,ACK] >**ACK**
- >**RST**,[ACK] [<RST,[ACK]]

Súčasťou kontroly, či ide o kompletnú komunikáciu, je aj **korektnosť IP adries**, ktoré sú naznačené intervalovými zátvorkami (<>). To znamená, že > je napr. odosielateľ a < je príjemca.

Ak sa nájde kompletná (alebo aj nekompletná) komunikácia, vypíše sa celý jej priebeh a prepínač complete (incomplete) sa nastaví na True, aby sa zabezpečila funkčnosť zo zadania, ktorá hovorí, aby sa vypísala len **prvá kompletná a prvá nekompletná** komunikácia.

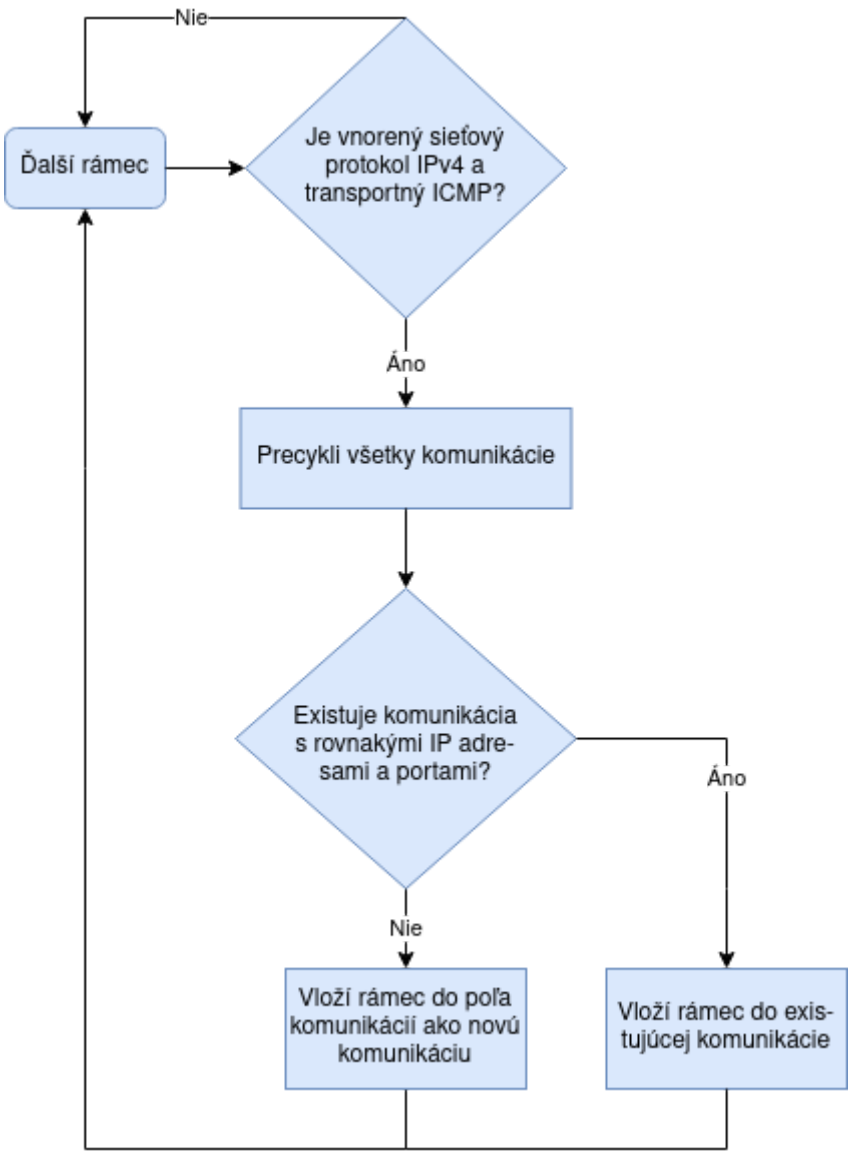
b) Pridávanie rámcov do **UDP** komunikácie je identické ako aj pri TCP s tým, že nekontroluje, či je vnorený protokol TCP, ale UDP. Tak či onak, princíp a diagram je rovnaký. Proces po precyklení všetkých rámcov sa ale mení takto:



Následne sa už len vypíšu všetky rámce v komunikáciách

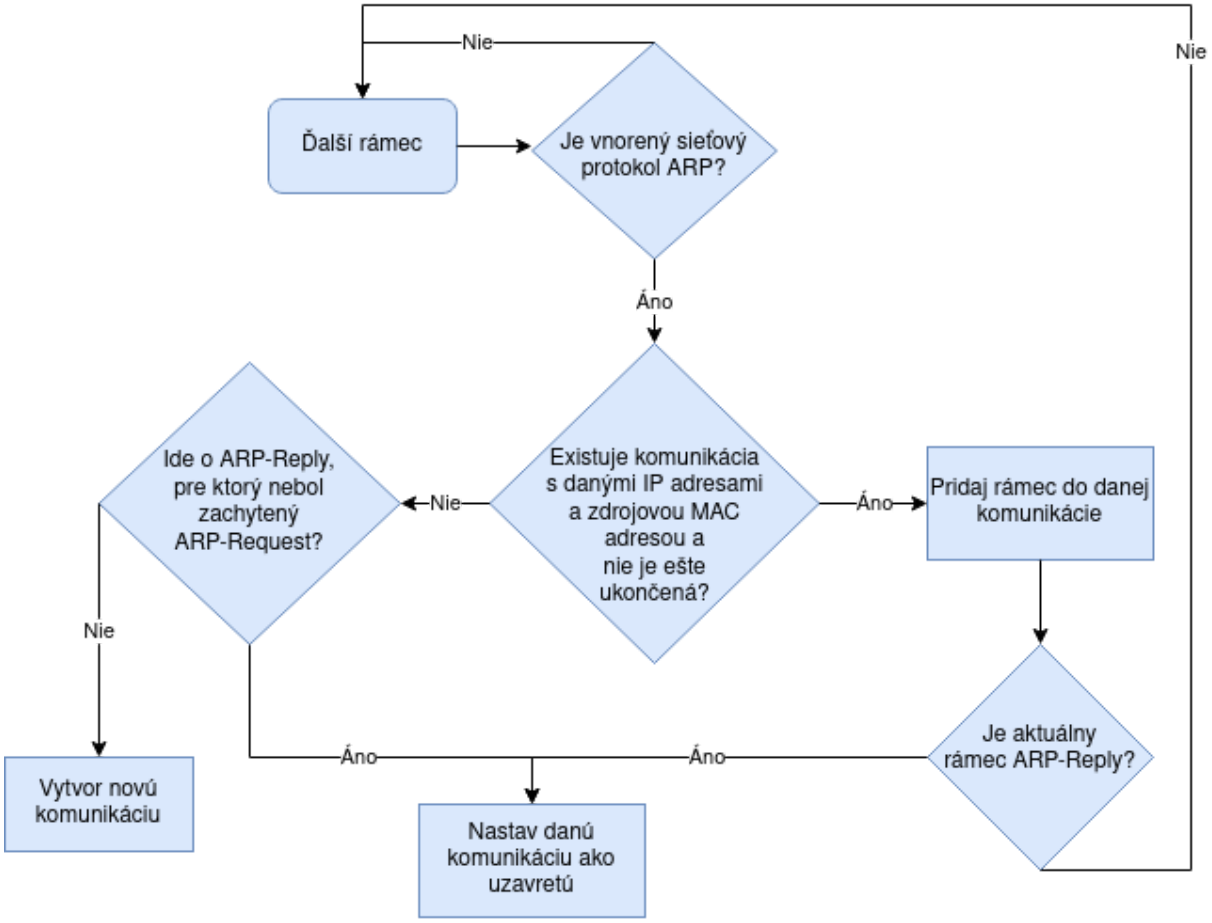
Meno: Marko Stahovec
ID: 110897

c) V rámci **ICMP** komunikácii spočíva celý princíp v takomto priebehu:



Následne po priradení všetkých rámcov do korešpondujúcich komunikácií sa **vypíšu všetky komunikácie**. Ako môžeme vidieť na diagrame, proces priradzovania do komunikácií je vo svojej podstate identický, no rozdiel nastáva pri vypisovaní komunikácií, kde pri ICMP sa nezohľadňujú nijaké flagy a vypisujú sa všetky komunikácie bez ohľadu na kompletnosť. Tento proces zabezpečí správne párovanie pre Echo request a Echo reply dvojice do streamov a ďalšie zgrupovanie rôznych rámcov.

d) **ARP** komunikácie fungujú na kompletne odlišnom princípe ako predchádzajúce tri. Nižšie je priložený diagram, ktorý zobrazuje postup pri **priradzovaní rámcov do komunikácií**.



Pri výpise ARP komunikácií je jedno rozhodovanie navyše, ktoré delí výpis podľa jedného kritéria, ktoré hovorí, aby sa najprv vypísali tie ARP komunikácie, ktoré sú **kompletné**, a teda obsahujú aspoň jeden ARP-Request a jeden ARP-Reply s rovnakými IP adresami a korešpondujúcimi MAC adresami.
Tie, ktoré sú tvorené iba requestmi a replymi, sú vypísané samostatne na konci ako bolo udané v zadaní.

6. Príklady štruktúry externých súborov

Na čítanie protokolov a rôznych hodnôt som si zvolil prístup, v ktorom rozdeľujem všetky súvisiace záležitosti do **osobitných súborov**. Hodnoty v súboroch sú

rozdelené do dvoch stĺpcov, kde prvý je hexadecimálna formát pre danú hodnotu (bez prípony 0x) a druhý stĺpec predstavuje samotnú hodnotu korešpondujúcu pre dané číslo. Taký spôsob zaručuje, že si môžem dané hodnoty načítať do slovníka veľmi jednoducho.

```
01 ICMP
02 IGMP
06 TCP
09 IGRP
11 UDP
32 ESP
58 EIGRP
59 OSPF
73 L2TP
```

Toto je súbor **ipprotocols.txt**, z ktorého si načítavam všetky známe IPv4 protokoly.

```
010b PVSTP+
0200 XEROX PUP
0201 PUP Addr Trans
0800 IPv4
0801 X.75 Internet
0805 X.25 Level 3
0806 ARP
8035 Reverse ARP
809b Appletalk
80f3 AppleTalk AARP
8100 IEEE 802.1Q VLAN-tagged frames
8137 Novell IPX
86dd IPv6
880b PPP
8847 MPLS
8848 MPLS with upstream-assigned label
8863 PPPoE Discovery Stage
8864 PPPoE Session Stage
```

Súbor **ethertypes.txt** používaný pri analýze vnoreného sieťového protokolu pre Ethernet II a IEEE 802.3 LLC + SNAP.

Meno: Marko Stahovec
ID: 110897

7. Používateľské rozhranie

Používateľské rozhranie je jednoduché - orientované len na operácie v **konzole**.

```
Analyzátor sieťovej komunikácie
Autor: Marko Stahovec
Zadaj q pre ukončenie programu
Názov súboru (napr. 'eth-2'):
```

Hneď po spustení programu sa objaví takáto správa, ktorá je samovysvetľujúca. Štvrtý riadok je samotný vstup, cez ktorý používateľ zadáva **názov súboru**, ktorý sa bude spracovávať.

```
Analyzátor sieťovej komunikácie
Autor: Marko Stahovec
Zadaj q pre ukončenie programu
Názov súboru (napr. 'eth-2'): eth-2

Zadaj c pre zmenu súboru
Zadaj q pre ukončenie programu
Zadaj 1 pre výpis prvých troch úloh
Zadaj 2 pre výpis HTTP
Zadaj 3 pre výpis HTTPS
Zadaj 4 pre výpis Telnet
Zadaj 5 pre výpis SSH
Zadaj 6 pre výpis FTP riadiacich rámcov
Zadaj 7 pre výpis FTP dátových rámcov
Zadaj 8 pre výpis TFTP
Zadaj 9 pre výpis ICMP
Zadaj 10 pre výpis ARP

Výber:
```

Po vybratí súboru (napr. eth-2) je používateľ ponúknutý niekoľkými možnosťami, či už zmenením súboru pri pomýlení sa, vypnutím celého programu alebo výberom 1-10, kde 1 je prepínač na výpis prvých troch úloh a zvyšných 9 slúži na analýzu konkrétnych protokolov.

Meno: Marko Stahovec
ID: 110897

```
Zadaj 7 pre výpis FTP dátových rámcov
Zadaj 8 pre výpis TFTP
Zadaj 9 pre výpis ICMP
Zadaj 10 pre výpis ARP

Výber: 1
Výpis do súboru? [a/n]: n
------------------------------------------------------------------------------------------
                        Analyzing file eth-2
------------------------------------------------------------------------------------------

Rámec 1
Dĺžka rámca poskytnutá pcap API - 54 B
Dĺžka rámca prenášaného po médiu - 64 B
Ethernet II
Zdrojová MAC adresa: B4 B5 2F 74 CB AE
Cieľová MAC adresa: 00 02 CF AB A2 4C
IPv4
Zdrojová IP adresa: 192.168.1.33
Cieľová IP adresa: 147.175.1.18
TCP
HTTP
Zdrojový port:  49950
Cieľový port:  80
00 02 CF AB A2 4C B4 B5  2F 74 CB AE 08 00 45 00
00 28 0E D7 40 00 80 06  00 00 C0 A8 01 21 93 AF
01 12 C3 1E 00 50 C6 DA  F9 B9 F9 68 11 5A 50 14
00 00 56 A5 00 00
```

Po výbere výpisu je užívateľovi ponúknutá možnosť **výpisu do súboru** alebo do konzole. Výpis do súboru je odporúčaný v prípade, ak je výstup príliš obsiahly, keďže konzola ho nemusí vypísať celý. Daný prepínač funguje na princípe, že do súboru “appenduje”, vďaka čomu **nebude výstup v súbore prepísaný** ani pri viacerých analýzach.
Následne už pokračuje **samotný výpis**, ktorý bol nakonfigurovaný užívateľovým výberom prepínačov.

```
IP adresy vysielajúcich uzlov:
192.168.1.33
2.20.182.123
173.194.70.190
173.194.44.39
147.175.1.18
173.252.110.27
147.251.48.205

Adresa uzla s najväčším počtom odoslaných paketov:
192.168.1.33
118 paketov

end

Zadaj c pre zmenu súboru
Zadaj q pre ukončenie programu
Zadaj 1 pre výpis prvých troch úloh
Zadaj 2 pre výpis HTTP
Zadaj 3 pre výpis HTTPS
Zadaj 4 pre výpis Telnet
Zadaj 5 pre výpis SSH
Zadaj 6 pre výpis FTP riadiacich rámcov
Zadaj 7 pre výpis FTP dátových rámcov
Zadaj 8 pre výpis TFTP
Zadaj 9 pre výpis ICMP
Zadaj 10 pre výpis ARP

Výber:
```

Po výpise sa užívateľ ocitne znova v menu, kde môže pokračovať v ďalších analýzach podľa jeho želania.

Meno: Marko Stahovec
ID: 110897

8. Implementačné prostredie

Na implementáciu tohto zadania som si zvolil jazyk **Python**, ktorý mi vďaka svojej jednoduchosti a prístupnosti pripadal ako najlepšia voľba. Veľké množstvo informácií na dohľadanie a fakt, že toto zadanie nie je výpočtovo náročné ma presvedčili, že Python by mal byť najlepšou voľbou.
Tento projekt som realizoval vo vývojovom programovacom prostredí od spoločnosti Jetbrains - **Pycharme**.
Z externých knižníc som použil len dve:

- **scapy** - na otvorenie .pcap súboru
- **binascii** - na prekonvertovanie byte streamu na hexadecimálny formát.

Ďalej som použil aj **os.path** na overenie existencie externého súboru a **codecs** na prekonvertovanie bajtov.

9. Záver

Myslím si, že moje riešenie vyhovuje všetkým podmienkam a je dostatočne čitateľné a robustné, keďže používateľské rozhranie je samovysvetľujúce a samotný program primerane efektívny vzhľadom na všetky aspekty zadania.