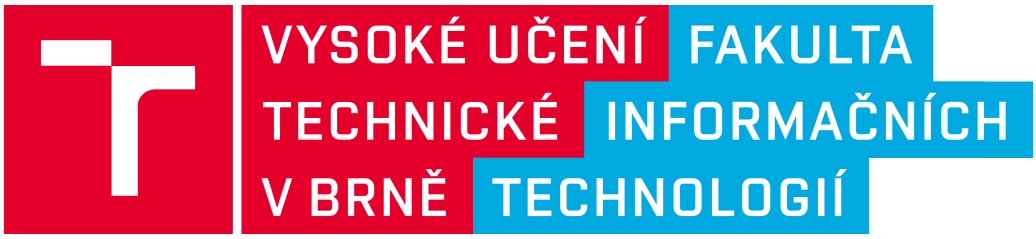


Počítačová komunikácia a siete

Packet sniffer

24. apríla 2022

Adam Rajko (xrajko00)

Obsah

1 Úvod **2**

2 Spustenie programu **2**

3 Implementácia **3**
3.1 Použité knižnice 3
3.2 Spracovanie argumentov 3
3.3 Zostavenie sieťového adaptéru 4
3.4 Zachtávanie a analýza paketov 4
3.4.1 Spracovanie IPv4 a IPv6 4
3.4.2 Spracovanie TCP a UDP 5
3.5 Výpis paketu na štandartný výstup 5

4 Testovanie **5**

5 Použité zdroje **8**

1 Úvod

Cielom projektu bol návrh a implementácia sieťového analyzátoru, ktorý slúži na zachytávanie a filtrovanie paketov v sieti. Zachytávanie paketov je možné na rôznych sieťových zariadeniach, aktuálna implementácia podporuje len zariadania ethernetového typu. Riešenie podporuje zachytávanie paketov podľa protokolu paketu, ich kombináciou a podľa portu na ktorom komunikácia prebieha. Informácie o zachytenom pakete sa vypisujú na štandarntý výstup `stdout`.

2 Spustenie programu

V priečinku projektu sa nachádza Makefile, ktorý zostaví projekt použitím:
`$ make`
Pre odstránenie zkompilovaného progamu `sniffer` a objektov `*.o` je možné pomocou:
`$ make clean`
Program sa spúšta pomocou, (program je potrebné spúšťať ako administrátor):
`$ ./ipk-sniffer [-i rozhranie | --interface rozhranie] {-p port} {[--tcp|-t] [--udp|-u] [--arp] [--icmp]} {-n num}`

- `--interface <rozhranie> | -i <rozhranie>` - určuje rozhranie, na ktorom na ktorom sa budú zachytávať pakety. V prípade, že sa tento argument nevyskytuje, vypíše sa zoznam dostupných rozhraní.
- `-p <port>` - voliteľný parameter, filtruje pakety na danom rozhraní podľa portu (source aj destination). V prípade neuvedenia parametru sa uvažujú všetky porty.
- `--tcp | -t` - voliteľný parameter, budú zobrazované len TCP pakety
- `--udp | -u` - voliteľný parameter, budú zobrazované len UDP pakety
- `--icmp` - voliteľný parameter, budú zobrazované len ICMPv4 a ICMPv6 pakety
- `--arp` - voliteľný parameter, budú zobrazované len `ARP` rámce
- `-n <nummber>` - určuje počet paketov, ktoré sa majú zobraziť. V prípade, že sa argument nevyskytuje, zobrazí sa 1 paket.

V prípade chybných argumentov sa program skončí s návratovou hodnotou **1**. V prípade úspechu vráti hodnotu **0**. V prípade zlyhania súčastí knižnice PCAP vráti hodnotu **2**.

3 Implementácia

3.1 Použité knižnice

Na monitorovanie paketov je použitá knižnica `pcap.h` [1]. Zoznam použitých knižníc:

```
#include <arpa/inet.h>   // inet_ntop
#include <getopt.h>      // struct option, getopt_long
#include <netinet/ether.h> // struct ether_header
#include <netinet/in.h>  // ntohs
#include <netinet/ip.h>  // struct ip
#include <netinet/ip6.h> // struct ip6_hdr
#include <netinet/tcp.h> // struct tcphdr
#include <netinet/udp.h> // struct udphdr
#include <pcap.h>
#include <time.h> // localtime, strftime

#include <cmath>   // round
#include <iomanip> // setfill, setw
#include <iostream> // cout
#include <sstream> // stringstream
#include <string>  // string
```

3.2 Spracovanie argumentov

Na spracovanie argumentov je použitá knižnica `getopt.h` [2], ktorá podporuje parsovanie dlhých aj krátkych parametrov. Implementácia dlhých parametrov pomocou štruktúry `option` a krátkych parametrov:

```
// long options
struct option long_options[] = {
    {"interface", optional_argument, NULL, 'i'},
    {"tcp", no_argument, &tcp, 1},
    {"udp", no_argument, &udp, 1},
    {"icmp", no_argument, &icmp, 1},
    {"arp", no_argument, &arp, 1},
    {0, 0, 0, 0}};

// short options
// ::   - optional argument
// :    - required argument
const char *short_options = "i::p:tun:";
```

Následne samotné parsovanie je implementované pomocou funkcie `getopt_long`, ktorá vracia znak v prípade nájdenia krátkeho parametra:

```
c = getopt_long(argc, argv, short_options, long_options, nullptr)
```

3.3 Zostavenie sieťového adaptéru

Implementácia sieťového adaptéru využíva funkcie knižnice `pcap.h`. Najprv je potrebné nastaviť interface, ktorý je získaný zo vstupného paramatru. V prípade neprítomnosti parametru je program ukončený s hodnotou `0`.

Potom je možné priradiť zariadeniu na ktorom sa pracuje masku siete a ip adresu pomocou funkcie `pcap_lookupnet`.

Ak všetko prebehne v poriadku je možné otvoriť zariadenie na zachytávanie paketov, na čo slúži funkcia `pcap_open_live`, kde návratová hodota sa uloží do `pcap_t *handle`.

Následne je potrebné skontrolovať podporu ethernetovej hlavicky pomcou funkcie `pcap_datalink`, ktorej návratovú hodnotu je potrebné porovnať s konštantou `DLT_EN10MB` [5].

Potom je zostavený výraz zo vstupných parametrov, pre filtrovanie paketov pomocou funkcie `filter_expression_init`.

Na kompiláciu adaptéru slúži funkcia `pcap_compile`, kde potom skompilovaný adaptér pomocu funkcie `pcap_setfilter` aplikujeme.

Nakoniec zachytávanie paketov prebieha pomocou funkcie `pcap_loop`, kde obsluha paketov prebieha potom v callback funkcii `packet_handler` [4].

3.4 Zachtávanie a analýza paketov

Na obsluhu paketov slúži funkcia `packet_handler`. Prototyp funkcie:

```
void packet_handler(u_char *user_data, const struct pcap_pkthdr *pkthdr,
                    const u_char *packet);
```

3.4.1 Spracovanie IPv4 a IPv6

Spracovanie paketov typu UDP, TCP aj ICMP prebieha rovnako v oboch protokoloch. Je potrebné ale rozlíšiť hlavičky, z ktorých je možné zistiť IP adresu zdroja a aj cieľa [3].

```
const struct ip *ip_header = (struct ip *)(packet + sizeof(struct ether_header));
const struct ip6_hdr *ip6_header = (struct ip6_hdr *)(packet +
                                    sizeof(struct ether_header));
```

Získanie IP adries z hlavičiek (IPv4):

```
char source_ip[INET_ADDRSTRLEN];
char dest_ip[INET_ADDRSTRLEN];

inet_ntop(AF_INET, &(ip_header->ip_src), source_ip, INET_ADDRSTRLEN);
inet_ntop(AF_INET, &(ip_header->ip_dst), dest_ip, INET_ADDRSTRLEN);
```

3.4.2 Spracovanie TCP a UDP

Na zistenie hlavičiek, z ktorých je možné získať port zdroja a aj cieľa sú použité štruktúry z knižníc. Zistenie TCP a UDP hlavičky (IPv4) [3]:

```
struct tcphdr *tcp_header = (tcphdr *)(packet + sizeof(struct ether_header) +
                            sizeof(struct ip));
struct udphdr *udp_header = (udphdr *)(packet + sizeof(struct ether_header) +
                            sizeof(struct ip));
```

Zistenie portov z hlavičky (TCP):

```
std::string source_port = std::to_string(ntohs(tcp_header->source));
std::string dest_port = std::to_string(ntohs(tcp_header->dest));
```

3.5 Výpis paketu na štandartný výstup

Na získanie dát v požadovanom formáte na výstup zabezpečuje funkcia `get_data`. Prototyp funkcie:

```
std::string get_data(u_char *data, u_int32_t data_length);
```

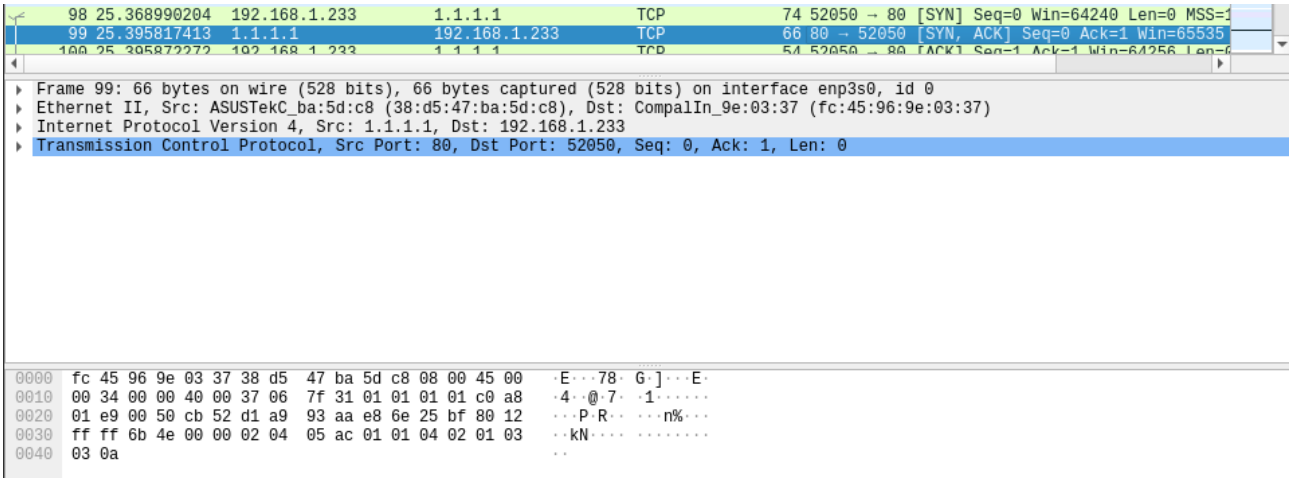
Paket sa vypíše na konci callback funkcie `packet_handler`.

```
timestamp:      2022-04-20T20:21:33.357+02:00
src MAC:        d0:f8:8c:fd:fb:ed
dst MAC:        01:00:5e:7f:ff:fa
frame length:   164 bytes
src IP:         192.168.1.192
dst IP:         239.255.255.250
src port:       45858
dst port:       1900

0x0000: 01 00 5e 7f ff fa d0 f8  8c fd fb ed 08 00 45 00        ..^.....  ......E.
0x0010: 00 96 6d cb 40 00 01 11  59 29 c0 a8 01 c0 ef ff        ..m.@...  Y)......
0x0020: ff fa b3 22 07 6c 00 82  ad 44 4d 2d 53 45 41 52        ...".l..  .DM-SEAR
0x0030: 43 48 20 2a 20 48 54 54  50 2f 31 2e 31 0d 0a 48        CH * HTT  P/1.1..H
0x0040: 4f 53 54 3a 20 32 33 39  2e 32 35 35 2e 32 35 35        OST: 239  .255.255
0x0050: 2e 32 35 30 3a 31 39 30  30 0d 0a 4d 41 4e 3a 20        .250:190  0..MAN:
0x0060: 22 73 73 64 70 3a 64 69  73 63 6f 76 65 72 22 0d        "ssdp:di  scover".
0x0070: 0a 53 54 3a 20 75 72 6e  3a 6d 64 78 2d 6e 65 74        .ST: urn  :mdx-net
0x0080: 66 6c 69 78 2d 63 6f 6d  3a 73 65 72 76 69 63 65        flix-com  :service
0x0090: 3a 74 61 72 67 65 74 3a  31 0d 0a 4d 58 3a 20 32        :target:  1..MX: 2
0x00a0: 0d 0a 0d 0a                                             ....
```

4 Testovanie

Testovanie prebiehalo na referenčnom stroji PDS-VM s OS Ubuntu 20.04.2 LTS. Správnosť zachytených paketov bola kontrolovaná pomocou softwéru Wireshark.

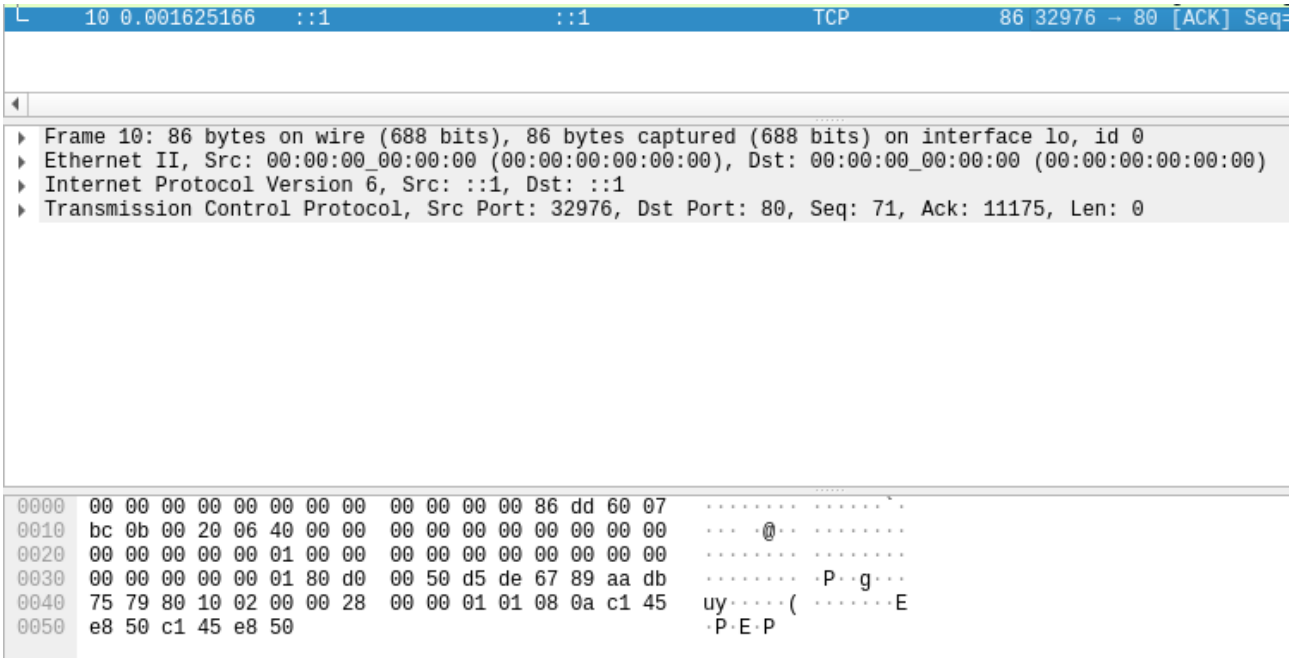


Obr. 1: Zachytený IPv4 paket programom Wireshark

```
timestamp:      2022-04-20T21:58:14.549+02:00
src MAC:        38:d5:47:ba:5d:c8
dst MAC:        fc:45:96:9e:03:37
frame length:   66 bytes
src IP:         1.1.1.1
dst IP:         192.168.1.233
src port:       80
dst port:       52050

0x0000: fc 45 96 9e 03 37 38 d5  47 ba 5d c8 08 00 45 00          .E...78.  G.]...E.
0x0010: 00 34 00 00 40 00 37 06  7f 31 01 01 01 01 c0 a8          .4..@.7.  .1......
0x0020: 01 e9 00 50 cb 52 d1 a9  93 aa e8 6e 25 bf 80 12          ...P.R..  ...n%...
0x0030: ff ff 6b 4e 00 00 02 04  05 ac 01 01 04 02 01 03          ..kN....  ........
0x0040: 03 0a                                                     ..
```

Obr. 2: Zachytený IPv4 paket programom `ipk-sniffer`



Obr. 3: Zachytený IPv6 paket programom Wireshark

```
timestamp:      2022-04-20T22:50:02.487+02:00
src MAC:        00:00:00:00:00:00
dst MAC:        00:00:00:00:00:00
frame length:   86 bytes
src IP:         ::1
dst IP:         ::1
src port:       32976
dst port:       80

0x0000: 00 00 00 00 00 00 00 00  00 00 00 00 86 dd 60 07          ........  ......`.
0x0010: bc 0b 00 20 06 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 d0  00 50 d5 de 67 89 aa db          ........  .P..g...
0x0040: 75 79 80 10 02 00 00 28  00 00 01 01 08 0a c1 45          uy.....(  .......E
0x0050: e8 50 c1 45 e8 50                                         .P.E.P
```

Obr. 4: Zachytený IPv6 paket programom `ipk-sniffer`

5 Použité zdroje

Literatúra

[1] Carstens, T.: pcap-linktype(7). [online], [vid. 2022-04-23].
URL <https://www.tcpdump.org/pcap.html>

[2] GNU: Getopt Long Option Example. [online], [vid. 2022-04-23].
URL <https://www.gnu.org/software/libc/manual/html_node/Getopt-Long-Option-Example.html>

[3] Lukasavage, T.: Offline packet capture analysis with C/C++ & libpcap. [online], [vid. 2022-04-23].
URL <http://tonylukasavage.com/blog/2010/12/19/offline-packet-capture-analysis-with-c-c----amp--libpcap/>

[4] Manual: Man page of pcap_loop. [online], [vid. 2022-04-23].
URL <https://www.tcpdump.org/manpages/pcap_loop.3pcap.html>

[5] Manual: Programming with pcap. [online], [vid. 2022-04-23].
URL <https://linux.die.net/man/7/pcap-linktype>