Projektová dokumentace

IPK 2. projekt

Varianta ZETA: Sniffer paketů

Karel Norek, xnorek01

24. dubna 2021

Obsah

1 Úvod

Cílem projektu bylo vytvořit sniffer packetů, který bude podporovat jak IPv4 tak IPv6 společně s protokoly TCP, UDP, ARP, ICMP. Projekt je implementován v jayzce C++.

2 Packety

2.1 TCP - Transmission Control Protocol

- spojení point-to-point
- spolehlivý přenos dat

2.2 UDP - User Datagram Protocol

- Jednoduchý protokol transportní vrstvy
- Datagramová služba bez záruky doručení, pořadí paketů
- Nespojovaná služba (connection-less)

2.3 ARP - Address Resolution Protocol

- Slouží k získání linkové adresy síťového rozhraní protistrany ve stejné podsíti

2.4 ICMP - Internet Control Message Protocol

- Používán hosty, směrovači pro oznámení chyb, diagnostiku sítě
- Obecně 2 typy zpráv - oznámení chyby, dotaz na službu

3 Implementace

3.1 Parsování argumentů

Program začne zpracováním argumentů. Pro tenhle účel je vytvrořena třída *ArgumentParser*. Ta používá funkci *getopt* z knihovny *getopt.h*.
Mezi možné argumenty patří:

- -i rozhraní | –interface rozhraní : Pokud je zadán parametr rozhraní, bude program uvažovat rozhraní kde bude chytat packety. Pokud není zadán, nebo je program spuštěn bez argumentů, vypíše se seznam aktivních rozhraní
- -p port : Filtruje packety, které mají daný port buď v source nebo v destination části.
- - -tcp nebo -t : Filtruje pouze TCP packety
- - -udp nebo -u : Filtruje pouze UDP packety
- - -arp : Filtruje pouze ARP packety
- - -icmp : Filtruje pouze ICMP packety
- -n num : Počet chytaných packetů

Argumenty –tcp, –udp, –arp a –icmp se mohou kombinovat. To znamená že se bude filtrovat jejich sjednocení.

3.2 Výpis rozhraní

Pro výpis rozhraní se používá funkce *pcap_findalldevs* z knihovny *pcap/pcap.h*. Pokud nenastane chyba, vypíší se všechna aktivní rozhraní.

3.3 Chytání packetů

Pro chytání packtů je využita knihovna *pcap/pcap.h*.
Nejdříve se sestaví filtr podle zadaných argumentů ve funkci *buildFilter*. Zde se postupně kontrolují argumenty a tvoří se řetězec, který se později převede na filtr.
Dále se použije upravený kód z [1]. Změnou je použití *pcap_loop* místo *pcap_next*. Pokud nenaskytne chyba během otevíraní rozhraní, případně nastavovaní filtru, volá se funkce *pcal_loop*, která když zachytí packet volá callback funkci *gotPacket*.

3.4 Zpracování packetu

Jakmile se zachytí požadovaný packet, volá se funkce *gotPacket*. V funkci se používá knihovna *if_ether.h*, která obsahuje strukturu *ether_header*. Do této struktury přetypujeme packet, který se nachází v argumentu funkce jako *const u_char *packet*. Díky tomuto přetypování můžeme zjistit typ packetu. Máme 3 možnosti: Typ je *ETHERTYPE_IP*, neboli IPv4 packet. To může znamenat buď TCP, UDP nebo ICMP packety. Dalším typem je *ETHERTYPE_ARP*, který nám řekne že packet je typu ARP. Posledním typem je *ETHERTYPE_IPV6*. To znamená že packet je IPvž packet a tím pádem packet bude buď TCP, UDP nebo ICMPv6.
Typ zjistíme pomocí funkce ntohs, do ktere jako argument vložíme část struktury s názvem ether_type. Jakmile víme typ, můžeme začít řešit daný typ packetu.

