



Sniffer paketů

Manuál

David Chocholatý (xchoch09)

Brno, 5. března 2022

Obsah

1	Úvod	3
2	Návod použití	3
2.1	Předpoklady	3
2.2	Vytvoření projektu	3
2.3	Spuštění snifferu	3
2.3.1	Argumenty	3
2.3.2	Příklad spuštění - Obecný zápis volání programu	4
2.3.3	Příklad spuštění - Ukázkové příklady	4
3	Implementace	5
3.1	Použité knihovny	5
3.1.1	Knihovny pro práci se síťovými prvky	5
3.1.2	Ostatní knihovny	5
3.2	Zpracování argumentů	6
3.3	Vytvoření filtru	6
3.4	Analýza paketů	6
3.4.1	Funkce <i>create_pcap_handler()</i>	6
3.4.2	Funkce <i>packet_handler()</i>	7
3.4.3	Funkce <i>handle_ipv4_packet()</i>	8
3.4.4	Funkce <i>handle_ipv6_packet()</i>	8
3.4.5	Funkce <i>stop_capture()</i>	9
3.5	Zobrazení dat	9
4	Testování	9
4.1	TCP	9
4.1.1	Wireshark	10
4.1.2	Sniffer	10
4.1.3	Správně odfiltrované UDP pakety	11
4.2	UDP	11
4.2.1	Wireshark	12
4.2.2	Sniffer	12
4.2.3	Data paketu	13
4.3	ICMP	14
4.3.1	Příkaz ping	15
4.3.2	Wireshark	15
4.3.3	Sniffer	16
4.4	ARP	16
4.4.1	<i>arping</i>	16
4.4.2	Wireshark	17
4.4.3	Sniffer	17
4.5	ICMPv6	17
4.5.1	Příkazy	18
4.5.2	Wireshark	18
4.5.3	Sniffer	20
4.6	Podpora IPv6	20

4.6.1	<i>curl</i>	20
4.6.2	Wireshark	21
4.6.3	Sniffer	21
4.7	Kombinace typů paketů	21
4.7.1	TCP a UDP - Wireshark	22
4.7.2	TCP a UDP - Sniffer	22
4.7.3	TCP a UDP - Wireshark	23
4.7.4	TCP a UDP - Sniffer	23
4.7.5	ICMP a ARP - ping	24
4.7.6	ICMP a ARP - Wireshark	24
4.7.7	ICMP a ARP - Sniffer	25
4.7.8	ICMP a ARP - arping	25
4.7.9	ICMP a ARP - Wireshark	26
4.7.10	ICMP a ARP - Sniffer	26
4.8	Argument <i>port</i>	27
4.8.1	Testovací příkazy	27
4.8.2	Wireshark	27
4.8.3	Sniffer	28
4.9	Argument <i>num</i>	28
4.9.1	Wireshark	28
4.9.2	Sniffer	31
4.9.3	Spuštění snifferu bez určení rozhraní	32

1 Úvod

Dokumentace popisuje použití, implementaci a testování síťového analyzátoru pro zachytávání a filtrování paketů na síťovém rozhraní.

2 Návod použití

2.1 Předpoklady

- gcc
- GNU Make
- tar (Nutné pouze při vytváření archivu)

2.2 Vytvoření projektu

Projekt lze vytvořit pomocí Makefile následujícím příkazem

```
# make
```

2.3 Spuštění snifferu

2.3.1 Argumenty

Sniffer lze spustit s následujícími argumenty:

Argument	Popis	Dlouhá varianta
-h	výpis nápovědy	--help
-i rozhraní	rozhraní, na kterém se bude poslouchat	--interface
-p port	filtrování paketů na daném rozhraní podle portu	----
-t	zobrazení pouze TCP paketů	--tcp
-u	zobrazení pouze UDP paketů	--udp
--icmp	zobrazení pouze ICMPv4 a ICMPv6 paketů	pouze
--arp	zobrazení pouze ARP rámců	pouze
-n num	počet paketů pro zobrazení, výchozí hodnota 1	----

2.3.2 Příklad spuštění - Obecný zápis volání programu

```
# ./ipk-sniffer [-i rozhrani | interface rozhrani] {-p port} {[--tcp|-t] [--udp|-u]  
[--arp] [--icmp] } {-n num}
```

Při výskytu chyby je nutné spouštět projekt s **rootovskými privilegii**.

2.3.3 Příklad spuštění - Ukázkové příklady

Vytvoření projektu

```
# make
```

Filtrování tcp a udp paketů na rozhraní eth0 a zobrazení 5 paketů

```
# sudo ./ipk-sniffer -i eth0 --tcp -u -n 5
```

Filtrování icmp paketů na rozhraní lo a zobrazení 1 paketu

```
# sudo ./ipk-sniffer --interface lo --icmp
```

Filtrování paketů na rozhraní eth0 dle portu 443 a zobrazení 1 paketu

```
# sudo ./ipk-sniffer -p 443 --interface eth0
```

Filtrování arp rámců na rozhraní eth0 a zobrazení 1 paketu

```
# sudo ./ipk-sniffer --arp -i eth0
```

Spuštění snifferu bez určení rozhraní - výpis aktivních rozhraní

```
# sudo ./ipk-sniffer
```

3 Implementace

Sniffer paketů je implementovaný v jazyce C. Při překladu je používána verze C99. Tento program podporuje jak IPv4, tak i IPv6. Projekt využívá knihovnu *libpcap*. Implementace je vytvořena v následujících souborech:

- Makefile - překlad projektu
- error.c (.h) - práce s chybovými kódy, výpis chybových zpráv na *stderr*
- ipk-sniffer.c (.h) - hlavní soubor snifferu paketů obsluhující logiku programu
- option.c (.h) - obsluha argumentů
- packet-print.c (.h) - výpis dat paketu a dalších souvisejících informací na *stdout*

3.1 Použité knihovny

3.1.1 Knihovny pro práci se síťovými prvky

- <pcap.h> - funkce pro vytvoření adaptéru a zachytávání paketů
- <linux/if_ether.h> - struktura hlavičky ethernetového rámce *ethhdr*
- <netinet/in.h> - struktura *sockaddr_in*
- <netinet/ip.h> - struktura IP hlavičky *ip*
- <netinet/ip6.h> - struktura IPv6 hlavičky *ip6_hdr*
- <netinet/tcp.h> - struktura TCP hlavičky *tcphdr*
- <netinet/udp.h> - struktura UDP hlavičky *udphdr*
- <netinet/ip_icmp.h> - struktura ICMP hlavičky *icmp_hdr*
- <netinet/icmp6.h> - struktura ICMPv6 hlavičky *icmp6_hdr*
- <arpa/inet.h> - funkce *ntohs()*, *inet_ntoa()*, *inet_ntop()*

3.1.2 Ostatní knihovny

- <stdio.h> - práce se vstupem a výstupem programu
- <stdlib.h> - *EXIT_SUCCESS*, *EXIT_FAILURE*, *strtoul()*, *exit()*
- <stdbool.h> - datový typ *bool*
- <stdint.h> - datový typ *uint_8t*
- <ctype.h> - funkce *isprint()*
- <string.h> - funkce pro práci s řetězci: *strcmp()*, *strcpy()*, *strcat()*
- <limits.h> - rozsahy datových typů: použit rozsah *USHRT_MAX*
- <unistd.h> - funkce *getopt()*
- <time.h> - práce s časem pro vytvoření časového razítka
- <signal.h> - signál pro ukončení programu pomocí *Ctrl+C*

3.2 Zpracování argumentů

Zpracování argumentů je implementováno v souboru ***option.c (.h)***. Hlavní funkcí pro obsluhu argumentů je funkce *parse_args()*. Zmíněná funkce nadále pracuje s funkcí *parse_long_opt()* pro zpracování dlouhých verzí argumentů (*anglicky long options*), a s funkcí *parse_short_opt()* pro zpracování verzí krátkých (*anglicky short options*). Při zpracování krátkých verzí argumentů je použita standardní funkce *getopt()*.

3.3 Vytvoření filtru

Vytvoření filtru je implementováno ve funkci *create_filter()* nacházející se ve zdrojovém souboru ***ipk-sniffer.c (.h)***. Ve zmíněné funkci je pomocí maker přidán filtr pro daný typ paketů:

- TCP
- UDP
- ICMPv4 a ICMPv6
- ARP

Výše zmíněné typy filtrů lze libovolně kombinovat. Syntaxe filtru je vytvořena dle syntaxe filtrů paketů: *pcap_filter*[7].

3.4 Analýza paketů

Hlavní logika snifferu paketů se nachází v následujících funkcích implementovaných v souboru *ipk-sniffer.c*:

- *create_pcap_handler()* - sestavení síťového adaptéru
- *packet_handler()* - hlavní funkce pro analýzu zachyceného paketu
- *handle_ipv4_packet()* - analýza IPv4 paketu a volání funkcí pro výpis dat
- *handle_ipv6_packet()* - analýza IPv6 paketu a volání funkcí pro výpis dat
- *stop_capture()* - ukončení zachytávání paketů

3.4.1 Funkce *create_pcap_handler()*

Implementovaný algoritmus nejprve začíná použitím funkce *pcap_lookupnet()* pro získání IP adresy a masky pro síťové zařízení. Dále je použita funkce *pcap_open_live()* pro otevření síťového zařízení v promiskuitním módu. Algoritmus pokračuje funkcí *pcap_compile()* pro kompilaci výrazu filtru do binární verze a pomocí funkce *pcap_setfilter()* je filtr nastaven pro zachytávání paketů.

Zmíněnou logiku implementuje následující úsek zdrojového kódu:

```
pcap_t *create_pcap_handle (char *device, const char *filter)
{
    char err_buf[PCAP_ERRBUF_SIZE];
    pcap_t *handle = NULL;
    struct bpf_program bpf;
    bpf_u_int32 netmask;
    bpf_u_int32 src_ip;

    /* Get network device source IP address and netmask */
    if (pcap_lookupnet(device, &src_ip, &netmask, err_buf) == PCAP_ERROR)
    {
        print_error(PCAP_LOOKUPNET_ERR);
        return NULL;
    }

    /* Open the device for live capture in promiscuous mode */
    if ((handle = pcap_open_live(device, BUFSIZ, 1, 1000, err_buf)) == NULL)
    {
        print_error(PCAP_OPEN_LIVE_ERR);
        return NULL;
    }

    /* Convert the packet filter expression into a packet filter binary */
    if (pcap_compile(handle, &bpf, (char *)filter, 0, netmask) == PCAP_ERROR)
    {
        print_error(PCAP_COMPILE_ERR);
        return NULL;
    }

    /* Bind the packet filter to the libpcap handle */
    if (pcap_setfilter(handle, &bpf) == PCAP_ERROR)
    {
        print_error(PCAP_SETFILTER_ERR);
        return NULL;
    }

    return handle;
}
```

Po sestavení síťového adaptéru lze zachytávat pakety pomocí funkce *pcap_loop()*.

3.4.2 Funkce *packet_handler()*

Funkce slouží pro zpracování zachycených paketů. Hlavním účelem této funkce je určení IP verze pro další práci s paketem. Typ paketu je určen následujícím výpočtem:

```
int packet_type = ((int)packet_ptr[12] << 8) | (int)packet_ptr[13];
```


U ethernetového paketu je cílová ethernet adresa v bajtech 0 až 5, zdrojová ethernet adresa je v bajtech 6 až 11 a typ/délka pole je v bajtech 12 a 13. Jedná se o big-endian hodnotu, takže je načten první byte na offsetu 12 a vložen do horních 8 bitů hodnoty a poté je načten druhý byte s posunem 13 a vložen do spodních 8 bitů hodnoty[6].

3.4.3 Funkce *handle_ipv4_packet()*

Funkce slouží pro zpracování IPv4 paketů. Její hlavní úlohou je určení typu paketu (TCP, UDP, ICMP nebo ARP rámeček) pro určení další funkce pro výpis dat.

