VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

PŘEDMĚT ISA

APLIKACE PRO ZÍSKÁNÍ STATISTIK O SÍŤOVÉM PROVOZU

ŠKOLNÍ PROJEKT

AUTOR PROJEKTU **SAMUEL HEJNÍČEK**

VEDOUCÍ PROJEKTU **Ing. MATĚJ GRÉGR, Ph.D.**

BRNO 2024

Obsah

Kapitola 1

Úvod

Tento text slouží jako dokumentace k projektu do předmětu ISA akademického roku 2024/2025, jehož tématem je vypisování statistik pro jednotlivá komunikující spojení, Program je napsán v jazyce C++ standardu C++20 a určen pro operační systém Linux. Pro správné fungování je nutné jej spouštět s oprávněním administrátora.

1.1 Stručný popis programu

Program zachytává jednotlivé síťové pakety, na základě kterých utváří síťová spojení (přes které byly pakety přeneseny) a tato spojení si ukládá. Program podporuje IPv4 i IPv6 spojení. Mezi ukládané vlastnosti patří komunikující IP adresy, porty, protokol a dále parametry získané z paketu jako je velikost paketu a počet přenesených dat. V každém daném časovém intervalu (ve výchozím stavu 1 vteřina) jsou jednotlivá komunikující spojení včetně jejich statistik v podobě přenesených bytů a počtů paketů vypsána na standardní výstup (ve výchozím stavu jsou seřazená dle počtu přenesených bytů). Program vždy vypisuje aktuálně komunikující spojení, která průběžně aktualizuje. Program končí stisknutím klávesové zkratky CTRL+C.

1.2 Spouštění programu

Ke kompilaci programu je nutné v kořenové složce zavolat příkaz **make**, který vytvoří spustitelný soubor **isa-top**. Ten lze spustit s následujícími parametry:

Argument	Hodnota	Význam	Popis
`-i`	Od uživatele/chybí*	Název rozhraní	Rozhraní, na kterém bude program zachytávat pakety.
`-s`	b nebo p	Seřazení výstupu	Seřadí výpis jednotlivých spojení (sestupně) dle hodnoty. Pokud je zadáno písmeno b, jsou spojení vypsána dle množství přenesených bytů, v případě písmena p dle počtu přenesených paketů.
`-t`	Od uživatele/chybí**	Časový interval aktualizace	Specifikuje interval, ve kterém bude docházet k aktualizaci statistik síťových spojení.
`-l`		Výpis síťových rozhraní	Při zapnutí programu vypíše dostupná síťová rozhraní pro záchyt paketů a ukončí program.
`-h`		Nápověda	Vypíše nápovědu a ukončí program.

Tabulka 1.1: Přehled vstupních argumentů

Na pořadí zadaných argumentů programu nezáleží. Časový interval aktualizace lze zadat jako celé kladné číslo. Argument -i včetně korektní hodnoty je povinný.

Pozn. * Pokud není specifikováno rozhraní, je program ukončen s chybou. Pro výpis dostupných rozhraní lze použít přepínač -l.

Pozn. ** Pokud není zadána hodnota od uživatele, je výchozí doba aktualizace 1 vteřina.

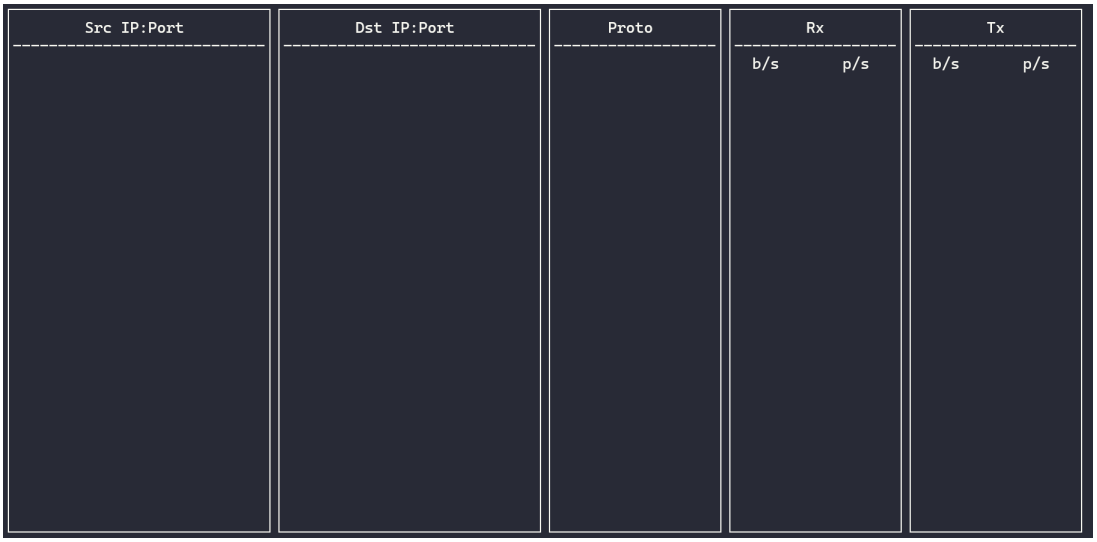
Příklad spouštění programu (program pracuje na rozhraní eth0, k aktualizaci dochází každých 5 vteřin a statistiky jsou vypisované dle počtu přenesených paketů):
./isa-top -i eth0 -s p -t 5

Pro smazání objektových souborů a spustitelného souboru programu lze využít příkaz **make clean**.

1.3 Výstup programu

Výstup programu v podobě jednotlivých spojení včetně jejich statistik je realizován pomocí knihovny ncurses. [5]

Hlavní okno je rozděleno na 5 sloupců, ve kterých je ve směru zleva doprava zobrazena zdrojová komunikující IP adresa a port, cílová IP adresa a port (v případě, že port má hodnotu nula není k žádné z IP adres vypsán), protokol a následně statistiky, kdy sloupec Tx označuje statistiky pro komunikaci směrem od zdrojové adrese k cílové a sloupec Rx naopak. Oba sloupce zobrazují jak počet přenesených paketů, tak počet přenesených bytů za sekundu (vždy probíhá přepočet dle aktuálně zvoleného intervalu). Velikost paketu je získána z proměnné **pkthdr->len**. Statistiky jsou zaokrouhlované na jedno desetinné místo. Pro účely přepočtu bytů na dané jednotky rozumíme, že 1 kilobyte má 1000 bytů. Výstup zobrazuje obrázek níže.



Obrázek 1.1: Výstup programu pomocí knihovny ncurses

Kapitola 2

Stručná teorie k programu

2.1 Síťový model a jeho vztah k programu

Dle modelu ISO/OSI lze síťovou komunikaci rozdělit do sedmi vrstev. Dle modelu TCP/IP do 4 vrstev. V našem případě můžeme zůstat u modelu TCP/IP - nezajímá nás prezentační ani relační vrstva. Mezi důležizé vrstvy pro síťový analyzátor patří druhá vrstva (L2), tedy linková, třetí vrstva označovaná jako síťová (L3) a čtvrtá vrstva označená jako transportní (L4). Data, která chce uživatel poslat po síti, prochází zapouzdřením, kdy na každé síťové vrstvě je přidána hlavička odpovídajícího protokolu (může jít například o TCP hlavičku či IP hlavičku), což se liší dle typu posílaných dat respektive způsobu komunikace. Tento proces postupuje vzhledem k modelu shora dolů, tedy od aplikační vrstvy po fyzickou vrstvu. [10] [11] [1].

Síťový analyzátor implementovaný v tomto programu zachycuje pakety, základní jednotky dat přenášené po síti, u kterých postupuje opačným směrem, tedy dochází k rozbalování jednotlivých hlaviček a výčtu důležitých informací. Nejprve je tedy rozbalena hlavička Ethernetu (na 2. síťové vrstvě) a následně je rozbalena hlavička IP, ze které lze například vyčíst zdrojovou a cílovou IP adresu paketu. Toto postupné rozbalování je důležité, jelikož ne všechny informace se nacházejí ve stejné hlavičce (porty transportních protokolů nenajdeme v hlavičce IP). [16]

2.2 Knihovna Libcap

Jako knihovna pro zachytávání paketů síťovým analyzátorem byla zvolena knihovna Libcap zajišťující vysokoúrovňové rozhraní a umožňující i odchyt paketů, které nejsou cílené přímo pro daný počítač a také jejich čtení/ukládání do souboru. [6]

Mezi podporované protokoly patří:

- **TCP** - Protokol transportní vrstvy, který zajišťuje spojově orientovaný a spolehlivý přenos dat. Aplikace komunikující přes TCP posílá proud bytů rozdělený do segmentů. Tato data jsou na zařízení přijata ve stejném pořadí jako byla odeslána. Spojení je navázáno pomocí tří stupňového "handshake"mechanismu. [13] [4]

- **UDP** - Protokol transportní vrstvy, který nezajišťuje spolehlivý přenos dat. U tohoto protokolu se bez pomocných mechanismů nelze dozvědět, že data protistraně dorazila v pořádku a ani není zajištěno, že přijdou ve správném pořadí. [14]
- **ICMPv4 a ICMPv6** - Komunikační protokoly, které slouží primárně k odesílání chybových zpráv popřípadě zpráv o úspešné komunikaci mezi zařízeními (typicky se tyto zprávy generují pomocí příkazu ping). ICMPv6 navíc zajišťuje podporu Multicastu pro IPv6 (typ MLD) nebo podporu autokonfigurace (typ NDP). [8] [9]

2.3 Knihovna ncurses

Jako knihovna pro výpis informací o spojeních a jejich statistikách byla zvolena knihovna ncurses, která slouží jako "wrapper"nad prací s terminálem, kdy uživateli poskytuje vysoce flexibilní API zahrnující funkce jako práce s kurzorem, vytvoření menších oken v rámci hlavního okna nebo práce s barvami textu. [5]

2.4 Existující nástroje

2.4.1 Nástroj iftop

Nástroj iftop stejně jako isa-top zachycuje provoz na specifikovaném síťovém rozhraní a zobrazuje tabulku s šířkou pásma zabraného dvojící komunikujících zařízení. [7]

2.4.2 Nástroj Wireshark

Nástroj Wireshark se řadí mezi pokročilé nástroje pro monitorování komunikace na síťovém rozhraní v reálném čase. Program podporuje až stovky protokolů, je multiplatformní a obsahuje přehledné a obsáhlé grafické rozhraní. [15]

Kapitola 3

Návrh a implementace programu

3.1 Návrh programu a rozdělení do souborů

Z důvodu volby programovacího jazyka C++ se návrh snaží o objektově orientované rozvržení programu, tedy aby celky programu zajišťující nějakou funkční část byly rozdělené do tříd, které spolu komunikují.

Jednotlivé soubory programu jsou:

- soubory "cliParser.cpp"a "cliParser.hpp", obsahující třídu pro zpracování a validaci vstupních argumentů.
- soubory "packetSniffer.cpp"a "packetSniffer.hpp", obsahující třídu, která slouží pro zachytávání paketů a extrakci informací z nich
- soubory "packetDisplay.cpp"a "packetDisplay.hpp", obsahující třídu, která slouží k výpisu spojení a jeho statistik na standardní výstup
- soubory "connectionManager.cpp"a "connectionManager.hpp", obsahující třídu, která ukládá a spravuje jednotlivá spojení
- soubory "exceptions.cpp"a "exceptions.hpp", definující vlastní výjimky v programu vyvolané při vzniku nějaké chybové události
- soubory "main.cpp"a "main.hpp", definující hlavní logiku programu a jeho ukončení

3.2 Vlastní implementace

3.2.1 Hlavní logika

Program je realizován jako vícevláknový. Hlavní vlákno obsahuje cyklus while, ve kterém je obstarávána hlavní logika programu a sekundární vlákno využívá síťový čmuchač, který bez ohledu na stav hlavního vlákna zajišťuje zachytávání a zpracování paketů.

V hlavním vláknu dochází k vytvoření instancí jednotlivých tříd a zpracování argumentů, následně je vytvořeno vlákno síťového čmuchače, který začíná pracovat. Poté se vlákno dostane ke hlavní logice programu, kdy v cyklu while načte spojení uložená v mapě do vektoru, který je předán třídě pro výpis spojení. Tato spojení jsou následně v čitelném

formátu (viz kapitola výstup programu) vypsána na obrazovku. Vektor je následně smazán a hlavní vlákno spí po dobu danou intervalem aktualizace spojení (vlákno síťového analyzátoru mezitím ukládá nová spojení do mapy k budoucímu výpisu).

3.2.2 Zpracování vstupů

Zpracování vstupních argumentů probíhá ve třídě **"cliParser"**, kde figuruje funkce getopt, která umožňuje zadávání argumentů v libovolném pořadí a umožňuje automatickou kontrolu existence povinné hodnoty argumentu. Pokud nebyly zadány některé volitelné paremetry jako typ třídění nebo interval aktualizace, jsou použity výchozí hodnoty.

3.2.3 Zachytávání paketů a jejich zpracování

Zachytávání paketů zajišťuje třída **"packetSniffer"**. Síťový "čmuchač", který je instancí této třídy otevře síťové rozhraní pro zachytávání paketů a následně začně pakety v cyklu zachytávat. Toto zachytávání probíhá do doby, než uživatel zadá CTRL+C. Každý jednotlivý paket je poslán ke zpracování metodě **packetParser**, která ověří, že jde o paket, který je pro náš projekt zajímavý (IPv4 a IPv6 pakety s protokolem TCP, UDP, ICMPv4 a ICMPv6) a informace uložené z paketů uloží do struktury reprezentující nové spojení.

3.2.4 Tvorba a aktualizace spojení

Správu existujících spojení zajišťuje třída **"connectionManager"**. Do její instance se snaží vložit nové spojení síťový čmuchač po každém novém zachyceném paketu pomocí metody **addConnection**. Z důležitých prvků spojení je utvořen unikátní klíč (IP adresy, porty a protokol), kterým je dané spojení vyhledáváno v mapě. Pokud bylo spojení nalezeno, dojde pouze k aktualizaci statistik, v opačném případě je přidáno nové spojení. Spojení je také vyhledáváno "opačně", tedy s přehozenými IP adresami a porty, aby byly statistiky aktualizovány správně vzhledem ke směru toku. Po uplynutí intervalu aktualizace jsou vždy pomocí metody **parseConnectionVector** všechny spojení z mapy extrahovány do vektoru, odkud s nimi může pracovat třída pro výpis spojení. Mapa je následně smazána a připravena na nová spojení.

3.2.5 Výpis statistik

Výpis statistik realizuje třída **"packetDisplay"**. Metoda **windowRefresh**, která je volána v hlavním cyklu programu zajišťuje překreslení hlavního okna včetně menších oken vždy s aktualizovanými statistikami. Mezi další pomocné metody patří **printTextCenter**, která daný text vždy vytiskne uprostřed specifikovaného okna nebo metoda **printVectorConnections**, která v oknech zobrazí jednotlivé informace o spojeních a zajistí správnou interpretaci počtu bytů a paketů ve formátu, který je pro člověka snadno čitelný. Po výpisu statistik v okně je vektor spojení smazán, aby do něj následně mohla být vložena´nová spojení k výpisu.

Kapitola 4

Testování aplikace

Veškeré testování probíhalo na systému Linux Ubuntu s administrátorskými právy.

4.1 Validace vstupních argumentů

Začátek testování proběhl hned při spouštění programu, kdy byla otestována reakce programu na neočekávané argumenty a také zkontrolován výpis chybových hlášení v případě zadání špatného argumentu nebo pokud došlo k chybě v průběhu práce programu (např. nenalazení daného síťového rozhraní).

4.1.1 Vstupní argumenty programu

Jednotlivě zadané vstupy programu byly:

- ./isa-top -i rozhranicko reprezentující zadání nevalidního jména rozhraní
- ./isa-top -i eth0 -s b -s p reprezentující zadání obou parametrů pro třídění naráz
- ./isa-top -i eth0 -t -5 reprezentující spuštění programu se záporným časovým intervalem aktualizace

4.1.2 Očekávané výstupy programu

Ve všech třech případech je očekáváno ukončení programu s vypsánim specifické chybové hlášky na standardní chybový výstup.

4.1.3 Reálné výstupy programu

Výstupy programu po provedení testovacích příkazů pro spuštění jsou tyto:

- ERROR [PCAP_OPEN_LIVE] Interface name rozhranicko: No such device exists
- ERR: You can sort only by 1 parameter at once
- ERR: Time interval must be positive a number

Ze všech výstupů lze vidět, že program zareagoval adekvátně na všechny nestandardní vstupy.

4.2 Test počtu vypisovaných vypisovaných paketů

4.2.1 Příkaz ping

Jedním z prvních způsobů testování bylo použití příkazu ping, který ověřuje dostupnost zařízení na IP síti. Použití tohoto příkazu bylo zvoleno z důvodu neustáleho přenosu stejného množství paketů mezi oběma stranami. [12]

Vstup testu

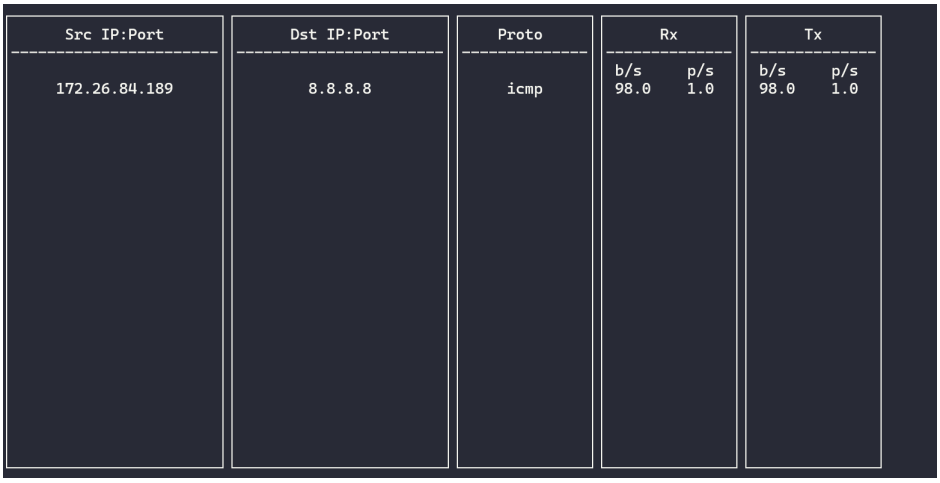
Pro ověření IPv4 konektivity byl zadán příkaz **ping 8.8.8.8**, který posílá zprávy ICMP typu Echo Request na DNS servery společnosti Google. Program byl spuštěn pomocí příkazu **./isa-top eth0**, tedy pakety byly zachytávány na rozhraní eth0 s intervalem aktualizace jedné sekundy. Pro oveření IPv6 konektivity byl zadán přkaz **ping ::1**, který posílá zprávy ICMP typu Echo Request na lokální rozhraní loopback. Program byl spuštěn pomocí stejného příkazu jako pro v minulém případě.

Očekávaný výstup testu

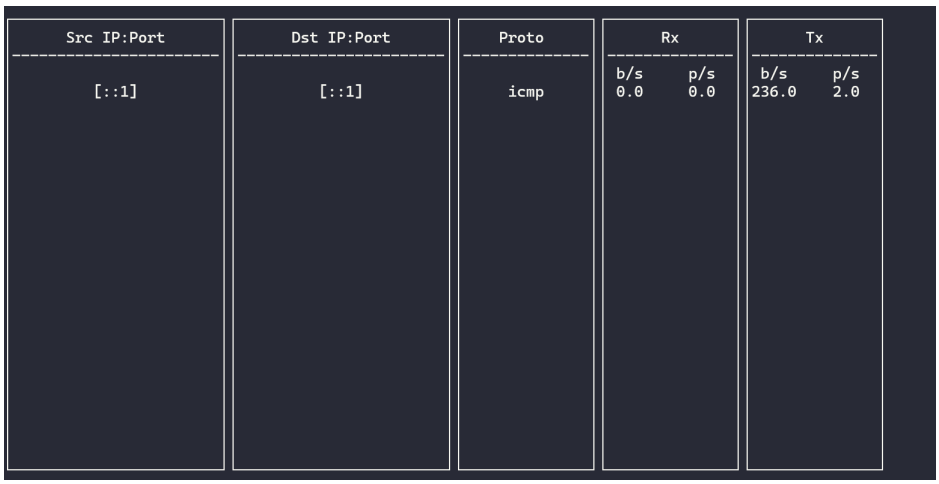
Vzhledem k hodnotě intervalu aktualizace příkazu ping, který je shodný s intervalem pro opětovné zaslání zprávy ICMP, lze při dosažení konektivity očekávat na výpisu programu spojení obsahující jeden paket poslaný směrem k zařízení, jehož dostupnost chceme zjistit (typ Echo Request) a jeden přijatý paket (typ Echo Reply), který potvrzuje dostupnost konektivity vůči cílovému zařízení. V obou případech by velikost paketu reprezentující spojení měla být stejná.

Reálný výstup testu

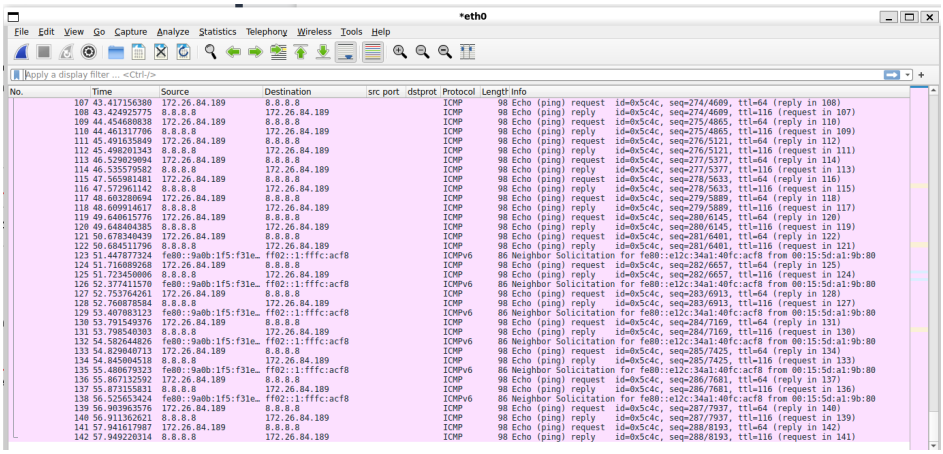
Výsledek testování jak pro IPv4 tak pro IPv6 spojení zachycují obrázky níže. V případě IPv4 se test úspěšně podařil a v okně lze vidět spojení obsahující jeden přenesený paket vždy v daném směru se stejnou velikostí. Ve výstupu nelze vidět porty, jelikož protokol ICMP porty nepoužívá (je na třetí síťové vrstvě). U IPv6 lze rovněž hovořit o úspěšném testu, jediným rozdílem z důvodu použití loopback rozhraní je existence obou paketů v kolonce Tx, jelikož kvůli identickým IP adresám a protokolům se programu jeví komunikace tohoto spojení jako jednostranná.
Pro IPv4 verzi je ještě přiložen snímek z programu Wireshark, kde je patrná komunikace ICMP probíhající každou vteřinu s velikostí paketu 98 bytů, která je shodná s velikostí vypsanou programem na výstup.



Obrázek 4.1: Výstup testu pro IPv4 spojení



Obrázek 4.2: Výstup testu pro IPv6 spojení



Obrázek 4.3: Kontrola výpisu pomocí programu Wireshark

4.2.2 Python skript využívající knihovnu scapy

Pro testování většího počtu přenesených paketů byla využita knihovna **Scapy**, která umožňuje vytvoření a poslání libovolného množství paketů s podporou všech klíčových protokolů [3]

Test výpisu počtu přenesených paketů

Vstup testu

Program byl spuštěn příkazem **./isa-top -i eth0**, tedy aby zachytával pakety na rozhraní ethernetu. Testovací python skript měl pouze dva funkční řádky. Jeden vytváří paket se zvolenou zdrojovou a cílovou IP adresou (pro účely testu 1.2.3.4 a 5.6.7.8) se zvolenými porty (4567 a 4568) a protolem TCP. Na dalším řádku je celkem 100 těchto paketů odesláno. Skript byl spuštěn příkazem **python3 script.py** a jeho zdrojový kód je na obrázku níže.

```
if __name__ == "__main__":
    pkt = IP(src="1.2.3.4", dst="5.6.7.8" ) / TCP(sport=4567, dport=4568)
    send(pkt, count=100)
```

Obrázek 4.4: Python skript pro odeslání paketů

Očekávaný výstup testu

V případě tohoto testu je očekáván výstup v podobě spojení obsahující IP adresy a porty zmíněné ve vstupu testu a u tohoto spojení je také očekáváno odeslání 100 paketů od IP adresy 1.2.3.4. Množství dat v tomto testu není podstatné.

Reálný výstup testu

Obrázek níže dokládá výsledek testu, který je uspokojivý. Na výstupu programu **isa-top** lze vidět že ze zdrojové IP adresy 1.2.3.4 bylo posláno celkem 100 paketů směrem k cílové adrese 5.6.7.8, což bylo cílem tohoto testu.

Src IP:Port	Dst IP:Port	Proto	Rx		Tx	
			b/s	p/s	b/s	p/s
1.2.3.4:4567	5.6.7.8:4568	tcp	0.0	0.0	5.4K	100.0

Obrázek 4.5: Výstup testu objemu přenesených dat

4.3 Test výpisu objemu dat

4.3.1 Příkaz hping3

Pro testování správného výpisu objemu přenesených dat byl zvolen síťový nástroj hping3 umožňující posílat vlastní TCP/UDP/ICMP pakety, na které cíl reaguje stejně jako by se jednalo o příkaz ping zmiňovaný výše. Volba tohoto nástroje byla zvolena, jelikož umožňuje posílat data o definovaném objemu i definovaném počtu paketů a lze tak sledovat, že program správně zachytí a vypíše velikost a počet přenesených dat. [2]

Vstup testu

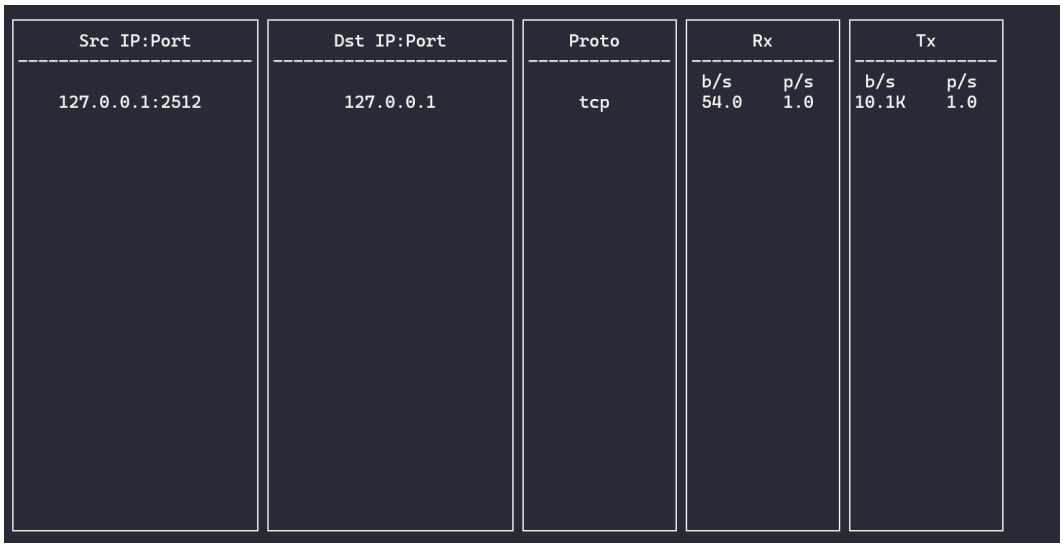
Program byl spuštěn příkazem **./isa-top -i lo**, tedy aby zachytával pakety na rozhraní loopback. Program hping3 byl spuštěn pomocí příkazu **./hping3 -c 5 -d 10000 localhost**.

Očekávaný výstup testu

Vzhledem k zadanému příkazu hping je v programu **isa-top** očekáván výpis spojení mezi nějakými porty na rozhraní loopback, u něhož bude objem data činit asi 10 000 bytů (velikost dat a hlaviček) a tato data budou vypsána po dobu 5 sekund, jelikož program hping posílá těchto paketů celkem 5.

Reálný výstup testu

Výsledek testu se opět ukázal jako úspěšný a na výstupu lze pozorovat komunikaci na loopback rozhraní, při které došlo v poslední vteřině k přenosu 10.1K dat tedy asi 10 kilobytů. Tento výstup byl zobrazen po dobu 5 sekund, nicméně kvůli nemožnosti vložení videa výsledek testu dokládá jeden snímek níže.



Obrázek 4.6: Výstup testu objemu přenesených dat

4.4 Test výpisu s použitím argumentu pro řazení spojení

4.4.1 Použití internetového prohlížeče

Pro testování správného seřazení spojení na standardním výstupu programu byl použit internetový prohlížeč firefox, který při vstupu na různé stránky generuje několik spojení s různým počtem i délkou paketů. V takovém případě lze snadno demonstrovat použití přepínače pro třídění.

Vstup testu

Při testu byl program spuštěn celkem dvakrát, a to následovně:

- **./isa-top -i eth0 -s p**, kdy dochází k řazení na základě počtu paketů vždy dohromady pro Tx a Rx
- **./isa-top -i eth0 -s b**, kdy dochází k řazení na základě počtu přenesených bytů vždy dohromady pro Rx a Tx

Očekávaný výstup testu

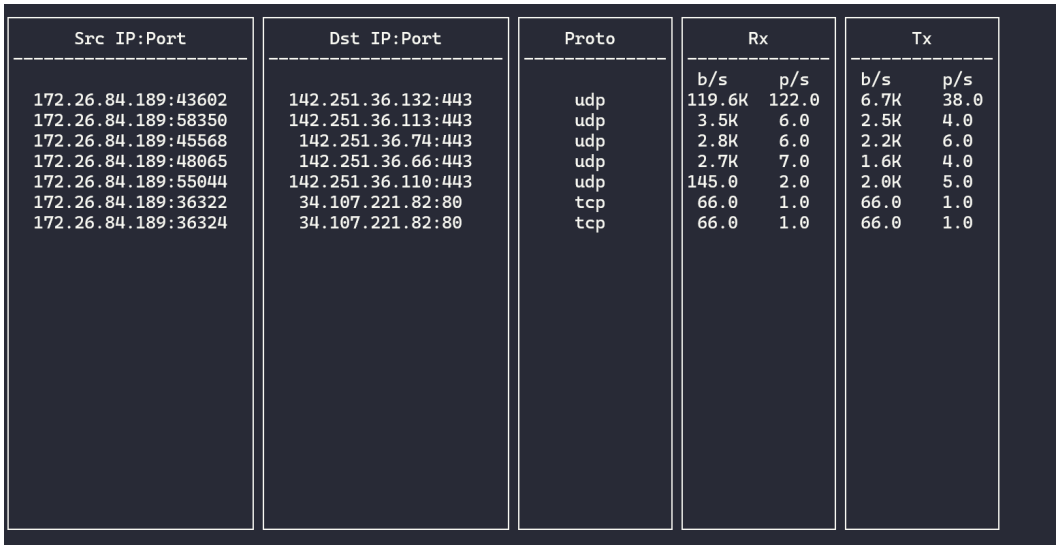
V prvním případě je očekáván seřazený výpis spojení sestupně dle počtu přenesených paketů, v druhém případě dle počtu přenesených bytů.

Reálný výstup testu

Poslední test tohoto programu dopadl opět úspěšně a na obrázcích níže lze v prvním případě vidět korektně seřazené spojení dle počtu paketů a v druhém případě dle počtu bytů.

Src IP:Port	Dst IP:Port	Proto	Rx		Tx	
			b/s	p/s	b/s	p/s
172.26.84.189:45342	157.240.30.54:443	udp	1.8M	1.4K	16.0K	80.0
172.26.84.189:36910	157.240.30.54:443	tcp	135.9K	43.0	5.1K	34.0
172.26.84.189:36896	157.240.30.54:443	tcp	67.1K	32.0	4.8K	34.0
172.26.84.189:36902	157.240.30.54:443	tcp	3.6K	10.0	3.0K	14.0
172.26.84.189:36906	157.240.30.54:443	tcp	3.6K	11.0	2.9K	12.0
172.26.84.189:36996	157.240.30.54:443	tcp	3.6K	10.0	2.9K	13.0
172.26.84.189:36990	157.240.30.54:443	tcp	3.6K	10.0	2.9K	13.0
172.26.84.189:36972	157.240.30.54:443	tcp	3.5K	9.0	2.9K	13.0
172.26.84.189:36988	157.240.30.54:443	tcp	3.6K	10.0	2.9K	12.0
172.26.84.189:36880	157.240.30.54:443	tcp	3.6K	10.0	2.8K	11.0

Obrázek 4.7: Výstup programu se spojeními seřazenými dle množství přenesených paketů



Src IP:Port	Dst IP:Port	Proto	Rx		Tx	
			b/s	p/s	b/s	p/s
172.26.84.189:43602	142.251.36.132:443	udp	119.6K	122.0	6.7K	38.0
172.26.84.189:58350	142.251.36.113:443	udp	3.5K	6.0	2.5K	4.0
172.26.84.189:45568	142.251.36.74:443	udp	2.8K	6.0	2.2K	6.0
172.26.84.189:48065	142.251.36.66:443	udp	2.7K	7.0	1.6K	4.0
172.26.84.189:55044	142.251.36.110:443	udp	145.0	2.0	2.0K	5.0
172.26.84.189:36322	34.107.221.82:80	tcp	66.0	1.0	66.0	1.0
172.26.84.189:36324	34.107.221.82:80	tcp	66.0	1.0	66.0	1.0

Obrázek 4.8: Výstup programu se spojeními seřazenými dle množství přenesených bytů

Literatura

[1] *Referenční model ISO/OSI* online. Dostupné z: https://ijs2.8u.cz/index.php?option=com_content&view=article&id=13&Itemid=119. [cit. 2024-11-17].

[2] *Tool documentation* online. Květen 2024. Dostupné z: https://www.kali.org/tools/hping3/. [cit. 2024-11-17].

[3] Biondi, P. a the Scapy community. *Welcome to Scapy's documentation!* online. 2024. Dostupné z: https://scapy.readthedocs.io/en/latest/. [cit. 2024-11-17].

[4] Khan Academy. *Transmission Control Protocol (TCP)* online. 2024. Dostupné z: https://cs.khanacademy.org/computing/informatika-pocitace-a-internet/x8887af37e7f1189a:internet/x8887af37e7f1189a:tcp-protokol/a/transmission-control-protocol--tcp. [cit. 2024-11-17].

[5] Padala, P. *NCURSES Programming HOWTO* online. 2001. Dostupné z: https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/intro.html#WHATIS. [cit. 2024-11-17].

[6] The Tcpdump Group. *Pcap(3PCAP) man page* online. Září 2023. Dostupné z: https://www.tcpdump.org/manpages/pcap.3pcap.html. [cit. 2024-11-17].

[7] Warren, P. a Lightfoot, C. *Iftop: display bandwidth usage on an interface* online. Leden 2017. Dostupné z: https://pdw.ex-parrot.com/iftop/. [cit. 2024-11-17].

[8] Wikipedia contributors. *ICMPv6* online. Říjen 2024. Dostupné z: https://en.wikipedia.org/wiki/ICMPv6. [cit. 2024-11-17].

[9] Wikipedia contributors. *Internet Control Message Protocol* online. Listopad 2024. Dostupné z: https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol. [cit. 2024-11-17].

[10] Wikipedia contributors. *Internet protocol suite* online. Listopad 2024. Dostupné z: https://en.wikipedia.org/wiki/Internet_protocol_suite. [cit. 2024-11-17].

[11] Wikipedia contributors. *OSI model* online. Listopad 2024. Dostupné z: https://en.wikipedia.org/wiki/OSI_model. [cit. 2024-11-17].

[12] Wikipedia contributors. *Ping (networking utility)* online. Listopad 2024. Dostupné z: https://en.wikipedia.org/wiki/Ping_(networking_utility). [cit. 2024-11-17].

[13] Wikipedia contributors. *Transmission Control Protocol* online. Listopad 2024. Dostupné z: https://en.wikipedia.org/wiki/Transmission_Control_Protocol. [cit. 2024-11-17].

[14] Wikipedia contributors. *User Datagram Protocol* online. Listopad 2024. Dostupné z: https://en.wikipedia.org/wiki/User_Datagram_Protocol. [cit. 2024-11-17].

[15] Wireshark foundation. *About Wireshark* online. Dostupné z: https://www.wireshark.org/about.html. [cit. 2024-11-17].

[16] Zaoral, K. *Lekce 5 - Sítě - Přenos informací (paketů)* online. Dostupné z: https://www.itnetwork.cz/site/zaklady/site-prenos-informaci-paketu. [cit. 2024-11-17].