3.4.1 IPv4 packety

Pro typy TCP, UDP a ICMP použijeme další knihovnu se strukturou. Struktura se nachází v knihovně *netinet/ip.h*. Znovu přetypujeme packet na sturkturu, ale tentokrát přičteme dělku ethernetové hlavičky (14). Poté se zavolá funkce *printHeader*.
Funkce *printHeader* nejdříve zjistí source IP adresu a destination IP adresu ze struktury ip poslanou skrze argument funkce. Na převod se používá funkce *inet_ntoa* z knihovny *arpa/inet.h*. Dále se postup liší podle typu packetů.
Pro TCP packet se přetypuje packet na strukturu *tcphdr* z knihovny *netinet/tcp.h*, ale přičte se velikost IPv4 hlavičky a ethernet hlavičky. Díky této struktuře můžeme dostat port pro source a destination pomocí již zmíněné funkce *ntohs*. Dále se z hlavičky dostane čas pomocí funkce *getTime*. Tato funkce vrací globalní čas (časové pásmo +00:00). Jakmile máme všechny potřebné informace, hlavička packetu se vypíše.
Pro UDP packet je stejný postup jako u TCP, s tím rozdílem, že se používá knihovna *netinet/udp.h*.
Pro ICMP packet se pouze získá čas pomocí funkce *getTime* a vypíše se hlavička, protože ICMP packet nemá žádný port.
Po vytisknutí hlavičky následuje tisk samotného packetu. To probíha ve funkci *printPacket*. Funkce je převzatá z [1], ale přepsaná do C++ a mírně upravená.

3.4.2 ARP packety

ARP packety se přetypují do struktury *ether_arp* z knihovny *if_ether.h*. Poté se volá funkci *printHeaderARP*.
Tato funkce vytiskne MAC adresy source a destination místo IP adres.
Po vytisknutí hlavičky se volá funkce *printPacket* pro vytisknutí packetu.

3.4.3 IPv6 packety

Pro IPv6 packety se nejdřív přetypuje packet na strukturu *ip6_hdr* z knihovny *netinet/ip6.h*. Díky které zjistíme typ packetu.
Pro vypsání IPvž hlavičky se volá funkce *printHeaderIPV6*. Ta nejdříve zjistí IPvž adresu pomocí funkce *inet_ntop* z knihovny *arpa/inet.h*.
Dále se postupuje jako u IPv4 s jediným rozdílem: U přetypování do TCP a UDP struktur se jako velikost IPv6 hlavičky používá fixní velikost 40.
Po vypsání hlavičky se vypíše packet pomocí *printPacket*.

3.5 Ukončení

Po zpracování všech packetů se uvolní zdroje pomocí _freecode a *pcap_close*. Poté program úspěšně končí.

3.6 Signál CTRL+C

Program korektně chytá signál CTRL + C a ukončí program s návratovým kódem 1.

4 Testování

Program se testoval metodou porovnání výstupu s referenčním zdrojem. Pro generovaní různých packetů je využit příkaz *nping* (*ping -6* nebo *netcat* pro IPv6).
Jako referenční zdroj byl zvolen program wireshark.

4.1 ARP

```
student@student-vm:~$ sudo nping --arp 1.1.1.1

Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2021-04-24 20:54 CEST
SENT (0.0037s) ARP who has 1.1.1.1? Tell 192.168.45.128
SENT (1.0040s) ARP who has 1.1.1.1? Tell 192.168.45.128
SENT (2.0066s) ARP who has 1.1.1.1? Tell 192.168.45.128
```

Obrázek 1: Generovaní ARP packetu

```
student@student-vm:~/IPK$ sudo ./ipk-sniffer -i ens33 --arp
2021-4-24T18:54:29.911057z 00:0c:29:42:fb:fc > ff:ff:ff:ff:ff:ff, length 42 bytes

0x0000: ff ff ff ff ff ff 00 0c 29 42 fb fc 08 06 00 01  ........)B......
0x0010: 08 00 06 04 00 01 00 0c 29 42 fb fc c0 a8 2d 80  ........)B....-.
0x0020: 00 00 00 00 00 00 01 01 01 01                    ..........
```

Obrázek 2: IPK-sniffer výstup

```
▸ Frame 3: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface ens33, id 0
▾ Ethernet II, Src: VMware_42:fb:fc (00:0c:29:42:fb:fc), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▸ Source: VMware_42:fb:fc (00:0c:29:42:fb:fc)
    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: VMware_42:fb:fc (00:0c:29:42:fb:fc)
    Sender IP address: 192.168.45.128
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 1.1.1.1

0000  ff ff ff ff ff ff 00 0c  29 42 fb fc 08 06 00 01   ........ )B......
0010  08 00 06 04 00 01 00 0c  29 42 fb fc c0 a8 2d 80   ........ )B....-.
0020  00 00 00 00 00 00 01 01  01 01                    ........ ..
```

Obrázek 3: Wireshark výstup

4.2 IPv4 TCP

```
student@student-vm:~$ sudo nping --tcp 1.1.1.1

Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2021-04-24 20:58 CEST
SENT (0.0029s) TCP 192.168.45.128:32850 > 1.1.1.1:80 S ttl=64 id=64737 iplen=40
 seq=2703813459 win=1480
```

