

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové komunikace a sítě - 2.projekt
Varianta ZETA: Sniffer paketů

Obsah

1	Teoretická část	2
1.1	IPv4	2
1.2	IPv6	2
1.3	ARP	2
1.4	TCP	2
1.5	UDP	2
1.6	ICMP	2
2	Implementace	3
2.1	Převzaté části kódu	3
2.2	Kroky implementace	4
2.2.1	Začátek a zpracování všech parametrů	4
2.2.2	Průběh programu	4
2.2.3	Vypisování výstupu	4
2.2.4	Zpracovávání bitů pro výpis dat z pakety	4
3	Testování	5

1 Teoretická část

1.1 IPv4

IPv4 je protokol používaný pro přepojování paket v síti. Dnes je ze všech datkových protokolů nejpoužívanější. Používá 32-bitovou adresu. jeho vlastností je, že neodpovídá za přepravovaná data. Může se stát, že data přijdou v jiném pořadí, než byly posílány, nebo můžou být duplikovány, či nepřijdou vůbec. [11]

1.2 IPv6

Je to nejnovější verze a nahrazuje IPv4 díky svojí 128-adresy, díky tomu má každé zařízení připojené na internet svoji unikátní adresu. Narozdíl od MAC adres, se IPv6 adresy se časem mění. Jejich zápis je 8krát 4 hexadecimální čísla [9]

1.3 ARP

ARP je komunikační protokol používaný k přistupování do síťové vrstvy k např MAC adresám, nebo IPv4. Po odesílání zpráv, které jsou zapouzdřeny přímo protokolem síťové vrstvy, čeká na odpověď. Tato komunikace probíhá mezi hranicemi jediné sítě a nikdy neprobíhá přes mezisíťové uzly. [12]

1.4 TCP

TCP je nejpoužívanější protokol na transportní vrstvě mezi TCP/IP. Díky tomuto protokolu je možné navázat spojení mezi dvěma zařízeními ve stejné síti a tímto způsobem mezi sebou přenášet data. Odpovídá za doručení a za doručení ve správném pořadí. Používá se tam, kde je nutné potvrzení přijetí dat, např. emaily, internetové konverzace, stahování souborů. [8]

1.5 UDP

Hlavním rozdílem UDP od TCP je, že UDP neručí za odesílané data, tzv. nečeká na zpětnou vazbu, jestli byl přenos úspěšný. používá se tam, kde není čas čekat na zpětnou vazbu o doručení dat, takže u audio/video hovorů, hraní her. [10]

1.6 ICMP

Patří do skupiny protokolů TCP/IP a byl vytvořen pro posílání chybových zpráv, hlášení a informací, jestli byla daná operace úspěšná, či ne. Od TCP a UDP se liší tak, že je pouze vytvořen jako následek nějaké vzniklé události. Lze na něj použít funkci ping, která slouží pro kontrolu, zda je zařízení v síti a jak dlouho trvá přenos dat paketům. [13]

2 Implementace

2.1 Převzaté části kódu

Inspiraci na strukturu programu je převzatá z [2], stejně tak jako funkce: `print_hex_ascii_line`, `print_payload` a `got_packet`. Tyto funkce jsou dále upravované, podle potřeb projektu.

Příložená licence pro použití částí kódu z tohoto zdroje:

`sniffex.c`

Sniffer example of TCP/IP packet capture using libpcap.

Version 0.1.1 (2005-07-05)

Copyright (c) 2005 The Tcpdump Group

This software is intended to be used as a practical example and demonstration of the libpcap library; available at:

<http://www.tcpdump.org/>

Použití v `main` funkce (`pcap_loop`, `pcap_setfilter`, `pcap_compile`, `pcap_lookupnet`, `pcap_open_live`, `pcap_findalldevs`) funkci jsou převzaté z [1] a dále upravované podle specifikace projektu.

Formát filtru protokolů ve vlastní funkci `check_arg` vytvořen podle [5].

Úprava na výpis správného formátu času ve funkci `print_time` je inspirována [3].

Použití knihoven pro UDP a IPv6 protocol je použito z [6] a [4].

