

Slovenská technická univerzita
Fakulta infromatiky a infromačných technológií
Ilkovičova 3, 842 19 Bratislava 4

Zadanie č. 1 analyzátor sieťovej komunikácie

Autor: Michal Greguš

Predmet: Počítačové Komunikačné siete

Cvičiaci: Ing. Rastislav Bencel, PhD.

Cvičenie: Štvrtok 8:00

Zadanie úlohy

Zadanie 1: Analýzátor sieťovej komunikácie

Zadanie úlohy

Navrhните 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:

- Poradové číslo rámca v analyzovanom súbore.
- Dĺžku rámca v bajtoch poskytnutú pcap API, ako aj dĺžku tohto rámca prenášaného po médiu.
- Typ rámca – Ethernet II, IEEE 802.3 (IEEE 802.3 s LLC, IEEE 802.3 s LLC a SNAP, IEEE 802.3 – Raw).
- 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:

- Zoznam IP adries všetkých prijímajúcich uzlov,
- IP adresu uzla, ktorý sumárne prijal (bez ohľadu na príjemcu) 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:

- HTTP
- HTTPS
- TELNET
- SSH
- FTP riadiace

- f) FTP dátové
- g) TFTP, **uvedte všetky rámce komunikácie**, nielen prvý rámec na UDP port 69
- h) ICMP, uvedte aj typ ICMP správy (pole Type v hlavičke ICMP), napr. Echo request, Echo reply, Time exceeded, a pod.
- i) **Všetky ARP dvojice (request – reply)**, uvedte aj IP adresu, ku ktorej sa hľadá MAC (fyzická) adresa a pri ARP-Reply uvedte 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.

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.

- 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:

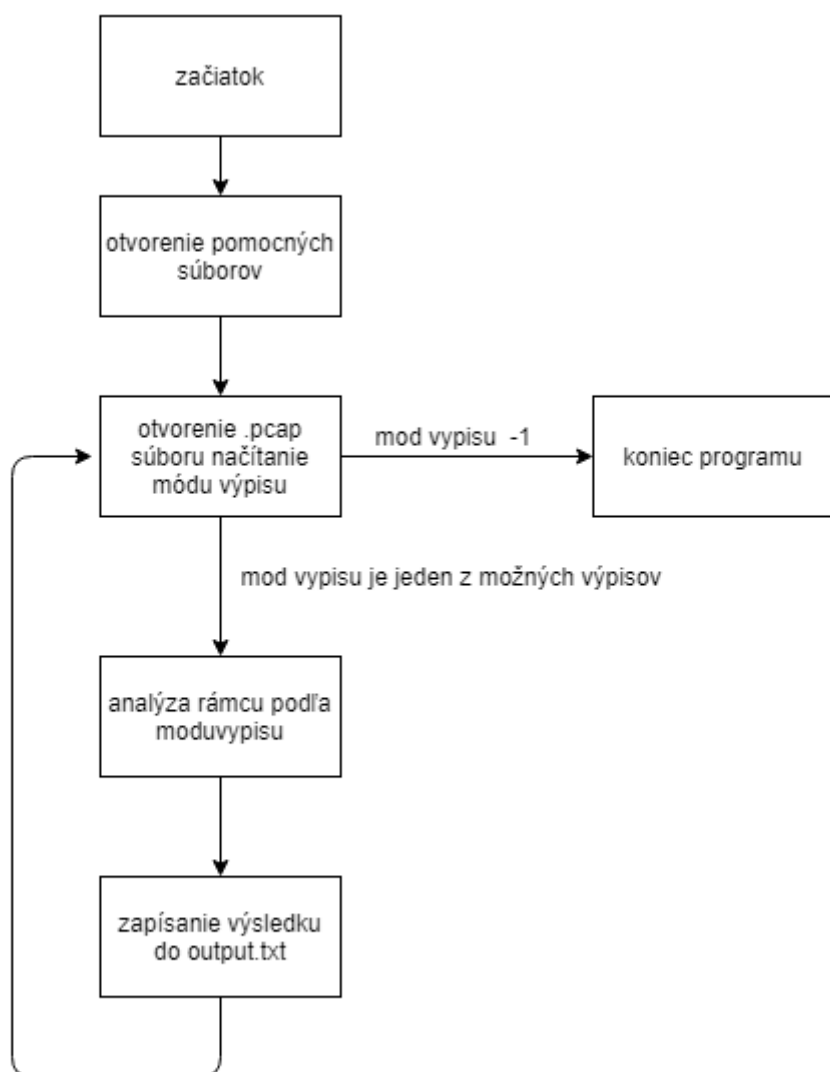
```
pcap_createsrcstr()  
pcap_open()  
pcap_open_offline()  
pcap_close()  
pcap_next_ex()  
pcap_loop()
```

