**Slovenská technická univerzita v Bratislave**

Fakulta informatiky a informačných technológií

**Počítačové a komunikačné siete**

**Analyzátor sieťovej komunikácie**

Jozef Varga

Dr. Ing. Michal Ries

2017/2018

**Riešenie 1. zadania**

Navrhnite a implementujte programový “post” analyzátor Ethernet siete, ktorý analyzuje

komunikácie v sieti zaznamenané v súbore a poskytuje nasledujúce informácie

o komunikáciách. Vypracované zadanie musí spĺňať nasledujúce body:

1. Výpis všetkých komunikácií, t.j. všetkých rámcov v hexadecimálnom tvare postupne tak, ako boli zaznamenané v súbore.  
   Pre každý rámec uveďte:
   1. - Poradové číslo rámca v analyzovanom súbore.
   2. - Dĺžku rámca v bajtoch poskytnutú pcap API, ako aj dĺžku tohto rámca prenášaného po médiu.
   3. - Typ rámca – Ethernet II, IEEE 802.3 (IEEE 802.3 - LLC, IEEE 802.3- LLC - SNAP, IEEE 802.3 – Raw).
   4. - Zdrojovú a cieľovú fyzickú (MAC) adresu uzlov, medzi ktorými je rámec prenášaný.

Vo výpise jednotlivé bajty rámca usporiadajte po 8, 16 alebo 32 v jednom riadku. Pre prehľadnosť výpisu je vhodné použiť neproporcionálny (monospace) font.

1. Študent musí vedieť vysvetliť, aké informácie sú uvedené v jednotlivých rámcoch Ethernet II, t.j. vnáranie protokolov ako aj ozrejmiť dĺžky týchto rámcov.
2. Analýzu cez jednotlivé vrstvy vykonajte len pre rámce Ethernet II a protokoly rodiny  
   TCP/IPv4. Na konci výpisu z bodu 1) uveďte IP adresy všetkých vysielajúcich uzlov, ako aj IP adresu uzla, ktorý sumárne odvysielal (bez ohľadu na príjemcu) najväčší počet bajtov (vypíšte).  
   V danom súbore analyzujte zadané komunikácie:
   1. HTTP komunikácie
   2. HTTPS komunikácie
   3. TELNET komunikácie
   4. SSH komunikácie
   5. FTP riadiace komunikácie
   6. FTP dátové komunikácie
   7. Všetky TFTP komunikácie
   8. Všetky ICMP komunikácie
   9. Všetky ARP dvojice (request – reply).

Vo všetkých výpisoch treba uviesť aj IP adresy a pri transportných protokoloch aj

porty komunikujúcich uzlov.V prípade výpisu h) uveďte aj typ ICMP správy (pole Type v hlavičke ICMP), napr. Echo request, Echo reply, Time exceeded, a pod.V prípade výpisu i) uveďte pri ARP-Request 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 rovnakých rámcov ARP-Request, vypíšte všetky. Ak počet rámcov danej komunikácie je väčší ako 20, vypíšte iba 10 prvých a 10 posledných rámcov. Pri všetkých výpisoch musí byť poradové číslo rámca zhodné s číslom rámca v analyzovanom súbore.

1. Program musí byť organizovaný tak, aby čísla protokolov v rámci Ethernet II a v IP pakete ako aj čísla portov v transportných protokoloch boli programom určené z externého súboru a pre známe protokoly a porty boli uvedené aj ich názvy.
2. V procese analýzy rámcov pri identifikovaní jednotlivých polí rámca ako aj polí hlavičiek vnorených protokolov nie je povolené použiť funkcie poskytované použitým programovacím jazykom. Celý rámec je potrebné spracovať postupne po bajtoch.
3. Program musí byť organizovaný tak, aby bolo možné jednoducho rozširovať jeho funkčnosť o výpis rámcov podľa ďalších požiadaviek na protokoly v bode 3) – pri doimplementovaní jednoduchej funkčnosti na cvičení.
4. Študent musí byť schopný preložiť a spustiť program v miestnosti, v ktorej má cvičenia!