3.4.4 Funkce *handle_ipv6_packet()*

Funkce slouží pro zpracování IPv6 paketů. Její hlavní úlohou je určení typu paketu (TCP, UDP nebo ICMPv6) pro určení další funkce pro výpis dat. Před samotným určení typu je paket testován na rozšířený typ hlavičky[8] a případně jsou provedeny potřebné výpočty pro posunutí ukazatele na paket[4].

```
int next_header = ipv6_header->ip6_nxt;

/* Determining if the packet has an extended header */
switch (next_header)
{
/* Routing */
case IPPROTO_ROUTING:;
    struct ip6_rthdr *header_r = (struct ip6_rthdr*)packet_ptr;
    packet_ptr += sizeof(struct ip6_rthdr);
    next_header = header_r->ip6r_nxt;
    break;

/* Hop by hop */
case IPPROTO_HOPOPTS:;
    struct ip6_hbh *header_h = (struct ip6_hbh*)packet_ptr;
    packet_ptr += sizeof(struct ip6_hbh);
    next_header = header_h->ip6h_nxt;
    break;

/* Fragmentation */
case IPPROTO_FRAGMENT:;
    struct ip6_frag *header_f = (struct ip6_frag*)packet_ptr;
    packet_ptr += sizeof(struct ip6_frag);
    next_header = header_f->ip6f_nxt;
    break;

/* Destination options */
case IPPROTO_DSTOPTS:;
    struct ip6_dest *header_d = (struct ip6_dest*)packet_ptr;
    packet_ptr += sizeof(struct ip6_dest);
    next_header = header_d->ip6d_nxt;
    break;
```

```
default:
    break;
}
```

3.4.5 Funkce *stop_capture()*

Funkce slouží pro ukončení zachytávání paketů.

3.5 Zobrazení dat

Pro zobrazení dat slouží především následující funkce implementované v souboru *packet-print.c*:

- *print_tcp_packet()* - výpis informací a dat pro TCP paket
- *print_udp_packet()* - výpis informací a dat pro UDP paket
- *print_icmp_packet()* - výpis informací a dat pro ICMP paket
- *print_arp_frame()* - výpis informací a dat pro ARP rámec
- *print_ipv6_tcp_packet()* - výpis informací a dat pro IPv6 TCP paket
- *print_ipv6_udp_packet()* - výpis informací a dat pro IPv6 UDP paket
- *print_ipv6_icmp_packet()* - výpis informací a dat pro IPv6 ICMPv6 paket

Pro výpis dat paketu v hexadecimálním a ascii tvaru slouží funkce *print_data()*[5] implementovaná v téže souboru. Funkce se dá logicky rozdělit na dvě části kdy se vypisují všechna data kromě posledního řádku a výpis posledního řádku, při kterém je ještě ošetřen případ, kdy data nezaplní celý řádek a tudíž je potřeba pomocí mezer doplnit zbývajících část pro dodržení formátu výpisu. Všechna data jsou vypisována po bytech. Při výpisu dat v ascii formátu jsou netisknutelné znaky nahrazeny znakem '.'.

Výpis časového razítka je implementovaný ve funkci *print_timestamp()*[1] dle RFC3339[3].

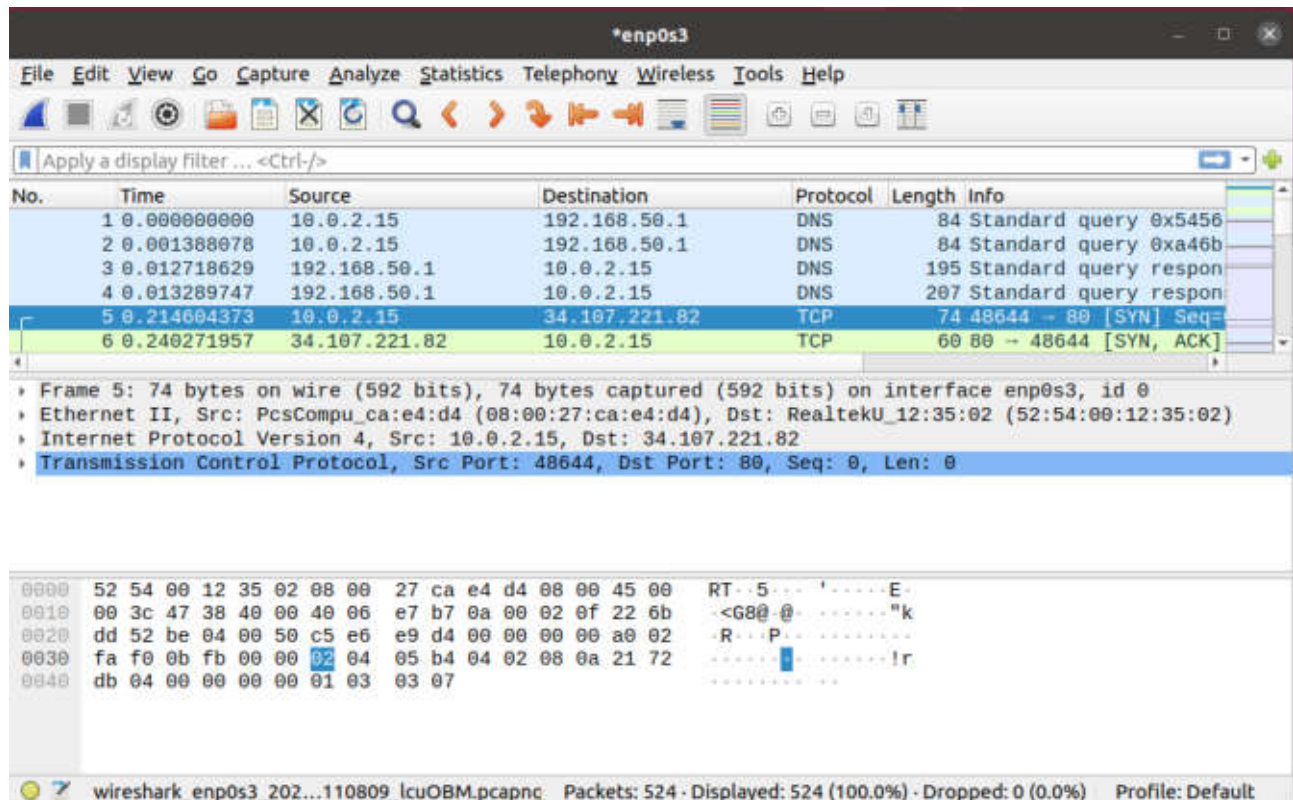
4 Testování

Testování bylo prováděno na referenčním virtuálním stroji s operačním systémem Ubuntu 20.04.2 LTS. Při testování byl použit open-source nástroj Wireshark[2].

4.1 TCP

Při testování filtrování TCP komunikace byla využita práce s prohlížečem Firefox 97.0 (64 bit), kdy při komunikaci se nejčastěji používají UDP nebo TCP protokoly pro komunikaci. Následující příklad poukazuje na správný výstup snifferu, kdy pakety s protokolem DNS využívající protokol UDP byly ignorovány a zachycen byl až TCP paket.

4.1.1 Wireshark



4.1.2 Sniffer

```
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i enp0s3 --tcp
[sudo] password for student:
timestamp: 2022-03-05T11:08:25.466296+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 74 bytes
src IP: 10.0.2.15
dst IP: 34.107.221.82
src port: 48644
dst port: 80

0x0000: 52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00 RT..5... '.....E.
0x0010: 00 3c 47 38 40 00 40 06 e7 b7 0a 00 02 0f 22 6b .<G8@.@. ...."k
0x0020: dd 52 be 04 00 50 c5 e6 e9 d4 00 00 00 00 a0 02 .R...P.. ....
0x0030: fa f0 0b fb 00 00 02 04 05 b4 04 02 08 0a 21 72 .....!r
0x0040: db 04 00 00 00 00 01 03 03 07 ..... ..
```

4.1.3 Správně odfiltrované UDP pakety

The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture control, and analysis. A filter bar at the top of the packet list shows 'Apply a display filter ... <Ctrl-/>'. The packet list table contains the following data:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.50.1	DNS	84	Standard query 0x5456
2	0.001388078	10.0.2.15	192.168.50.1	DNS	84	Standard query 0xa46b
3	0.012718629	192.168.50.1	10.0.2.15	DNS	195	Standard query response
4	0.013289747	192.168.50.1	10.0.2.15	DNS	207	Standard query response
5	0.214604373	10.0.2.15	34.107.221.82	TCP	74	48644 → 80 [SYN] Seq=
6	0.240271957	34.107.221.82	10.0.2.15	TCP	60	80 → 48644 [SYN, ACK]

The details pane for the selected packet (No. 4) shows the following structure:

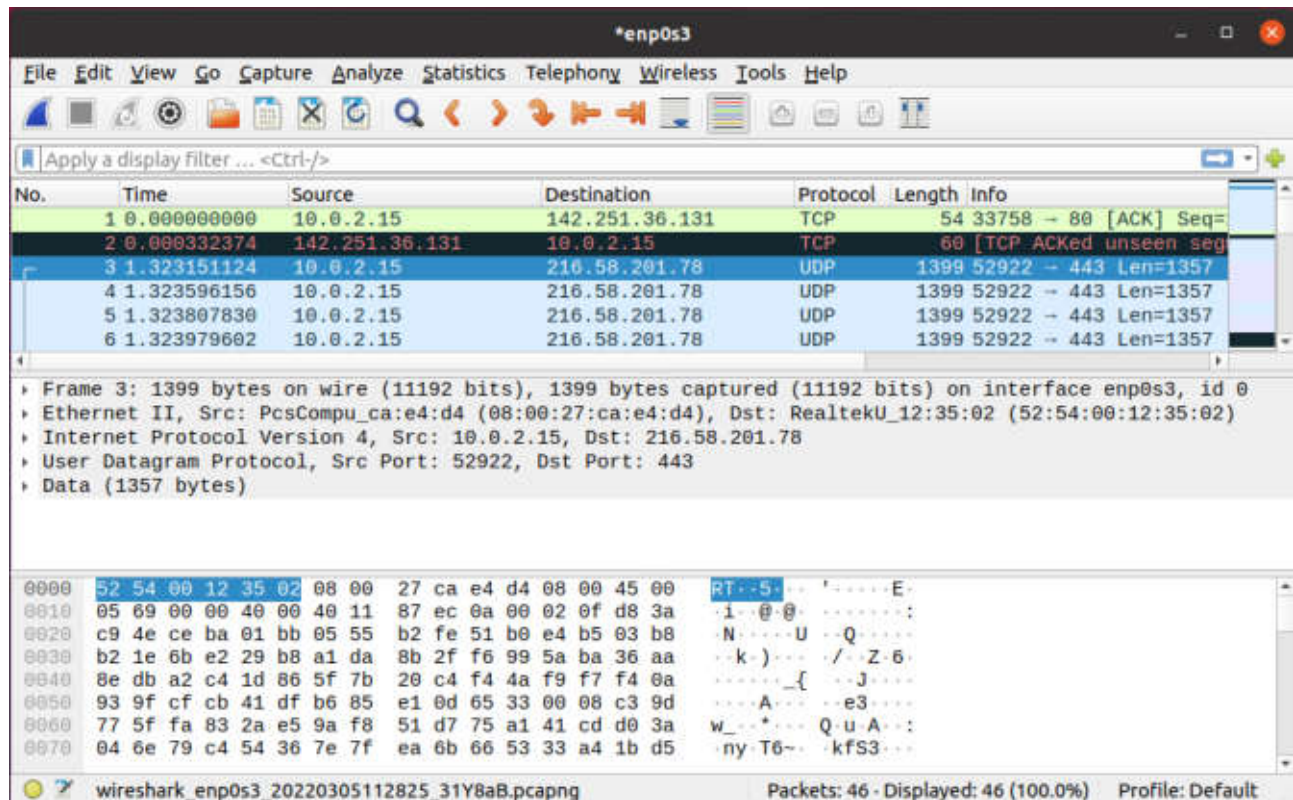
- Frame 4: 207 bytes on wire (1656 bits), 207 bytes captured (1656 bits) on interface enp0s3, id 0
- Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_ca:e4:d4 (08:00:27:ca:e4:d4)
- Internet Protocol Version 4, Src: 192.168.50.1, Dst: 10.0.2.15
- User Datagram Protocol, Src Port: 53, Dst Port: 44134
- Domain Name System (response)

The packet bytes pane shows the raw data in hexadecimal and ASCII. The total length of the packet is 207 bytes. The status bar at the bottom indicates: Total Length (ip.len), 2 bytes; Packets: 524 - Displayed: 524 (100.0%) - Dropped: 0 (0.0%) - Profile: Default.

