Autor: Martin Pirkovský
AIS ID: 103095

FIIT STU

Ilkovičova 2, 842 16 Karlova Ves

Analyzátor sieťovej komunikácie

Autor: Martin Pirkovský

2020/2021

Autor: Martin Pirkovský
AIS ID: 103095

1) Zadanie úlohy:

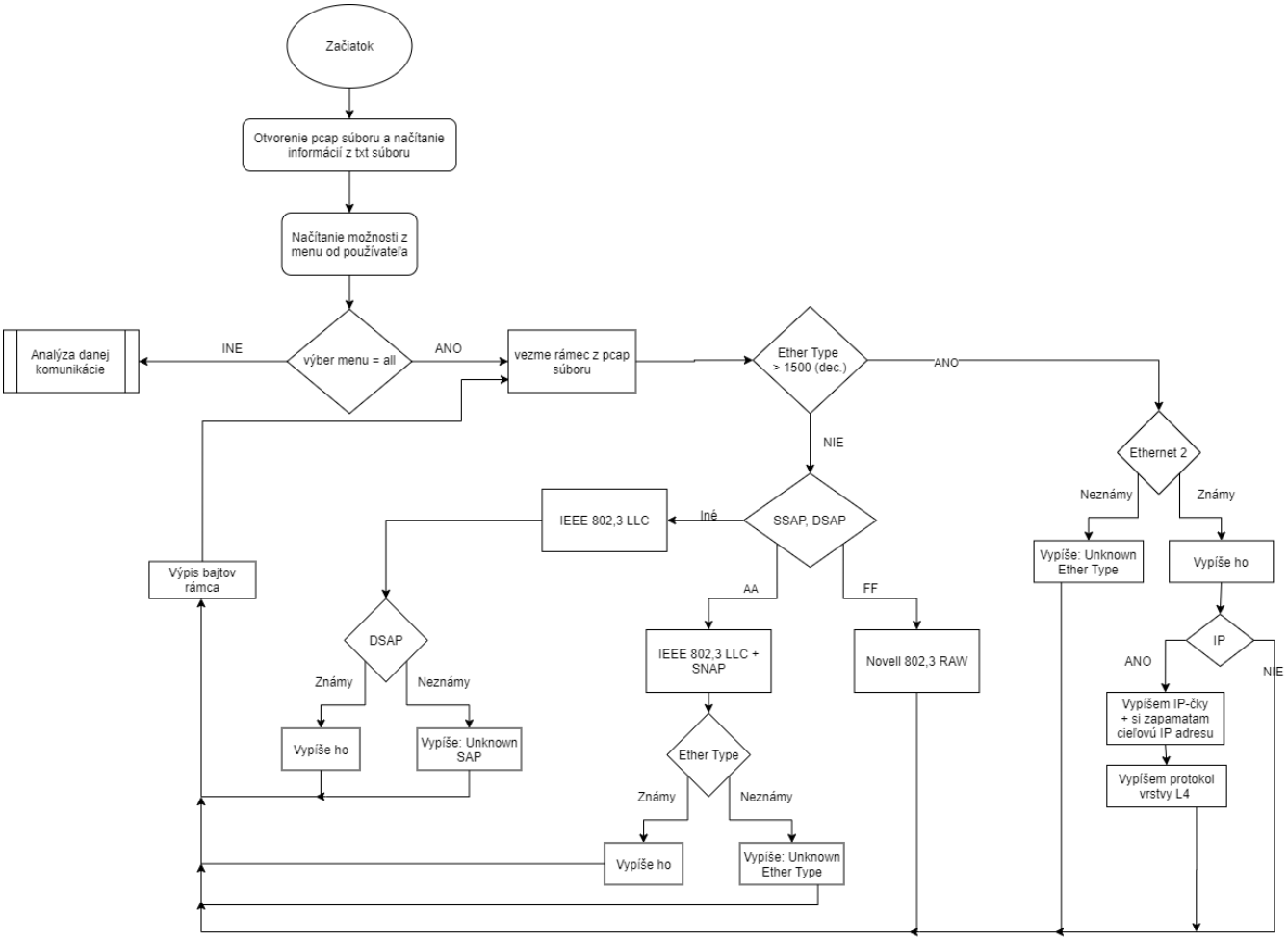
Navrhnite a implementujte programový analyzátor Ethernet siete, ktorý analyzuje komunikácie v sieti zaznamenané v pcap súbore a poskytuje nasledujúce informácie o komunikáciách.
Vypracované zadanie musí spĺňať nasledujúce body:
1) **Výpis všetkých rámcov v hexadecimálnom tvare** postupne tak, ako boli 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.
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 prijímajúcich uzlov,
b) IP adresu uzla, ktorý sumárne prijal (bez ohľadu na odosielateľa) najväčší počet paketov a koľko paketov prijal (berte do úvahy iba IPv4 pakety).
IP adresy a počet poslaných 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 ARPReply bez ARP-Request), vypíšte ich samostatne.
Vo všetkých výpisoch treba uviesť aj IP adresy a pri transportných protokoloch TCP a UDP aj porty komunikujúcich uzlov.
V prípadoch komunikácií so spojením vypíšte iba jednu kompletnú komunikáciu - obsahuje otvorenie (SYN) a ukončenie (FIN na oboch stranách alebo ukončenie FIN a RST alebo ukončenie iba s RST) spojenia a aj prvú nekompletnú komunikáciu, ktorá obsahuje iba otvorenie spojenia. Pri výpisoch vyznačte, ktorá komunikácia je kompletná.
Ak počet rámcov komunikácie niektorého z protokolov z bodu 4 je väčší ako 20, vypíšte iba 10 prvých a 10 posledných rámcov tejto komunikácie. **(Pozor: toto sa nevzťahuje na bod 1, program musí byť schopný vypísať všetky rámce zo súboru podľa bodu 1.)** Pri všetkých výpisoch musí byť poradové číslo rámca zhodné s číslom rámca v analyzovanom súbore.

Autor: Martin Pirkovský
AIS ID: 103095

5) Program musí byť organizovaný tak, aby čísla protokolov v rámci Ethernet II (pole Ethertype), IEEE 802.3 (polia DSAP a SSAP), v IP pakete (pole Protocol), ako aj čísla portov v transportných protokoloch boli programom **načítané z jedného alebo viacerých externých textových súborov.** Pre známe protokoly a porty (minimálne protokoly v bodoch 1) a 4) budú uvedené aj ich názvy. Program bude schopný uviesť k rámcu názov vnoreného protokolu po doplnení názvu k číslu protokolu, resp. portu do externého súboru. Za externý súbor sa nepovažuje súbor knižnice, ktorá je vložená do programu.
6) V procese analýzy rámcov pri identifikovaní jednotlivých polí rámca ako aj polí hlavičiek vnorených protokolov nie je povolené použiť funkcie poskytované použitým programovacím jazykom alebo knižnicou. **Celý rámec je potrebné spracovať postupne po bajtoch.**
7) Program musí byť organizovaný tak, aby bolo možné jednoducho rozširovať jeho funkčnosť výpisu rámcov pri doimplementovaní jednoduchej funkčnosti na cvičení.
8) Študent musí byť schopný preložiť a spustiť program v miestnosti, v ktorej má cvičenia. V prípade dištančnej výučby musí byť študent schopný prezentovať podľa pokynov cvičiaceho program online, napr. cez Webex, Meet, etc.
V danom týždni, podľa harmonogramu cvičení, musí študent priamo na cvičení doimplementovať do funkčného programu (podľa vyššie uvedených požiadaviek) ďalšiu prídavnú funkčnosť.
Program musí mať nasledovné vlastnosti (minimálne):
1) Program musí byť implementovaný v jazykoch C/C++ alebo Python s využitím knižnice pcap, skompilovateľný a spustiteľný v učebniach. Na otvorenie pcap súborov použite knižnice *libpcap* pre linux/BSD a *winpcap/ npcap* pre Windows. Použité knižnice a funkcie musia byť schválené cvičiacim. V programe môžu byť použité údaje o dĺžke rámca zo struct pcap_pkthdr a funkcie na prácu s pcap súborom a načítanie rámcov:
pcap_createsrcstr()
pcap_open()
pcap_open_offline()
pcap_close()
pcap_next_ex()
pcap_loop()
Použitie funkcionality *libpcap* na priamy výpis konkrétnych polí rámca (napr. ih->saddr) bude mať za následok nulové hodnotenie celého zadania.
2) Program musí pracovať s dátami optimálne (napr. neukladať MAC adresy do 6x int).
3) Poradové číslo rámca vo výpise programu musí byť zhodné s číslom rámca v analyzovanom súbore.
4) Pri finálnom odovzdaní, pre každý rámec vo všetkých výpisoch uviesť použitý protokol na 2. - 4. vrstve OSI modelu. (ak existuje)
5) Pri finálnom odovzdaní, pre každý rámec vo všetkých výpisoch uviesť zdrojovú a cieľovú adresu / port na 2. - 4. vrstve OSI modelu. (ak existuje)
Nesplnenie ktoréhokoľvek bodu minimálnych požiadaviek znamená neakceptovanie riešenia cvičiacim.

Autor: Martin Pirkovský
AIS ID: 103095

2) Blokový návrh (koncepcia) fungovania riešenia spolu s návrhom mechanizmu analyzovania protokolov na jednotlivých vrstvách

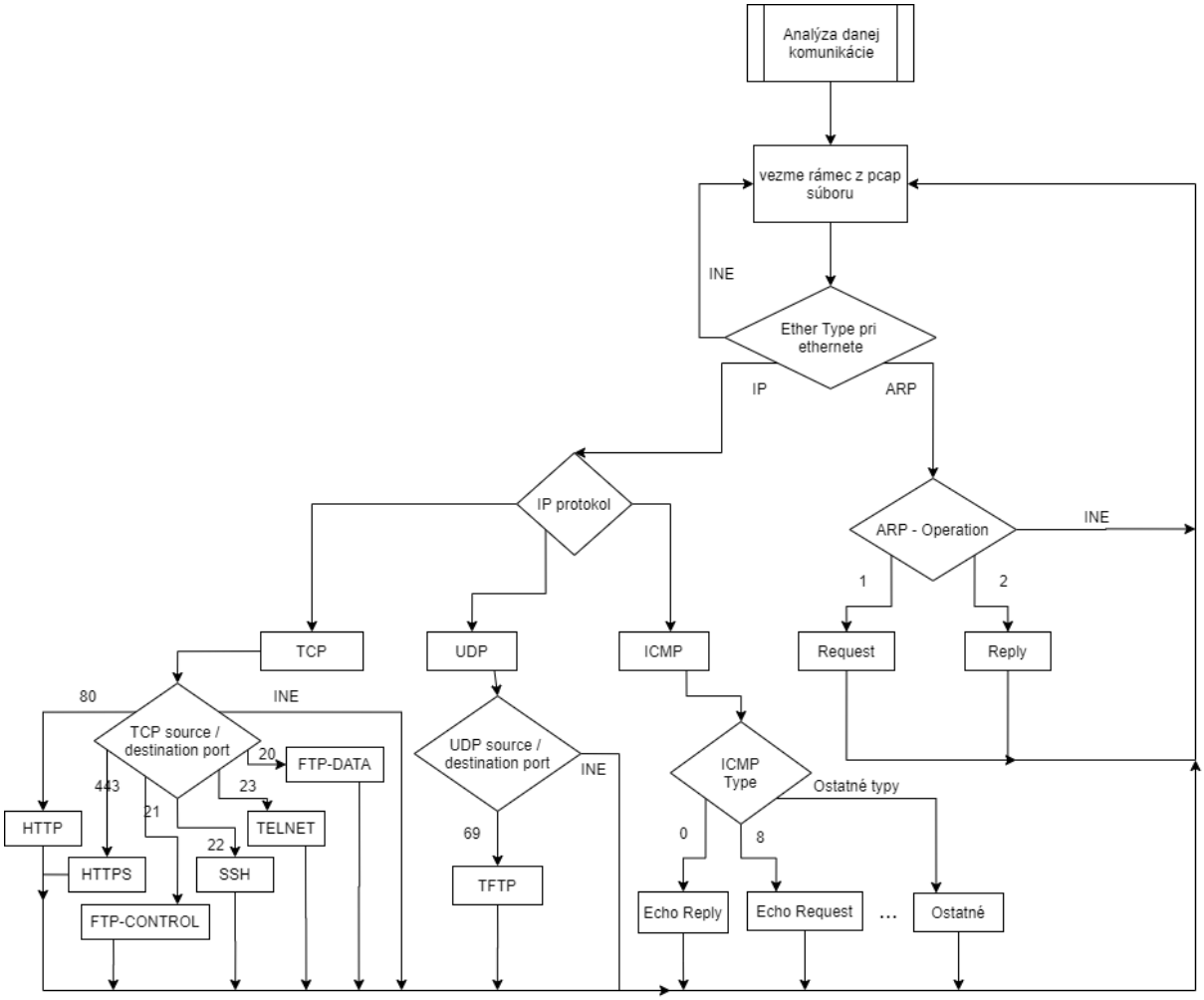


Na začiatku ako prvé si načítam z externého súboru potrebné dáta akými sú napríklad Ethertype-y, LLC SAP typy, protokoly v IP hlavičke a ďalšie, spolu s tým načítam aj pcap súbor, ktorý užívateľ zadal, zároveň si užívateľ vyberie, či chce výstup do súboru alebo do konzoly. Ďalej si užívateľ vyberie jednu z možností a po správnom zadaní program začne vykonávať danú vec. Ak si užívateľ vybral možnosť „all“, tak prebehne celý výpis bodov 1-3 zo zadania, teda program zistí, či ide pri jednotlivých rámcoch o Ethernet alebo o jednu z 802,3.

Ako prvé vypíšem číslo rámca, ktoré je nasledované informáciou o veľkosti rámca. Tú zistím za pomoci len funkcie v pythone. Ak je dĺžka prenášaného po médiu menšia ako 64 vypíšem 64 B. Ďalej zistím o aký typ protokolu ide na linkovej vrstve. Pri 802,3 na základe DSAP-u a SSAP-u zistím, či ide o Novell (na mieste DSAP a SSAP sa nachádza FFFF), LLC + SNAP (na mieste DSAP sa nachádza AA) alebo o jeden zo SAP typov z tabuľky. Pri SNAP-e sa ešte pokúsim zistiť, či sa tam nachádza nejaký známy Ethertype a vypísať ho. Pri neúspechu vypíšem „Unknown Ethertype“. Pri známom SAP type z tabuľky vypíšem o aký vnorený protokol sa jedná. Na konci ešte vypíšem celý rámec po bajtoch.

Autor: Martin Pirkovský
AIS ID: 103095