2.2 Kroky implementace

2.2.1 Začátek a zpracování všech parametrů

Předání potřebných hodnot do pomocných proměnných a deklarace ostatních proměnných.

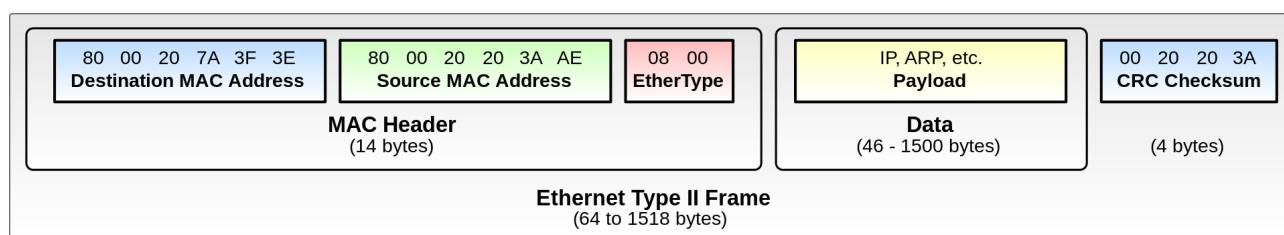
Funkce `check_interface` spracuje rozhraní, `check_arg` zpracuje protokoly do filtru protokolů a ještě změní hodnotu počtu `n` vypisovaných paket.

2.2.2 Průběh programu

Pokračování s funkcí `pcap_findalldevs`, která zjistí všechny možné zařízení, které jsou možné pro běh programu. `pcap_lookupnet` zjistí číslo IPv4 a maskuje čísla spojené se zařízením síťového zařízení. Dále `pcap_open_live` umožní začátek zachytávání paketů v síti. Funkce `pcap_compile` konvertuje string do podoby nutné pro filtraci protokolů. Jako předposlední funkci použité v `main()` je `pcap_setfilter`, která zpracuje filtr. Pokud je zpracování úspěšné, tak vrátí 0. Nakonec se zavolá `pcap_loop`, která vytvoří loop s parametrem `n` a všechno další zpracovávání protokolů a vypisování probíhá díky ní ve funkci `got_packet`.

2.2.3 Vypisování výstupu

Po vypsání času příchodu, díky struktuře `ethernet`, kam uložíme bity dané pakety si vytáhneme potřebné informace díky poloze, kde se jednotlivé informace nachází.



[7]

Z obrázku lze vidět, že prvních 6 bitů je pro destination MAC adresu a dalších 6 pro source MAC adresu. Pomocí 12. a 13. bitu (indexování od 0) lze zjistit, jestli protokol pracuje na IPv4, IPv6 nebo ARP protokolu. Protokol ARP nemá svůj port, proto podmínka v další části kódu, aby se v tomto případě nic nevypsalo. ICMP má speciální vypisování jiné pro IPv4 a IPv6.