4.2 UDP

Testování filtrování UDP paketů bylo opět založeno na práci se zmíněným prohlížečem Firefox. Následující příklad demonstruje správné filtrování a zachycení až UDP paketu. Zároveň lze poukázat, že sniffer správně zpracovává i pakety s větším objemem dat. Tento příklad testuje také validní nastavení ofsetů pro každý vypisovaný řádek v hexadecimálním tvaru.

4.2.1 Wireshark



4.2.2 Sniffer

```

student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i enp0s3 --udp
timestamp: 2022-03-05T11:28:30.312451+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 1399 bytes
src IP: 10.0.2.15
dst IP: 216.58.201.78
src port: 52922
dst port: 443

0x0000:  52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00  RT..5... '.....E.
0x0010:  05 69 00 00 40 00 40 11 87 ec 0a 00 02 0f d8 3a  .i...@.@. ....:
0x0020:  c9 4e ce ba 01 bb 05 55 b2 fe 51 b0 e4 b5 03 b8  .N.....U ..Q.....
0x0030:  b2 1e 6b e2 29 b8 a1 da 8b 2f f6 99 5a ba 36 aa  ..k.)... ./...Z.6.
0x0040:  8e db a2 c4 1d 86 5f 7b 20 c4 f4 4a f9 f7 f4 0a  ....._{ ..J....
0x0050:  93 9f cf cb 41 df b6 85 e1 0d 65 33 00 08 c3 9d  ....A... ..e3....
0x0060:  77 5f fa 83 2a e5 9a f8 51 d7 75 a1 41 cd d0 3a  w_...*... Q.u.A...:
0x0070:  04 6e 79 c4 54 36 7e 7f ea 6b 66 53 33 a4 1b d5  .ny.T6~. .kfs3...
0x0080:  14 9e 40 e1 0b d3 b8 d4 e9 5b 9f e6 11 b1 dd 9e  ..@..... .[.....
0x0090:  7f dd 64 cd 9a 73 b9 1b af 97 89 f5 d8 7f 06 9d  ..d..s.. ....:
0x00a0:  8f 81 76 73 30 6f bc ae af b0 45 2c c1 8d 56 01  ..vs0o.. ..E,..V.
0x00b0:  02 3d 65 d7 fb 82 8d 98 3d 23 d7 65 7e c7 42 b1  .=e..... =#.e~.B.
  
```

4.2.3 Data paketu

udpData.txt

```

0x0000: 52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00 RT..5... '....E.
0x0010: 05 69 00 00 40 00 40 11 87 ec 0a 00 02 0f d8 3a .i...@.@. ....:
0x0020: c9 4e ce ba 01 bb 05 55 b2 fe 51 b0 e4 b5 03 b8 .N....U ..Q....
0x0030: b2 1e 6b e2 29 b8 a1 da 8b 2f f6 99 5a ba 36 aa ..k.)... ./..Z.6.
0x0040: 8e db a2 c4 1d 86 5f 7b 20 c4 f4 4a f9 f7 f4 0a ....._{ ...J....
0x0050: 93 9f cf cb 41 df b6 85 e1 0d 65 33 00 08 c3 9d ....A.... .e3....
0x0060: 77 5f fa 83 2a e5 9a f8 51 d7 75 a1 41 cd d0 3a w_...*... Q.u.A.:
0x0070: 04 6e 79 c4 54 36 7e 7f ea 6b 66 53 33 a4 1b d5 .ny.T6~. .kfS3...
0x0080: 14 9e 40 e1 0b d3 b8 d4 e9 5b 9f e6 11 b1 dd 9e ..@..... .[.....
0x0090: 7f dd 64 cd 9a 73 b9 1b af 97 89 f5 d8 7f 06 9d ..d...s... ..
0x00a0: 8f 81 76 73 30 6f bc ae af b0 45 2c c1 8d 56 01 ..vs0o... .E,..V.
0x00b0: 02 3d 65 d7 fb 82 8d 98 3d 23 d7 65 7e c7 42 b1 .e..... =#e~.B.
0x00c0: 9d 39 63 65 ed 68 37 9e b8 4f f4 96 06 e3 5c aa .9ce.h7. .0....\
0x00d0: 56 8f 59 31 85 77 17 59 3e ba e3 62 39 ee b9 33 V.Y1.w.Y >..b9..3
0x00e0: d6 2c 9d 59 f1 1f ea 9e 84 47 3e 23 58 10 1a 98 .,.Y.... .G>#X...
0x00f0: ce fd c5 86 73 33 51 cd 08 1d 1f 13 4e 19 e9 55 ....s3Q. ....N..U
0x0100: 38 bb 4f 10 a6 40 e9 cc f8 72 42 1d 6d b3 27 db 8.O...@... rB.m.'
0x0110: 81 67 9b ad 46 2c 6d ab 5f a1 7b 38 90 3f e3 d2 .g..F,m. _{8.?..
0x0120: 00 33 69 6e 2d 03 2b 38 45 81 79 76 36 8b 9d 42 .3in-..+8 E.yv6..B
0x0130: db f8 0c 1c fc a7 96 c6 f3 e5 4b ba 54 6b ea b3 ..... .K.Tk..
0x0140: 95 45 63 b2 0e 01 af 0a 6f d3 a1 a1 7f 3c 12 43 .Ec..... o....<.C
0x0150: 7a 33 86 81 ce b4 c8 94 5e 85 cc 01 af 1b 25 a9 z3..... ^.....%.
0x0160: f9 eb d7 d5 89 c7 4f df 12 5d bc a8 1a 4d 74 34 .....0. .]...Mt4
0x0170: f0 25 d7 72 d4 94 f8 11 96 9b 66 15 98 16 44 b5 .%.r.... .f...D.
0x0180: ed db b4 a0 9b f8 73 67 e2 90 ff 17 4d 99 bb ed .....sg ....M...
0x0190: 60 1c af 48 78 84 02 b4 49 20 dd 98 78 21 c6 9d '..Hx... I ..x!..
0x01a0: 1a 48 c5 6f 60 d5 82 13 d7 0c 68 0a 45 ed 18 15 .H.o'... ..h.E...
0x01b0: 32 cd d8 24 bb ea 4d e8 1c 38 43 ea 37 1f 33 c8 2..$.M. .8C.7.3.
0x01c0: 4c 30 62 49 69 a4 1c e0 3b c2 a1 9e 4f f9 f9 4d L0bIi... ;...0..M
0x01d0: df a2 ca dd d7 80 1e 6f c9 8a db 83 39 db ef d0 .....o ....9...
0x01e0: bd d6 75 c3 82 a8 0a 76 87 3f 84 a2 5f 40 b9 38 ..u....v .?..@.8
0x01f0: 50 0a 34 31 ef fd 02 a5 84 c0 36 36 47 8c 12 9d P.41.... ..66G...
0x0200: 6a 09 91 37 c5 22 2d e8 df 42 61 07 82 66 db c1 j..7."-. .Ba..f..
0x0210: e9 11 ea 9d 2a 1c 18 19 c2 52 c3 a1 7f d0 1f 56 ....*.... .R....V
0x0220: 5d d0 3e f1 37 48 8c 4d ae e3 52 22 28 31 f5 c4 ].>.7H.M ..R"(1..
0x0230: 6d ea 79 aa f3 cf b0 2a 06 5e ea 8b 96 25 66 97 m.y....* .^...%f.
0x0240: 46 3e 65 c5 ab 4c 4f ff 5c 0b 15 c3 7b d8 c1 5d F>e..L0. \...{..]
0x0250: d9 e8 0c f7 1f 88 b8 31 4b 42 6a ae 45 7e 52 c8 .....1 KBj.E~R.
0x0260: 1b bd 02 0a aa 08 67 0d ec 88 77 06 19 88 d0 f7 .....g. ..w.....
0x0270: e1 15 0c b1 05 72 a4 b4 a2 cf d7 ce 40 e7 5e 42 .....r... ..@.^B
0x0280: fc dd a7 e6 0f 25 f3 6d c6 1f db 40 1e 85 26 1c .....%.m ...@..&.
0x0290: f1 81 c0 6a 75 d2 03 c8 27 2b 31 21 48 e2 fe 06 ...ju... '+1!H...
0x02a0: c0 58 ca f8 e9 3b 45 f7 4c 8c 73 24 d8 1b 45 ec .X...;E. L.s$.E.
0x02b0: d9 b0 c0 25 a8 a8 d3 17 82 db 49 54 c4 55 a2 d3 ...%. .... .IT.U..
0x02c0: e8 9d 56 33 67 63 13 b8 5e 5f 5d 26 1a 50 a4 5c ..V3gc.. ^_]&.P.\
0x02d0: a4 e1 4c 0e 75 b8 89 d4 e4 48 7a c8 ee cc 52 7d ..L.u.... .Hz...R}
0x02e0: 9c 26 a2 d6 82 b2 1f 0a 4c 40 2f 7a 11 f7 bb 1f .&..... L@/z....
0x02f0: 3a c2 ac a7 5e 55 92 66 15 0d f7 47 09 50 ba 55 :...^U.f ...G.P.U
0x0300: 1c 67 9b a6 b0 bd b8 85 65 49 31 ae b9 23 05 29 .g..... eI1..#.)
0x0310: c3 15 bc f2 20 78 a7 39 ec ff ab 5c 12 89 9b f7 .... x.9 ...\.
0x0320: 52 a1 80 12 da a4 43 a2 4f 2c 13 9d 57 29 ad 37 R....C. 0,..W).7
0x0330: 7e 1c 5f 8d 14 5f bd 70 ff 4e e6 46 ae 3d 41 84 ~._..._p .N.F.=A.

```

```

0x0340: 02 03 8c 0f 22 3a d3 d3 67 97 2e 64 b1 09 8f 88 ....":... g..d....
0x0350: 9d 17 cc 32 03 b7 e5 19 5a 6a 0c 55 14 15 ee f8 ...2.... Zj.U...
0x0360: f6 3f db 29 6e 29 6d 2b 24 03 95 fe b9 02 50 39 .?.)n)m+ $.....P9
0x0370: 22 19 0e 07 59 73 97 9f 00 34 71 d3 39 34 44 d3 "...Ys... .4q.94D.
0x0380: c9 ea fd ce b8 24 25 45 40 19 68 8d 4a 8f f6 c3 .....$%E @.h.J...
0x0390: 4d b9 6a 40 e0 a3 62 da 26 7c 60 bc c6 53 60 68 M.j@..b. &|'..S'h
0x03a0: c5 8a 76 75 6b da 89 4d 92 9c 81 01 51 c7 56 10 ..vuk..M ....Q.V.
0x03b0: e5 e2 39 bc d8 02 88 bc e4 69 86 61 24 92 99 aa ..9..... .i.a$...
0x03c0: 5c 09 42 49 c5 d2 03 66 9e 50 78 01 18 ec e7 0d \.BI...f .Px.....
0x03d0: ff 46 f3 e1 10 33 45 5b f9 42 17 75 5a 83 be 55 .F...3E[ .B.uZ..U
0x03e0: 30 d8 da 76 65 54 ec 09 00 e9 f4 0b 7b 3f 17 80 0..veT... ..{?...
0x03f0: a6 ca a2 d9 85 98 a7 5c 73 71 7f d9 e9 cb a9 a3 ..... \ sq.....
0x0400: 82 14 ea e6 ed 6c e2 94 2a 8f 65 05 4f 13 e5 8e .....l... *.e.O...
0x0410: ae 14 db 98 da 6a 10 49 ac 91 73 25 fe e9 b4 75 .....j.I ..s%...u
0x0420: 67 9c 68 8d c9 c4 90 4a a6 de 60 28 8a a4 10 e5 g.h....J ..'(...
0x0430: b2 e7 42 f4 fa e5 a6 68 4b 59 fc f2 9e 31 75 5a ..B....h KY...luZ
0x0440: f6 4c fc 99 2e b7 ef 96 24 48 ba 30 31 12 27 82 .L..... $H.01.'
0x0450: f8 dd 9f 53 5a 50 14 d3 ba e2 77 fb 5c 63 2a 77 ...SZP... ..w.\c*w
0x0460: b0 29 23 46 96 bf 4d b7 b6 ff 08 98 7b 6e 2e 7e .)#F..M. ....{n.~
0x0470: bf 53 8b bb 62 cc 90 22 bd 5f 07 13 1e 70 e9 9d .S..b.." _....p..
0x0480: d3 34 63 39 02 3d ed be 48 6c d3 a5 58 d5 d5 5e .4c9.=... Hl..X..^
0x0490: 74 9b ca c6 ce ac 76 78 76 d0 29 fe c5 be a6 6d t.....vx v.)....m
0x04a0: bd 12 b8 a4 19 03 a6 99 91 c9 c2 88 20 35 6e 23 ..... 5n#
0x04b0: e4 c9 7a 89 67 b6 6c f0 2f b2 c5 79 c3 fa 50 35 ..z.g.l. /..y..P5
0x04c0: 07 41 92 f4 44 c9 8d 2b 7f 41 ad fc 18 b6 38 0e .A..D...+ .A....8.
0x04d0: 8b 40 b4 44 26 50 48 30 73 03 cc 4a d5 3d 24 65 .@.D&PHO s..J.=$e
0x04e0: da 54 54 ef a7 4a 0e b0 8f e2 6a 08 c3 34 04 fd .TT..J... ..j..4..
0x04f0: 91 57 68 c0 ba 34 d6 f8 f3 80 03 21 ab 3d e4 b4 .Wh..4... ..!..=..
0x0500: 69 a9 2c f5 6c 36 a1 f3 af 13 ad 27 eb ed 0d 8b i.,.l6... ..'....
0x0510: 5f 2a 7a 5c 5d c8 e1 c2 ce 18 d7 1e f6 12 2b a8 _*z\]... ..+..
0x0520: 3c 69 1e 2a c8 98 7f 61 55 d4 c6 01 c4 17 7a ec <i.*...a U....z.
0x0530: 4e bf 9d 53 7d a1 bc 26 1f 59 ec 86 83 56 f3 4f N..S}...& .Y...V.O
0x0540: a1 ec 14 d0 02 ed 97 af 08 bc 36 62 5f ea 3a 65 ..... ..6b_..:e
0x0550: cd 88 d5 cd 3e c3 72 73 c5 4d e0 61 9d 29 81 36 ....>.rs .M.a.) .6
0x0560: ab da 1a d7 a5 2a b7 12 3e d8 eb b3 69 a2 f3 72 .....*.. >...i..r
0x0570: fb 24 3f ae f8 90 a0 . $?....