Ak sa jedná o Ethernet II, program rovnako ako pri 802,3 sa pokúsi zistiť vnorený protokol a vypísať ho. Ak nájde známy Ethertype (políčko Ethertype obsahuje číslo väčšie ako 1500) vypíšem ho. V prípade, že sa jedná o Internet Protokol, tak vypíšem aj IP adresy spolu s protokolom vrstvy L4 a uložím si cieľovú IP adresu, ak je to jedinečná adresa, akú ešte nemám v zozname IP adries, ktorý využívam pre bod 3 zo zadania. Nakoniec vypíšem celý rámec po bajtoch. Keď som už prešiel celý pcap súbor, tak ešte vypíšem všetky jedinečné cieľové IP adresy, spolu s najpočetnejšou IP adresou, ku ktorej uvediem aj jej početnosť.



Ak si užívateľ na začiatku zvolí zobraziť danú komunikáciu, tak postupujem nasledovne. Ak si vybral jednu z TCP komunikácií, tak najprv vyfiltrujem dané rámce s daným protokolom. Tie následne pozgrupujem podľa socketov a potom keď už mám dané komunikácie pri sebe, tak už iba označím, ktoré sú kompletné (majú 3-way SYN handshake na začiatku a končia buď 4 alebo 3-way FIN handshakom alebo končia resetom), a ktoré sú nekompletné (majú iba 3-way SYN handshake na začiatku). Na to sledujem v TCP protokole políčko „Flags“, pričom FIN reprezentuje 1, SYN reprezentuje 2, RST je 4 a ACK je 16. Pre kombináciu flagov tieto čísla zrátam a má sa zhodovať s číslom, ktoré reprezentuje „Flags“ políčko. Všetky rámce s well-known portom na TCP (HTTP, HTTPS, FTP-CONTROL, FTP-DATA, SSH, TELNET), ktoré máme

analyzovať, fungujú na rovnakom princípe a teda jediné čo sa v postupe mení je protokol na aplikačnej vrstve, na základe ktorého filtrujem dané rámce ako som už spomenul vyššie.

