



# Projektová dokumentácia

Implementácia sniffera packetov

IPK, projekt č.2

# Obsah

Úvod	3
Teória	3
Sniffer	3
Paket	3
Implementácia	4
Implementačné detaily	4
Knížnice	4
Dôležité funkcie	5
void print_error_and_exit(string string)	5
void print_interfaces(pcap_if_t *interface_list)	5
pcap_if_t *get_interfaces()	5
void check_existing_interface(string interface, pcap_if_t *interface_list)	5
void parse_args(int argc, char *argv[])	5
void add_to_filter(string str)	5
void create_filter()	5
void print_packet_data(const void *addr, int len)	6
void print_ipv6(const struct in6_addr *addr)	6
void print_formatted_time(const struct pcap_pkthdr *header)	6
void print_mac_address(const struct ether_header *ether_header)	6
void print_ipv4(struct ip *ip)	6
void print_mac_address(const struct ether_header *ether_header)	6
void sniffer(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)	6
void catch_sigint(int signal)	6
Testovanie	7
Popis testovania	7
Priebeh testovania	7
Report z testovania	7
ARP	7
UDP	7
TCP	8
ICMP	8
ICMPv6	8
UDP – Ipv6	8
Bibliografia	9

# Úvod

Cieľom projektu bolo implementovať program, ktorý dokáže odchytať sieťovú komunikáciu na danom rozhraní, pomocou filtra získať len zhodné pakety a následne ich vypísať v požadovanom formáte. Projekt bol naimplementovaný v jazyku C++ hlavne pre jeho rozšírenie STD oproti jazyku C, ktoré umožňuje jednoduchšiu prácu so 'stringami'. Program umožňuje odchytať a filtrovať pakety na rôznych portoch a s rôznymi protokolmi. Medzi podporované protokoly patria UDP, TCP, ARP, ICMP.

## Teória

Pri študovaní problematiky boli použité ako zdroje informácií tcpdump<sup>1</sup> a stackoverflow<sup>2</sup>.

### Sniffer

Sniffer paketov – tiež známy ako analyzátor paketov, analyzátor protokolov alebo sieťový analyzátor – je softvér používaný na monitorovanie sieťovej prevádzky. Sniffery fungujú tak, že skúmajú toky dátových paketov, ktoré prechádzajú medzi počítačmi v sieti, ako aj medzi počítačmi v sieti a väčším internetom. Tieto pakety sú určené a adresované konkrétnym počítačom, ale používanie paketového sniffera v „transparentnom režime“ umožňuje IT profesionálom, koncovým používateľom alebo útočníkom preskúmať akýkoľvek paket bez ohľadu na miesto určenia.

### Paket

Paket označuje v informatike blok dát prenášaný v počítačových sieťach založených na prepojení paketov. Paket sa skladá z riadiacich dát (metadát) a z užívateľských dát (užitočné zaťaženie, anglicky payload). Riadiace dáta poskytujú sieti potrebné dáta na doručenie paketu, napríklad adresu zdroja a cieľa, kódy pre detekciu chýb, kontrolné súčty a informácie o poradí. Obvykle sa riadiace dáta nachádzajú v hlavičkách paketov a na ich konci, pričom užívateľské dáta sú medzi nimi.

Rôzne komunikačné protokoly používajú rôzne konvencie na rozlišovanie medzi riadiacimi prvkami a dátami. V binárnych synchrónnych prenosoch používajú pakety 8-bitové skupiny (bajty) a na vymedzenie jednotlivých prvkov sú použité špeciálne znaky. Ďalšie protokoly (napríklad Ethernet), definujú začiatok hlavičky a dátových prvkov ako pozíciu vzhľadom na začiatok paketu. Niektoré protokoly formátujú informácie na úrovni bitov namiesto použitia bajtovej úrovne [1].