```

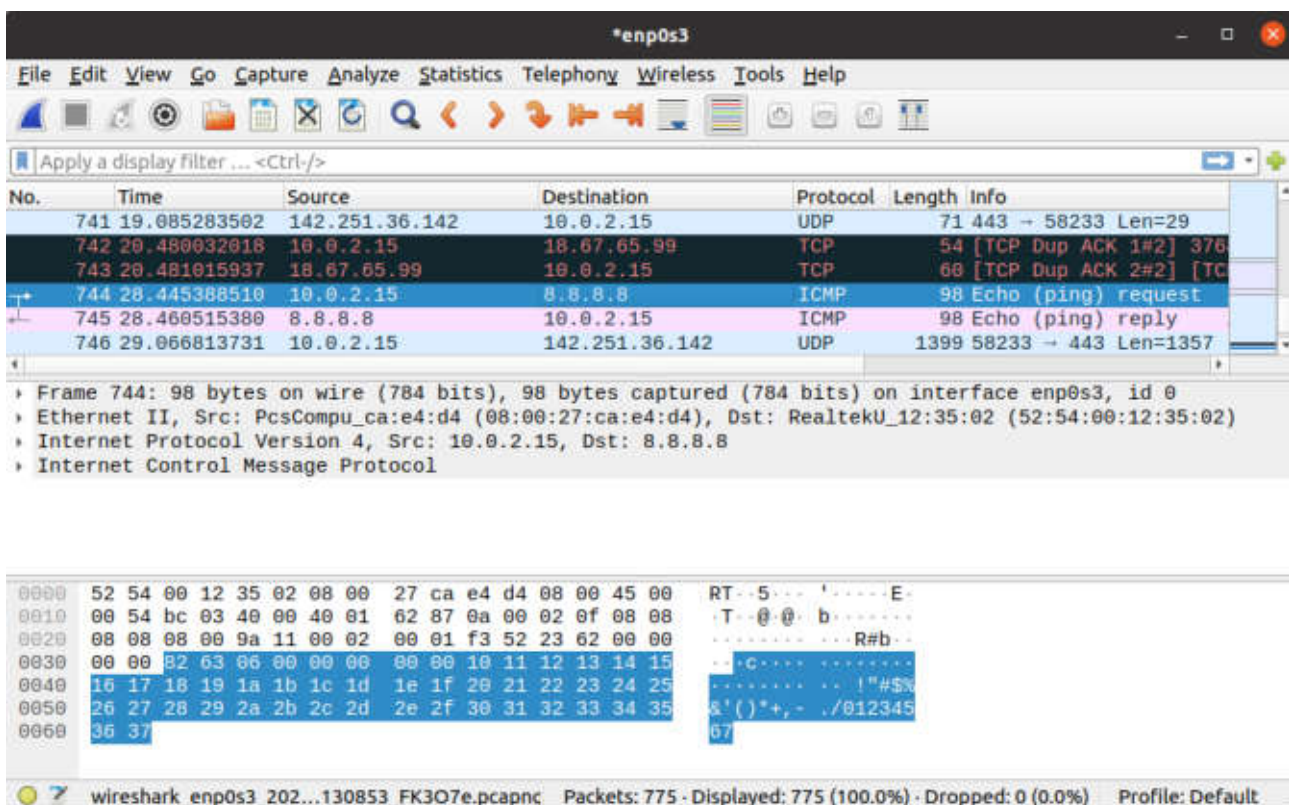
4.3 ICMP

ICMP pakety byly testovány pomocí příkazu *ping*. Konkrétně bylo provedeno testování na IP adresu 8.8.8.8, která je adresou primárního DNS serveru pro Google DNS. Před samotným spuštěním příkazu *ping* byla prováděna práce s prohlížečem Firefox a testování, zda filtr správně odfiltruje nepožadované protokoly. Tento test sniffer úspěšně obstál a zachycený paket je až ICMP paketem.

4.3.1 Příkaz ping

```
student@student-vm:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=15.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=12.4 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=28.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=10.6 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 10.628/16.775/28.930/7.199 ms
```

4.3.2 Wireshark



The image shows a Wireshark packet capture window titled '*enp0s3'. The packet list pane displays several packets, with packet 744 selected. The packet details pane shows the structure of the selected packet: Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol. The packet bytes pane shows the raw data of the selected packet, with the ICMP Echo (ping) request highlighted.

No.	Time	Source	Destination	Protocol	Length	Info
741	19.085283502	142.251.36.142	10.0.2.15	UDP	71	443 → 58233 Len=29
742	20.480032018	10.0.2.15	18.67.65.99	TCP	54	[TCP Dup ACK 1#2] 376
743	20.481015937	18.67.65.99	10.0.2.15	TCP	60	[TCP Dup ACK 2#2] [TC
744	28.445388510	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
745	28.460515380	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
746	29.066813731	10.0.2.15	142.251.36.142	UDP	1399	58233 → 443 Len=1357

Frame 744: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_ca:e4:d4 (08:00:27:ca:e4:d4), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 8.8.8.8
 Internet Control Message Protocol

0000 52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00 RT..5... '....E-
 0010 00 54 bc 03 40 00 40 01 62 87 0a 00 02 0f 08 08 .T--@.@. b.....
 0020 08 08 08 00 9a 11 00 02 00 01 f3 52 23 62 00 00 R#b..
 0030 00 00 82 63 06 00 00 00 00 00 10 11 12 13 14 15 ..-c.....
 0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 !"#\$%
 0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &'()*+,-./012345
 0060 36 37 67

wireshark_enp0s3_202...130853_FK3O7e.pcapng Packets: 775 - Displayed: 775 (100.0%) - Dropped: 0 (0.0%) Profile: Default

4.3.3 Sniffer

```
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i enp0s3 --icmp
timestamp: 2022-03-05T13:09:23.418704+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 98 bytes
src IP: 10.0.2.15
dst IP: 8.8.8.8

0x0000:  52 54 00 12 35 02 08 00  27 ca e4 d4 08 00 45 00  RT..5... '.....E.
0x0010:  00 54 bc 03 40 00 40 01  62 87 0a 00 02 0f 08 08  .T..@.@. b.....
0x0020:  08 08 08 00 9a 11 00 02  00 01 f3 52 23 62 00 00  ..... ..R#b..
0x0030:  00 00 82 63 06 00 00 00  00 00 10 11 12 13 14 15  ...C.... ....
0x0040:  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25  ..... .. !"#$%
0x0050:  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35  &'()*+,- ./012345
0x0060:  36 37                                     67
```

4.4 ARP

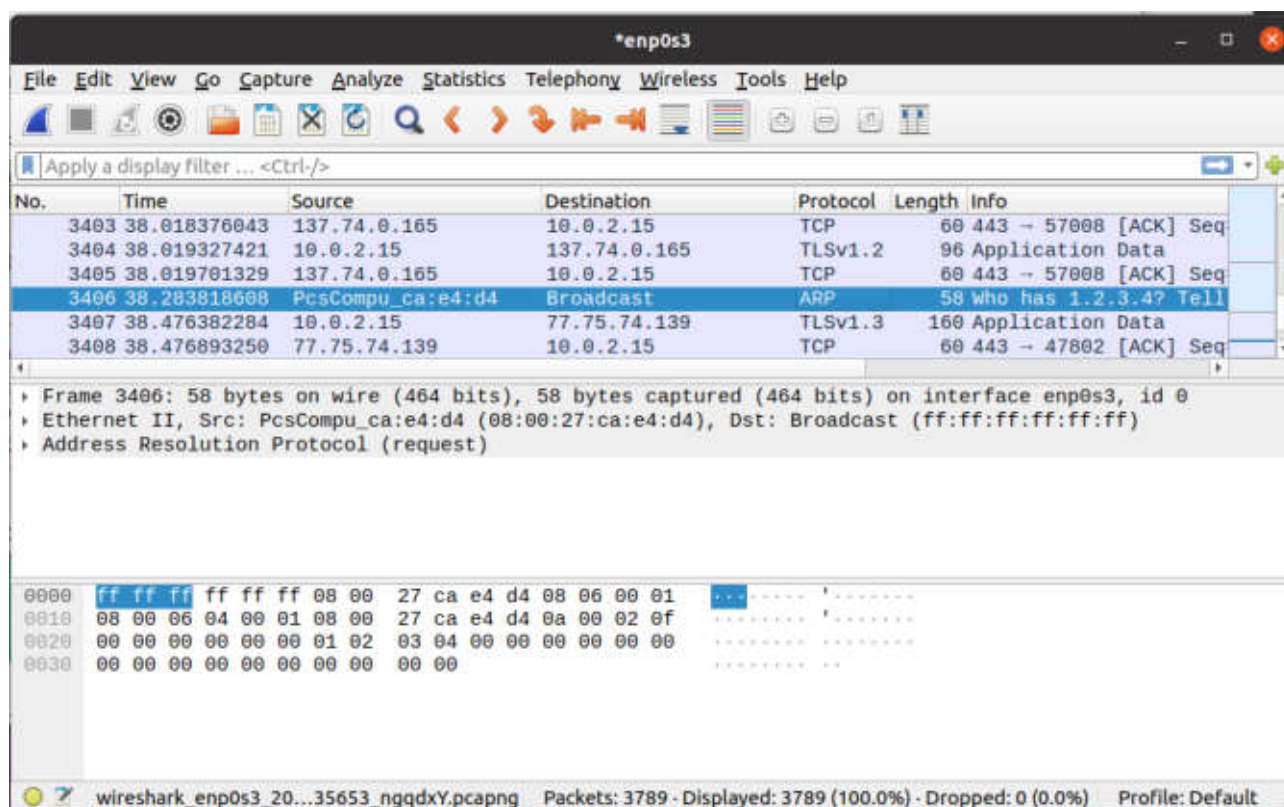
Testování ARP rámců bylo provedeno pomocí *arping*. Testována byla IP adresa 1.2.3.4, která se samozřejmě v síti nenachází a tudíž nedošla žádná odpověď na ARP dotaz. Ovšem tímto testem byl otestován právě ARP dotaz (*anglicky ARP request*), který používá protokol ARP. Následující příklad poukazuje na správné odfiltrování paketů a zachycení ARP paketu.

4.4.1 *arping*

```
student@student-vm:~$ sudo arping -c 4 -A -I enp0s3 1.2.3.4
ARPING 1.2.3.4
Timeout
Timeout
Timeout
Timeout

--- 1.2.3.4 statistics ---
4 packets transmitted, 0 packets received, 100% unanswered (0 extra)
```

4.4.2 Wireshark



4.4.3 Sniffer



4.5 ICMPv6

Hlavní rozdíl testování ICMPv6 paketů je testování na rozhraní loopback. ICMPv6 pakety byly testovány následujícím postupem. Nejprve byl manuálně zaslán UDP paket. Tento paket byl ignorován a byl zachycen až paket informující o nedostupnosti destinace. Poté bylo provedeno testování pomocí příkazu *ping* pro IPv6. Z tohoto důvodu byl sniffer spuštěn s argumentem *-n 2* pro zobrazení 2 paketů.

4.5.1 Příkazy

```

student@student-vm:~$ echo "This is my data" > /dev/udp/::1/3000
student@student-vm:~$ ping -6 ::1
PING ::1(::1) 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.030 ms
64 bytes from ::1: icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from ::1: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from ::1: icmp_seq=4 ttl=64 time=0.038 ms
^C
--- ::1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3064ms
rtt min/avg/max/mdev = 0.030/0.036/0.039/0.003 ms

```

4.5.2 Wireshark

The screenshot shows the Wireshark interface with a packet capture on the loopback interface `lo`. The packet list pane displays the following packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	::1	::1	DIS	78	PDUType: 105 Unknown
2	0.000016393	::1	::1	ICMPv6	126	Destination Unreachable (
3	8.852529340	::1	::1	ICMPv6	118	Echo (ping) request id=0x0
4	8.852540383	::1	::1	ICMPv6	118	Echo (ping) reply id=0x00
5	9.868832140	::1	::1	ICMPv6	118	Echo (ping) request id=0x0
6	9.868844563	::1	::1	ICMPv6	118	Echo (ping) reply id=0x00

The packet details pane for the selected packet (Frame 2) shows the following structure:

- Frame 2: 126 bytes on wire (1008 bits), 126 bytes captured (1008 bits) on interface `lo`, id 0
- Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
- Internet Protocol Version 6, Src: ::1, Dst: ::1
- Internet Control Message Protocol v6

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```

0000  00 00 00 00 00 00 00 00 00 00 00 86 dd 60 0d  ....H:
0010  cd 10 00 48 3a 40 00 00 00 00 00 00 00 00 00 00  --X-@
0020  00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00  hN
0030  00 00 00 00 00 01 01 04 68 4e 00 00 00 00 60 0e  -X-@
0040  e9 58 00 18 11 40 00 00 00 00 00 00 00 00 00 00  +Th
0050  00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00  is is my data
0060  00 00 00 00 00 01 8d ab 0b b8 00 18 00 2b 54 68
0070  69 73 20 69 73 20 6d 79 20 64 61 74 61 0a

```


Capturing from Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	:::1	:::1	DNS	78	PDUType: 105 Unknown
2	0.000016393	:::1	:::1	ICMPv6	126	Destination Unreachable (
3	0.852529340	:::1	:::1	ICMPv6	118	Echo (ping) request id=0x
4	0.852540383	:::1	:::1	ICMPv6	118	Echo (ping) reply id=0x00
5	0.868832140	:::1	:::1	ICMPv6	118	Echo (ping) request id=0x
6	0.868844563	:::1	:::1	ICMPv6	118	Echo (ping) reply id=0x00

Frame 3: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface lo, id 0

- Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
- Internet Protocol Version 6, Src: :::1, Dst: :::1
- Internet Control Message Protocol v6

```

0000  00 00 00 00 00 00 00 00 00 00 00 00 86 dd 60 0d  .....
0010  71 d7 00 40 3a 40 00 00 00 00 00 00 00 00 00 00  q-@:
0020  00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 01 80 00 23 d4 00 06 00 01 87 58  .....#...X
0040  23 62 00 00 00 00 ee 1a 04 00 00 00 00 00 10 11  #b...
0050  12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21  .....!
0060  22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31  "##$%'()*+,-./01
0070  32 33 34 35 36 37 234567

```

Loopback: lo: <live capture in progress> Packets: 20 - Displayed: 20 (100.0%) Profile: Default

4.5.3 Sniffer

```
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i lo --icmp -n 2
timestamp: 2022-03-05T13:33:02.416544+01:00
src MAC: 00:00:00:00:00:00
dst MAC: 00:00:00:00:00:00
frame length: 126 bytes
src IP: ::1
dst IP: ::1

0x0000:  00 00 00 00 00 00 00 00  00 00 00 00 86 dd 60 0d  .....
0x0010:  cd 10 00 48 3a 40 00 00  00 00 00 00 00 00 00 00  ...H:@..
0x0020:  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  .....
0x0030:  00 00 00 00 00 01 01 04  68 4e 00 00 00 00 60 0e  ..... hN....
0x0040:  e9 58 00 18 11 40 00 00  00 00 00 00 00 00 00 00  .X...@..
0x0050:  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  .....
0x0060:  00 00 00 00 00 01 8d ab  0b b8 00 18 00 2b 54 68  ..... +Th
0x0070:  69 73 20 69 73 20 6d 79  20 64 61 74 61 0a       is is my data.

timestamp: 2022-03-05T13:33:11.269057+01:00
src MAC: 00:00:00:00:00:00
dst MAC: 00:00:00:00:00:00
frame length: 118 bytes
src IP: ::1
dst IP: ::1

0x0000:  00 00 00 00 00 00 00 00  00 00 00 00 86 dd 60 0d  .....
0x0010:  71 d7 00 40 3a 40 00 00  00 00 00 00 00 00 00 00  q..@:@..
0x0020:  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  .....
0x0030:  00 00 00 00 00 01 80 00  23 d4 00 06 00 01 87 58  ..... #.....X
0x0040:  23 62 00 00 00 00 ee 1a  04 00 00 00 00 00 10 11  #b.....
0x0050:  12 13 14 15 16 17 18 19  1a 1b 1c 1d 1e 1f 20 21  ..... !
0x0060:  22 23 24 25 26 27 28 29  2a 2b 2c 2d 2e 2f 30 31  "#$%&'()*+,-./01
0x0070:  32 33 34 35 36 37       234567
```

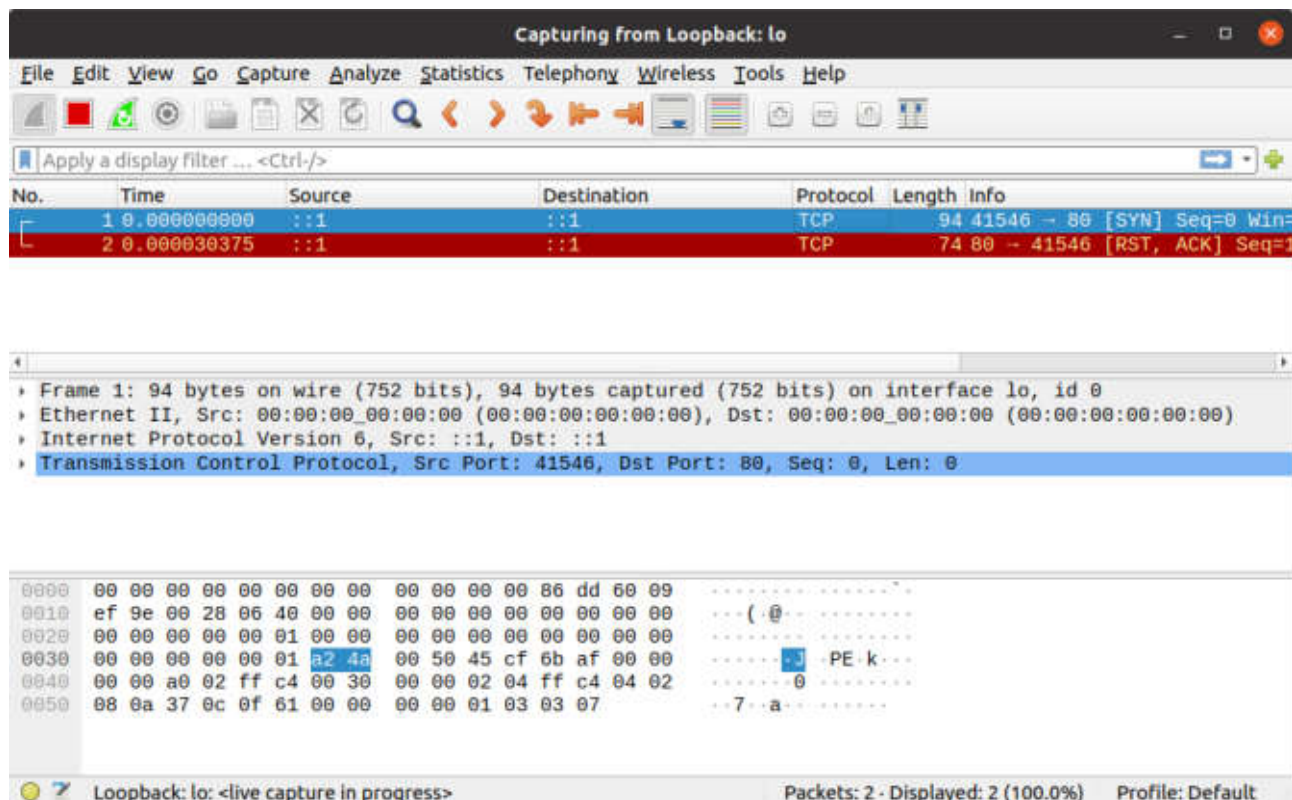
4.6 Podpora IPv6

Podpora IPv6 byla testována pomocí programu *curl*.

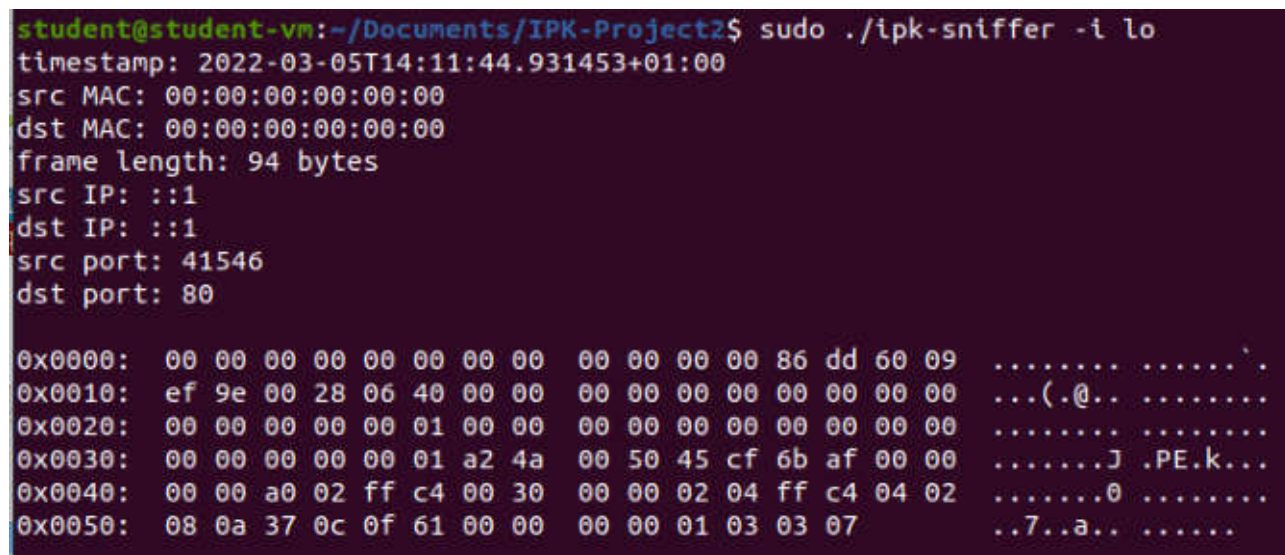
4.6.1 *curl*