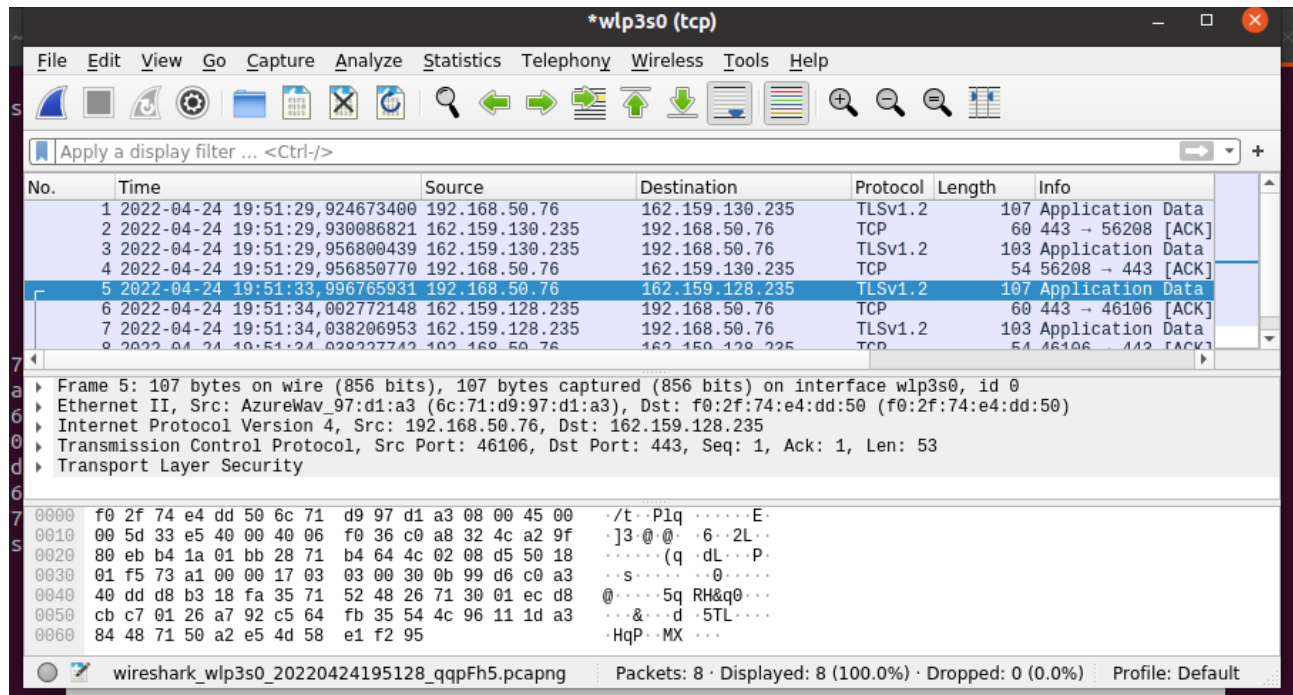
2.2.4 Zpracovávání bitů pro výpis dat z pakety

Ve funkci `print_payload` se bity zpracovávají po 16 bitech, kdy se nejdříve zjistí, či počet bitů je větších než 16, pokud ano tak se offset posune na 16 a těch prvních 16 bitů se zpracuje. Takto to jde postupně až do té doby, než zůstane méně než 16 bitů a už se dál nepokračuje.

O samotný výpis se stará `print_hex_ascii_line`.

3 Testování

Pomocí aplikace Wireshark, jsem si našla danou packetu, kterou jsem zachytila pomocí `ipk-sniffer.cpp`. Pro kontrolu vidím, že se jejich obsah shoduje.



The screenshot shows the Wireshark interface with the following details:

- Packet List:** Shows several packets. Packet 5 is selected, which is a TLSv1.2 packet of length 107 bytes.
- Packet Details:** Shows the structure of the selected packet: Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Transport Layer Security.
- Packet Bytes:** Displays the raw packet data in hexadecimal and ASCII format.

```
simir@simir-X550CC:~/Documents/IPK$ sudo ./sniffer --tcp -i wlp3s0
timestamp: 2022-04-24T19:51:33.996
frame length: 107 bytes
src MAC: 6c:71:d9:97:d1:a3
dst MAC: f0:2f:74:e4:dd:50
src IP: 192.168.50.76
dst IP: 162.159.128.235
src port: 46106
dst port: 443

0x0000  f0 2f 74 e4 dd 50 6c 71 d9 97 d1 a3 08 00 45 00  ./t..Plq.....E.
0x0010  00 5d 33 e5 40 00 40 06 f0 36 c0 a8 32 4c a2 9f  .]3.@.@..6..2L..
0x0020  80 eb b4 1a 01 bb 28 71 b4 64 4c 02 08 d5 50 18  ....(q.dL...P.
0x0030  01 f5 73 a1 00 00 17 03 03 00 30 0b 99 d6 c0 a3  ..s.....0.....
0x0040  40 dd d8 b3 18 fa 35 71 52 48 26 71 30 01 ec d8  @.....5qRH&q0...
0x0050  cb c7 01 26 a7 92 c5 64 fb 35 54 4c 96 11 1d a3  ...&...d.5TL....
0x0060  84 48 71 50 a2 e5 4d 58 e1 f2 95  .HqP..MX...
```