**Program musí mať nasledovné vlastnosti (minimálne):**

1. Program musí byť implementovaný v jazyku C/C++ s využitím definovaných knižníc

a skompilovateľný a spustiteľný na PC v učebniach. Použité knižnice musia byť

schválené cvičiacim

1. Poradové číslo rámca vo výpise programu musí byť zhodné s číslom rámca

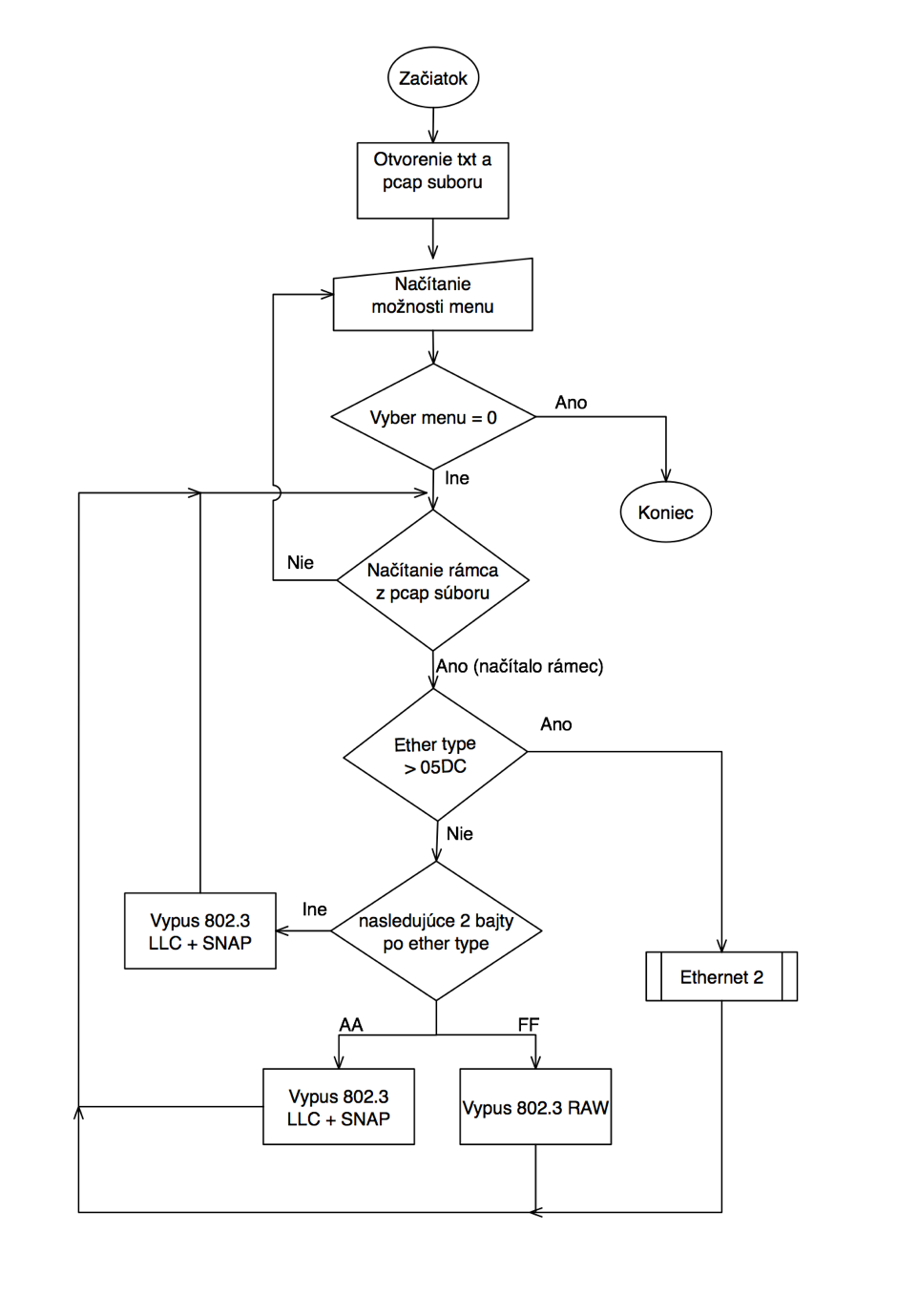
v analyzovanom súbore.