Použitie funkcionality *libpcap* na priamy výpis konkrétnych polí rámca (napr. *ih->saddr*) bude mať za následok nulové hodnotenie celého zadania.

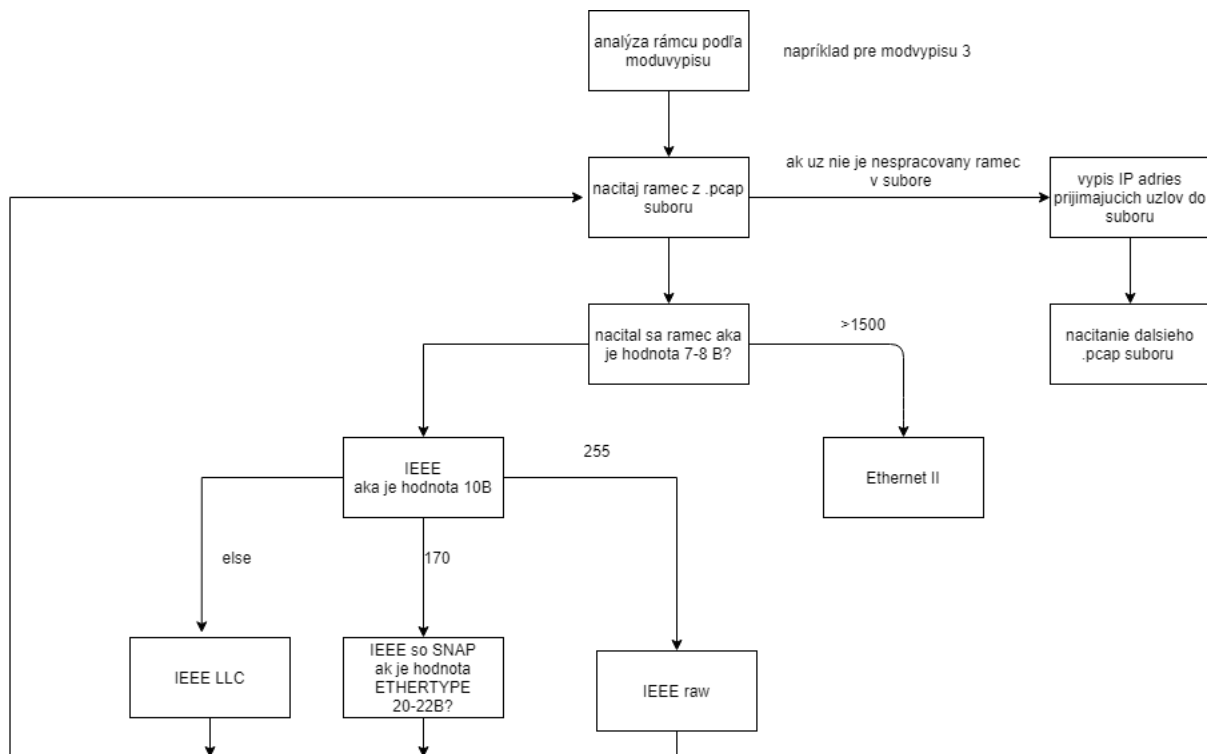
- 2) Program musí pracovať s dátami optimálne (napr. neukladať MAC adresy do 6x int).
- 3) Poradové číslo rámca vo výpise programu musí byť zhodné s číslom rámca v analyzovanom súbore.
- 4) Pri finálnom odovzdaní, pre každý rámec vo všetkých výpisoch uviesť použitý protokol na 2. - 4. vrstve OSI modelu. (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.