Obrázek 4: Generovaní TCP packetu

```
student@student-vm:~/IPK$ sudo ./ipk-sniffer -i ens33 --tcp
2021-4-24T18:58:34.122778z 192.168.45.128 : 32850 > 1.1.1.1 : 80, length 54 bytes

0x0000: 00 50 56 ec a2 04 00 0c 29 42 fb fc 08 00 45 00  .PV.....)B....E.
0x0010: 00 28 fc e1 00 00 40 06 8d c4 c0 a8 2d 80 01 01  .(....@.....-...
0x0020: 01 01 80 52 00 50 a1 28 eb 53 00 00 00 00 50 02  ...R.P.(.S....P.
0x0030: 05 c8 ac d1 00 00                                ......
```

Obrázek 5: IPK-sniffer výstup

```
▸ Frame 1: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface ens33, id 0
▾ Ethernet II, Src: VMware_42:fb:fc (00:0c:29:42:fb:fc), Dst: VMware_ec:a2:04 (00:50:56:ec:a2:04)
    ▸ Destination: VMware_ec:a2:04 (00:50:56:ec:a2:04)
    ▸ Source: VMware_42:fb:fc (00:0c:29:42:fb:fc)
      Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 192.168.45.128, Dst: 1.1.1.1
▸ Transmission Control Protocol, Src Port: 32850, Dst Port: 80, Seq: 0, Len: 0
```

```
0000  00 50 56 ec a2 04 00 0c  29 42 fb fc 08 00 45 00   ·PV·····  )B····E·
0010  00 28 fc e1 00 00 40 06  8d c4 c0 a8 2d 80 01 01   ·(····@·  ····-···
0020  01 01 80 52 00 50 a1 28  eb 53 00 00 00 00 50 02   ···R·P·(  ·S····P·
0030  05 c8 ac d1 00 00                                 ·····
```

Obrázek 6: Wireshark výstup

4.3 IPv4 UDP

```
student@student-vm:~$ sudo nping --udp 1.1.1.1

Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2021-04-24 21:01 CEST
SENT (0.0022s) UDP 192.168.45.128:53 > 1.1.1.1:40125 ttl=64 id=35587 iplen=28
```

Obrázek 7: Generovaní UDP packetu

```
student@student-vm:~/IPK$ sudo ./ipk-sniffer -i ens33 --udp
2021-4-24T19:1:21.9505z 192.168.45.128 : 53 > 1.1.1.1 : 40125, length 42 bytes

0x0000: 00 50 56 ec a2 04 00 0c 29 42 fb fc 08 00 45 00  .PV.....)B....E.
0x0010: 00 1c 8b 03 00 00 40 11 ff a3 c0 a8 2d 80 01 01  ......@.....-...
0x0020: 01 01 00 35 9c bd 00 08 72 c1                    ...5....r.
```

Obrázek 8: IPK-sniffer výstup

```
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface ens33, id 0
▾ Ethernet II, Src: VMware_42:fb:fc (00:0c:29:42:fb:fc), Dst: VMware_ec:a2:04 (00:50:56:ec:a2:04)
  ▸ Destination: VMware_ec:a2:04 (00:50:56:ec:a2:04)
  ▸ Source: VMware_42:fb:fc (00:0c:29:42:fb:fc)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 192.168.45.128, Dst: 1.1.1.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 28
    Identification: 0x8b03 (35587)
  ▸ Flags: 0x0000
    Fragment offset: 0
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0xffa3 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.45.128
    Destination: 1.1.1.1

0000  00 50 56 ec a2 04 00 0c  29 42 fb fc 08 00 45 00   ·PV·····  )B····E·
0010  00 1c 8b 03 00 00 40 11  ff a3 c0 a8 2d 80 01 01   ·····@·  ····-···
0020  01 01 00 35 9c bd 00 08  72 c1                    ···5····  r·
```

Obrázek 9: Wireshark výstup

4.4 ICMP

```
student@student-vm:~$ sudo nping --icmp 1.1.1.1

Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2021-04-24 21:03 CEST
SENT (0.0032s) ICMP [192.168.45.128 > 1.1.1.1 Echo request (type=8/code=0) id=16
258 seq=1] IP [ttl=64 id=64526 iplen=28 ]
```