---

<sup>1</sup> <https://www.tcpdump.org/manpages/pcap.3pcap.html>

<sup>2</sup> <https://stackoverflow.com/questions/tagged/pcap>

# Implementácia

## Implementačné detaily

Implementácia začína spracovaním argumentov, kde sa roztriedia argumenty a zároveň sa nastaví 'flagy' pre jednotlivé protokoly. Následne sa vytvorí filter za pomoci spomínaných flagov a voliteľného portu, ktorý bude použitý na odchytenie paketov. Tento port sa používa len pri UDP a TCP, keďže ARP ani ICMP nemajú protokoly. Tento filter zatiaľ v tvare 'stringu' je skompilovaný pomocou funkcie `pcap_compile`. Následne je filter aplikovaný pomocou `pcap_setfilter` a začnú sa odchyťvať pakety pomocou funkcie `pcap_loop`. Keď už sú pakety odchytené je zavolaná funkcia, ktorá paket spracuje cez switch-case konštrukciu a podľa príslušného protokolu sú dáta v pakete naformátované a vytlačené na STDOUT. Hrubšia štruktúra programu bola inšpirovaná `sniffex-om`<sup>3</sup> a návodom na prácu s `pcap` [2].

## Knižnice

Na implementáciu sniffera boli použité nasledovné knižnice na správu paketov, hlavne knižnica `pcap.h`

- `<iostream>`
- `<string>`
- `<string.h>`
- `<stdio.h>`
- `<unistd.h>`
- `<ctype.h>`
- `<cmath>`
- `<sstream>`
- `<pcap.h>`
- `<net/ethernet.h>`
- `<netinet/ip.h>`
- `<netinet/ip_icmp.h>`
- `<netinet/in.h>`
- `<netinet/tcp.h>`
- `<netinet/udp.h>`
- `<netinet/if_ether.h>`
- `<netinet/ether.h>`
- `<netinet/ip6.h>`
- `<signal.h>`

Netinet knižnice boli použité na prácu s hlavičkami paketov a prácu s nimi.

Knižnica `signal.h` bola použitá na odchytenie signálu SIGINT a následnému korektnému uvoľneniu.

---

<sup>3</sup> <https://www.tcpdump.org/other/sniffex.c>

## Dôležité funkcie

Funkcie ktoré boli naimplementované na spracovávanie a prácu s dátami paketu a ktoré zaisťujú celkovú funkčnosť programu.

`void print_error_and_exit(string string)`

Funkcia vytlačí chybovú hlášku na STDERR a ukončí program s chybou.

`void print_interfaces(pcap_if_t *interface_list)`

Funkcia vytlačí všetky dostupné rozhrania a uvoľní zoznam pomocou `pcap_freealldevs()`. Rozhrania sú uložené v dátovom type zoznam cez ktorý sa iteruje pomocou `->next` a dáta sa vytlačia na STDOUT.

`pcap_if_t *get_interfaces()`

Funkcia nájde všetky rozhrania pomocou `pcap_findalldevs()` a vráti zoznam.

`void check_existing_interface(string interface, pcap_if_t *interface_list)`

Funkcia kontroluje, či je dané rozhranie v premennej `interface` platné a či je v zozname rozhraní. Ak nie je, program sa ukončí s chybou.

`void parse_args(int argc, char *argv[])`

Funkcia skontroluje, či sú nastavené nejaké argumenty, ak nie, vypíše všetky dostupné rozhrania. Ak sú nastavené argumenty, skontroluje `-i` alebo `--interface`. Ak nie je zadané, vytlačia sa všetky dostupné rozhrania. Ak je dané rozhranie, uloží ho. Potom funkcia skontroluje `-h` alebo `--help` argument a ak ho nájde, vypíše nápovedu. Potom funkcia skontroluje argumenty ako `{-p port} [--tcp|-t] [--udp|-u] [--arp] [--icmp] [--igmp] {-n num}` a nastaví booleovské príznaky alebo zodpovedajúce argumenty. Funkcia kontroluje správne argumenty 'num' a 'port'. Ak nie je nastavený žiadny protokol, príznaky pre všetky protokoly sú nastavené na hodnotu TRUE.

