VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové komunikace a sítě – 2. projekt

Sniffer paketov – variant Zeta

23. apríla 2022

Martin Kozák

Obsah

1 Úvod

Cieľom druhého projektu v predmete Počítačové siete a komunikácia bolo vytvoriť program na zachytávanie packetov z daného internetového rozhrania a následne zistiť bližšie informácie z packetu.

Zadaním bolo vytvoriť packet sniffer, alebo analyzátor packetov. Pozostáva z dvoch častí a to zachytávač paketov a analyzátor paketov. Zachytávač paketov v tomto projekte umožnuje nastavenia filtrov pre porty, `ipv4` a `ipv6` a 5 protokolov: `TCP`, `UDP`, `ICMP`, `ICMPV6`.

2 Použitie

Zdrojový kód je možné preložiť pomocou priloženého makefile, príkazom `make`. Program je možné spustiť pomocou:

```
$ sudo ./ipk-sniffer [-i rozhraní | --interface rozhraní] -p port [--tcp|-t]
[--udp|-u] [--arp] [--icmp]  -n num
```

`-i rozhranie | --interface rozhranie` - názov rozhrania
`-p port` - bude filtrovať pakety na danom rozhraní podľa portu
`--tcp|-t` - budú sa zobrazovať len TCP pakety
`--tcp|-t` - budú sa zobrazovať len UDP pakety
`--icmp` - budú sa zobrazovať len ICMPv4 a ICMPv6 pakety
`arp` - bude zobrazovať pouze ARP rámce
`-n num` - určuje počet paketov, ktoré se majú zobraziť

Pokiaľ nebudú konkrétne protokoly špecifikované, uvažujú sa k výpisu všetky. V prípade úspešného vykonania programu sa vráti hodnota `0`. V prípade neúspešného vykonania programu sa vráti hodnota `1`.

3 Implementačné detaily

Celý program je napísaný v jazyku C. Celý kód je napísaný v súbore `ipk-sniffer.c`. Na implementáciu bola použitá knihovňa `libpcap`, ktorá poskytuje funkcie na zachytávanie paketov pre UNIXové systémy. Konkrétne je využité API `pcap`, teda dostupné funkcie z hlavičkového súboru `pcap.h`. Na spracovanie hlavičiek protokolov získaných z paketov sú použité štruktúry z hlavičkových súborov `netinet`.

3.1 Vytvorenie sieťového deskriptoru

Vytvorenie sieťového deskriptoru prebieha s využitím funkcií z `pcap.h`. Celý priebeh vytvorenia prebieha vo funkcii `create_set_pcap_descriptor()`, ktorá vracia ukazovateľ na tento deskriptor. Parametrami funkcie sú meno rozhrania a filter. Filter sa vytvára vo funkcii `create_filter()`. Vo funkcii sa kontroluje rozhranie, pripravuje sa filter, vytvára sa deskriptor a nastavuje filter. Používajú sa funkcie a postup z [1].

Po vytvorení zadaného deskriptoru, ktorý sa vráti do funkcie `main`, sa následne volá funkcia `pcap_loop()` [2], ktorá vytvára konečný cyklus s počtom cyklov `n_packets` na zbieranie paketov, volá pri tom funkciu `packet_parser`, ktorá príjma reťazcový literál reprezentujúci daný paket. V tejto funkcii, už prebieha parsovanie paketu.

3.2 Parsovanie paketu - packet_parser()

Zistenie všetkých potrebných informácií a protokolov na výpis prebieha práve v tejto funkcii. Program je vďaka použitej funkcie `signal` z hlavičkového súboru `signal.h` možné hocikedy ukončiť pomocou `ctrl + C`.

Najprv sa načíta hlavička ethernethového rámca pomocou štruktúry `ether_header`. Z nej je podľa dokumentácie možné zistiť, cieľovú, zdrojovú MAC adresu a typ protokolu `ether_type`, podľa ktorého sa následne rozhoduje, či nasledujúci protokol v hlavičke je `ipv4`, `ipv6`, alebo `arp`. Ak sa jedná o protokol typu `arp`, tak sa hneď volá funkcia `arp_packet()`. Ak sa jedná o typ pripojenia `ipv4` alebo `ipv6`, tak sa zistuju z hlavičky protokolu ip adresy a typ nasledujúceho protokolu/hlavičky.

Protokol typu `ipv4` sa načítava do štruktúry `ip` z `ip.h` a protokol typu `ipv6` do štruktúry `ip6_hdr` z `ip6.h`. Najprv sa zistí dĺžka hlavičiek. Pre `ipv6` je daná pevne a to 40 bytov[3]. Pre `ipv4` sa musí dopočítať pomocou dĺžky, ktorá je v hodnote `ip_hl`*4[4], na posledných štyroch bitoch prvého bajtu [].