Obrázek 10: Generovaní ICMP packetu

```
student@student-vm:~/IPK$ sudo ./ipk-sniffer -i ens33 --icmp
2021-4-24T19:3:18.736121z 192.168.45.128 > 1.1.1.1, length 42 bytes

0x0000: 00 50 56 ec a2 04 00 0c 29 42 fb fc 08 00 45 00  .PV.....)B....E.
0x0010: 00 1c fc 0e 00 00 40 01 8e a8 c0 a8 2d 80 01 01  ......@.....-...
0x0020: 01 01 08 00 b8 7c 3f 82 00 01                    .....|?...
```

Obrázek 11: IPK-sniffer výstup

```
▸ Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface ens33, id 0
▾ Ethernet II, Src: VMware_42:fb:fc (00:0c:29:42:fb:fc), Dst: VMware_ec:a2:04 (00:50:56:ec:a2:04)
  ▸ Destination: VMware_ec:a2:04 (00:50:56:ec:a2:04)
  ▸ Source: VMware_42:fb:fc (00:0c:29:42:fb:fc)
    Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 192.168.45.128, Dst: 1.1.1.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 28
    Identification: 0xfc0e (64526)
  ▸ Flags: 0x0000
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x8ea8 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.45.128
    Destination: 1.1.1.1
0000  00 50 56 ec a2 04 00 0c  29 42 fb fc 08 00 45 00   ·PV····· )B····E·
0010  00 1c fc 0e 00 00 40 01  8e a8 c0 a8 2d 80 01 01   ·····@· ····-···
0020  01 01 08 00 b8 7c 3f 82  00 01                     ·····|?· ··
```

Obrázek 12: Wireshark výstup

4.5 ICMPV6

```
student@student-vm:~$ sudo ping -6 ::1
PING ::1(::1) 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.028 ms
```

Obrázek 13: Generovaní ICMPv6 packetu

```
student@student-vm:~/IPK$ sudo ./ipk-sniffer -i lo --icmp
2021-4-24T19:30:15.373668z ::1 > ::1, length 118 bytes

0x0000: 00 00 00 00 00 00 00 00 00 00 00 00 86 dd 60 0a  ..............`.
0x0010: e7 5b 00 40 3a 40 00 00 00 00 00 00 00 00 00 00  .[.@:@..........
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 df 22 00 07 00 01 c7 71  .........".....q
0x0040: 84 60 00 00 00 00 90 b3 05 00 00 00 00 00 10 11  .`..............
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
```

Obrázek 14: IPK-sniffer výstup

```
▸ Frame 1: 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)
  ▸ 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: IPv6 (0x86dd)
▾ Internet Protocol Version 6, Src: ::1, Dst: ::1
    0110 .... = Version: 6
  ▸ .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 1010 1110 0111 0101 1011 = Flow Label: 0xae75b
    Payload Length: 64
    Next Header: ICMPv6 (58)
    Hop Limit: 64
    Source: ::1
    Destination: ::1
▸ Internet Control Message Protocol v6
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 86 dd 60 0a   ........ ......`.
0010  e7 5b 00 40 3a 40 00 00  00 00 00 00 00 00 00 00   .[.@:@.. ........
0020  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 01 80 00  df 22 00 07 00 01 c7 71   ........ ."......q
0040  84 60 00 00 00 00 90 b3  05 00 00 00 00 00 10 11   .`...... ........
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
```

Obrázek 15: Wireshark výstup

4.6 IPv6 TCP

```
student@student-vm:~$ sudo netcat ::1 5000
```

Obrázek 16: Generovaní TCPv6 packetu

```
student@student-vm:~/IPK$ sudo ./ipk-sniffer -i lo --tcp
2021-4-24T19:32:32.114680z ::1 : 43626 > ::1 : 5000, length 94 bytes

0x0000: 00 00 00 00 00 00 00 00 00 00 00 00 86 dd 60 0e  ..............`.
0x0010: a1 41 00 28 06 40 00 00 00 00 00 00 00 00 00 00  .A.(.@..........
0x0020: 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00  ................
0x0030: 00 00 00 00 00 01 aa 6a 13 88 03 54 89 70 00 00  .......j...T.p..
0x0040: 00 00 a0 02 ff c4 00 30 00 00 02 04 ff c4 04 02  .......0........
0x0050: 08 0a be 87 af fb 00 00 00 00 01 03 03 07        ..............
```