`void add_to_filter(string str)`

Funkcia pripojí `or` a 'str' do premennej `filter`. Ak je premenná `filter` prázdna, funkcia nepripojí prvé „or“.

`void create_filter()`

Funkcia vytvorí filter, ktorý sa potom použije vo funkcii `pcap_compile()` na vytvorenie filtra používaného na filtrovanie paketov. Funkcia sa ovláda pomocou booleovských príznakov protokolov. Ak je príznak nastavený na hodnotu true, zavolá sa funkcia `add_to_filter()` s príslušným protokolom, ktorý sa má pridať k filtru. Funkcia spracováva aj okrajové prípady, ako je parameter `-p num` set bez možnosti `udp` alebo `tcp`, takže sa automaticky pridajú na filtrovanie. Funkcia pracuje s globálnou premennou `filter`.

`void print_packet_data(const void *addr, int len)`

Funkcia vytlačí paketové dáta vo formáte 'hexdump' a zmení všetky netlačiteľné znaky na bodky [3] [4].

`void print_ipv6(const struct in6_addr *addr)`

Funkcia vytlačí IPv6 adresu v správnom formáte [5].

`void print_formatted_time(const struct pcap_pkthdr *header)`

Funkcia preberá dve hodnoty z hlavičky paketu - tv\_sec a tv\_usec a transformuje ich do formátu času RC3339 [6].

`void print_mac_address(const struct ether_header *ether_header)`

Funkcia vytlačí zdrojové a cieľové MAC adresy z ethernetovej hlavičky.

`void print_ipv4(struct ip *ip)`

Funkcia vytlačí IPv4 zo štruktúry IP.

`void print_mac_address(const struct ether_header *ether_header)`

Funkcia vytlačí zdrojové a cieľové MAC adresy z ethernetovej hlavičky [7].

`void sniffer(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)`

Funkcia nájde paket a vytlačí jeho obsah. Prvé správne hodnoty sú priradené k štruktúram ako ip, iphdr, ip6\_hdr, ether\_header, udphdr, tcphdr. Potom sa pomocou prepínača nájde a spracuje správny protokol.

Podporované protokoly sú UDP, TCP, ARP, ICMP, ICMPv6.

Nájdenny protokol sa potom vytlačí vytlačením času zachytenia paketu, zdrojovej a cieľovej MAC adresy, dĺžky rámca, zdrojovej a cieľovej IP adresy a následne naformátovaných údajov [8] .

`void catch_sigint(int signal)`

Funkcia spracuje SIGINT, uvoľní všetky pridelené zdroje a skončí s chybou.

# Testovanie

## Popis testovania

Testovanie slúžilo na overenie, či sa požadované dáta vypisujú v správnom formáte. Pri implementácii bol použitý princíp TDD (Test Driven Development), pri ktorom sa kontrolovala správnosť za pomoci nástroja Wireshark<sup>4</sup>. Pre generáciu paketov bol použitý príkaz nping<sup>5</sup>.

## Priebeh testovania

Pri testovaní bol použitý vyššie spomínaný nástroj nping, ktorý dokázal generovať ARP, UDP, TCP a ICMP. Nástroj bohužiaľ nevedel generovať ICMPv6 pakety ani UDP alebo TCP pakety IPv6. ICMPv6 pakety a UDPv6 pakety bolo možné odchytiť z štandardného rozhrania, ale TCPv6 pakety nebolo možné nájsť a odchytiť a teda ani otestovať.

## Report z testovania

Ako report z testovania sú priložené screenshots z nástroja Wireshark, nástroja pre generovanie paketov a implementovaného sniffera pre všetky možné protokoly.<sup>6</sup>

### ARP

▼ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface lo, id 0  
▶ Interface id: 0 (lo)