```
student@student-vm:~$ curl -g -6 'http://[::1]:80/'
curl: (7) Failed to connect to ::1 port 80: Connection refused
student@student-vm:~$
```

4.6.2 Wireshark



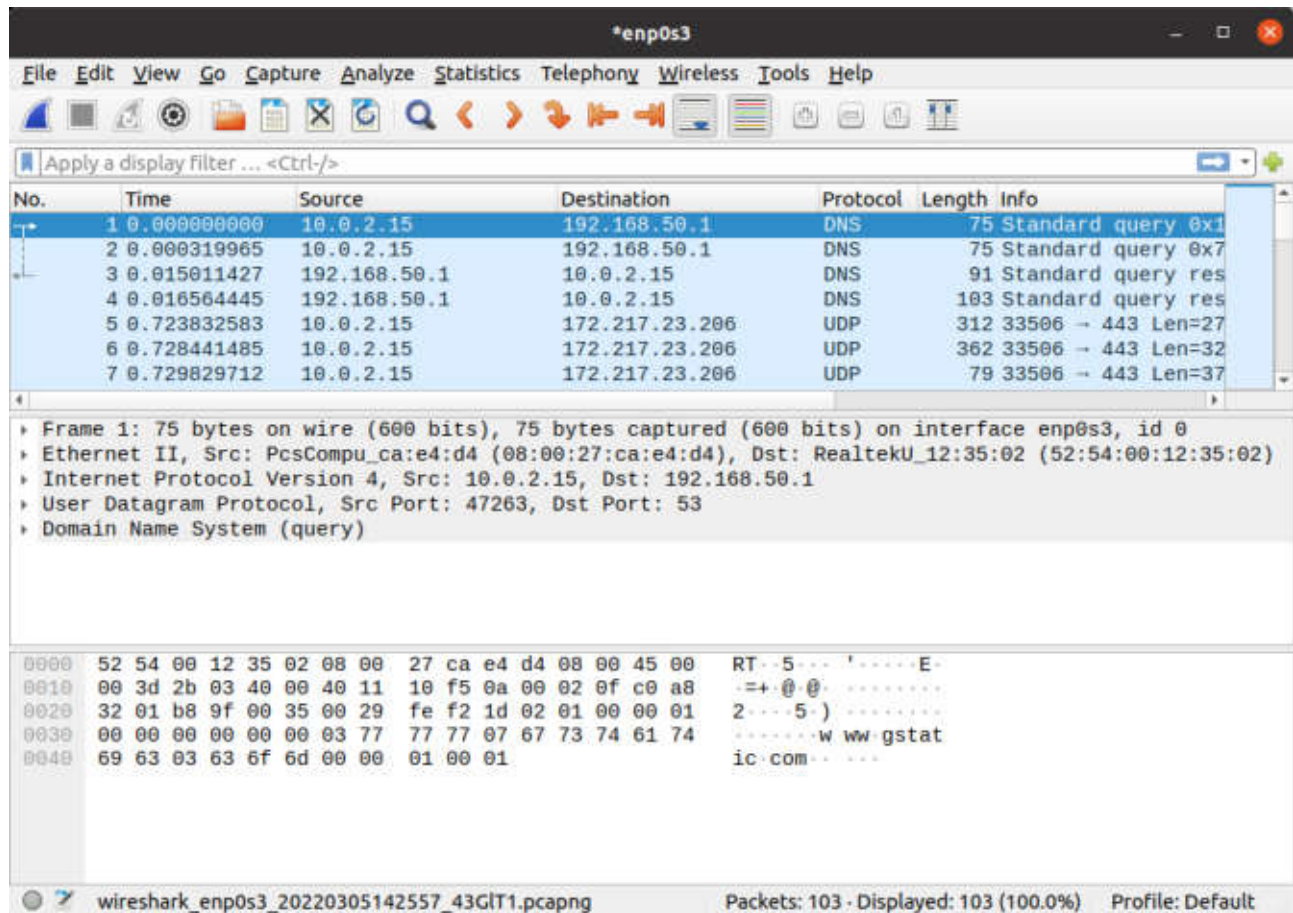
4.6.3 Sniffer



4.7 Kombinace typů paketů

Následující ukázkové příklady testují kombinaci TCP a UDP paketů a kombinaci ICMP a ARP paketů.

4.7.1 TCP a UDP - Wireshark

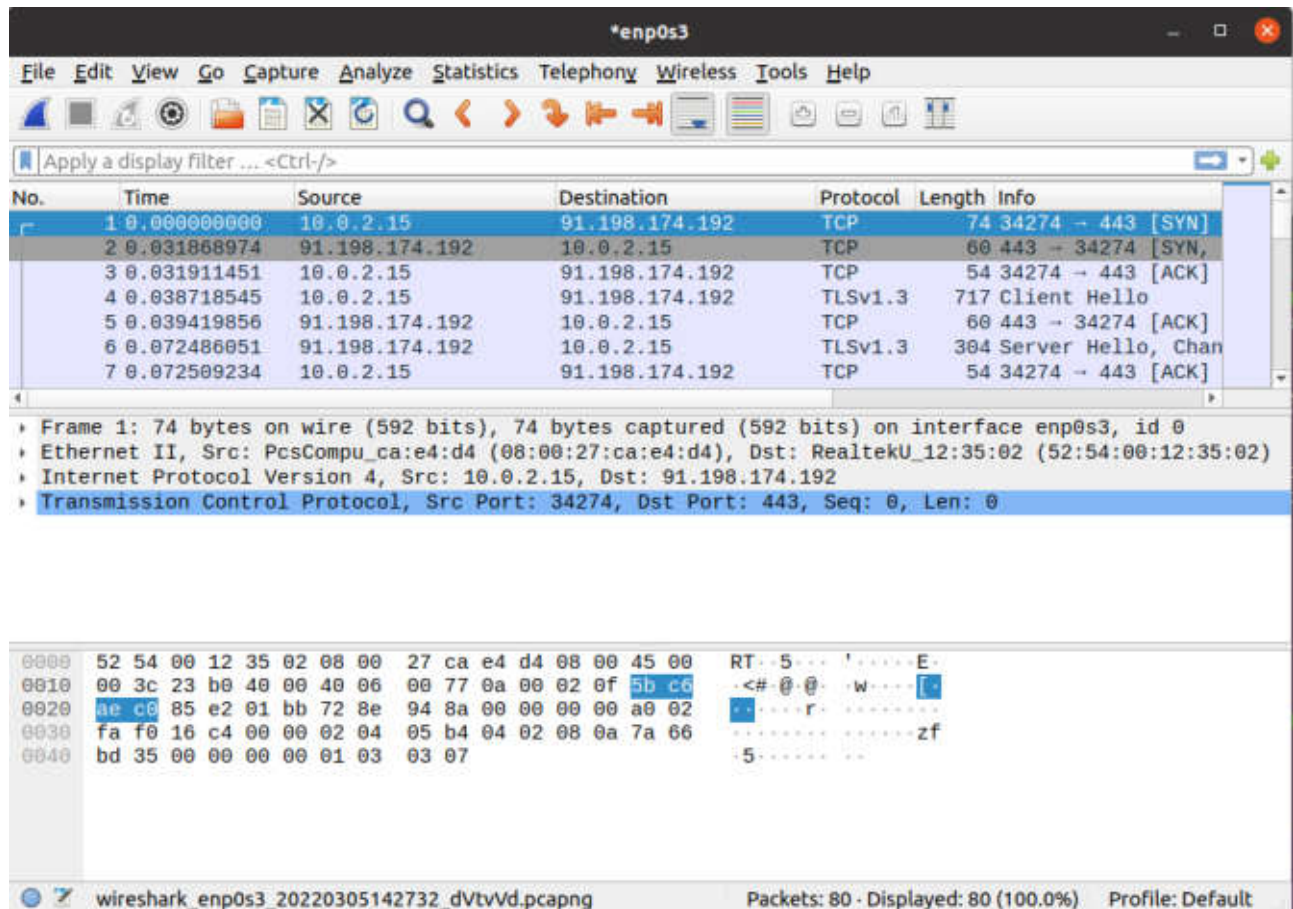


4.7.2 TCP a UDP - Sniffer

```
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i enp0s3 -t -u
timestamp: 2022-03-05T14:26:01.323670+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 75 bytes
src IP: 10.0.2.15
dst IP: 192.168.50.1
src port: 47263
dst port: 53

0x0000: 52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00 RT..5... '.....E.
0x0010: 00 3d 2b 03 40 00 40 11 10 f5 0a 00 02 0f c0 a8 .+=.@.@. ....
0x0020: 32 01 b8 9f 00 35 00 29 fe f2 1d 02 01 00 00 01 2....5.) ....
0x0030: 00 00 00 00 00 00 03 77 77 77 07 67 73 74 61 74 .....w ww.gstat
0x0040: 69 63 03 63 6f 6d 00 00 01 00 01 ic.com.. ...
```

4.7.3 TCP a UDP - Wireshark



The screenshot shows the Wireshark interface with a capture on interface `enp0s3`. The packet list shows a TCP SYN packet (No. 1) from 10.0.2.15 to 91.198.174.192 on port 443. The packet details pane shows the frame structure: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol (Seq: 0, Len: 0). The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	91.198.174.192	TCP	74	34274 → 443 [SYN]
2	0.031868974	91.198.174.192	10.0.2.15	TCP	60	443 → 34274 [SYN, ...]
3	0.031911451	10.0.2.15	91.198.174.192	TCP	54	34274 → 443 [ACK]
4	0.038718545	10.0.2.15	91.198.174.192	TLSv1.3	717	Client Hello
5	0.039419856	91.198.174.192	10.0.2.15	TCP	60	443 → 34274 [ACK]
6	0.072486051	91.198.174.192	10.0.2.15	TLSv1.3	304	Server Hello, Chan...
7	0.072509234	10.0.2.15	91.198.174.192	TCP	54	34274 → 443 [ACK]

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_ca:e4:d4 (08:00:27:ca:e4:d4), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Internet Protocol Version 4, Src: 10.0.2.15, Dst: 91.198.174.192
 Transmission Control Protocol, Src Port: 34274, Dst Port: 443, Seq: 0, Len: 0

0000 52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00 RT..5... '.....E.
 0010 00 3c 23 b0 40 00 40 06 00 77 0a 00 02 0f 5b c6 .<#.@.@. .w....[.
 0020 ae c0 85 e2 01 bb 72 8e 94 8a 00 00 00 00 a0 02r.
 0030 fa f0 16 c4 00 00 02 04 05 b4 04 02 08 0a 7a 66zf
 0040 bd 35 00 00 00 00 01 03 03 07 .5..... ..

wireshark_enp0s3_20220305142732_dVtvVd.pcapng Packets: 80 - Displayed: 80 (100.0%) Profile: Default

4.7.4 TCP a UDP - Sniffer

```
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i enp0s3 -t -u
timestamp: 2022-03-05T14:27:37.750461+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 74 bytes
src IP: 10.0.2.15
dst IP: 91.198.174.192
src port: 34274
dst port: 443

0x0000: 52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00 RT..5... '.....E.
0x0010: 00 3c 23 b0 40 00 40 06 00 77 0a 00 02 0f 5b c6 .<#.@.@. .w....[.
0x0020: ae c0 85 e2 01 bb 72 8e 94 8a 00 00 00 00 a0 02 .....r. ....
0x0030: fa f0 16 c4 00 00 02 04 05 b4 04 02 08 0a 7a 66 .....zf
0x0040: bd 35 00 00 00 00 01 03 03 07 .5..... ..
```


4.7.5 ICMP a ARP - ping