Obrázek 17: IPK-sniffer výstup

```
▸ Frame 1: 94 bytes on wire (752 bits), 94 bytes captured (752 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)
    ▸ 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: IPv6 (0x86dd)
▾ Internet Protocol Version 6, Src: ::1, Dst: ::1
      0110 .... = Version: 6
    ▸ .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
      .... .... .... 1110 1010 0001 0100 0001 = Flow Label: 0xea141
      Payload Length: 40
      Next Header: TCP (6)
      Hop Limit: 64
      Source: ::1
      Destination: ::1
▸ Transmission Control Protocol, Src Port: 43626, Dst Port: 5000, Seq: 0, Len: 0
```

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 86 dd 60 0e   ........ ......`.
0010  a1 41 00 28 06 40 00 00  00 00 00 00 00 00 00 00   .A.(.@.. ........
0020  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 01 aa 6a  13 88 03 54 89 70 00 00   .......j ...T.p..
0040  00 00 a0 02 ff c4 00 30  00 00 02 04 ff c4 04 02   .......0 ........
0050  08 0a be 87 af fb 00 00  00 00 01 03 03 07         ........ ......
```

Obrázek 18: Wireshark výstup

4.7 TCP packet s portem

```
student@student-vm:~$ sudo nping --tcp -p 23 1.1.1.1

Starting Nping 0.7.80 ( https://nmap.org/nping ) at 2021-04-24 21:52 CEST
SENT (0.0022s) TCP 192.168.45.128:41943 > 1.1.1.1:23 S ttl=64 id=33016 iplen=40
 seq=2451374927 win=1480
```

Obrázek 19: Generovaní TCP packetu s portem 23

```
student@student-vm:~/IPK$ sudo ./ipk-sniffer -i ens33 -p 23
2021-4-24T19:52:7.72502z 1.1.1.1 : 23 > 192.168.45.128 : 29153, length 60 bytes

0x0000: 00 0c 29 42 fb fc 00 50 56 ec a2 04 08 00 45 00  ..)B...PV.....E.
0x0010: 00 28 a7 db 00 00 80 06 a2 ca 01 01 01 01 c0 a8  .(..............
0x0020: 2d 80 00 17 71 e1 0b d8 50 2b 2c 99 de 65 50 14  -...q...P+,..eP.
0x0030: fa f0 eb ba 00 00 00 00 00 00 00 00              ............
```

Obrázek 20: IPK-sniffer výstup

```
▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface ens33, id 0
▾ Ethernet II, Src: VMware_ec:a2:04 (00:50:56:ec:a2:04), Dst: VMware_42:fb:fc (00:0c:29:42:fb:fc)
    ▸ Destination: VMware_42:fb:fc (00:0c:29:42:fb:fc)
    ▸ Source: VMware_ec:a2:04 (00:50:56:ec:a2:04)
      Type: IPv4 (0x0800)
      Padding: 000000000000
▾ Internet Protocol Version 4, Src: 1.1.1.1, Dst: 192.168.45.128
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 40
      Identification: 0xa7db (42971)
    ▸ Flags: 0x0000
      Fragment offset: 0
      Time to live: 128
      Protocol: TCP (6)
      Header checksum: 0xa2ca [validation disabled]
      [Header checksum status: Unverified]
      Source: 1.1.1.1
0000  00 0c 29 42 fb fc 00 50  56 ec a2 04 08 00 45 00   ··)B···P V·····E·
0010  00 28 a7 db 00 00 80 06  a2 ca 01 01 01 01 c0 a8   ·(···· ····· ····
0020  2d 80 00 17 71 e1 0b d8  50 2b 2c 99 de 65 50 14   -···q··· P+,··eP·
0030  fa f0 eb ba 00 00 00 00  00 00 00 00               ···· ···· ····
```

Obrázek 21: Wireshark výstup

5 Závěr

Program odchytáva určené packety jak má.

6 Použité zdroje

[1] CARSTENS, T.: Programming with pcap. [online], 2012, [viděno 20. 4. 2021].
URL https://www.tcpdump.org/pcap.html

[2] TORVALDS, L.: [online], 2013, [viděno 20. 4. 2021].
URL https://sites.uclouvain.be/SystInfo/usr/include/linux/

[3] VESELÝ, V.: Síťová vrstva. [online], 2020, [viděno 10. 4. 2021].
URL https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIPK-IT%2Flectures%2FIPK2020-04-IPv4.pdf&cid=14005

[4] VESELÝ, V.: Transportní vrstva. [online], 2020, [viděno 10. 4. 2021].
URL https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIPK-IT%2Flectures%2FIPK2020-03-TRANSPORT.pdf&cid=14005