Encapsulation type: Ethernet (1)  
Arrival Time: Apr 23, 2022 22:41:57.874837725 CEST  
[Time shift for this packet: 0.000000000 seconds]  
Epoch Time: 1650746517.874837725 seconds  
[Time delta from previous captured frame: 0.000000000 seconds]  
[Time delta from previous displayed frame: 0.000000000 seconds]  
[Time since reference or first frame: 0.000000000 seconds]  
Frame Number: 1  
Frame Length: 42 bytes (336 bits)  
Capture Length: 42 bytes (336 bits)  
[Frame is marked: False]  
[Frame is ignored: False]  
[Protocols in frame: eth:ethertype:arp]  
[Coloring Rule Name: ARP]  
[Coloring Rule String: arp]  
▼ Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)  
▶ Source: 00:00:00:00:00:00 (00:00:00:00:00:00)  
Type: ARP (0x0806)  
▼ Address Resolution Protocol (request)  
Hardware type: Ethernet (1)  
Protocol type: IPv4 (0x0800)  
Hardware size: 6  
Protocol size: 4  
Opcode: request (1)  
Sender MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)  
Sender IP address: 192.168.0.1  
Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)  
Target IP address: 127.0.0.1

! k ~/D/1/Proj2 k sudo nping --arp 127.0.0.1 --source-ip 192.168.0.1

Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2022-04-23 22:41 CEST  
SENT (0.0578s) ARP who has 127.0.0.1? Tell 192.168.0.1

ARP  
timestamp: 2022-04-23T22:41:57.874+02:00  
src MAC: 00:00:00:00:00:00  
dst MAC: ff:ff:ff:ff:ff:ff  
frame length: 42 bytes  
src IP: 192.168.0.1  
dst IP: 127.0.0.1

0x0000: ff ff ff ff ff ff 00 00 00 00 08 06 00 01 .....  
0x0010: 08 00 06 04 00 01 00 00 00 00 00 c0 a8 00 01 .....  
0x0020: 00 00 00 00 00 00 7f 00 00 01 ..... \*

### UDP

▼ Frame 14: 50 bytes on wire (400 bits), 50 bytes captured (400 bits) on interface lo, id 0  
▶ Interface id: 0 (lo)

Encapsulation type: Ethernet (1)  
Arrival Time: Apr 23, 2022 22:44:47.994160307 CEST  
[Time shift for this packet: 0.000000000 seconds]  
Epoch Time: 1650746607.994160307 seconds  
[Time delta from previous captured frame: 11.520744750 seconds]  
[Time delta from previous displayed frame: 11.520744750 seconds]  
[Time since reference or first frame: 35.754575467 seconds]  
Frame Number: 14  
Frame Length: 50 bytes (400 bits)  
Capture Length: 50 bytes (400 bits)  
[Frame is marked: False]  
[Frame is ignored: False]  
[Protocols in frame: eth:ethertype:ip:udp:cipio]  
[Coloring Rule Name: UDP]  
[Coloring Rule String: udp]  
▼ 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)  
▶ Destination: 00:00:00:00:00:00 (00:00:00:00:00:00)  
▶ Source: 00:00:00:00:00:00 (00:00:00:00:00:00)  
Type: IPv4 (0x0800)  
▼ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
User Datagram Protocol, Src Port: 2222, Dst Port: 1111  
Source Port: 2222  
Destination Port: 1111  
Length: 16  
Checksum: 0x8807 [unverified]  
[Checksum Status: Unverified]  
[Stream Index: 3]  
[Timestamps]

~/D/1/Proj2 k sudo nping --udp -p 1111 -g 2222 127.0.0.1 --data-string "test udp"

Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2022-04-23 22:44 CEST  
SENT (0.0613s) UDP 127.0.0.1:2222 > 127.0.0.1:1111 ttl=64 id=53819 iplen=36

UDP  
timestamp: 2022-04-23T22:44:47.994+02:00  
src MAC: 00:00:00:00:00:00  
dst MAC: 00:00:00:00:00:00  
frame length: 50 bytes  
src IP: 127.0.0.1  
dst IP: 127.0.0.1  
src port: 2222  
dst port: 1111

0x0000: 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 .....E.  
0x0010: 00 24 cd 45 00 00 40 11 af 81 7f 00 00 01 7f 00 ..\$.E..e.....  
0x0020: 00 01 08 ae 04 57 00 10 88 07 74 65 73 74 20 75 .....W...test u  
0x0030: 64 70 dp

<sup>4</sup> Nástroj Wireshark z webovej lokality <https://www.wireshark.org/>

<sup>5</sup> Nástroj nping z webovej lokality <https://nmap.org/nping/#:~:text=Nping%20is%20an%20open%20source,full%20control%20over%20protocol%20headers.>

<sup>6</sup> Názov protokolu pri sniffer-e (práva strana testovacieho reportu pre jednotlivý protokoly) je len z informačných dôvodov a vo finálnej verzii sa nenachádza.

## TCP

<div>▼ Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface lo, id 0</div> <div>► Interface id: 0 (lo)</div> <div>Encapsulation type: Ethernet (1)</div> <div>Arrival Time: Apr 23, 2022 22:46:13.122167397 CEST</div> <div>[Time shift for this packet: 0.000000000 seconds]</div> <div>Epoch Time: 1650746773.122167397 seconds</div> <div>[Time delta from previous captured frame: 0.000000000 seconds]</div> <div>[Time delta from previous displayed frame: 0.000000000 seconds]</div> <div>[Time since reference or first frame: 0.000000000 seconds]</div> <div>Frame Number: 1</div> <div>Frame Length: 62 bytes (496 bits)</div> <div>Capture Length: 62 bytes (496 bits)</div> <div>[Frame is marked: False]</div> <div>[Frame is ignored: False]</div> <div>Protocols in frame: eth:ethertype:ip:tcp:data</div> <div>[Coloring Rule Name: TCP SYN/FIN]</div> <div>[Coloring Rule String: tcp.flags &amp; 0x02    tcp.flags.fin == 1]</div> <div>► 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)</div> <div>► Destination: 00:00:00:00:00:00 (00:00:00:00:00:00)</div> <div>► Source: 00:00:00:00:00:00 (00:00:00:00:00:00)</div> <div>Type: IPv4 (0x0800)</div> <div>► Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1</div> <div>► Transmission Control Protocol, Src Port: 2222, Dst Port: 1111, Seq: 0, Len: 8</div> <div>► Data (8 bytes)</div>	<div>~/D1/Proj2 [1] sudo nping --tcp -p 1111 -g 2222 127.0.0.1 --data-string "test tcp" 4110ms + So ...</div> <div>Starting Nping 0.7.80 ( https://mmap.org/nping ) at 2022-04-23 22:46 CEST</div> <div>SENT (0.0710s) TCP 127.0.0.1:2222 &gt; 127.0.0.1:1111 S ttl=64 id=57503 iplen=48 seq=3241075256 win=1480</div>
	<div>TCP</div> <div>timestamp: 2022-04-23T22:46:13.122+02:00</div> <div>src MAC: 00:00:00:00:00:00</div> <div>dst MAC: 00:00:00:00:00:00</div> <div>frame length: 62 bytes</div> <div>src IP: 127.0.0.1</div> <div>dst IP: 127.0.0.1</div> <div>src port: 2222</div> <div>dst port: 1111</div> <div>0x0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00 .....E.</div> <div>0x0010: 00 30 e0 9f 00 00 40 06 9c 26 7f 00 00 01 7f 00 .0....@. &amp;.....</div> <div>0x0020: 00 01 08 ae 04 57 c1 2e e2 38 00 00 00 00 50 02 ....W...&amp;8....P.</div> <div>0x0030: 05 c8 8f e5 00 00 74 65 73 74 20 74 63 70 .....te st tcp</div>