simir@simir-X550CC:~/Documents/IPK\$

Filtrování TCP pakety

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-04-24 20:09:00,712485359	188.122.76.11	192.168.50.76	RTCP	94	Receiver Report
2	2022-04-24 20:09:00,801587525	192.168.50.76	188.122.76.11	UDP	50	41753 → 50003 Len=8
3	2022-04-24 20:09:00,824353639	188.122.76.11	192.168.50.76	UDP	60	50003 → 41753 Len=8
4	2022-04-24 20:09:01,712601409	188.122.76.11	192.168.50.76	RTCP	94	Receiver Report
5	2022-04-24 20:09:02,263474996	192.168.50.76	188.122.76.11	RTCP	70	Receiver Report
6	2022-04-24 20:09:02,712681232	188.122.76.11	192.168.50.76	RTCP	94	Receiver Report
7	2022-04-24 20:09:02,734916797	192.168.50.76	188.122.76.11	RTCP	70	Receiver Report
8	2022-04-24 20:09:03,262198379	192.168.50.76	188.122.76.11	RTCP	70	Receiver Report
9	2022-04-24 20:09:03,712729442	188.122.76.11	192.168.50.76	RTCP	94	Receiver Report

Frame 5: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface wlp3s0, id 0
 Ethernet II, Src: AzureWav_97:d1:a3 (6c:71:d9:97:d1:a3), Dst: f0:2f:74:e4:dd:50 (f0:2f:74:e4:dd:50)
 Internet Protocol Version 4, Src: 192.168.50.76, Dst: 188.122.76.11
 User Datagram Protocol, Src Port: 41753, Dst Port: 50003
 Real-time Transport Control Protocol (Receiver Report)

```

0000  f0 2f 74 e4 dd 50 6c 71 d9 97 d1 a3 08 00 45 00  ./t..Plq.....E.
0010  00 38 0a 03 40 00 40 11 35 38 c0 a8 32 4c bc 7a  .8..@.@.58..2L.z
0020  4c 0b a3 19 c3 53 00 24 d9 ad 80 c9 00 01 00 09  L....S.$.....
0030  e5 28 79 d8 67 26 94 15 fd 14 64 65 be 49 6c ea  .(y.g&....de.Il.
0040  f8 c4 63 0c 00 80  ..c...
  
```

wireshark_wlp3s0_20220424200900_fej8m3.pcapng Packets: 9 · Displayed: 9 (100.0%) · Dropped: 0 (0.0%) Profile: Default

```

simir@simir-X550CC:~/Documents/IPK$ sudo ./sniffer --udp -i wlp3s0
timestamp: 2022-04-24T20:09:02.263
frame length: 70 bytes
src MAC: 6c:71:d9:97:d1:a3
dst MAC: f0:2f:74:e4:dd:50
src IP: 192.168.50.76
dst IP: 188.122.76.11
src port: 41753
dst port: 50003

0x0000  f0 2f 74 e4 dd 50 6c 71 d9 97 d1 a3 08 00 45 00  ./t..Plq.....E.
0x0010  00 38 0a 03 40 00 40 11 35 38 c0 a8 32 4c bc 7a  .8..@.@.58..2L.z
0x0020  4c 0b a3 19 c3 53 00 24 d9 ad 80 c9 00 01 00 09  L....S.$.....
0x0030  e5 28 79 d8 67 26 94 15 fd 14 64 65 be 49 6c ea  .(y.g&....de.Il.
0x0040  f8 c4 63 0c 00 80  ..c...
simir@simir-X550CC:~/Documents/IPK$
  
```

Filtrování UDP pakety

Capturing from wlp3s0 (arp)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-04-24 20:10:02.644656613	IntelCor_18:4b:38	Broadcast	ARP	42	Who has 192.168.50.1? To
2	2022-04-24 20:10:20.509832697	f0:2f:74:e4:dd:50	AzureWav_97:d1:a3	ARP	42	Who has 192.168.50.76?
3	2022-04-24 20:10:20.509854623	AzureWav_97:d1:a3	f0:2f:74:e4:dd:50	ARP	42	192.168.50.76 is at 6c:
4	2022-04-24 20:10:23.534124123	IntelCor_18:4b:38	Broadcast	ARP	42	Who has 192.168.50.1? To