```

student@student-vm:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=11.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=11.1 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=15.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=15.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=117 time=14.5 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=117 time=12.6 ms
^C
--- 8.8.8.8 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 11.056/13.466/15.594/1.740 ms

```

4.7.6 ICMP a ARP - Wireshark

Wireshark capture showing ICMP ping packets. The packet list displays frames 4011 to 4017. Frame 4015 is selected, showing an ICMP Echo (ping) request. The packet details pane shows the Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol layers. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
4011	61.739538779	10.0.2.15	216.58.201.78	UDP	80	39503 → 443 Len=38
4012	61.772409228	216.58.201.78	10.0.2.15	UDP	71	443 → 39503 Len=29
4013	64.622718960	10.0.2.15	93.184.220.29	TCP	54	[TCP Keep-Alive] 4
4014	64.623100613	93.184.220.29	10.0.2.15	TCP	60	[TCP Keep-Alive AC
4015	64.890377703	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request
4016	64.902175554	8.8.8.8	10.0.2.15	ICMP	98	Echo (ping) reply
4017	65.892417856	10.0.2.15	8.8.8.8	ICMP	98	Echo (ping) request

Frame 4015: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp0s3, id 0

- Ethernet II, Src: PcsCompu_ca:e4:d4 (08:00:27:ca:e4:d4), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
- Internet Protocol Version 4, Src: 10.0.2.15, Dst: 8.8.8.8
- Internet Control Message Protocol

Packet bytes (hex): 52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00 RT..5...E..

Packet bytes (hex): 00 54 01 cd 40 00 40 01 1c be 0a 00 02 0f 08 08 T..@..e#b..

Packet bytes (hex): 08 08 08 00 3b d9 00 09 00 01 dc 65 23 62 00 00;...e#b..

Packet bytes (hex): 00 00 f8 81 05 00 00 00 00 00 10 11 12 13 14 15! "\$%&

Packet bytes (hex): 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25&'()*+,-./012345

Packet bytes (hex): 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35

Packet bytes (hex): 36 37 67

Wireshark_enp0s3_...856_qU2cYp.pcapng Packets: 4075 - Displayed: 4075 (100.0%) - Dropped: 0 (0.0%) Profile: Default

4.7.7 ICMP a ARP - Sniffer

```
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i enp0s3 --icmp --arp
timestamp: 2022-03-05T14:30:04.360966+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 98 bytes
src IP: 10.0.2.15
dst IP: 8.8.8.8

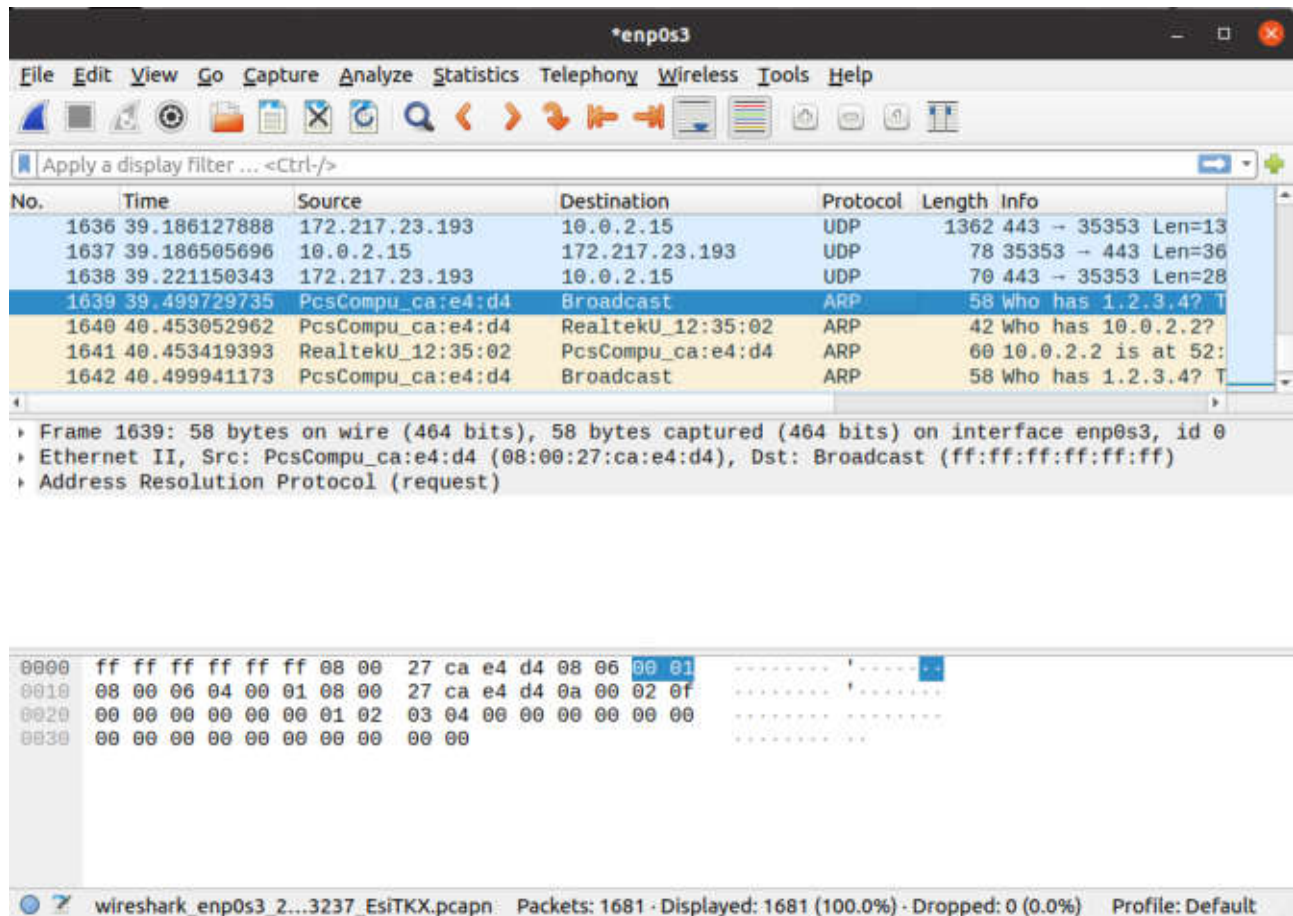
0x0000:  52 54 00 12 35 02 08 00  27 ca e4 d4 08 00 45 00  RT..5... '.....E.
0x0010:  00 54 01 cd 40 00 40 01  1c be 0a 00 02 0f 08 08  .T..@.@. ....
0x0020:  08 08 08 00 3b d9 00 09  00 01 dc 65 23 62 00 00  ....;... ..e#b..
0x0030:  00 00 f8 81 05 00 00 00  00 00 10 11 12 13 14 15  ....
0x0040:  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25  .... ..!"#$%
0x0050:  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35  8'()*+,- ./012345
0x0060:  36 37                                     67
```

4.7.8 ICMP a ARP - arping

```
student@student-vm:~$ sudo arping -c 4 -A -I enp0s3 1.2.3.4
ARPING 1.2.3.4
Timeout
Timeout
Timeout
Timeout

--- 1.2.3.4 statistics ---
4 packets transmitted, 0 packets received, 100% unanswered (0 extra)
```

4.7.9 ICMP a ARP - Wireshark



The image shows a Wireshark capture window titled '*enp0s3'. The packet list pane displays several packets, with packet 1639 selected. The packet details pane shows the structure of the selected ARP request packet. The packet bytes pane shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
1636	39.186127888	172.217.23.193	10.0.2.15	UDP	1362	443 → 35353 Len=13
1637	39.186505696	10.0.2.15	172.217.23.193	UDP	78	35353 → 443 Len=36
1638	39.221150343	172.217.23.193	10.0.2.15	UDP	70	443 → 35353 Len=28
1639	39.499729735	PcsCompu_ca:e4:d4	Broadcast	ARP	58	who has 1.2.3.4? T
1640	40.453052962	PcsCompu_ca:e4:d4	RealtekU_12:35:02	ARP	42	who has 10.0.2.2?
1641	40.453419393	RealtekU_12:35:02	PcsCompu_ca:e4:d4	ARP	60	10.0.2.2 is at 52:
1642	40.499941173	PcsCompu_ca:e4:d4	Broadcast	ARP	58	who has 1.2.3.4? T

Frame 1639: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_ca:e4:d4 (08:00:27:ca:e4:d4), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 Address Resolution Protocol (request)

```

0000  ff ff ff ff ff ff 08 00 27 ca e4 d4 08 06 00 01  .....
0010  08 00 06 04 00 01 08 00 27 ca e4 d4 0a 00 02 0f  .....
0020  00 00 00 00 00 00 01 02 03 04 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00  .....
  
```

wireshark_enp0s3_2...3237_EsITKX.pcapn Packets: 1681 · Displayed: 1681 (100.0%) · Dropped: 0 (0.0%) Profile: Default

4.7.10 ICMP a ARP - Sniffer

```

student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i enp0s3 --icmp --arp
timestamp: 2022-03-05T14:33:19.794388+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: ff:ff:ff:ff:ff:ff
frame length: 58 bytes

0x0000:  ff ff ff ff ff ff 08 00 27 ca e4 d4 08 06 00 01  .....
0x0010:  08 00 06 04 00 01 08 00 27 ca e4 d4 0a 00 02 0f  .....
0x0020:  00 00 00 00 00 00 01 02 03 04 00 00 00 00 00 00  .....
0x0030:  00 00 00 00 00 00 00 00 00 00  .....
  
```


4.8 Argument *port*

Dále byl testován argument *port* zadáný ve tvaru *-p port*. Test byl proveden pomocí vytvoření vlastních UDP paketů zasílaných na různé porty. Sniffer úspěšně zachytil až paket s portem zadaným vstupním argumentem.

4.8.1 Testovací příkazy

```
student@student-vm:~$ echo "This is my data" > /dev/udp/127.0.0.1/80
student@student-vm:~$ echo "This is my data" > /dev/udp/127.0.0.1/443
student@student-vm:~$ echo "This is my data" > /dev/udp/127.0.0.1/5000
student@student-vm:~$ echo "This is my data" > /dev/udp/127.0.0.1/2999
student@student-vm:~$ echo "This is my data" > /dev/udp/127.0.0.1/3001
student@student-vm:~$ echo "This is my data" > /dev/udp/127.0.0.1/3000
student@student-vm:~$
```

4.8.2 Wireshark

The screenshot shows the Wireshark interface with the following details:

- Packet List:**

No.	Time	Source	Destination	Protocol	Length	Info
6	6.543891651	127.0.0.1	127.0.0.1	ICMP	86	Destination unread...
7	9.807611866	127.0.0.1	127.0.0.1	UDP	58	42890 → 2999 Len=1
8	9.807627938	127.0.0.1	127.0.0.1	ICMP	86	Destination unread...
9	12.832014820	127.0.0.1	127.0.0.1	UDP	58	49050 → 3001 Len=1
10	12.832053259	127.0.0.1	127.0.0.1	ICMP	86	Destination unread...
11	16.096276297	127.0.0.1	127.0.0.1	DIS	58	PDUType: 105
12	16.096294260	127.0.0.1	127.0.0.1	ICMP	86	Destination unread...
- Packet Details (Frame 11):**
 - Frame 11: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface lo, id 0
 - Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
 - Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
 - User Datagram Protocol, Src Port: 38996, Dst Port: 3000
 - Distributed Interactive Simulation
- Packet Bytes:**

```
0000  00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  -----E-
0010  00 2c 72 6a 40 00 40 11 ca 54 7f 00 00 01 7f 00  ,rj@-@-T-----
0020  00 01 98 54 0b b8 00 18 fe 2b 54 68 69 73 20 69  ...T....+This i
0030  73 20 6d 79 20 64 61 74 61 0a                      s my dat a
```

The status bar at the bottom indicates: **wireshark_lo_20220305154300_hFRFR5.pcapng** Packets: 14 - Displayed: 14 (100.0%) - Dropped: 0 (0.0%) Profile: Default

4.8.3 Sniffer

```
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -u -i lo -p 3000
timestamp: 2022-03-05T15:43:19.903793+01:00
src MAC: 00:00:00:00:00:00
dst MAC: 00:00:00:00:00:00
frame length: 58 bytes
src IP: 127.0.0.1
dst IP: 127.0.0.1
src port: 38996
dst port: 3000

0x0000:  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  .....E.
0x0010:  00 2c 72 6a 40 00 40 11 ca 54 7f 00 00 01 7f 00  .,rj@.@. .T.....
0x0020:  00 01 98 54 0b b8 00 18 fe 2b 54 68 69 73 20 69  ...T.... .+This i
0x0030:  73 20 6d 79 20 64 61 74 61 0a                      s my dat a.
```

4.9 Argument *num*

Testování argumentu *num* bylo testováno pro zobrazení 3 paketů.

4.9.1 Wireshark

The screenshot shows the Wireshark interface with the following details:

- Packet List:**

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	18.67.65.28	TCP	54	48254 → 443 [ACK]
2	0.000249953	18.67.65.28	10.0.2.15	TCP	60	[TCP ACKed unseen]
3	5.120978937	PcsCompu_ca:e4:d4	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2?
4	5.121191164	RealtekU_12:35:02	PcsCompu_ca:e4:d4	ARP	60	10.0.2.2 is at 52:
5	7.983496949	10.0.2.15	34.120.208.123	TLSv1.2	93	Application Data
6	7.983850907	34.120.208.123	10.0.2.15	TCP	60	443 → 39404 [ACK]
7	8.010819441	34.120.208.123	10.0.2.15	TLSv1.2	93	Application Data
- Packet Details (Frame 1):**
 - Ethernet II, Src: PcsCompu_ca:e4:d4 (08:00:27:ca:e4:d4), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 - Internet Protocol Version 4, Src: 10.0.2.15, Dst: 18.67.65.28
 - Transmission Control Protocol, Src Port: 48254, Dst Port: 443, Seq: 1, Ack: 1, Len: 0
- Packet Bytes:**