## ICMP

<div>▼ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface lo, id 0</div> <div>► Interface id: 0 (lo)</div> <div>Encapsulation type: Ethernet (1)</div> <div>Arrival Time: Apr 23, 2022 22:48:40.250054971 CEST</div> <div>[Time shift for this packet: 0.000000000 seconds]</div> <div>Epoch Time: 1650746920.250054971 seconds</div> <div>[Time delta from previous captured frame: 0.000000000 seconds]</div> <div>[Time delta from previous displayed frame: 0.000000000 seconds]</div> <div>[Time since reference or first frame: 0.000000000 seconds]</div> <div>Frame Number: 1</div> <div>Frame Length: 42 bytes (336 bits)</div> <div>Capture Length: 42 bytes (336 bits)</div> <div>[Frame is marked: False]</div> <div>[Frame is ignored: False]</div> <div>Protocols in frame: eth:ethertype:ip:icmp</div> <div>[Coloring Rule Name: ICMP]</div> <div>[Coloring Rule String: icmp    icmpv6]</div> <div>► 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)</div> <div>► Destination: 00:00:00:00:00:00 (00:00:00:00:00:00)</div> <div>► Source: 00:00:00:00:00:00 (00:00:00:00:00:00)</div> <div>Type: IPv4 (0x0800)</div> <div>► Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1</div> <div>► Internet Control Message Protocol</div>	<div>~/D1/Proj2 [1] sudo nping --icmp 127.0.0.1 4113ms + So 23. april 2022,</div> <div>Starting Nping 0.7.80 ( https://mmap.org/nping ) at 2022-04-23 22:48 CEST</div> <div>SENT (0.0626s) ICMP [127.0.0.1 &gt; 127.0.0.1 Echo request (type=8/code=0) id=45509 seq=1] IP [ttl=64 id=1112 iplen=28 ]</div>
	<div>ICMP</div> <div>timestamp: 2022-04-23T22:48:40.250+02:00</div> <div>src MAC: 00:00:00:00:00:00</div> <div>dst MAC: 00:00:00:00:00:00</div> <div>frame length: 42 bytes</div> <div>src IP: 127.0.0.1</div> <div>dst IP: 127.0.0.1</div> <div>0x0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00 .....E.</div> <div>0x0010: 00 1c eb a5 00 00 40 01 91 39 7f 00 00 01 7f 00 .....@. .9.....</div> <div>0x0020: 00 01 08 00 86 91 71 6d 00 01 .....qm ..</div>

## ICMPv6

<div>▼ Frame 3: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface enxfe43b420fa8, id 0</div> <div>► Interface id: 0 (enxfe43b420fa8)</div> <div>Encapsulation type: Ethernet (1)</div> <div>Arrival Time: Apr 23, 2022 22:50:07.150380757 CEST</div> <div>[Time shift for this packet: 0.000000000 seconds]</div> <div>Epoch Time: 1650747007.150380757 seconds</div> <div>[Time delta from previous captured frame: 1.106000914 seconds]</div> <div>[Time delta from previous displayed frame: 1.106000914 seconds]</div> <div>[Time since reference or first frame: 2.100084742 seconds]</div> <div>Frame Number: 3</div> <div>Frame Length: 70 bytes (560 bits)</div> <div>Capture Length: 70 bytes (560 bits)</div> <div>[Frame is marked: False]</div> <div>[Frame is ignored: False]</div> <div>Protocols in frame: eth:ethertype:ipv6:icmpv6</div> <div>[Coloring Rule Name: ICMPv6]</div> <div>[Coloring Rule String: icmp    icmpv6]</div> <div>► Ethernet II, Src: Tp-LinkT_d5:1e:25 (08:ff:7b:d5:1e:25), Dst: IPv6mcast_02 (33:33:00:00:00:02)</div> <div>► Destination: IPv6mcast_02 (33:33:00:00:00:02)</div> <div>► Source: Tp-LinkT_d5:1e:25 (08:ff:7b:d5:1e:25)</div> <div>Type: IPv6 (0x8600)</div> <div>► Internet Protocol Version 6, Src: fe80:6aff:7bff:fed5:1e25, Dst: ff02::2</div> <div>► Internet Control Message Protocol v6</div>	<div>ICMPv6</div> <div>timestamp: 2022-04-23T22:50:07.150+02:00</div> <div>src MAC: 68:ff:7b:d5:1e:25</div> <div>dst MAC: 33:33:00:00:00:02</div> <div>frame length: 70 bytes</div> <div>src IP: fe80:0000:0000:0000:6aff:7bff:fed5:1e25</div> <div>dst IP: ff02:0000:0000:0000:0000:0000:0000:0002</div> <div>0x0000: 33 33 00 00 00 02 68 ff 7b d5 1e 25 86 dd 60 00 33....h. {.%..}</div> <div>0x0010: 00 00 00 10 3a ff fe 80 00 00 00 00 00 00 6a ff .....j.</div> <div>0x0020: 7b ff fe d5 1e 25 ff 02 00 00 00 00 00 00 00 00 {....%. ....</div> <div>0x0030: 00 00 00 00 02 85 00 75 3a 00 00 00 00 01 01 ..... u:.....</div> <div>0x0040: 68 ff 7b d5 1e 25 h.{.%</div>
<div>0000 33 33 00 00 00 02 68 ff 7b d5 1e 25 86 dd 60 00 33....h. {.%..</div> <div>0010 00 00 00 10 3a ff fe 80 00 00 00 00 00 00 6a ff .....j.</div> <div>0020 7b ff fe d5 1e 25 ff 02 00 00 00 00 00 00 00 00 {....%. ....</div> <div>0030 00 00 00 00 02 85 00 75 3a 00 00 00 00 01 01 ..... u:.....</div> <div>0040 68 ff 7b d5 1e 25 h.{.%</div>	