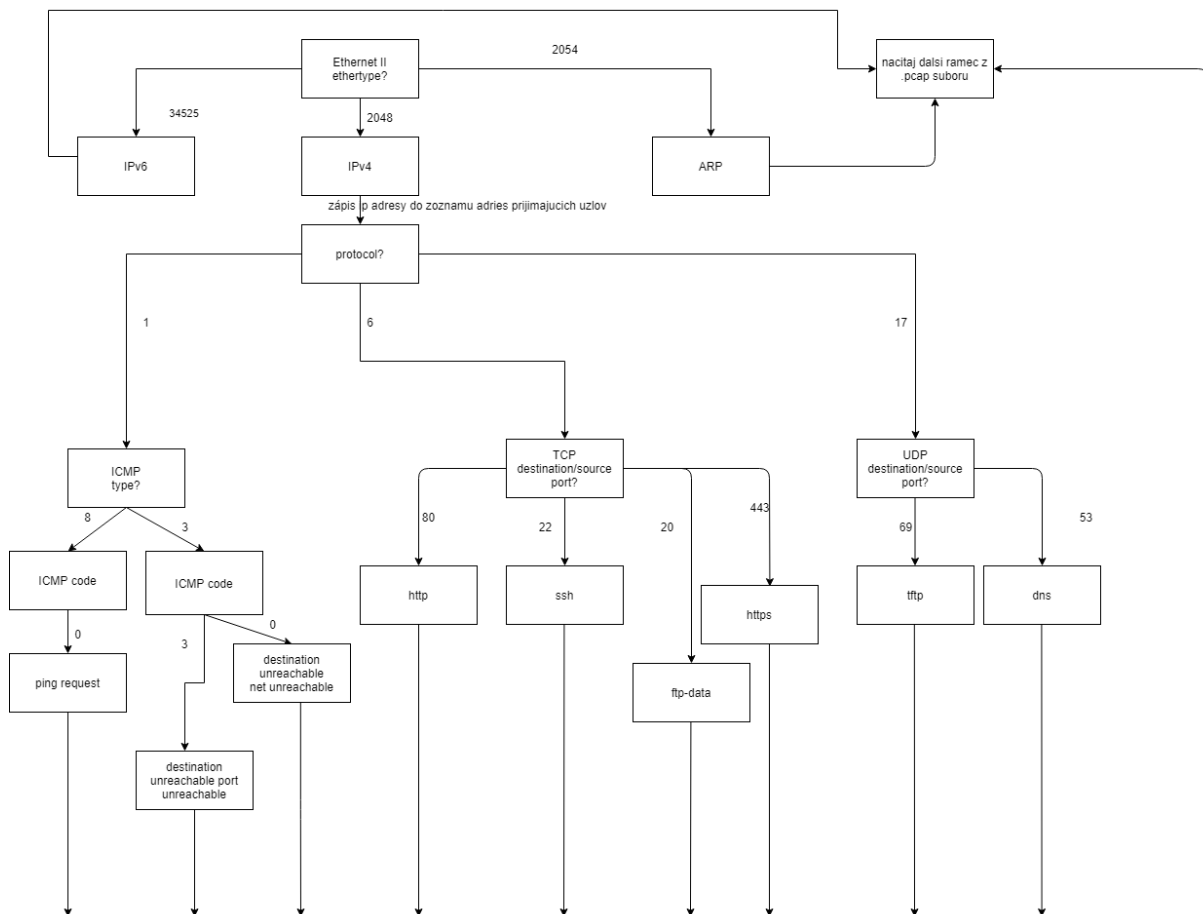
Blokový návrh programu a koncepcia fungovanie riešenia