1. Pre každý rámec uviesť použitý protokol na 2. - 4. vrstve OSI modelu.
2. Pre každý rámec uviesť zdrojovú a cieľovú adresu / port na 2. - 4. vrstve OSI modelu.

**Riešenie**

Blokový návrh (koncepcia) riešenia

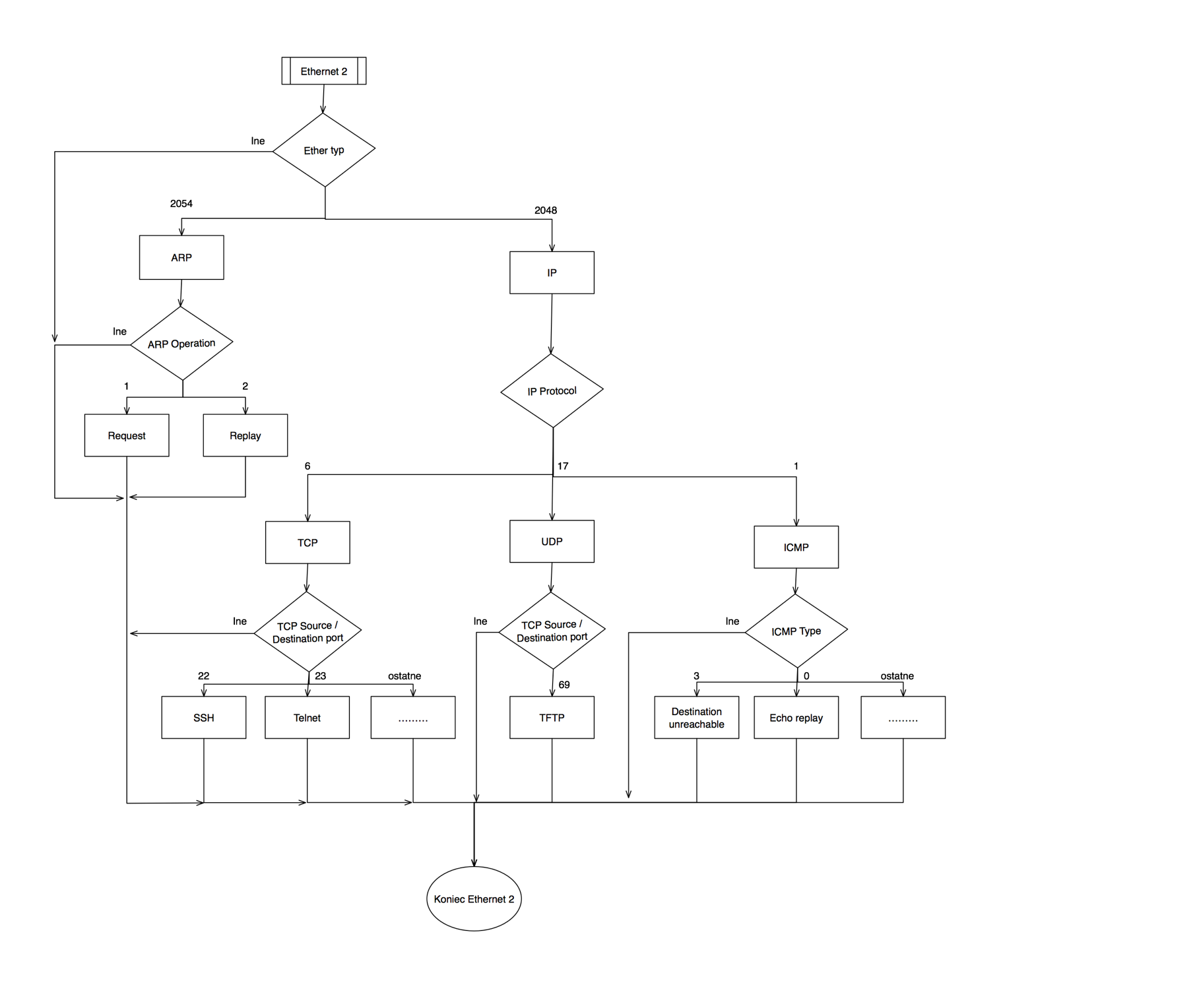
Obr. č 1.