## UDP – Ipv6

<div>▼ Frame 81: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface enxfe43b420fa8, id 0</div> <div>► Interface id: 0 (enxfe43b420fa8)</div> <div>Encapsulation type: Ethernet (1)</div> <div>Arrival Time: Apr 23, 2022 22:50:48.626358669 CEST</div> <div>[Time shift for this packet: 0.000000000 seconds]</div> <div>Epoch Time: 1650747048.626358669 seconds</div> <div>[Time delta from previous captured frame: 0.000578994 seconds]</div> <div>[Time delta from previous displayed frame: 0.000578994 seconds]</div> <div>[Time since reference or first frame: 1.745028048 seconds]</div> <div>Frame Number: 81</div> <div>Frame Length: 122 bytes (976 bits)</div> <div>Capture Length: 122 bytes (976 bits)</div> <div>[Frame is marked: False]</div> <div>[Frame is ignored: False]</div> <div>Protocols in frame: eth:ethertype:ipv6:udp:mdns</div> <div>[Coloring Rule Name: UDP]</div> <div>[Coloring Rule String: udp]</div> <div>► Ethernet II, Src: 08:8f:c3:0b:54:3a (08:8f:c3:0b:54:3a), Dst: IPv6mcast_fb (33:33:00:00:00:fb)</div> <div>► Destination: IPv6mcast_fb (33:33:00:00:00:fb)</div> <div>► Source: 08:8f:c3:0b:54:3a (08:8f:c3:0b:54:3a)</div> <div>Type: IPv6 (0x8600)</div> <div>► Internet Protocol Version 6, Src: fe80:c918:5b30:305c:7ef1, Dst: ff02::fb</div> <div>► User Datagram Protocol, Src Port: 5353, Dst Port: 5353</div> <div>Source Port: 5353</div> <div>Destination Port: 5353</div> <div>Length: 60</div> <div>Checksum: 0xae fb (unverified)</div> <div>[Checksum Status: Unverified]</div> <div>[Stream Index: 41]</div> <div>► [Timestamps]</div>	<div>UDP – IPv6</div> <div>timestamp: 2022-04-23T22:50:48.626+02:00</div> <div>src MAC: 08:8f:c3:0b:54:3a</div> <div>dst MAC: 33:33:00:00:00:fb</div> <div>frame length: 122 bytes</div> <div>src IP: fe80:0000:0000:0000:c918:5b30:305c:7ef1</div> <div>dst IP: ff02:0000:0000:0000:0000:0000:0000:00fb</div> <div>src port: 5353</div> <div>dst port: 5353</div> <div>0x0000: 33 33 00 00 00 fb 08 8f c3 0b 54 3a 86 dd 60 0a 33.....T:..</div> <div>0x0010: 69 ab 00 44 11 01 fe 80 00 00 00 00 00 c9 18 i..D.....</div> <div>0x0020: 5b 30 5c 7e f1 ff 02 00 00 00 00 00 00 00 00 [00~.....</div> <div>0x0030: 00 00 00 00 00 fb 14 e9 14 e9 00 44 ae fb 00 00 .....D.....</div> <div>0x0040: 00 00 00 01 00 00 00 00 00 24 33 37 37 62 30 .....\$377b0</div> <div>0x0050: 65 31 38 2d 30 31 66 37 2d 34 37 32 61 2d 62 62 e18-01f7 -472a-bb</div> <div>0x0060: 32 61 2d 31 37 63 32 64 38 33 35 62 65 62 63 05 2a-17c2d 835becb.</div> <div>0x0070: 6c 6f 63 61 6c 00 00 01 00 01 local....</div>
<div>0000 33 33 00 00 00 fb 08 8f c3 0b 54 3a 86 dd 60 0a 33.....T:..</div> <div>0010 69 ab 00 44 11 01 fe 80 00 00 00 00 00 c9 18 i..D.....</div> <div>0020 5b 30 5c 7e f1 ff 02 00 00 00 00 00 00 00 00 [00~.....</div> <div>0030 00 00 00 00 00 fb 14 e9 14 e9 00 44 ae fb 00 00 .....D.....</div> <div>0040 00 00 00 01 00 00 00 00 00 24 33 37 37 62 30 .....\$377b0</div> <div>0050 65 31 38 2d 30 31 66 37 2d 34 37 32 61 2d 62 62 e18-01f7 -472a-bb</div> <div>0060 32 61 2d 31 37 63 32 64 38 33 35 62 65 62 63 05 2a-17c2d 835becb.</div> <div>0070 6c 6f 63 61 6c 00 00 01 00 01 local....</div>	