Otvorenie spojenia: 3-way SYN handshake: 1. rámec má iba SYN

2. rámec má SYN+ACK

3. rámec má iba ACK

Zatvorenie spojenia: 4-way FIN handshake (ktoré má „n“ rámcov):

n-3. rámec má FIN+ACK

n-2. rámec má iba ACK

n-1. rámec má FIN+ACK

n. rámec má iba ACK

3-way FIN handshake (ktoré má „n“ rámcov):

n-2. rámec má FIN+ACK

n-1. rámec má FIN+ACK

n. rámec má iba ACK

RST zatvorenie:

n. rámec má RST alebo RST+ACK

Ak chce užívateľ zobraziť TFTP komunikáciu, tak program pracuje nasledovne. Najprv hľadám rámec s UDP protokolom na transportnej vrstve, ktorý má port s „well-known“ hodnotou 69, ktorá označuje TFTP a zároveň označuje začiatok novej TFTP komunikácie, nakoľko má opcode 1, čo je read request. Zapamätám si zdrojový port, na základe ktorého viem určiť, či sa nejaký ďalší rámec s UDP protokolom týka danej komunikácie, nakoľko cieľový port daného UDP protokolu musí byť rovnaký ako môj zdrojový. Keď takýto nájdem zapamätám si zdrojový port a uložím si ho pretože odteraz až po koniec bude komunikácia prebiehať na týchto dvoch portoch (port 69 sa v danej komunikácií už neobjaví). Ďalšiu vec, ktorú si zároveň uložím pri príchode druhého rámca z danej komunikácie je údaj „length“ z UDP protokolu, ak sa opcode (tento údaj sú prvé dva bajty hneď ako skončí UDP hlavička) rovná 3, čo nám značí data. Táto dĺžka sa na data protokoloch komunikácie nebude meniť počas celej komunikácie až pokiaľ táto dĺžka nebude zrazu menšia, čo nám značí, že komunikácia končí a už do nej patrí iba jeden rámec s hodnotou 4 na políčku opcode, čo značí „Acknowledgement“. Komunikácia ešte môže skončiť, ak je na políčku opcode hodnota 5, čo znamená „Error“, a teda komunikácia zlyhala, vo väčšine prípadov už komunikácia neprijíma ďalšie rámce a pokladá sa za ukončenú.

Ďalej si užívateľ môže vybrať ICMP komunikáciu. Pri tej si za pomoci filtra vyhľadám Ethernet II, IP protokol a v ňom ICMP. Všetky potrebné informácie o rámci, ako MAC adresy, IP adresy a pod. vypíšem a pridám k tomu aký „type“ sa nachádza v ICMP hlavičke. Tieto všetky rámce vypíšem a to je všetko k danému protokolu.

Poslednou voľbou v menu je výpis ARP komunikácie. Prechádzam všetky rámce a keď nájdem rámec s ARP protokolom, tak sa pozriem, či nemám otvorenú komunikáciu s rovnakými alebo prehodenými IP adresami plus má buď svoju zdrojovú alebo cieľovú adresu rovnakú ako je MAC adresa komunikácie. Ak je to reply, tak komunikáciu zatvorím, ak je to request, tak iba daný rámec pridám do konkrétnej komunikácie. Ak nenájdem žiadnu komunikáciu, s ktorou by sa mi zhodovali údaje, tak vytvorím novú s údajmi zo súčasného rámca. Následne už iba komunikácie vypíšem, pričom sú vo výpise requesty vypísané spolu a k ním je vypísaný jeden alebo žiadny reply.

3) Príklad štruktúry externých súborov pre určenie protokolov a portov

```
0:Null SAP
2:LLC Sublayer Management / Individual
3:LLC Sublayer Management / Group
6:IP (DoD Internet Protocol)
14:PROWAY (IEC 955) Network Management, Maintenance and Installation
66:BPDU (Bridge PDU / 802.1 Spanning Tree)
78:MMS (Manufacturing Message Service) EIA-RS 511
94:ISI IP
126:X.25 PLP (ISO 8208)
142:PROWAY (IEC 955) Active Station List Maintenance
224:IPX(Novell Netware)
244:LAN Management
254:ISO Network Layer Protocols
2048:Internet IP (IPv4)
2049:X.75 Internet
2053:X.25 Level 3
2054:ARP (Address Resolution Protocol)
32821:Reverse ARP
32923:Appletalk
33011:AppleTalk AARP (Kinetics)
33024:IEEE 802.1Q VLAN-tagged frames
33079:Novell IPX
34525:IPv6
34827:PPP
34887:MPLS
34888:MPLS wit upstream-assigned label
34915:PPPoE Discovery Stage
34916:PPPoE Session Stage
```

Vyššie je uvedený príklad ako načítavam SAP-y a Ethertype-y. Dané riadky rozdeľujem za pomoci funkcie split(), ktorej dám ako oddeľovač dvojbodku. Tá mi vlastne vytvorí pole s dvoma prvkami. Prvok na nultej pozícií je hodnota v decimálnom tvare aká sa môže nachádzať v rámci na mieste Ethertype-u alebo na pozícii DSAP, SSAP. Či sa jedná o SAP alebo Ethertype rozlišujem podľa toho, že ak je hodnota väčšia ako 1500 jedná sa o Ethertype a teda túto dvojicu prvkov z poľa priradím do slovníka „ethernet“. Ak je táto hodnota menšia ako 256, tak sa jedná o SAP a teda pridám danú dvojicu do slovníka „ieee“. Rovnako

Autor: Martin Pirkovský
AIS ID: 103095

pristupujem k dátam z textových súborov aj pri načítavaní protokolov a portov, avšak rozdiel je iba v tom, že už sú všetky v samostatnom súbore .txt a teda ich bez akýchkoľvek podmienok iba rozdelím a pridám do slovníka nultý prvok ako „key“ a prvý prvok poľa ako „value“ danej dvojice.

4) Používateľské rozhranie

```
Zadaj cestu k pcap súboru: vzorky_pcap_na_analyzu/trace-15.pcap
```

Užívateľ ako prvé musí zadať cestu k pcap súboru, ktorý chce analyzovať.

```
Pre daný výpis napíš:
        all - pre zobrazenie všetkých rámcov a jedinečných IP adries
        http - pre výpis HTTP komunikácie
        https - pre výpis HTTPS komunikácie
        telnet - pre výpis TELNET komunikácie
        ssh - pre výpis SSH komunikácie
        ftp-control - pre výpis FTP riadiace komunikácie
        ftp-data - pre výpis FTP dátové komunikácie
        tftp - pre výpis TFTP komunikácie
        icmp - pre výpis ICMP komunikácie
        arp - pre výpis ARP dvojíc komunikácie
icmp
```

Následne mu program ukáže tabuľku, s možnosťami výpisu. Ak zadá „all“ vypíšu sa mu všetky údaje potrebné pre bod 1-3 zo zadania. Pri možnostiach „http“, „https“, „telnet“, „ssh“, „ftp-control“, „ftp-data“ sa vypíše prvá nekompletná komunikácia a jedna kompletná komunikácia. Pre možnosť „tftp“ sa vypíšu všetky TFTP komunikácie. Pri možnosti „icmp“ sa vypíšu všetky ICMP protokoly spolu s typom aký majú. A ak užívateľ napíše možnosť „arp“, tak sa mu vypíšu všetky arp komunikácie, aj také, ktoré nemajú reply alebo request a sú samostatné.

```
Stlač:  1 pre výstup do konzoly
        2 pre výstup do súboru
1
```

Ako posledné sa program používateľa spýta, kam chce výstup, či do konzoly alebo do súboru. V prípade výberu možnosti súbor sa výstup zapíše do súboru „output.txt“. Po tomto prebehne analýza na daný výstup, a následne program skončí.

Autor: Martin Pirkovský
AIS ID: 103095

5) **Voľba implementačného prostredia**

Ako implementačný jazyk som si zvolil Python, a ako prostredie som využil program PyCharm od českej firmy JetBrains. V pythone som programoval na strednej a nakoľko podporuje všetko čo je potrebné pre zadanie tak sa mi to zdalo ako dobrá voľba. Vďaka PyCharmu som nemal žiadny problém s nainštalovaním knižnice scapy, ktorú som tam nainštaloval troma klikmi. Zároveň nakoľko je Python jeden z vyšších programovacích jazykov, tak sa mi zdalo vhodné využiť niektoré z jeho možností, ktoré si napríklad v jazyku C je nutné doimplementovať sám. Takou vecou sú napríklad slovníky, ktoré pokiaľ viem nie sú štandardom v céčku.