Rozhodovanie o type na 2 vrstve



Analýza ethernet rámca cez 3-4 vrstvu



Mechanizmus analyzovanie rámcov

Program na začiatku otvorí zvolený .pcap súbor a následne ho analyzuje tak, aby výsledný výpis zodpovedal požadovanému výpisu. V zásade pre jednotlivé výpisy 4-14 si najskôr prejdem celý súbor aby som si poznačil potrebné údaje pre výpis. A v druhom prechode vypisujem potrebné rámce.

Jednotlivé rámce v súbore analyzujem postupne po bajtoch, pričom potrebné resp. pre analýzu významné bajty si ukladám do príslušných štruktúr a ich polí určených pre tieto údaje. Tieto štruktúry boli vytvorené na mieru pre jednotlivé protokoly podľa ich hlavičiek.

Opis procesu spracovania jedného rámcu:

Na začiatku zo štruktúry typu pcap_pkthdr zistím dĺžku rámca, a následne vypočítam reálnu dĺžku rámca prenášaného po médiu pomocou mojej funkcie dlzka_paketu_po_mediu(). Ďalej analyzujem rámec. Následne analyzujem rámec po bajtoch z údajov uložených v poli data_packetu. Významné údaje si kopírujem počas analýzy do príslušných štruktúr pre potreby ďalšej analýzy alebo výpisu.

1. načítam rámec z prvých 12B zistím mac adresy

2. zo 13,14B zistím ethertype

ak je ethertype > 1500 ide o ethernet II hlavičku

inak je to IEEE hlavička

IEEE hlavička

Zisťujem podľa 15B pole SAP o aký konkrétny typ IEEE hlavičky sa jedná

SAP = 255 IEEE raw

SAP = 170 IEEE s LLC a SNAP header

Inak IEEE s LLC

Následne pre všetky nižšie spomenuté protokoly sledujem už len časti hlavičky, ktoré sú definované v rámci štruktúr pre jednotlivé protokoly

3. ETHERNET hlavička na základe pola ethertype zistím o aký vnorený protokol ide

Ethertype == 2054 ARP

Ethertype == 2048 Ipv4

Ethertype == 34525 Ipv6

4. Protokol 3. vrstvy

Ak IPv6 – zistím zdrojovú a cieľovú IP adresu a podľa pola next header aj protokol vyššej vrstvy

Ak ARP – podľa pola operation určím či ide o request alebo reply. Ďalej z rámcu zisťujem source/target hardware a protocol adresu.

Ak IPv4 – zistím zdrojovú a cieľovú IP adresu, podľa pola protokol protokol vyššej vrstvy, a taktiež IHL aby som sa vedel správne posunúť za IP hlavičku.

5. Protokol 4. vrstvy

Podľa poľa protokol v IPv4 hlavičke zistíme o aký protokol ide.

Ak protocol == 6 TCP

Ak protocol == 17 UDP

Ak protocol == 1 ICMP

Pre ICMP zisťujem následne pole type a code, ktoré potrebujem pre výpis ICMP správy.

Pri TCP/UDP sa zameriavam na zdrojový a cieľový port.

Pri TCP navyše zisťujem aj hodnoty v poli FLAG pre potreby určenia kompletnosti komunikácie.

6. Port identifikujúci aplikáciu na 5./7. vrstve

Pri TCP a UDP na základe čísiel portov určím aká aplikácia používa daný protokol.

7. Výpis rámcu v hexa formáte – tento výpis vypíše len samotný rámec v hexa formáte, jednotlivé potrebné údaje ako napr. názov protokolu alebo adresy sú vypisované priebežne počas procesu analýzy rámcu.