# Bibliografia

- [1] „Wikipedia,“ 8 8 2021. [Online]. Available: <https://cs.wikipedia.org/wiki/Paket>. [Cit. 23 4 2022].
- [2] S. Moon, „binarytides,“ 31 7 2020. [Online]. Available: <https://www.binarytides.com/packet-sniffer-code-c-libpcap-linux-sockets/>. [Cit. 23 4 2022].
- [3] „programcreek,“ [Online]. Available: [https://www.programcreek.com/cpp/?code=mq1n%2FNoMercy%2FNoMercy-master%2FSource%2FClient%2FNM\\_Engine%2FINetworkScanner.cpp](https://www.programcreek.com/cpp/?code=mq1n%2FNoMercy%2FNoMercy-master%2FSource%2FClient%2FNM_Engine%2FINetworkScanner.cpp). [Cit. 23 4 2022].
- [4] paxdiablo, „stackoverflow,“ 15 10 2011. [Online]. Available: <https://stackoverflow.com/questions/7775991/how-to-get-hexdump-of-a-structure-data>. [Cit. 23 4 2022].
- [5] P. Buddy a nategoose, „errorsfixing,“ 28 10 2021. [Online]. Available: <https://errorsfixing.com/expand-an-ipv6-address-so-i-can-print-it-to-stdout/>. [Cit. 23 4 2022].
- [6] alk, „stackoverflow,“ 15 9 2017. [Online]. Available: <https://stackoverflow.com/questions/46757406/how-to-print-timestamp-of-a-packet-read-from-pcap-file>. [Cit. 23 4 2022].
- [7] user257111, „stackoverflow,“ 24 12 2010. [Online]. Available: <https://stackoverflow.com/questions/4526576/how-do-i-capture-mac-address-of-access-points-and-hosts-connected-to-it>. [Cit. 23 4 2022].
- [8] T. T. Group, „tcpdump,“ 5 7 2005. [Online]. Available: <https://www.tcpdump.org/other/sniffex.c>. [Cit. 23 4 2022].