Na tomto mieste sa vypisujú ip adresy. Pre `ipv4` adresu je použitá funkcia `inet_ntoa()`, ktorá berie premenné `ip_src` a `ip_dst`. Pre `ipv6` adresu je použitá funkcia `inet_ntop()`, ktorá berie premenné `ip6_src` a `ip6_dst`.

Na rozhodnutie o tom ktorý protokol nasleduje po `ipv4` a `ipv6` sa použijú informácie z 10 bytu z hlavičky pre `ipv4`[4] a z 7 bytu z hlavičky pre `ipv6`[3].

ARP protokol

Pri type protokolu `arp` sa načíta hlavička do štruktúry `ether_arp`, odkiaľ sa získa IP adresa z premennej `arp_tpa` a `arp_spa`

TCP protokol

Pri type protokolu `tcp` sa načíta hlavička do štruktúry `tcphdr`, odkiaľ sa získa IP adresa z premennej `source` a `dest`.

UDP protokol

Pri type protokolu `udp` sa načíta hlavička do štruktúry `udphdr`, odkiaľ sa získa IP adresa z premennej `source` a `dest`.

ICMP a ICMPv6 protokol

Pri type protokolu `ICMP` a `ICMPv6` sa nede

3.3 Funkcie na výpis informácii na štandardný výstup

print_all_devices()

Funckia vypíše dostupné rozhrania na štandartný výstup na ktorých je možné odchytávať pakety. Využíva k tomu funkciu `pcap_find_alldevs`[5].

print_packet_head()

Najprv sa vypisujú informácie z hlavičky ethernetového rámca. Vypisuje zdrojovú MAC adresu a cieľovú adresu z štruktúry `ether_header`.

print_packet()

Funkcia vypíše celý paket v hexadecimálnom kóde a ASCII kóde. Netisknuteľné znaky sú nahradené vo výpise bodkou. na zistenie netiskuteľného znaku je použitá funkcia `isprint()`.

Celkovo sa výpis uskutočňuje volaním funkcii vo funkcii `packet_parser()`, najprv sa vypíše čas podľa RFC3339, potom MAC adresy z ethernethovej hlavičky, IP adresy, poprípade porty ak existujú a typ protokolu naviac. Nasleduje kompletný výpis paketu v hexadecimálnom a ASCII tvare.

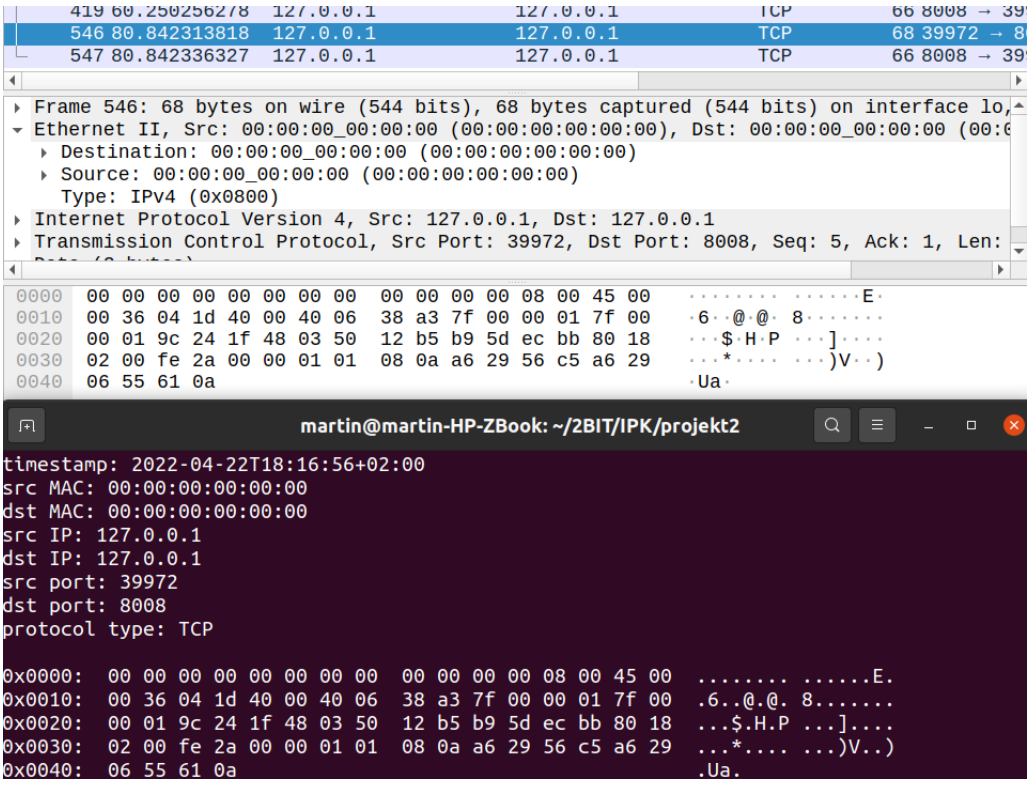
```
timestamp: 2022-04-22T17:47:04+02:00
src MAC: 84:c5:a6:e1:f9:10
dst MAC: 00:1a:1e:06:87:10
src IP: 100.64.202.79
dst IP: 162.159.130.234
src port: 55320
dst port: 443
protocol type: TCP

0x0000:  00 1a 1e 06 87 10 84 c5  a6 e1 f9 10 08 00 45 00  ........ ......E.
0x0010:  00 28 ff 41 40 00 40 06  e7 74 64 40 ca 4f a2 9f  .(.A@.@. .td@.O..
0x0020:  82 ea d8 18 01 bb c5 d0  00 2c c8 6c 64 98 50 10  ........ .,.ld.P.
0x0030:  13 1c 54 34 00 00                                ..T4..
```

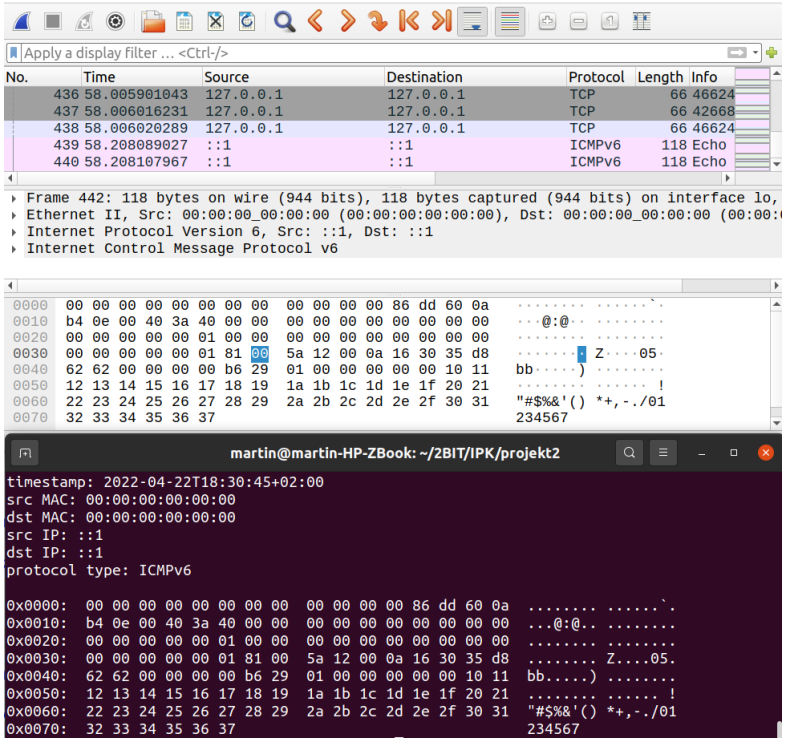
Obr. 1: príklad výpisu programu

4 Testovanie

Testovanie prebiehalo pomocou porovnávania výstupu programu `ipk-sniffer` a verejne dostupného programu `wireshark`. Pakety som generoval pomocou programu `ping` a `nc` na vytvorenie jednoduchej komunikácie client/server. Na otestovanie `imcp` a `icmpv6` protokolov som použil program `ping`.



Obr. 2: príklad porovnania výstupu medzi wireshark a mojim programom TCP/ipv4



Obr. 3: príklad porovnania výstupu medzi wireshark a mojim programom ICMPv6

5 Záver

Programovanie sniffera ma veľmi bavilo. Vyskúšal som si prácu s paketami a ich hlavičkami. Dokážem rozlíšiť v ethernetovom rámci MAC adresy, aky typ protokolu je využitý, poprípade kde hľadať ip adresy. Projekt som riadne zdokumentoval a citoval. Je otestovaný na všetkých prepínačoch. Otestoval som aj príjmanie `ipv4` a `ipv6` protokolov.

Zdroje

[1] Carstens, T.: *PROGRAMMING WITH PCAP*. [online], [cit. 2022-04-23].
Dostupné z: `https://www.tcpdump.org/pcap.html`

[2] Group, T. T.: *MAN PAGE OF PCAP_LOOP*. [online], [cit. 2022-04-23].
Dostupné z: `https://www.tcpdump.org/manpages/pcap_loop.3pcap.html`

[3] GeeksforGeeks: *Internet Protocol version 6 (IPv6) Header*. [online], [cit. 2022-04-23].
Dostupné z: `https://www.geeksforgeeks.org/internet-protocol-version-6-ipv6-header/`

[4] GeeksforGeeks: *Introduction and IPv4 Datagram Header*. [online], [cit. 2022-04-23].
Dostupné z: `https://www.geeksforgeeks.org/introduction-and-ipv4-datagram-header/`

[5] Group, T. T.: *MAN PAGE OF PCAP_FINDALLDEVS*. [online], [cit. 2022-04-23].
Dostupné z: `https://www.tcpdump.org/manpages/pcap_findalldevs.3pcap.html`