```
0000  52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00  RT..5... '....E.
0010  00 28 71 a8 40 00 40 06 69 ba 0a 00 02 0f 12 43  .(q-@.@. i.....C
0020  41 1c bc 7e 01 bb 98 ca bb 08 73 5b 54 46 50 10  A-----s[TFP
0030  f9 9c 5f 88 00 00                                .....
```

The status bar at the bottom indicates: **wireshark_enp0s3_20220305152342_STI7IM.pcapng** | **Packets: 8 · Displayed: 8 (100.0%)** | **Profile: Default**

The image shows the Wireshark network protocol analyzer interface. The title bar indicates the capture is on interface *enp0s3. The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture control, and analysis. A display filter bar shows 'Apply a display filter ... <Ctrl-/>'. The packet list pane displays eight captured packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	18.67.65.28	TCP	54	48254 → 443 [ACK]
2	0.000249953	18.67.65.28	10.0.2.15	TCP	60	[TCP ACKed unseen
3	5.120978937	PcsCompu_ca:e4:d4	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2?
4	5.121191164	RealtekU_12:35:02	PcsCompu_ca:e4:d4	ARP	60	10.0.2.2 is at 52:
5	7.983496949	10.0.2.15	34.120.208.123	TLSv1.2	93	Application Data
6	7.983850907	34.120.208.123	10.0.2.15	TCP	60	443 → 39404 [ACK]
7	8.010819441	34.120.208.123	10.0.2.15	TLSv1.2	93	Application Data

The packet details pane for the selected packet (Frame 2) shows the following structure:

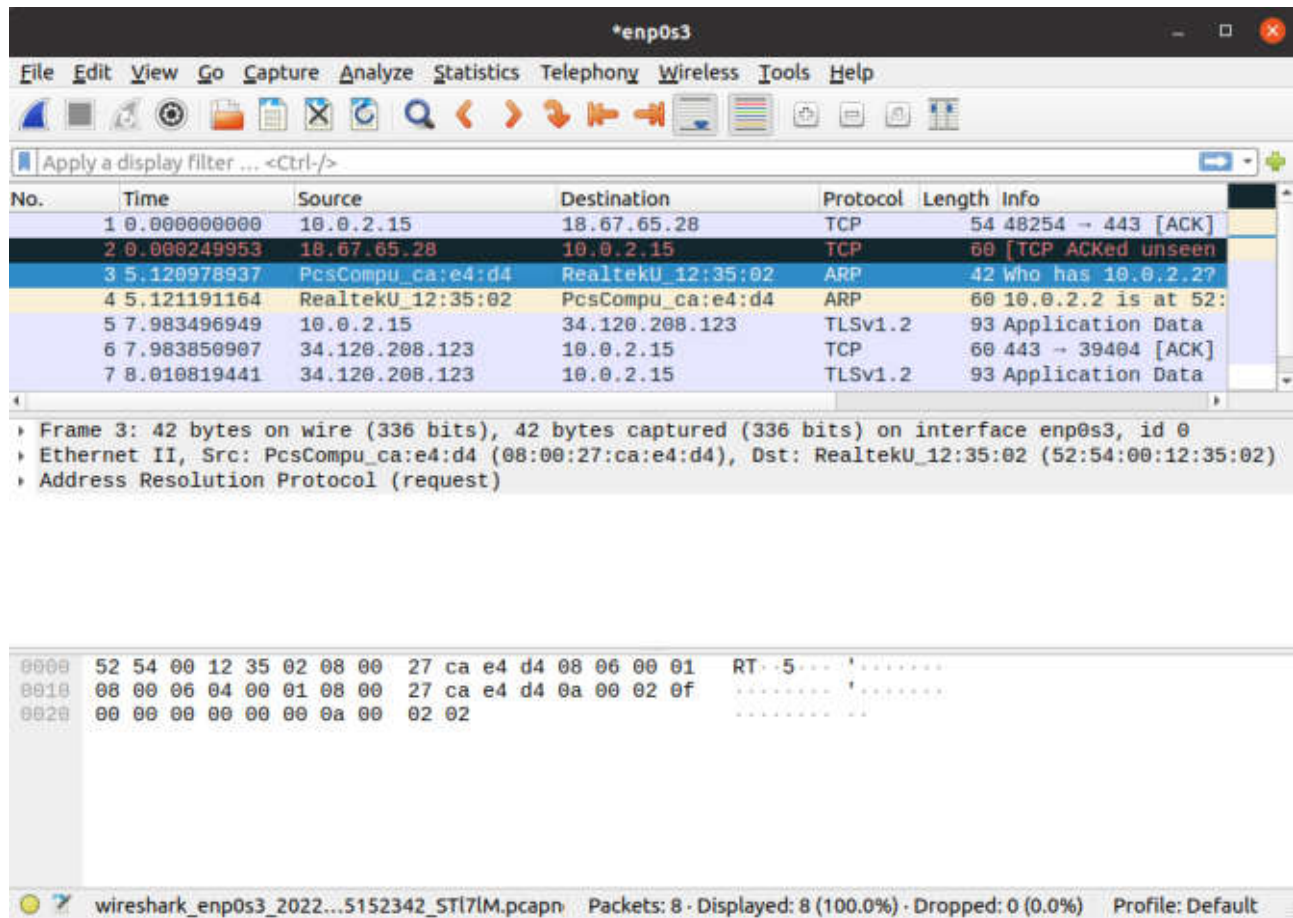
- Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface enp0s3, id 0
- Ethernet II, Src: RealtekU_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu_ca:e4:d4 (08:00:27:ca:e4:d4)
- Internet Protocol Version 4, Src: 18.67.65.28, Dst: 10.0.2.15
- Transmission Control Protocol, Src Port: 443, Dst Port: 48254, Seq: 1, Ack: 2, Len: 0

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```

0000  08 00 27 ca e4 d4 52 54 00 12 35 02 08 00 45 00  ..'.RT..5..E.
0010  00 28 97 47 00 00 40 06 84 1b 12 43 41 1c 0a 00  (.G.@...CA...
0020  02 0f 01 bb bc 7e 73 5b 54 46 98 ca bb 09 50 10  ....-s[TF...P.
0030  ff ff 76 b7 00 00 00 00 00 00 00 00          ...v.....
  
```

The status bar at the bottom indicates the capture file is 'wireshark_enp0s3_2022...5152342_STl7IM.pcapn', with 8 packets displayed (100.0%) and 0 dropped (0.0%). The profile is set to Default.



The image displays the Wireshark network protocol analyzer interface. The top section shows a list of captured packets on the interface `enp0s3`. The bottom section provides a detailed view of the selected packet (Frame 3), showing its structure and raw data.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	18.67.65.28	TCP	54	48254 → 443 [ACK]
2	0.000249953	18.67.65.28	10.0.2.15	TCP	60	[TCP ACKed unseen
3	5.120978937	PcsCompu_ca:e4:d4	RealtekU_12:35:02	ARP	42	Who has 10.0.2.2?
4	5.121191164	RealtekU_12:35:02	PcsCompu_ca:e4:d4	ARP	60	10.0.2.2 is at 52:
5	7.983496949	10.0.2.15	34.120.208.123	TLSv1.2	93	Application Data
6	7.983850907	34.120.208.123	10.0.2.15	TCP	60	443 → 39404 [ACK]
7	8.010819441	34.120.208.123	10.0.2.15	TLSv1.2	93	Application Data

Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3, id 0
 Ethernet II, Src: PcsCompu_ca:e4:d4 (08:00:27:ca:e4:d4), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
 Address Resolution Protocol (request)

```

0000  52 54 00 12 35 02 08 00 27 ca e4 d4 08 06 00 01  RT-5...
0010  08 00 06 04 00 01 08 00 27 ca e4 d4 0a 00 02 0f  .....
0020  00 00 00 00 00 00 0a 00 02 02  .....
  
```

wireshark_enp0s3_2022...5152342_STI7IM.pcapn Packets: 8 - Displayed: 8 (100.0%) - Dropped: 0 (0.0%) Profile: Default

4.9.2 Sniffer

```

student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i enp0s3 -n 3
timestamp: 2022-03-05T15:23:43.37330+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 54 bytes
src IP: 10.0.2.15
dst IP: 18.67.65.28
src port: 48254
dst port: 443

0x0000:  52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00  RT..5... '.....E.
0x0010:  00 28 71 a8 40 00 40 06 69 ba 0a 00 02 0f 12 43    .(q.@.@. i.....C
0x0020:  41 1c bc 7e 01 bb 98 ca bb 08 73 5b 54 46 50 10    A..~.... ..s[TFP.
0x0030:  f9 9c 5f 88 00 00                                .._...

timestamp: 2022-03-05T15:23:43.37580+01:00
src MAC: 52:54:00:12:35:02
dst MAC: 08:00:27:ca:e4:d4
frame length: 60 bytes
src IP: 18.67.65.28
dst IP: 10.0.2.15
src port: 443
dst port: 48254

0x0000:  08 00 27 ca e4 d4 52 54 00 12 35 02 08 00 45 00  ..'...RT ..5...E.
0x0010:  00 28 97 47 00 00 40 06 84 1b 12 43 41 1c 0a 00  .(.G..@. ...CA...
0x0020:  02 0f 01 bb bc 7e 73 5b 54 46 98 ca bb 09 50 10  ....~s[ TF....P.
0x0030:  ff ff 76 b7 00 00 00 00 00 00 00 00 00 00 00 00  ..V..... ....

timestamp: 2022-03-05T15:23:48.158309+01:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 42 bytes

0x0000:  52 54 00 12 35 02 08 00 27 ca e4 d4 08 06 00 01  RT..5... '.....
0x0010:  08 00 06 04 00 01 08 00 27 ca e4 d4 0a 00 02 0f  .... '.....
0x0020:  00 00 00 00 00 00 0a 00 02 02                    .... ..

student@student-vm:~/Documents/IPK-Project2$ █

```


4.9.3 Spuštění snifferu bez určení rozhraní

Spuštění snifferu bez určení rozhraní vypisuje aktivní rozhraní.

```
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer
enp0s3
lo
any
bluetooth-monitor
nflog
nfqueue
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -i
enp0s3
lo
any
bluetooth-monitor
nflog
nfqueue
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -t -u -n 10
enp0s3
lo
any
bluetooth-monitor
nflog
nfqueue
student@student-vm:~/Documents/IPK-Project2$ sudo ./ipk-sniffer -t -u -i -n 10
enp0s3
lo
any
bluetooth-monitor
nflog
nfqueue
```

Reference

- [1] TIMESTAMP. <https://gist.github.com/jedisct1/b7812ae9b4850e0053a21c922ed3e9dc>. [Online].
- [2] WIRESHARK. <https://www.wireshark.org/>. [Online].
- [3] RFC3339. <https://datatracker.ietf.org/doc/html/rfc3339>, 2002. [Online].
- [4] IPv6 EXTENDED HEADER CALCULATION. https://github.com/yuan901202/vuw_nwen302_ethernet_packet_sniffer/blob/master/eps.c, 2015. [Online].
- [5] PRINT DATA FUNCTION. <https://www.binarytides.com/package-sniffer-code-c-libpcap-linux-sockets/>, 2020. [Online].
- [6] IP ADDRESS OF PACKET. <https://stackoverflow.com/questions/21222369/getting-ip-address-of-a-packet-in-pcap-file>, 2021. [Online].
- [7] MAN PAGE OF PCAP-FILTER. <https://www.tcpdump.org/manpages/pcap-filter.7.html>, 2022. [Online].
- [8] VESELÝ, V. IPv6 Síťová vrstva. Univerzitní přednáška, 2022.