|  |
| --- |
| Slovenská Technická Univerzita v Bratislave, Fakulta informatiky a informačných technológií |
| Analyzátor sieťovej komunikácie |
| Počítačové a komunikačné siete |

|  |
| --- |
| Norbert Matuška  2023/24 |

Contents

[Zadanie úlohy 1](#_Toc148382911)

[Diagram pre čítanie zo súboru 2](#_Toc148382912)

[Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách 3](#_Toc148382913)

[Úloha 1 až 3 3](#_Toc148382914)

[Úloha 4 3](#_Toc148382915)

[TFTP 3](#_Toc148382916)

[Diagram pre fungovanie TFTP 4](#_Toc148382917)

[ARP 5](#_Toc148382918)

[Diagram pre fungovanie ARP 6](#_Toc148382919)

[ICMP + frag 7](#_Toc148382920)

[Diagram pre fungovanie ICMP + frag 8](#_Toc148382921)

[Príklad štruktúry externých súborov pre určenie protokolov a portov 9](#_Toc148382922)

[Opísanie používateľského prostredia 10](#_Toc148382923)

[Voľba implementačného prostredia 11](#_Toc148382924)

[Zhodnotenie a možnosti rozšírenia 11](#_Toc148382925)

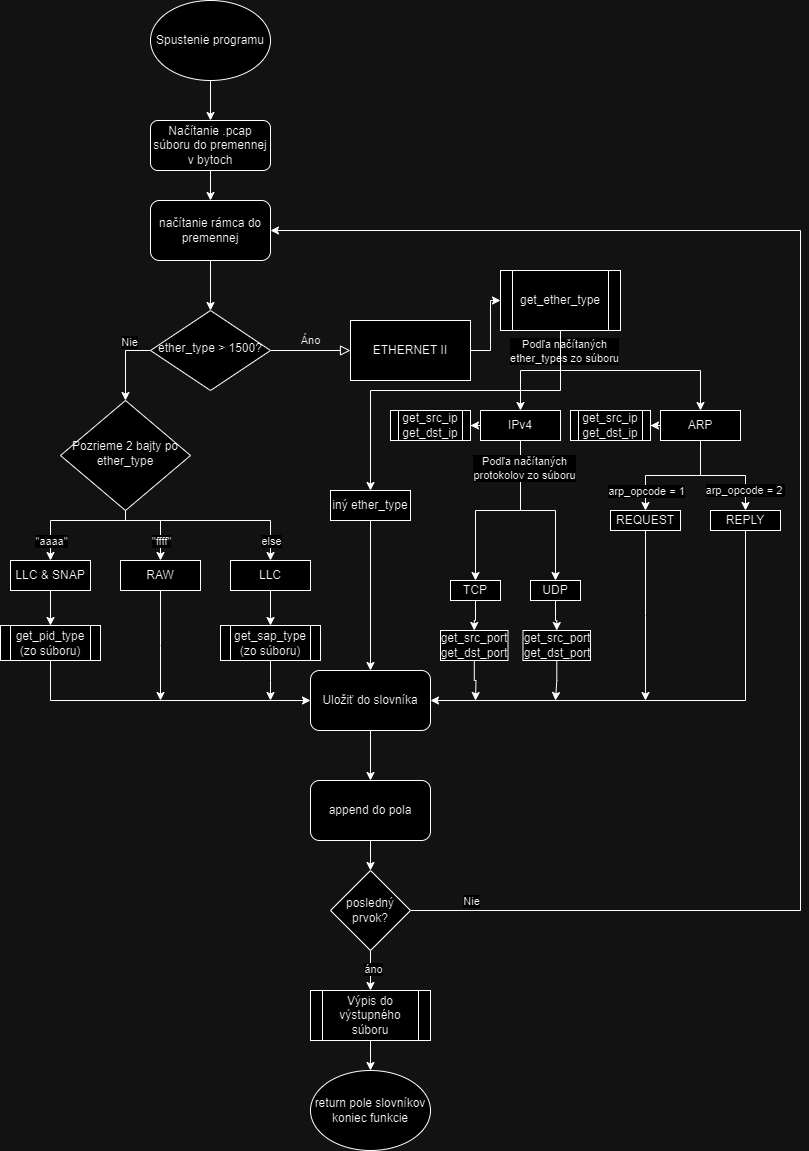
[Zdroje 12](#_Toc148382926)