Program na začiatok inicializuje potrebné premenné, otvorý txt a pcap súbor. Následne načíta hodnotu, ktorú mu zadáme z klávesnice (možnosť menu). V menu sa nachádza aj ukončenie aplikácie (0). Vyberieme si jednu z možností a program načita prvý rámec (program načítava rámce pokial súbor pcap nie je na konci) tým, že v našej aplikácii analyzujeme len Ethernet 2 zistíme, či Ether typ v ethernetovej hlavičke je väčší ako 05DC, ak je väčší, analyzujeme podrobnejšie a ak nie je, zistime podla nasledujúcich 2 bajtov o aký typ 802.3 protokolu sa jedná. 802.3 RAW by mal nasledujúce hodnoty FF, 802.3 LLC + SNAP by mal nasledujúce hodnoty AA a ak to nie je ani jeden z nich je to 802.3 LLC. Pri 802.3 vypisujeme len jej typ MAC adresy, číslo rámca, dĺžka PCAP api a dĺžka rámca prenášaného po médiu.

O MAC adresách vieme že prvých 6 bajtov je destination MAC ďalších 6 Bajtov je source MAC a posledné 2 bajty ethernetovej hlavičky je Ether typ. Dĺžku PCAP api nám dáva pcap.h knižnica. Dĺžku rámca prenášaného na médiu vypočítame ako PCAP api dĺžku + 4 bajty (FCS) a k tomu túto dĺžku musíme zaokrúhľovať na 64 bajtov kvôli minimálnej dĺžke ethernetovej hlavičky.

obr. č.2



Ak je to Ethernet 2 a v menu sme si vybrali inú hodnotu ako 0 alebo 1 (ukoncenie alebo základný výpis) tak prehľadávame podrobnejšie Ethernet 2.

Ako prvé sa pozeráme na Ether typ ktorý nám hovorý o protokole 3 vrstvy ak je:

* 2048 je to IP
  + pri IP sledujeme ako prvé jej verziu a IHL (2 Bajty). Zistime či daná hodnota je väčšia ako 45 (v hex) tak potrebujeme pri dalšom čítaní údajov počítať s väčšou hlavičkou IP o ten počet (8\*n pričom n je rozdiel od čísla 45 v hex.)
  + následne sledujeme protokol ktorý môže byt
    - 06 –TCP
      * pri tcp sledujeme tcp porty (destination aj source) a pozorujeme ich hodnotu ak ich hodnota je:
        + 20 – FTP DATA
        + 21 – FTP CONTROL
        + 22 – SSH
        + 23 – TELNET
        + 80 – HTTP
        + 443 – HTTPS
    - 11 –UDP
      * pri udp tak sledujeme udp porty (destination aj source) a pozorujeme ich hodnotu ak ich hodnota je:
        + 69 - TFTP
    - 01 –ICMP
      * pri ICMP sledujeme jeho typ (jeho hodnotu) :
        + 0 - Echo Reply
        + 3 - Destination Unrechable
        + 4 - Source Quench
        + 5 - Redirect
        + 8 - Echo
        + 9 - Router Advertisment
        + 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
* 2054 je to ARP pri ARP sledujeme tieto veci:
  + sledujeme operation:
    - 1 - Request
    - 2 – Replay
  + Source HW Address
    - MAC adresa odosielateľa
  + Source Protocol Address
    - IP adresa odosielateľa
  + Target HW Address
    - MAC adresa cieľa (ak je to request tak táto adresa sú samé 0)
  + Target Protocol Address
    - IP adresa cieľa

Náš program podľa výberu užívateľa sa zameria na jednotlivé protokoly a následne vypíše nami sledované údaje. K údajom sa program dostáva (a vypisuje ich) pomocou vytvorených štruktúr ktoré sme navrhli sami. Tieto štruktúry sme navrhli podľa hlavičiek jednotlivých protokolov a načítavame do nich jednotlivé bajty (informácie v štruktúre sú spracované bajt po bajte). Následne tieto informácie využívame (porovnávame/vypisujeme). Všetky tieto informácie sú poskytnuté v Hexa výpise rámca. Každý z týchto bajtov má určitý význam a my podľa jeho umiestnenia (poradie voči začiatku) tento význam skúmame a priraďujeme mu hodnoty.

**Funkcie štruktúry využité v kóde**

Mnou vytvorené štruktúry obsahujú premenné typu u\_char a u\_short (podľa počtu bajtov) do ktorých načítavam jednotlivé bajty z rámca. Po načítaní údajov skúmam údaje už iba z štruktúr. Štruktúry majú mená podľa ich obsahu.

Štruktúry:

* struct ethernet\_hlavicka
* struct ieee\_hlavicka
* struct ip\_hlavicka
* struct tcp\_hlavicka
* struct udp\_hlavicka
* struct arp\_hlavicka
* struct icmp\_hlavicka

Funkcie obsahujú rozdelenú základnú logigu programu.

Funkcie:

* u\_short vymena\_bytov(u\_short x)
  + nakoľko po načítaní štruktúr načítava susedné 2 bajty v opačnom poradí táto funkcia slúži na ich navrátenie do správnej polohy.
* int meno\_protokolu(FILE \*subor,char znak,u\_short ethernet\_typ\_cislo, int x)
  + táto funkcia nám vypisuje meno protokolu (zo súboru) po zaslaní jeho hodnoty.
* void meno\_portu(FILE \*subor, u\_short port\_zdroj, u\_short port\_ciel, char x)
  + funguje ako funkcia int meno\_protokolu akurád namiesto protokolu hľadá porty.
* void vypis\_mac\_addresy(struct ethernet\_hlavicka \*ethernet)
  + slúži na výpis MAC adries
* void hexa\_vypis(int i,const u\_char \*packet)
  + vypisuje celkový výpis v hexadecimálnom tvare
* void vypis\_zaklad(int cislo\_ramca, int i,int x)
  + vypisuje základný výpis ako dĺžku prenášanú na médiu, dĺžka pcap API.
* int velkost\_bytov(int x)
  + prepočítava dĺžku prenášanú po médiu
* void vypis\_ip(struct ip\_hlavicka \*ip)
  + slúži na výpis IP adries
* void vypis\_udp\_protocol(struct udp\_hlavicka \*udp)
  + slúži na výpis udp protokolu (jeho portov)
* void vypis\_tcp\_protocol(struct tcp\_hlavicka \*tcp)
  + slúži na výpis tcp protokolu (jeho portov)
* int vyber\_standart(struct ethernet\_hlavicka \*ethernet, struct ieee\_hlavicka \*ieee, FILE \*subor,int hodnota\_zo\_suboru,int \*dlzka\_ethernetovej\_hlavicky, const u\_char \*packet, int ci\_sa\_vypise)
  + vypisuje nám typ rámca (ETHERNET 2 / IEEE)
* int vypis\_menu()
  + výpis menu
* int main(int argc, const char \* argv[])
  + main nám spája všetky funkcie a vytvára jednotlivé prepojenia medzi nimi. Funkcionalita je vytvorená podľa blokovej schémy vyššie.

**Obsah súboru**

Typ prenosu

# 1Ethernet 2

# 0IEEE 802.3 LLC

# 255IEEE 802.3 Novell RAW (Novell properietary)

# 170IEEE 802.3 LLC + SNAP

Ether typ

| 2048IPv4

| 2054ARP

| 512XEROX PUP

| 513PUP Addr Trans

| 2049X.75 Internet

| 2053X.25 Level 3

| 32821Revers ARP

| 32923Appletalk

| 33011Appletalk AARP

| 33024IEEE 802.1Q VLAN tagged frames

| 33079Novell IPX

| 34525IPv6

| 34827PPP

| 34887MPLS

| 34888MPLS with upstream assigned label

| 34915PPPoE Discovery Stage

| 34916PPPoE Session Stage

SSAP

+ 0NULL SAP

+ 2LLC Sublayer Management or Individual

+ 3LLC Sublayer Management or Group

+ 6IP (DOD Internet Protocol)