Použité funkcie

V programe som použil

Funkcie z pcap.h knižnice:

pcap_next_ex() na načítanie ďalšieho rámcu

pcap_open_offline() na otvorenie pcap súborov

a tiež štruktúru struct pcap_pkthdr

Funkcie z knižnice string.h:

ako napríklad strcmp, alebo strstr;

Vlastné funkcie:

Mnou vytvorené funkcie najmä na výpis a čítanie zo súborov a na prácu so spájaným zoznamom ip adries uzlov, ktoré sú popísané v zdrojovom kóde programu.

Vlastné štruktúry: V programe používam aj vlastné štruktúry na uchovávanie potrebných dát pre analýzu resp. výpis informácií z rámcov


```

//struktura uchovavajúca mac adresy a ethertype
typedef struct Ethernet{
    u_char Dmac[6];
    u_char Smac[6];
    u_char type[2];
}Ethernet;

//struktura uchovavajúca dĺžku ipv4 hlavicky, ipv4 adresy, a protokol vyššej(L4) vrstvy
typedef struct IPv4{
    u_char ihlv[1];
    u_char protocol[1];
    u_char sourceip[4];
    u_char destip[4];
}IPv4;

//struktura uchovavajúca ipv4 adresy, a protokol vyššej(L4) vrstvy
typedef struct IPv6{
    u_char protocol[1];
    u_char sourceip[16];
    u_char destip[16];
}IPv6;

//struktura uchovavajúca potrebné údaje pre arp protokol
typedef struct ARP{
    //request /reply
    u_char operation[2];

    //request unicast mac of sender
    //reply unicast mac of reciever which was unknown before request was sent
    u_char sendermac[6];
    u_char senderip[4];

    //request broadcast ff:ff
    //reply unicast address
    u_char targetmac[6];
    u_char targetip[4];
}ARP;

//struktura uchovavajúca zdrojový a cieľový port a potrebné flagy nastavené v tcp hlav
typedef struct TCP{
    u_char sourceport[2];
    u_char destport[2];
    u_char flag[1];
}TCP;

```

Príklad štruktúry externých súborov, pre určenie protokolov a portov

Štruktúra súboru messages.txt

```

0 0 echo (ping) reply
3 0 Destination Unreachable net unreachable
3 1 Destination Unreachable host unreachable
3 2 Destination Unreachable protocol unreachable
3 3 Destination Unreachable port unreachable
3 4 Destination Unreachable fragmentation needed
3 5 Destination Unreachable source route failed

```

Príklad štruktúry súboru obsahujúceho icmp správy prvé číslo v riadku definuje ICMP type a druhé ICMP code nasledovaný samotným obsahom správy. Obe čísla sú v decimálnom tvare.

štruktúra súboru protokoly1.txt

```

2 007E X.25 PLP
2 0042 BPDU
2 00E0 Proway
3 0001 1 ICMP
3 003A ICMPv6
3 0002 2 IGMP
3 0006 6 TCP
3 0011 17 UDP
4 0007 7 echo
4 0014 20 ftp-data
4 0015 21 ftp-control
4 0016 22 SSH

```

Príklad štruktúry súboru pre čísla protokolov a portov kde prvé číslo(decimálne) určuje vrstvu v hlavičky protokolu v ktorej sa nachádza nasledujúce hexadecimálne číslo a to kód samotného protokolu nasledovaný názvom daného protokolu/portu. Priradí k číslu portu/protokolu názov.

štruktúra súboru ciska.txt

```

#####
http 80
https 443
telnet 23
ssh 22
ftp-r 21
ftp-d 20
tftp 69
icmp 1
igmp 2
tcp 6
udp 17
ipv4 2048
ipv6 34525
arp 2054

```

Príklad štruktúry súboru na priradenie čísla protokolu k názvu. Použité najmä pri modvypisi(4 - 9)

Prvý údaj je názov protokolu/portu nasledovaný decimálnym číslom daného portu.

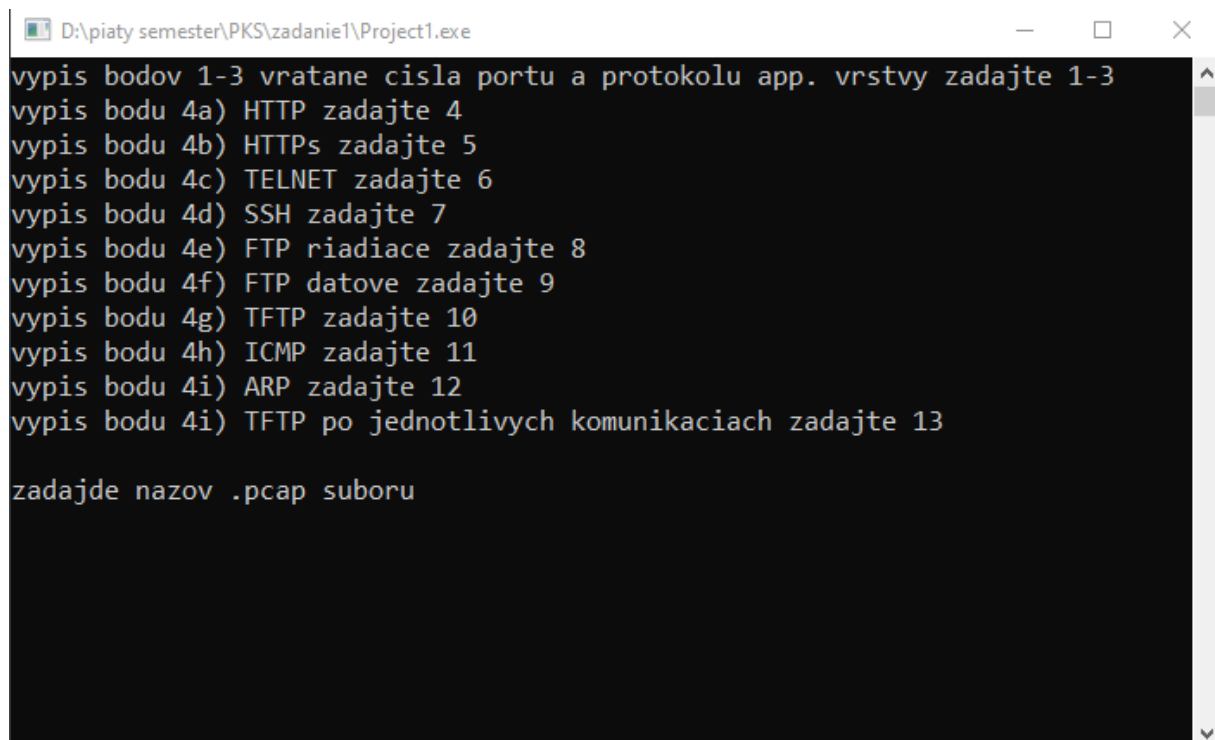
Opis používateľského rozhrania

Pred samotným spustením programu je potrebné vytvoriť 3 pomocné textové súbory. Prípadne skontrolovať správnosť ich názvu v programe.

```
FILE *protokoly;//mena protokolov a portov  
FILE *cisla;//cisla portov a aj niektorých protokolov  
FILE *messages;//icmp spravy
```

Ďalej treba nastaviť do premennej filepath cestu k .pcap súborom, ktoré chceme analyzovať. Momentálne je nastavená tak, aby program hľadal názov .pcap súboru zo vstupu v priečinku pcap nachádzajúcom sa v aktuálnom adresári.

```
sprintf(filepath,"pcap/");
```



```
D:\piaty semester\PKS\zadanie1\Project1.exe  
vypis bodov 1-3 vratane cisla portu a protokolu app. vrstvy zadajte 1-3  
vypis bodu 4a) HTTP zadajte 4  
vypis bodu 4b) HTTPs zadajte 5  
vypis bodu 4c) TELNET zadajte 6  
vypis bodu 4d) SSH zadajte 7  
vypis bodu 4e) FTP riadiace zadajte 8  
vypis bodu 4f) FTP datove zadajte 9  
vypis bodu 4g) TFTP zadajte 10  
vypis bodu 4h) ICMP zadajte 11  
vypis bodu 4i) ARP zadajte 12  
vypis bodu 4i) TFTP po jednotlivých komunikáciach zadajte 13  
  
zadajte nazov .pcap suboru
```