# Zadanie úlohy

Navrhnite a implementujte programový analyzátor Ethernet siete, ktorý analyzuje 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.
2. Výpis IP adries a vnorených protokol na 2-4 vrstve pre rámce Ethernet II.
3. Na konci výpisu z úlohy 2 uveďte pre IPv4 packety nasledujúcu štatistiku:
   1. Zoznam IP adries všetkých odosielajúcich uzlov a koľko paketov odoslali.
   2. IP adresu uzla, ktorý sumárne odoslal (bez ohľadu na prijímateľa) najväčší počet paketov a koľko paketov odoslal, ak ich je viac, tak uviesť všetky uzly.
4. Váš program rozšírte o analýzu komunikácie pre vybrané protokoly:
   1. TCP
   2. TFTP
   3. ICMP
   4. ARP
   5. Fragmentované ICMP
5. Súčasťou riešenia je aj dokumentácia

# Diagram pre čítanie zo súboru



# Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách

## Úloha 1 až 3

Po spustení programu zavolám funkciu, v ktorej sa vykonáva úloha 1 až 3. V tejto funkcii si na začiatku zavolám funkciu read\_file(), ktorá mi načíta obsah .pcap súboru do premennej packet\_bytes. Ďalej inicializujem objekt YAML pre neskoršie použitie, packets\_array čo bude pole slovníkov do ktorého budem ukladať spracované packety a získam plnú cestu k súboru (to je iba kvôli tomu aby som mohol vypísať názov testovacieho súboru aj do konzoly a aj do výstupného súboru). Vytvorím si taktiež inštanciu objektu IP\_Address\_Counter, ktorá počíta a ukladá ip adresy pre štatistiku z 3. úlohy. Ďalej iterujem cez každý jeden packet a vyťahujem potrebné informácie. Ak je ether\_type väčší ako 1500 viem, že to je ETHERNET II a môžem ďalej skúmať ether\_type. Ak je ether\_type IPv4, určím IPv4 protokol, a ak je to UDP alebo TCP, určím pri nich src\_port a dst\_port samozrejme s tým, že počítam dĺžku ip headeru. Ak je ether\_type ARP, určím či sa jedná o REQUEST alebo REPLY a taktiež určím src\_port a dst\_port. Ak ale ether\_type nieje väčší ako 1500 tak to bude IEEE 802.3. Tým pádom sa pozerám DSAP a SSAP a ak ich hexadecimálny tvar je „aaaa“, je to LLC & SNAP, alebo ak tam nájdem „ffff“, bude to RAW. Ak to nie je ani jedno z toho, bude to LLC. Nakoniec si celý packet uložím do slovníka a slovník appendnem do listu už analyzovaných packets. Po skončení iterovania vypíšem všetky zanalyzované packety do výstupného .yaml súboru spolu aj so štatistikou z 3. úlohy. Funkcia nakoniec vráti pole spracovaných packetov pre ďalšie použitie pri filtroch z úlohy 4.

## Úloha 4

### TFTP

Na začiatku vyberiem všetky packety z pola analyzovaných packetov, ktoré majú protokol UDP. Ďalej iteratívne prechádzam cez pole a porovnávam tftp\_opcode.

Ak nájdem 1, znamená to READ REQUEST a teda začiatok komunikácie, ktorú priradím do pola current\_communication.

Následne sa pozerám na ďalšie packety v UDP poli, kontrolujem src a dst ip, ak je zhoda pozriem či packeta má data opcode a hľadám k nej ACK.

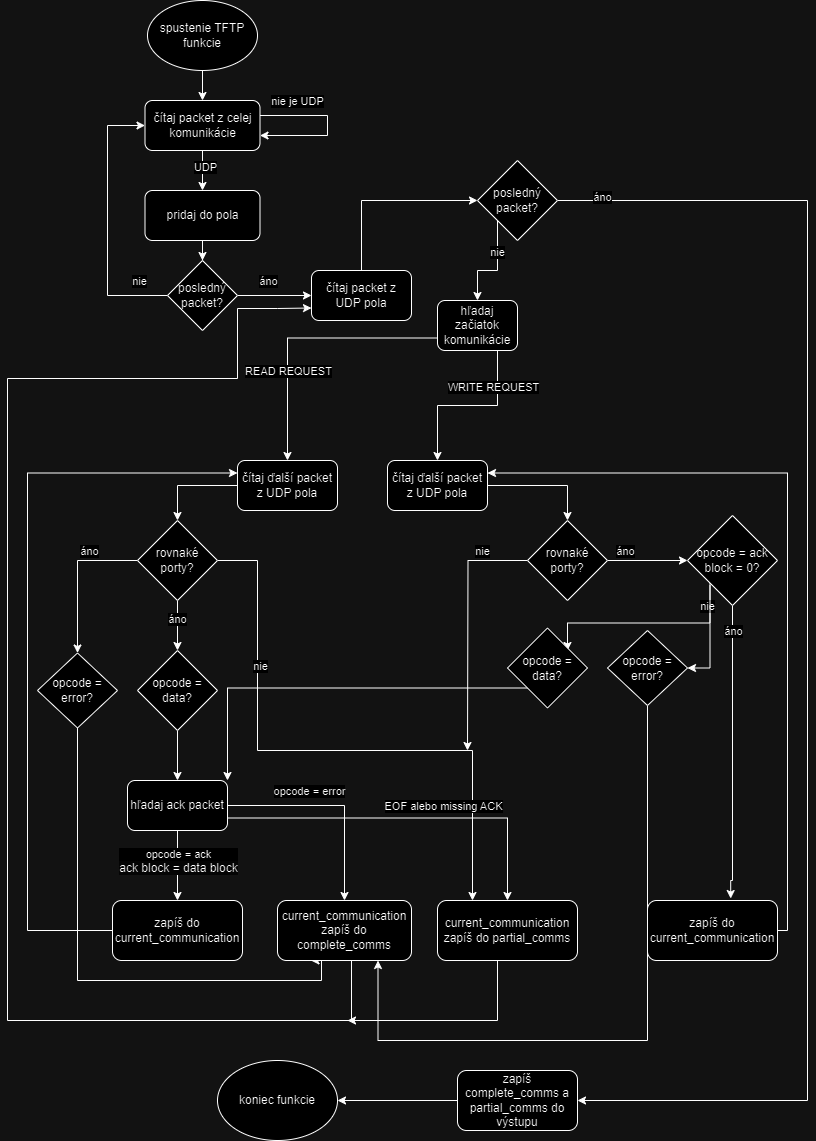
Ak sa nájde ACK k dátovej packete, pridám ju do current\_communication. Ak je zároveň posledná dátová packeta menšia ako predošlá dátova packeta, znamená to ukončenie komunikácie a priraďujem current communication do complete communications

Ak sa nájde packeta s error opcode, znamená to ukončenie komunikácie a priraďujem current communication do complete communications

Ak sa nenájde ani ACK packeta, ani error packeta, priraďujem current communication do partial communications.

Ak nájdem 2, znamená to WRITE REQUEST a teda začiatok komunikácie, ktorú priradím do pola current\_communication. Pri WRITE REQUEST to funguje podobne až na to že úplne na začiatku hľadám najprv ACK na WRITE REQUEST a potom to funguje na rovnakom princípe ako pri READ REQUEST.

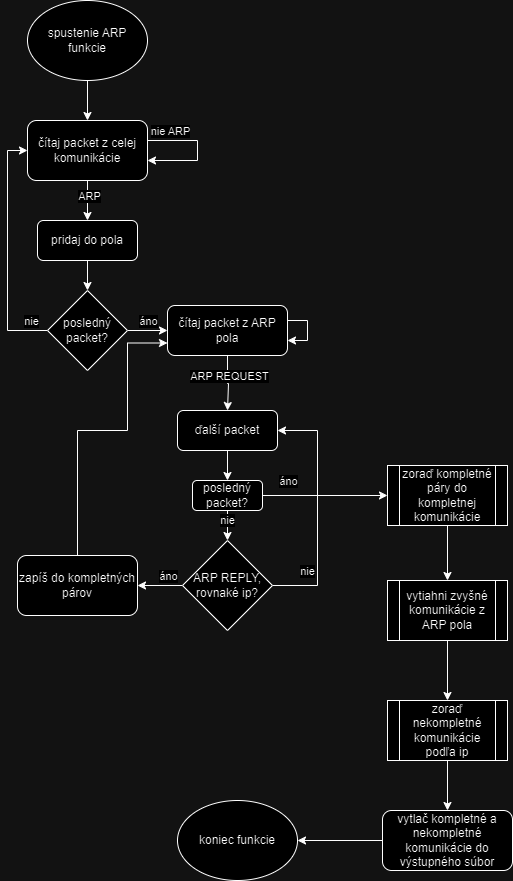
### Diagram pre fungovanie TFTP



### ARP

Na začiatku vyberiem všetky packety z pola analyzovaných packetov, ktoré majú protokol ARP. Následne spárujem všetky ARP requesty s ARP reply podľa ip komunikácie. Ďalej tieto páry spájam do kompletnej komunikácie taktiež podľa ip. Potom si vyberiem všetky zvyšné ARP komunikácie, vložím ich do pola nezoradených nekompletných komunikácii a prechádzam týmto polom a zoraďujem podľa ip do nekompletných komunikácií. Na konci vytlačím kompletné a nekompletné komunikácie do výstupného yaml súboru.

### Diagram pre fungovanie ARP



### ICMP + frag

Na začiatku vyberiem všetky packety z pola analyzovaných packetov, ktoré majú protokol ICMP. Prejdem celým ICMP polom a určím tam rôzne potrebné dáta ako napr. icmp\_id, icmp\_seq, flag, frag\_offset atď. Následne prechádzam celým polom a hľadám páry ECHO REQUEST a ECHO REPLY alebo ECHO REQUEST a TIME EXCEEDED podľa id, seqn. Ďalej tieto páry spojím do kompletných komunikácií podľa ip. Potom už len vytiahnem zvyšné komunikácie, ktoré nie sú v kompletných a zároveň majú nejaké icmp id a zoradím ich do nekompletných komunikácií. Komunikácie, ktoré ostali v ICMP poli môžu byť jedine fragmentované packety bez icmp id a tie ďalej porovnávam s ip id kompletných a nekompletných komunikácií a pripájam ich k jednotlivým packetom kde patria. Na konci už len vypíšem všetko zaradom.

### Diagram pre fungovanie ICMP + frag

A screenshot of a computer flowchart

Description automatically generated

# Príklad štruktúry externých súborov pre určenie protokolov a portov

Takto vyzerá môj súbor pre určenie TCP protokolu:

20 FTP-DATA  
21 FTP-CONTROL  
22 SSH  
23 TELNET  
25 SMTP  
53 DNS  
80 HTTP  
110 POP3  
119 NNTP  
137 NETBIOS-NS  
139 NETBIOS-SSN  
143 IMAP  
162 SNMP-TRAP  
179 BGP  
389 LDAP  
443 HTTPS  
514 SYSLOG  
17500 DB-LSP-DISC

Pre ICMP vyzerá zase troška inak ale na podobnom princípe

0:ECHO REPLY  
3:DESTINATION UNREACHABLE  
4:SOURCE QUENCH  
5:REDIRECT  
8:ECHO REQUEST  
9:ROUTER ADVERTISEMENT  
10:ROUTER SELECTION  
11:TIME EXCEEDED  
12:PARAMETER PROBLEM  
13:TIMESTAMP  
14:TIMESTAMP REPLY  
15:INFORMATION REQUEST  
16:INFORMATION REPLY  
17:ADDRESS MASK REQUEST  
18:ADDRESS MASK REPLY  
30:TRACEROUTE

# Opísanie používateľského prostredia

Funkcia na začiatku vykoná úlohy 1 až 3 a následne sa spýta usera na výber filtra

A screenshot of a computer

Description automatically generated

Testovací súbor sa dá vybrať hneď na začiatku kódu vo funkcii:

def testing\_file():  
 file\_path = "vzorky\_pcap\_na\_analyzu/"  
 file\_name = "trace-26.pcap"  
 full\_file\_path = file\_path + file\_name  
  
 return full\_file\_path

Do premennej file\_name napíšeme súbor, ktorý chceme testovať. Musí existovať.

# Voľba implementačného prostredia

Môj program som písal v jazyku Python v PyCharm IDE od JetBrains. Tento jazyk som si vybral kvôli zjednodušeniu práce s bajtmi a aby som využil jednoduché vstavané funkcie v Pythone ako je hexlify, int a str funkcie. Následne som využil knižnice ruamel.yaml a scapy kvôli rdpcap.

# Zhodnotenie a možnosti rozšírenia

Môj program nie je úplne najefektívnejšie spracovaný, ako možnosť rozšírenia by som celý program zefektívnil, pretože niektoré funkcie sa vykonávajú zbytočne/neefektívne. Taktiež by som program rozšíril o TCP filter, keby som mal viac času na pracovaní.

# Zdroje

Diagramy spracované na stránke:

<https://app.diagrams.net/>