Frame 2: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface wlp3s0, id 0

Ethernet II, Src: f0:2f:74:e4:dd:50 (f0:2f:74:e4:dd:50), Dst: AzureWav_97:d1:a3 (6c:71:d9:97:d1:a3)

Address Resolution Protocol (request)

```

0000 6c 71 d9 97 d1 a3 f0 2f 74 e4 dd 50 08 06 00 01 1q...../t..P....
0010 08 00 06 04 00 01 f0 2f 74 e4 dd 50 c0 a8 32 01 ...../t..P..2.
0020 00 00 00 00 00 00 c0 a8 32 4c .....2L

```

wlp3s0: <live capture in progress> Packets: 4 · Displayed: 4 (100.0%) Profile: Default

```

simir@simir-X550CC:~/Documents/IPK$ sudo ./sniffer --arp -i wlp3s0
timestamp: 2022-04-24T20:10:20.509
frame length: 42 bytes
src MAC: f0:2f:74:e4:dd:50
dst MAC: 6c:71:d9:97:d1:a3
0x0000 6c 71 d9 97 d1 a3 f0 2f 74 e4 dd 50 08 06 00 01 1q...../t..P....
0x0010 08 00 06 04 00 01 f0 2f 74 e4 dd 50 c0 a8 32 01 ...../t..P..2.
0x0020 00 00 00 00 00 00 c0 a8 32 4c .....2L
simir@simir-X550CC:~/Documents/IPK$

```

Filtrování ARP pakety

Literatura

- [1] Arora, H.: How to Perform Packet Sniffing Using Libpcap with C Example Code. [online], rev. 25. října 2012, [vid. 2022-04-24].
URL <https://www.thegeekstuff.com/2012/10/packet-sniffing-using-libpcap/>
- [2] Carstens, T.: PROGRAMMING WITH PCAP. [online], rev. 2002, [vid. 2022-04-24].
URL <https://www.tcpdump.org/pcap.html>
- [3] random dude: C Epoch Converter Routines. [online], rev. unknown, [vid. 2022-04-24].
URL <https://www.epochconverter.com/programming/c>
- [4] ip6h: ip6.h. [online], rev. 2006, [vid. 2022-04-24].
URL <https://sites.uclouvain.be/SystInfo/usr/include/netinet/ip6.h.html>
- [5] TheTcpdumpGrou: MAN PAGE OF PCAP-FILTER. [online], rev. 25. ledna 2022, [vid. 2022-04-24].
URL <https://www.tcpdump.org/manpages/pcap-filter.7.html>
- [6] udph: udp.h. [online], rev. 2009, [vid. 2022-04-24].
URL <https://sites.uclouvain.be/SystInfo/usr/include/netinet/udp.h.html>
- [7] WIKIPEDIA: Ethernet frame. [online], rev. 1. dubna 2022, [vid. 2022-04-24].
URL https://en.wikipedia.org/wiki/Ethernet_frame
- [8] WIKIPEDIA: Transmission Control Protocol. [online], rev. 19. dubna 2022, [vid. 2022-04-24].
URL https://en.wikipedia.org/wiki/Transmission_Control_Protocol
- [9] WIKIPEDIA: IPv6. [online], rev. 2. dubna 2022, [vid. 2022-04-24].
URL <https://en.wikipedia.org/wiki/IPv6>
- [10] WIKIPEDIA: User Datagram Protocol. [online], rev. 24. dubna 2022, [vid. 2022-04-24].
URL https://en.wikipedia.org/wiki/User_Datagram_Protocol
- [11] WIKIPEDIA: IPv4. [online], rev. 5 duben 2022, [vid. 2022-04-24].
URL <https://en.wikipedia.org/wiki/IPv4>
- [12] WIKIPEDIA: Address Resolution Protocol. [online], rev. 6. dubna 2022, [vid. 2022-04-24].
URL https://en.wikipedia.org/wiki/Address_Resolution_Protocol
- [13] WIKIPEDIA: Internet Control Message Protocol. [online], rev. 7. dubna 2022, [vid. 2022-04-24].
URL https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol