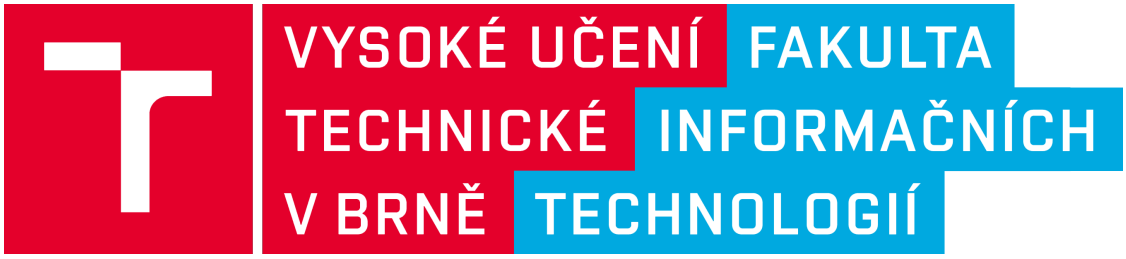


# Generování NetFlow dat ze zachycené síťové komunikace

Manuál

David Chocholatý (xchoch09)

Brno, 14. listopadu 2022

# Obsah

# 1 Úvod

Dokumentace popisuje úvod do problematiky, popis použití, návrh, implementaci a testování tzv. Flow exportéru (exportéru toků) [24].

# 2 Úvod do problematiky

NetFlow je síťový standard původně vyvinutý společností Cisco pro shromažďování informací o IP provozu a sledování síťových telemetrických dat. Přepínače nebo routery s podporou NetFlow, tzv. exportéry, generují tyto agregované statistiky provozu, které poskytují obraz o využití šířky pásma, komunikačních partnerech a aktivitách klientů[18].

Monitorování síťového provozu generuje statistiky jak o podkladových datových přenosech při abstrahování z paketů, tak o samotném předmětu komunikace (obsah komunikace se neukládá). Tyto statistiky představují údaje o toku v síti, které lze považovat za podobné seznamu telefonních hovorů. Síťové a bezpečnostní operace získají přehled o tom, kdo s kým komunikuje, kdy, jak dlouho a jak často. V jazyce prostředí datové sítě monitorují IP adresy, objemy dat, čas, porty, protokoly a to lze dále obohatit o měření latence a data aplikační vrstvy pro různé protokoly[18].

Typické nastavení monitorování toku (pomocí NetFlow) se skládá ze tří hlavních komponent: *Flow exportér*, *Flow kolektor* a *Analyzátor flow*.

- *Flow exportér* - zařízení nebo síťové zařízení (obvykle router nebo firewall), které má na starosti shromažďování informací o toku a jejich export do kolektoru toku[19].
- *Flow kolektor* - zařízení nebo server, který přijímá exportované informace o toku[19].
- *Analyzátor flow* - aplikace, která analyzuje informace o toku shromážděné kolektorem toku[19].

Nejčastěji používaným formátem je NetFlow v5. Následující dokumentace se vztahuje k programu *Flow exportéru*, který podporuje právě zmíněný formát.

# 3 Návod použití

## 3.1 Předpoklady

- gcc
- GNU Make
- tar (Nutné pouze při vytváření archivu)

## 3.2 Vytvoření projektu

Projekt lze vytvořit pomocí Makefile následujícím příkazem

```
# make
```

## 3.3 Spuštění Flow exportéru

### Argumenty

Flow exportér lze spustit s následujícími argumety:

| Argument | Popis |
|---|---|
| -f soubor | jméno analyzovaného souboru - ve formátu pcap (výchozí: STDIN) |
| -c netflow_kolektor:port | IP adresa nebo hostname NetFlow kolektoru (výchozí: 127.0.0.1:2055 ) |
| -a aktivní_časovač | interval v sekundách po kterém se exportují aktivní záznamy na kolektor (výchozí: 60) |
| -i sekundy | interval v sekundách po kterém se exportují neaktivní záznamy na kolektor (výchozí: 10) |
| -m počet | velikost mezipaměti (výchozí: 1024) |

### Příklad spuštění - Obecný zápis volání programu

```
# ./flow [–f <soubor>] [–c <netflow_kolektor>[:<port>]] [–a <aktivni_casovac>]
    [–i <neaktivni_casovac>] [–m <pocet>]
```

### Příklad spuštění - Ukázkové příklady

Vytvoření projektu

```
# make
```

Spuštění Flow exportéru s výchozím nastavením (soubor: *STDIN*, netflow_kolektor:port: *127.0.0.1:2055*, aktivní časovač: *60*, neaktivní časovač: *10*, velikost mezipaměti: *1024* ).

```
# ./flow
```

Výpis nápovědy.

```
# ./flow –h
```

Spuštění Flow exportéru s nastavením vstupního souboru na *input.pcap* a Flow kolektoru na *192.168.0.1:2055*. Ostatní parametry jsou ponechány ve výchozím nastavení.

```
# ./flow -f input.pcap -c 192.168.0.1:2055
```

Spuštění Flow exportéru s nastavením vstupního souboru na *input.pcap* a Flow kolektoru na *192.168.127.1* bez explicitního uvedení volitelného UDP portu. Ostatní parametry jsou ponechány ve výchozím nastavení.

```
# ./flow -f input.pcap -c 192.168.127.1
```

Spuštění Flow exportéru s nastavením vstupního souboru na *input.pcap*, Flow kolektoru na *192.168.0.1:2055* a aktivního časovače na 360 sekund. Ostatní parametry jsou ponechány ve výchozím nastavení.

```
# ./flow -f input.pcap -c 192.168.0.1:2055 -a 360
```

Spuštění Flow exportéru s nastavením vstupního souboru na *input.pcap*, Flow kolektoru na *192.168.0.1:2055* a neaktivního časovače na 120 sekund. Ostatní parametry jsou ponechány ve výchozím nastavení.

```
# ./flow -f input.pcap -c 192.168.0.1:2055 -i 120
```

Spuštění Flow exportéru s nastavením vstupního souboru na *input.pcap*, Flow kolektoru na *192.168.0.1:2055* a velikosti mezipaměti na 8192. Ostatní parametry jsou ponechány ve výchozím nastavení.

```
# ./flow -f input.pcap -c 192.168.0.1:2055 -m 8192
```

Spuštění Flow exportéru s nastavením vstupního souboru na *input.pcap*, Flow kolektoru na *192.168.0.1:2055*, aktivního časovače na 600 sekund, neaktivního časovače na 360 sekund a velikosti mezipaměti na 4096.

```
# ./flow -f input.pcap -c 192.168.0.1:2055 -a 600 -i 360 -m 4096
```

# 4 Návrh aplikace

Aplikace se skládá z následujících podstatných částí:

- Čtení vstupních dat ve formátu pcap (pcap.c (.h))
- Zpracování vstupních argumentů (option.c (.h))
- Zpracování zachycených paketů (netflow_v5.c (.h))
- Správa soketů (flow.c (.h))

- Správa paměti (memory.c (.h))
- Zpracování chybových stavů (error.c (.h))
- Správa mezipaměti (implementovaná pomocí binárního vyhledávacího stromu, tree.c (.h))
- Správa vytvořených toků (netflow_v5.c (.h))
- Exportér toků (netflow_v5.c (.h))

Pro popis bezchybného toku algoritmu program nejprve zpracuje vstupní uživatelské argumenty. Pokud zmíněné argumenty jsou validní, pokračuje čtením dat ze zadaného souboru či ze STDIN. Program podporuje pouze protokoly TCP, UDP a ICMP. Dále po zpracování paketů je paket zařazen do již existujícího toku, pokud se jeho klíč shoduje s klíčem daného toku. V opačném případě je vytvořen nový tok do kterého je zpracovaný paket zařazen. Program průběžně kontroluje hodnoty časovačů a zaplněnost mezipaměti. V případě vypršení některého z časovačů či "překročení"velikosti paměti (mezipaměť před zařazením nového toku již obsahuje zadaný počet povolených toků) jsou případné toky exportovány na kolektor. Po zpracování všech vstupních dat jsou zbylé toky exportovány na kolektor a program ukončen.

# 5 Implementace

Flow exportér je implementovaný v jazyce C. Při překladu je používána verze C99. Tento program podporuje NetFlow v5. Projekt využívá knihovnu *libpcap*[20]. Implementace je vytvořena v následujících souborech:

- Makefile - překlad projektu
- error.c (.h) - práce s chybovými kódy, výpis chybových zpráv na *stderr*
- flow.c (.h) - hlavní soubor Flow exportéru obsluhující logiku programu
- memory.c (.h) - obsluha paměti (alokace a uvolňování paměti)
- netflow_v5.c (.h) - obsluha Flow záznamů a exportování na kolektor
- option.c (.h) - obsluha argumentů
- pcap.c (.h) - obsluha vstupních paketů
- tree.c (.h) - binární vyhledávací strom implementující mezipaměť
- util.c (.h) - implementované pomocné funkce pro exportér

## 5.1 Použité knihovny

### Knihovny pro práci se síťovými prvky

- <arpa/inet.h> - funkce *ntohs(), htons(), ...*
- <linux/if_ether.h> - struktura hlavičky ethernetového rámce *ethhdr*
- <pcap.h> - funkce pro čtení dat ze souboru či STDIN ve formátu pcap
- <netdb.h> - definice funkce *gethostbyname()*
- <netinet/in.h> - struktura *sockaddr_in*
- <netinet/ip.h> - struktura IP hlavičky *ip*
- <netinet/ip_icmp.h> - struktura ICMP *icmp*
- <netinet/tcp.h> - struktura TCP hlavičky *tcphdr*
- <netinet/udp.h> - struktura UDP hlavičky *udphdr*

### Ostatní knihovny

- <stdio.h> - práce se vstupem a výstupem programu
- <stdlib.h> - *EXIT_SUCCESS, EXIT_FAILURE, strtoul(), exit()*
- <stdbool.h> - datový typ *bool*
- <stddef.h> - definice *NULL*
- <stdint.h> - datové typy *uint_8t*, *uint_16t*, *uint_32t* a *uint_64t*
- <string.h> - funkce pro práci s řetězci a pamětí: *strcpy(), ...*
- <limits.h> - rozsahy datových typů: *UINT16_MAX*, ...
- <unistd.h> - funkce *getopt()*

## 5.2 Zpracování argumentů

Zpracování argumentů je implementováno v souboru ***option.c (.h)***. Hlavní funkcí pro obsluhu argumentů je funkce *handle_options()*. Zmíněná funkce nadále pracuje s funkcí *init_options()* pro inicializaci argumentů a s funkcí *parse_options()* pro zpracování uživatelských parametrů. Pro zpracování argumentů je použita standardní funkce *getopt()*[11]. Pokud není uživatelem poskytnutý název a port kolektoru, je pomocí funkce *set_default_if_not_user_set()* nastavena výchozí hodnota *127.0.0.1:2055*. Případný výpis nápovědy je proveden pomocí funkce *print_help()*.

## 5.3 Zpracování vstupních dat

Zpracování vstupních dat je implementováno ve funkci *run_packets_processing()* nacházející se ve zdrojovém souboru ***pcap.c (.h)***. Zmíněná funkce využívá pro čtení dat ze souboru funkce *pcap_open_offline()*[21] a *pcap_next_ex()*[22]. Pokud není uživatelem poskytnutý vstupní soubor jsou vstupní data očekávána na STDIN. Pokud načtená data obsahují IPv4 paket, jsou zmíněná data dále podrobněji zpracována volanou funkcí *process_packet()* ze souboru *netflow_v5.c (.h)*. Speciální vlastností implementace zpracování vstupních dat je omezení povoleného rozsahu pro argumenty *aktivní časovač*, *neaktivní časovač* a *velikost mezipaměti*[2]:

- aktivní časovač (-a) 60 - 3600
- neaktivní časovač (-i) 10 - 600
- velikost mezipaměti (-m) 1024 - 524288

## 5.4 Správa soketů

Správa soketů je implementována ve funkcích *connect_socket()* a *disconnect_socket()* nacházejících se ve zdrojovém souboru ***flow.c (.h)***. Zmíněná funkce *connect_socket()* zajišťuje vytvoření soketu. K tomu jsou v průběhu vykonávání algoritmu použity funkce *socket()*[9] a *gethostbyname()*[10]. Funkce je inspirována poskytnutým zdrojovým kódem v souboru *echo-udp-client2.c* poskytnutým v kurzu ISA na VUT FIT [12]. Pro uzavření vytvořeného soketu je použita funkce *disconnect_socket()*.

## 5.5 Zpracování zachycených paketů

Zpracování zachycených paketů je implementováno ve funkci *process_packet()* nacházející se ve zdrojovém souboru ***netflow_v5.c (.h)***. Hlavní úlohou zmíněné funkce je zpracování dat paketu. Program podporuje pouze protokoly TCP, UDP a ICMP. Pakety s jinými protokoly nejsou dále zpracovávány. Speciální vlastností funkce je výpočet cílového portu pro protokol ICMP[4]. Ten je vypočítán pomocí ICMP typu a ICMP kódu pomocí následujícího příkazu:

```
packet_key->dst_port = my_icmp->icmp_type * 256 + my_icmp->icmp_code;
```

Po zpracování paketu je dále volána funkce *find_flow()* implementována ve stejném souboru pro zařazení paketu do toku.

## 5.6 Správa vytvořených toků

Správu toků zajišťují především funkce *find_flow()* a *compare_flows()* implementovány ve zdrojovém souboru ***netflow_v5.c (.h)***. Funkce *find_flow()* zajišťuje zařazení zpracovaného paketu do toku se shodným klíčem jako je klíč paketu. Pokud takový tok není v mezipaměti uložen, funkce vytváří nový tok, který je poté uložen do zmíněné mezipaměti. Funkce navíc zajišťuje kontrolu, zda není překročena povolená velikost mezipaměti. Pokud je vytvořen nový tok a mezipaměť již obsahuje povolený počet záznamů, je nejstarší záznam v mezipaměti exportován na kolektor a nově vytvořený tok je zařazen místo smazaného záznamu (paměť opět po vložení obsahuje maximální povolený počet záznamů v mezipaměti). Funkce *compare_flows()* zajišťuje porovnávání dvojicí klíčů. Jako unikatní hodnota klíče jednotlivých toků je použita následující sedmice údajů:

- zdrojové rozhraní (pozn. nastaveno vždy na 0)
- zdrojová IP adresa
- cílová IP adresa
- IP protokol
- zdrojový port
- cílový port

- IP typ služby

## 5.7 Exportér toků

Exportér toků je implementován pomocí funkcí *export_flows()*, *export_expired_flows()* a *export_all_flows_dispose_tree()* nacházejících se ve zdrojovém souboru ***netflow_v5.c (.h)***.

### Funkce *export_flows()*

Funkce *export_flows()* exportuje toky na příslušný Flow kolektor. Maximální počet toků, který funkce zasílá v jednom paketu je 30. Funkce nastavuje formát hlavičky a formát jednotlivých záznamů dle následující specifikace of společnosti Cisco pro verzi 5 (NetFlow v5)[3]:

Pozn.: V následujících tabulkách jsou hodnoty, které jsou při exportování nastavené jako nulové, zapsány *kurzívou*. Konkrétně se tedy jedná o hodnoty *engine_type*, *engine_id* a *sampling_interval* uvedené v hlavičce a hodnoty *nexthop*, *input*, *output*, *pad1*, *src_as*, *dst_as*, *src_mask*, *dst_mask* a *pad2* uvedené v samotném záznamu NetFlow v5.

*Z důvodu mnohdy zavádějícího významu při překladu do češtiny je obsah tabulek ponechán v původním znění v anglickém jazyce.*

| Bytes | Contents | Description |
|---|---|---|
| 0-1 | version | NetFlow export format version number |
| 2-3 | count | Number of flows exported in this packet (1-30) |
| 4-7 | SysUptime | Current time in milliseconds since the export device booted |
| 8-11 | unix_secs | Current count of seconds since 0000 UTC 1970 |
| 12-15 | unix_nsecs | Residual nanoseconds since 0000 UTC 1970 |
| 16-19 | flow_sequence | Sequence counter of total flows seen |
| *20* | *engine_type* | *Type of flow-switching engine* |
| *21* | *engine_id* | *Slot number of the flow-switching engine* |
| *22-23* | *sampling_interval* | *First two bits hold the sampling mode; remaining 14 bits hold value of sampling interval* |

Obrázek 1: Formát hlavičky NetFlow v5

| Bytes | Contents | Description |
|---|---|---|
| 0-3 | srcaddr | Source IP address |
| 4-7 | dstaddr | Destination IP address |
| *8-11* | *nexthop* | *IP address of next hop router* |
| *12-13* | *input* | *SNMP index of input interface* |
| *14-15* | *output* | *SNMP index of output interface* |
| 16-19 | dPkts | Packets in the flow |
| 20-23 | dOctets | Total number of Layer 3 bytes in the packets of the flow |
| 24-27 | First | SysUptime at start of flow |
| 28-31 | Last | SysUptime at the time the last packet of the flow was received |
| 32-33 | srcport | TCP/UDP source port number or equivalent |
| 34-35 | dstport | TCP/UDP destination port number or equivalent |
| *36* | *pad1* | *Unused (zero) bytes* |
| 37 | tcp_flags | Cumulative OR of TCP flags |
| 38 | prot | IP protocol type (for example, TCP = 6; UDP = 17) |
| 39 | tos | IP type of service (ToS) |
| *40-41* | *src_as* | *Autonomous system number of the source, either origin or peer* |
| *42-43* | *dst_as* | *Autonomous system number of the destination, either origin or peer* |
| *44* | *src_mask* | *Source address prefix mask bits* |
| *45* | *dst_mask* | *Destination address prefix mask bits* |
| *46-47* | *pad2* | *Unused (zero) bytes* |

Obrázek 2: Formát záznamu NetFlow v5

Uvedené informace jsou uloženy pomocí struktur *netflow_v5_header* a *netflow_v5_flow_record* nacházejících se ve zdrojovém hlavičkovém souboru ***netflow_v5.h***.

```
/*
 * Structure to store a NetFlow header.
 */
struct netflow_v5_header
{
    uint16_t version;
    uint16_t count;
    uint32_t sysuptime_ms;
    uint32_t unix_secs;
    uint32_t unix_nsecs;
    uint32_t flow_sequence;
    uint8_t engine_type;
    uint8_t engine_id;
    uint16_t sampling_interval;
};

/*
```

```
 * Structure to store exported NetFlow record.
 */
struct netflow_v5_flow_record
{
    uint32_t src_addr;
    uint32_t dst_addr;
    uint32_t nexthop;
    uint16_t input;
    uint16_t output;
    uint32_t packets;
    uint32_t octets;
    uint32_t first;
    uint32_t last;
    uint16_t src_port;
    uint16_t dst_port;
    uint8_t pad1;
    uint8_t tcp_flags;
    uint8_t prot;
    uint8_t tos;
    uint16_t src_as;
    uint16_t dst_as;
    uint8_t src_mask;
    uint8_t dst_mask;
    uint16_t pad2;
};
```

### Funkce *export_expired_flows()*

Funkce *export_expired_flows()* zajišťuje exportování expirovaných toků na kolektor. Pro tento účel je vytvořen dočasný binární vyhledávací strom do kterého jsou pomocí funkce *bst_find_expired()* vloženy expirované toky ze stromu reprezentujícího mezipaměť a obsahujícího uzly s jednotlivými toky. Dále pomocí volání funkce *export_all()* s poskytnutými expirovanými toky jsou tyto toky zaslány na kolektor pomocí paketů. Zmíněná logika funkce je zachycena následujícím úsekem zdrojového kódu:

```
/*
 * Function for exporting expired flows to collector.
 *
 * @param netflow_records   Pointer to pointer to the netflow recording system.
 * @param sending_system    Pointer to pointer to the sending system.
 * @param packet_time_stamp Current packet time stamp.
 * @param options           Pointer to options storage.
 * @return                  Status of function processing.
 */
uint8_t export_expired_flows(netflow_recording_system_t netflow_records,
                             netflow_sending_system_t sending_system,
                             struct timeval* packet_time_stamp,
                             options_t options)
{
```

```
    uint8_t status;
    bst_node_t expired_flows_tree;

    bst_init(&expired_flows_tree);

    // Add expired flows into the expired flows tree.
    status = bst_find_expired(&(netflow_records->tree),
                              &expired_flows_tree,
                              packet_time_stamp,
                              options);

    if (status != NO_ERROR)
    {
        return status;
    }

    // Export all flows from the expired flows tree by the oldest one.
    status = bst_export_all(netflow_records, sending_system, &expired_flows_tree);

    if (status != NO_ERROR)
    {
        bst_dispose(&(netflow_records->tree));
    }

    return status;
}
```

#### Funkce *export_all_flows_dispose_tree()*

Funkce *export_all_flows_dispose_tree()* zajišťuje exportování všech uzlů uchovaných v mezipaměti na kolektor a vymazání celého binárního vyhledávacího stromu reprezentujícího zmíněnou mezipamět.

## 5.8 Správa mezipaměti

Správa mezipaměti se nachází ve zdrojovém souboru ***tree.c (.h)***. Mezipamět uchovávající jednotlivé toky je reprezentována binárním vyhledávacím stromem[23], jehož spravující operace jsou implementovány následujícími funkcemi:

- *bst_init()* - inicializace stromu
- *bst_search()* - vyhledání uzlu ve stromu
- *bst_insert()* - vložení uzlu do stromu
- *bst_replace_by_rightmost()* - pomocná funkce pro funkci *bst_delete()*
- *bst_delete()* - vymazání uzlu ze stromu
- *bst_dispose()* - vymazání celého binárního vyhledávacího stromu
- *bst_move_node()* - přesun uzlu z jednoho stromu do druhého
- *bst_find_expired()* - nalezení expirovaných uzlů (toků)
- *bst_find_oldest()* - nalezení nejstaršího uzlu ve stromu
- *bst_export_oldest()* - exportování nejstaršího uzlu (toku) ve stromu
- *bst_export_all()* - exportování všech uzlů (toků) stromu

## 5.9 Správa paměti

Správa paměti (alokace a uvolnění paměti) je implementována v následujících funkcích nacházejících se v souboru ***memory.c (.h)***:

- Alokace paměti
  - *allocate_options()*
  - *allocate_string()*
  - *allocate_socket()*
  - *allocate_recording_system()*
  - *allocate_sending_system()*
  - *allocate_netflow_key()*
  - *allocate_flow_node()*
  - *allocate_tree_nod()*
- Uvolnění paměti
  - *free_options_mem()*
  - *free_netflow_key()*
  - *free_flow_node()*
  - *free_tree_node()*
  - *free_tree_node_keep_data()*
  - *free_flow_values_array()*
  - *free_string()*
  - *free_socket()*
  - *free_recording_system()*
  - *free_sending_system()*

- *free_allocated_mem()*

- Pomocné funkce
  - *is_allocated()*

# 6 Testování

Testování bylo prováděno na počítači s následující specifikací:

- Hardware
  - CPU: Intel Core i7-3520M
  - RAM: 8GiB (2x4 GiB SODIMM DDR3 1600MHz)
- Software
  - Ubuntu 22.04.1 LTS

Při testování byl použit open-source nástroj Wireshark[25] a webový prohlížeč Firefox 97.0 (64 bit)[6]. Nástroj Wireshark[25] byl využit na prohlížení komunikace uložené v souborech ve formátu pcap. Webový prohlížeč Firefox [6] byl využit pro generování TCP a UDP komunikace. Komunikace ICMP byla vytvářena pomocí příkazu *ping*. Dále pro vytváření souborů ve formátu pcap obsahující zachycenou komunikace byl použit nástroj *tcpdump*[20]. Flow kolektor zachycující exportované záznamy byl vytvořen pomocí nástroje *nfcapd*[15] a získaná data byla zobrazena pomocí nástroje *nfdump*[16]. Jako referenční aplikace sloužil především program *softflowd*[5]. Byly ovšem také využity nástroje *nProbe*[14] a *nfpcapd*[17].

## 6.1 Popis testování

Soubor ve formátu pcap byl vytvořen pomocí nástroje *tcpdump* například následujícím příkazem:

```
# tcpdump -i any -c 2000 -w tcp_udp_2000.pcap
```

Dále byl vytvořen NetFlow kolektor pomocí nástroje *nfcapd*[15] následujícím příkazem:

```
# sudo nfcapd -b 127.0.0.1 -D -T all -l netflow_results/ -I any -S 2 -p 2055 -t 10
```

Zrušení vytvořeného NetFlow kolektory je možné pomocí příkazu:

```
# sudo kill -9 'sudo lsof -t -i:2055'
```

Ověřit, zda je NetFlow kolektor spuštěn lze pomocí následujících příkazů s využitím utilit *netstat*[7] cite a *ps*[1]:

```
# netstat -n --udp --listen
```

```
# ps -ef | grep nfcapd
```

NetFlow exportér lze vytvořit pomocí příkazu make a následně spustit s případnými argumenty:

```
# make
```

```
# ./flow -f test_pcap_files/own/icmp_10.pcap
```

Vytvořené vlastní soubory ve formátu pcap lze nalézt na následujícím odkazu: https://drive.google.com/drive/folders/1DVWpOocwXnVylu7TYcSciOP-9ZBUpSA2?usp=share_link

Kromě vlastních vytvořených souborů ve formátu pcap byly také použity soubory ve stejné formátu poskytnuté v kurzu ISA na VUT FIT v roce 2022 [13] a soubory pro testování volně sdílené mezi studenty opět ve stejném formátu pcap.

Dále jsou pro ukázku uvedeny některé z testovaných případů.

*Pozn.: V následujících ukázkách testování jsou použity pouze vlastní vytvořené soubory ve formátu pcap.*

## 6.2 ICMP komunikace

ICMP komunikace byla vytvořena testovány pomocí příkazu *ping*. Konkrétně bylo provedeno testování na IP adresu *8.8.8.8*, která je adresou primárního DNS serveru pro Google DNS[8]. Pro testování byl využit jako referenční nástroj program softflowd[5]. Při testování byly referenční program a vlastní program spuštěny následujícími příkazy:

```
# sudo softflowd -v 5 -n 127.0.0.1:2055 -r test_pcap_files/own/icmp_10.pcap
```

```
# ./flow -f test_pcap_files/own/icmp_4.pcap
```

Data byla zobrazena následujícím příkazem s využití nástoje *nfdump*[16]:

```
# nfdump -r nfcapd.xxxxxxxxxxxxx
```

kde za *xxxxxxxxxxxx* je dosazeno datum a čas vytvoření souboru.

**Zachycené toky exportované pomocí nástroje softflowd[5]:**

```
Date first seen          Event  XEvent Proto      Src IP Addr:Port          Dst IP Addr:Port     X-Src IP Addr:Port        X-Dst IP Addr:Port   In Byte Out Byte
2022-11-13 21:22:06.389 INVALID  Ignore ICMP     192.168.50.54:0     ->         8.8.8.8:8.0           0.0.0.0:0     ->         0.0.0.0:0           84        0
2022-11-13 21:22:06.399 INVALID  Ignore ICMP           8.8.8.8:0     ->    192.168.50.54:0.0          0.0.0.0:0     ->         0.0.0.0:0           84        0
Summary: total flows: 2, total bytes: 168, total packets: 2, avg bps: 134400, avg pps: 200, avg bpp: 84
Time window: 2022-11-13 21:22:06 - 2022-11-13 21:22:06
Total flows processed: 2, Blocks skipped: 0, Bytes read: 288
Sys: 0.001s flows/second: 1805.1     Wall: 0.000s flows/second: 18348.6
```

Obrázek 3: Testování ICMP - softflowd

**Zachycené toky exportované pomocí vlastního nástroje:**

```
Date first seen          Event  XEvent Proto      Src IP Addr:Port          Dst IP Addr:Port     X-Src IP Addr:Port        X-Dst IP Addr:Port   In Byte Out Byte
2022-11-13 21:22:06.390 INVALID  Ignore ICMP     192.168.50.54:0     ->         8.8.8.8:8.0           0.0.0.0:0     ->         0.0.0.0:0           84        0
2022-11-13 21:22:06.399 INVALID  Ignore ICMP           8.8.8.8:0     ->    192.168.50.54:0.0          0.0.0.0:0     ->         0.0.0.0:0           84        0
Summary: total flows: 2, total bytes: 168, total packets: 2, avg bps: 149333, avg pps: 222, avg bpp: 84
Time window: 2022-11-13 21:22:06 - 2022-11-13 21:22:06
Total flows processed: 2, Blocks skipped: 0, Bytes read: 288
Sys: 0.001s flows/second: 1848.4     Wall: 0.000s flows/second: 16806.7
```

Obrázek 4: Testování ICMP - vlastní nástroj

## 6.3 TCP, UDP a ICMP komunikace

Při testování filtrování TCP komunikace byla využita práce s prohlížečem Firefox 97.0 (64 bit)[6], kdy při komunikaci se nejčastěji používají UDP nebo TCP protokoly. Zároveň pomocí příkazu *ping 8.8.8.8* byla generována ve stejný čas ICMP komunikace. Pro testování byl využit jako referenční nástroj program softflowd[5]. Při testování byly referenční program a vlastní program spuštěny následujícími příkazy:

```
# sudo softflowd -v 5 -n 127.0.0.1:2055 -r test_pcap_files/own/tcp_udp_icmp_100.pcap
```

```
# ./flow -f test_pcap_files/own/tcp_udp_icmp_100.pcap
```

Data byla zobrazena následujícím příkazem s využití nástoje *nfdump*[16]:

```
# nfdump -r nfcapd.xxxxxxxxxxxxx
```

kde za *xxxxxxxxxxxxx* je dosazeno datum a čas vytvoření souboru.

**Zachycené toky exportované pomocí nástroje softflowd[5]:**

```
Date first seen          Event  XEvent Proto      Src IP Addr:Port          Dst IP Addr:Port     X-Src IP Addr:Port        X-Dst IP Addr:Port   In Byte Out Byte
2022-11-13 21:14:27.094 INVALID  Ignore ICMP     192.168.50.54:0     ->         8.8.8.8:8.0           0.0.0.0:0     ->         0.0.0.0:0           84        0
2022-11-13 21:14:27.104 INVALID  Ignore ICMP          8.8.8.8:0     ->   192.168.50.54:0.0           0.0.0.0:0     ->         0.0.0.0:0           84        0
2022-11-13 21:14:27.962 INVALID  Ignore UDP      192.168.50.54:47855 ->  74.125.108.166:443           0.0.0.0:0     ->         0.0.0.0:0         3013        0
2022-11-13 21:14:27.975 INVALID  Ignore UDP     74.125.108.166:443   ->   192.168.50.54:47855         0.0.0.0:0     ->         0.0.0.0:0        90249        0
Summary: total flows: 4, total bytes: 93430, total packets: 100, avg bps: 835128, avg pps: 111, avg bpp: 934
Time window: 2022-11-13 21:14:27 - 2022-11-13 21:14:27
Total flows processed: 4, Blocks skipped: 0, Bytes read: 448
Sys: 0.000s flows/second: 4319.7     Wall: 0.000s flows/second: 34482.8
```

Obrázek 5: Testování TCP, UDP a ICMP - softflowd

**Zachycené toky exportované pomocí vlastního nástroje:**

```
Date first seen          Event  XEvent Proto      Src IP Addr:Port          Dst IP Addr:Port     X-Src IP Addr:Port        X-Dst IP Addr:Port   In Byte Out Byte
2022-11-13 21:14:27.094 INVALID  Ignore ICMP     192.168.50.54:0     ->         8.8.8.8:8.0           0.0.0.0:0     ->         0.0.0.0:0           84        0
2022-11-13 21:14:27.104 INVALID  Ignore ICMP          8.8.8.8:0     ->   192.168.50.54:0.0           0.0.0.0:0     ->         0.0.0.0:0           84        0
2022-11-13 21:14:27.962 INVALID  Ignore UDP      192.168.50.54:47855 ->  74.125.108.166:443           0.0.0.0:0     ->         0.0.0.0:0         3013        0
2022-11-13 21:14:27.975 INVALID  Ignore UDP     74.125.108.166:443   ->   192.168.50.54:47855         0.0.0.0:0     ->         0.0.0.0:0        90249        0
Summary: total flows: 4, total bytes: 93430, total packets: 100, avg bps: 835128, avg pps: 111, avg bpp: 934
Time window: 2022-11-13 21:14:27 - 2022-11-13 21:14:27
Total flows processed: 4, Blocks skipped: 0, Bytes read: 448
Sys: 0.001s flows/second: 3795.1     Wall: 0.000s flows/second: 30534.4
```

Obrázek 6: Testování TCP, UDP a ICMP - vlastní nástroj

## 6.4 Testování argumentů programu a nastavení výchozích hodnot

Dále bylo prováděno také testování příslušných argumentů programu. Nejprve pomocí testovacích výpisů a pomocí ladění bylo otestováno správné nastavení hodnot parametrů jak pro uživatelsky zadané argumenty, tak i pro výchozí hodnoty. Dále pomocí programu Wireshark[25], testovacích výpisů a pomocí ladění byla ověřena funkčnost jednotlivých parametrů programu (*aktivní časovač*, *neaktivní časovač* a *velikost mezipaměti*). Pro vytvoření Flow kolektoru a zobrazení obdržených dat byly využity stejné nástroje jako v předchozích příkladech.

# Použitá literatura

[1] Branko Lankester, Michael K. Johnson, Michael Shields, Charles Blake, David Mossberger-Tang, Albert Cahalan: ps(1) — Linux manual page. [online], [vid. 2022-10-17]. Dostupné z: https://man7.org/linux/man-pages/man1/ps.1.html

[2] Cisco: NetFlow Commands: cache through top. [online], [vid. 2022-10-2]. Dostupné z: https://www.cisco.com/en/US/docs/ios/12_3t/netflow/command/reference/nfl_a1gt_ps5207_TSD_Products_Command_Reference_Chapter.html

[3] Cisco: NetFlow Export Datagram Format. [online], rev. září 2007, [vid. 2022-10-15]. Dostupné z: https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html#wp1006108

[4] Damien Miller: [netflow-tools] Softflowd patches for ICMP type/code and DESTDIR support. [online], rev. březen 2006, [vid. 2022-11-10]. Dostupné z: https://marc.info/?l=netflow-tools&m=139653872523808&w=2

[5] Damien Miller, Hitoshi Irino (current maintainer): softflowd — Traffic flow monitoring. [online], [vid. 2022-10-15]. Dostupné z: https://manpages.ubuntu.com/manpages/jammy/man8/softflowd.8.html

[6] Firefox developers: Firefox Browser. [online], rev. 2022, [vid. 2022-03-23]. Dostupné z: https://www.mozilla.org/cs/firefox/

[7] Fred Baumgarten: netstat(8) - Linux man page. [online], [vid. 2022-10-17]. Dostupné z: https://linux.die.net/man/8/netstat

[8] Google Public DNS manual pages authors: *Google Public DNS*. rev. 2020, [vid. 2022-03-23]. Dostupné z: https://developers.google.com/speed/public-dns/docs/using

[9] Linux manual pages authors: *Linux manual pages*. rev. březen 2021, [vid. 2022-10-25]. Dostupné z: https://man7.org/linux/man-pages/man2/socket.2.html

[10] Linux manual pages authors: *Linux manual pages*. rev. březen 2021, [vid. 2022-10-25]. Dostupné z: https://man7.org/linux/man-pages/man3/gethostbyname.3.html

[11] Linux manual pages authors: *Linux manual pages*. rev. 2021, [vid. 2022-03-23]. Dostupné z: https://man7.org/linux/man-pages/man3/getopt.3.html

[12] Matoušek Petr, doc. Ing., Ph.D., M.A.: echo-udp-client2.c. Sdílené soubory pro kurz ISA na VUT FIT, 2022.

[13] Matoušek Petr, doc. Ing., Ph.D., M.A.: Síťové aplikace a správa sítí. Univerzitní kurz, 2022.

[14] ntop: nProbe. [online], [vid. 2022-10-17]. Dostupné z: https://www.ntop.org/products/netflow/nprobe/

[15] Peter Haag: nfcapd - netflow capture daemon. [online], [vid. 2022-10-17]. Dostupné z: https://manpages.ubuntu.com/manpages/jammy/man1/nfcapd.1.html

[16] Peter Haag: nfdump. [online], [vid. 2022-10-17].
Dostupné z: https://github.com/phaag/nfdump/tree/c45be96a9938a183f4f34a744fbee789573a8bee

[17] Peter Haag: nfdump. [online], [vid. 2022-10-18].
Dostupné z: https://github.com/phaag/nfdump/tree/c45be96a9938a183f4f34a744fbee789573a8bee

[18] Progress Software Corporation: IPFIX and NetFlow Monitoring. [online], [vid. 2022-11-10].
Dostupné z: https://www.flowmon.com/en/solutions/network-and-cloud-operations/netflow-ipfix

[19] Steve Petryschuk: NetFlow Basics: An Introduction to Monitoring Network Traffic. [online], rev. březen 2019, [vid. 2022-11-10].
Dostupné z: https://www.auvik.com/franklyit/blog/netflow-basics/

[20] The Tcpdump Group: TCPDUMP & LIBPCAP. [online], [vid. 2022-10-12].
Dostupné z: https://www.tcpdump.org/

[21] The Tcpdump Group: *pcap_open_offline(3PCAP) man page*. rev. srpen 2018, [vid. 2022-10-20].
Dostupné z: https://www.tcpdump.org/manpages/pcap_open_offline.3pcap.html

[22] The Tcpdump Group: *pcap_next_ex(3PCAP) man page*. rev. březen 2022, [vid. 2022-10-22].
Dostupné z: https://www.tcpdump.org/manpages/pcap_next_ex.3pcap.html

[23] Wikipedia komunita: Binary search tree. [online], rev. říjen 2022, [vid. 2022-11-1].
Dostupné z: https://en.wikipedia.org/wiki/Binary_search_tree

[24] Wikipedia komunita: NetFlow. [online], rev. říjen 2022, [vid. 2022-11-10].
Dostupné z: https://en.wikipedia.org/wiki/NetFlow

[25] Wireshark komunita: Wireshark. [online], rev. 2022, [vid. 2022-03-22].
Dostupné z: https://www.wireshark.org/