Po spustení programu možno zadať názov súboru, ktorý chceme analyzovať a následne aj jeho mód výpisu pre príslušný výpis ako vidieť na obrázku. Pre zvolenie módu 1,2,3 program vypíše vždy všetky požiadavky zo zadanie v bodoch 1 až 3. Teda vypíše všetky rámce analyzované cez 2-4 vrstvu a na konci sú uvedené ip adresy prijímajúcich uzlov. Následne pre jednotlivé konkrétne módy vypíše len konkrétnu komunikáciu.

Pri výpisoch 4-9 je najskôr vypísaná kompletná a následne nekompletná komunikácia.(Ak sa také nachádzajú v analyzovanom súbore)

TFTP 10 vypíše len jednu TFTP komunikáciu

TFTP 13 vypíše všetky TFTP komunikácie po jednotlivých komunikáciach.

Pri ARP výpise je pre každý request reply uvedené do ktorej komunikácie patria.

```
D:\piaty semester\PKS\zadanie1\Project1.exe
vypis bodov 1-3 vratane cisla portu a protokolu app. vrstvy zadajte 1-3
vypis bodu 4a) HTTP zadajte 4
vypis bodu 4b) HTTPS zadajte 5
vypis bodu 4c) TELNET zadajte 6
vypis bodu 4d) SSH zadajte 7
vypis bodu 4e) FTP riadiace zadajte 8
vypis bodu 4f) FTP datove zadajte 9
vypis bodu 4g) TFTP zadajte 10
vypis bodu 4h) ICMP zadajte 11
vypis bodu 4i) ARP zadajte 12
vypis bodu 4i) TFTP po jednotlivych komunikaciach zadajte 13

zadajte nazov .pcap suboru
trace-21.pcap
zadajte mod vypisu:
3
zadajte nazov .pcap suboru
```

Po zadaní módu výpisu program spracuje súbor a výpis zapíše do súboru output x(kde x je číslo súboru). Z každým ďalším načítaním a výpisom sa číslo x o jedna zväčší teda je zápis vykonaný vždy do ďalšieho súboru.

```
zadajte nazov .pcap suboru
trace-21.pcap
zadajte mod vypisu:
3
zadajte nazov .pcap suboru
lasldma
Chyba pri otvarani packet suboru: pcap/lasldma: No such file or directory
zadajte mod vypisu:
-1

-----
Process exited after 49.67 seconds with return value 0
Press any key to continue . . .
```

Ak chceme program ukončiť možno zadať ľubovoľný názov súboru a následne ako mód výpisu zvoliť

-1. V prípade ak sme omylom zadali nesprávny názov súboru treba zadať nejaký mód výpisu(okrem -1). Program sa následne automaticky opäť dopytuje na názov súboru na spracovanie.

Zvolené implementačné prostredie

Program som vytvoril v prostredí Dev-Cpp 5.11 s použitím 32-bit verzie TGM-GCC kompilera. Predtým som si na windowse musel nainštalovať knižnicu pcap(automaticky spolu s wiresharkom). Program wireshark som používal pri kontrole správností výpisu môjho programu.

Následne som podľa návodu vytvoril projekt, s príslušnými parametrami, ktorí umožňuje využitie funkcionalít z knižnice pcap.h potrebných na otvorenie .pcap súborov. Program možno následne štandardne spustiť na (windows) zariadení ak ho otvoríme v Dev-Cpp prostredí ako Dev-C++ project.

návod: <https://www.csie.nuk.edu.tw/~wuch/course/csc521/lab/ex1-wincap/>