+ 14PROWAY (IEC 955)

+ 66BPDU (Bridge PDU 802.1 Spanning tree)

+ 78MMS (Manufacturing Message Service)

+ 94 ISI IP

+ 126X.25 PLP (ISO 8208)

+ 142PROWAY (IEC 955) Active Station List Maintenance

+ 170SNAP

+ 224IPX (Novell NetWare)

+ 244LAN Management

+ 254ISO Network Layer Protocols

+ 255Global DSAP

IP

- 1ICMP

- 2IGMP

- 6TCP

- 9IGRP

- 17UDP

- 47GRE

- 50ESP

- 51AH

- 57SKIP

- 88EIGRP

- 89OSPF

- 115L2TP

TCP

/ 7ECHO

/ 19CHARGEN

/ 20FTP DATA

/ 21FTP CONTROL

/ 22SSH

/ 23TELNET

/ 25SMTP

/ 53DOMAIN

/ 79FINGER

/ 80HTTP

/ 110POP3

/ 111SUNRPC

/ 119NNTP

/ 139NETBIOS SSN

/ 143IMAP

/ 179BGP

/ 389LDAP

/ 443HTTPS

/ 445MICROSOFT DS

/ 1080SOCKS

UDP

< 7Echo

< 19Chargen

< 37Time

< 53Domain

< 67Bootps DHCP

< 68Bootpc DHCP

< 69TFTP

< 137Netbios ns

< 138Netbios dgm

< 161SNMP

< 162SNMP trap

< 500Isakmp

< 514syslog

< 520RIP

< 33434Traceroute

ICMP

\ 0Echo Reply

\ 3Destination Unrechable

\ 4Source Quench

\ 5Redirect

\ 8Echo

\ 9Router Advertisment

\ 10Router Selection

\ 11Time Exceeded

\ 12Parameter Problem

\ 13Timestamp

\ 14Timestamp Reply

\ 15Information Request

\ 16Information Reply

\ 17Address Mask Request

\ 18Address Mask Reply

\ 30Traceroute

ARP

\* 1Request

\* 2Reply

**Vysvetlenie funkcie súboru**

Pri akomkoľvek výpise protokolu alebo podrobnejšej analýze typu protokolu využívame čítanie zo súboru. V súbore sa každý podstatný riadok začína znakom, ktorý definuje jeho obsah:

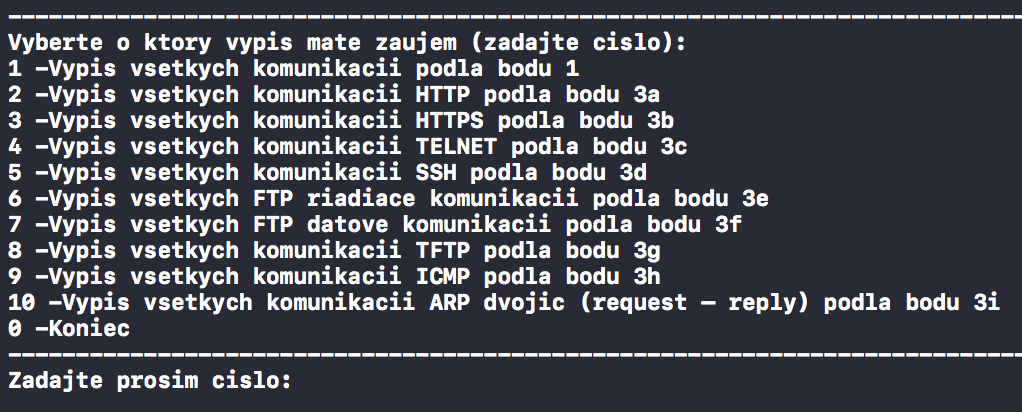
* # - Typ prenosu (Ethernet , 802.3 ....);
* | - Ether typ (IP, ARP ...)
* + - následné 2 bajty po ether type (LLC, LLC + snap ...)
* - IP (TCP, UDP ...)
* / - TCP porty (FTP, SSH ...)
* < - UDP porty (TFTP ...)
* \ - ICMP typ (echo ....)
* - ARP typ (request / replay)

Následne sa načíta číslo ktoré porovnáme z hodnotou v pcap súbore a ak je to vyhovujúce (rovnajú sa), vypíše celí riadok od miesta kde stojí.

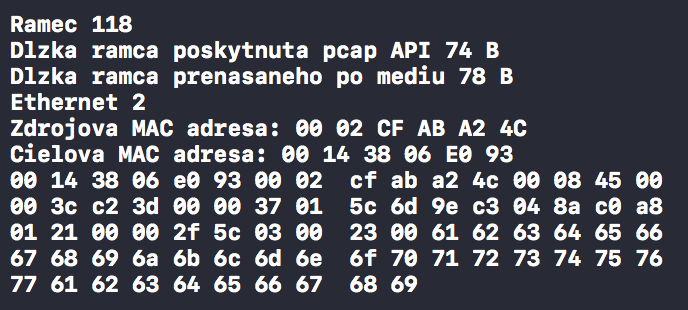
**Opis používateľského rozhrania**

Pred spustením aplikácie musíte mať pri exe súbore protocol.txt a pcap súbor s názvom aký je zadaný v programe (riadok v zdrojovom programe č. 220). Odovzdaný subor má nastavenú hodnotu subor.pcap .

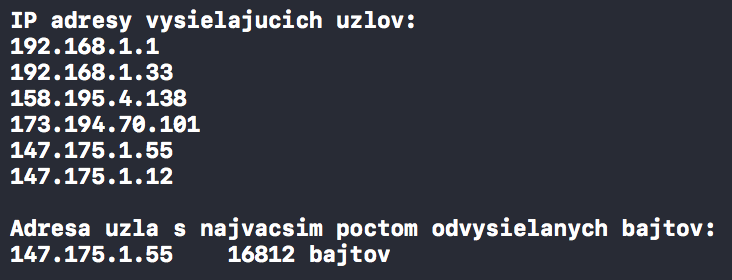
Po spustení aplikácie si vyberie z možností (podľa menu):



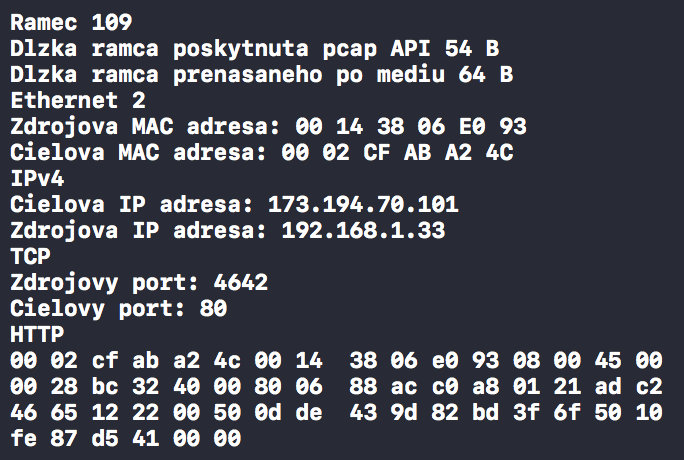
Výpis jedna mu vypíše všetky rámce a ich základné údaje



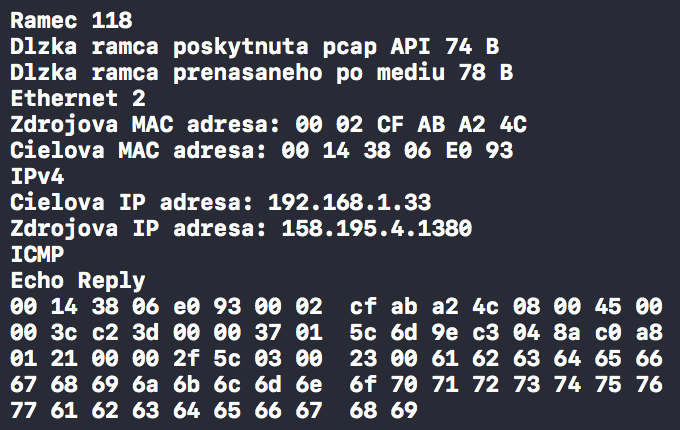
Na konci prvého výpisu sa zobrazia adresy vysielajúcich uzlov a adresa, ktorá odvysielala najviac bajtov (aj s počtom odvysielaných bajtov);



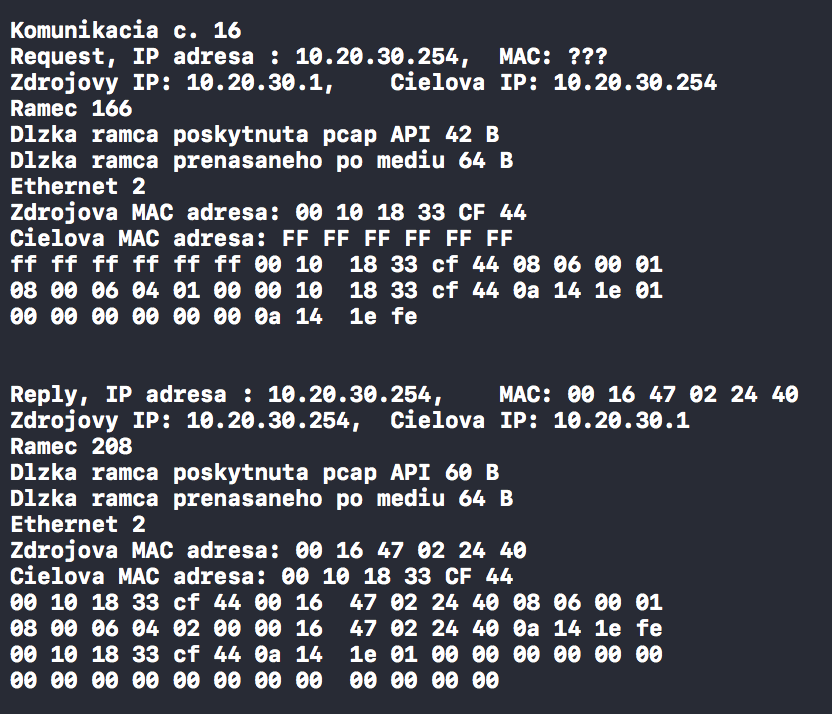
Po zadaní výpisu 2-8 vypíše sa nám už podrobnejší výpis rámca (pridajú sa IP, porty a protokol na 2-4 vrstve). Tento výpis nám ale vypíše len prvých a posledných 10 rámcov danej komunikácie.



Pri výpise 9 vypisujeme okrem predošlých vecí aj typ ICMP komunikácie



Výpis 10 je ARP komunikácia. Tu sa nám vypíše každý Request a ku nemu nájdene replay-e, ktoré sme zaznamenali po odvysielanom Requeste. Ak je Request bez odpovede, vypíše sa sám ako samostatná komunikácia, no bez odpovede. Výpis je formovaný podľa zadania. Vypisujeme adresu ktorá vysiela, IP adresu kam to vysiela a následne aj MAC adresy. Pri Requeste je MAC adresa cieľa neznáma preto sa rovná ??? a pri Replay je vidieť doplnená MAC adresa.



**Implementačné prostredie**

Nakoľko pracujem na macOS, Xcode 9.0 bol jasnou voľbou. Aplikácie ako Visual Studio tu nepracujú korektne a aplikáciu Xcode som využíval aj pred tým. Po nainštalovaní Xcode 9.0 treba vytvoriť projekt pre konzolu v jazyku C a následne skopírovať kód tam (dôvod je taký že operačný systém macOS nepodporuje natívne spúšťanie exe súborov) . Po skopírovaní kódu treba pridať knižnicu pcap.h. Kliknete na názov projektu a následne na spustiteľný súbor. Otvorí sa vám menu, v ktorom je lišta s názvom „Linked Frameworks and Libraries“, kde po kliknutí na tlačidlo + vyberiete knižnicu libpcap.tbd a libpcap.A.tbd. Následne je projekt spustiteľný.