VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií

Technická zpráva

Aplikace pro získání statistik o síťovém provozu

18. listopadu 2024 Adam Valík, xvalik05

Obsah

1 Uvedení do problematiky

Na fungování počítačových sítí je v dnešní době závislé téměř vše. Běžný člověk si nedokáže ani představit, jaké následky by nesl výpadek způsobený například lidskou chybou nebo kybernetickým útokem. Následující odstavce popisují v kostce související problematiku a shrnují motivaci vytváření aplikací získávajících statistiky o síťovém provozu.

1.1 Analýza síťového provozu

Monitorování sítě zahrnuje sledování statistik síťového provozu za účelem optimalizace výkonu, zajištění bezpečnosti a udržení provozu a dostupnosti. Síť se kontroluje na hardwarové i softwarové úrovni.

Monitoring začíná sběrem dat. Sleduje se vytíženost sítě, tj. využití šířky pásma, neboli jak efektivně je využita kapacita sítě, zdali je přetěžována nebo naopak nevyužita. Od toho se pak odvíjí výkon sítě, který je důležitý k rychlému přenosu dat. Mimo průtoková data se pak monitoruje stav uzlových zařízení, které jsou nezbytné pro správný chod sítě.

Nasbíraná data jsou pak předána analýze. Zkoumají se především trendy síťového provozu, ze kterého se odvozují závěry zmíněných motivací, kvůli kterým se data sbírají, například výkon nebo dostupnost sítě v závislosti na denní době. V nashromážděných datech se pozorují i anomálie, tedy data, která vybočují z běžného chodu. Mohou to být indikace výpadků nebo jiných neočekávaných dění, které běžně vedou k okamžitému upozornění správce sítě, aby mohl neprodleně jednat a předešlo se výpadku sítě.

Častá struktura aplikace pak obsahuje programy zachycující síťový provoz a uživatelské rozhraní prezentující výsledky analýzy. [2]

1.2 Principy měření přenosové rychlosti pomocí knihovny libcap

Knihovna libpcap je systémově nezávislé rozhraní pro zachytávání paketů na úrovni uživatele. Libcap poskytuje přenosný rámec pro nízkoúrovňové monitorování sítě a využívá se v aplikacích pro sběr a analýzu síťových statistik v reálném čase a aplikacích pro bezpečnostní monitorování či ladění sítí. Protože různí dodavatelé systémů často používají odlišná rozhraní pro zachytávání paketů, libpcap byl vytvořen jako jednotné API, které usnadňuje přenositelnost aplikací a eliminuje potřebu několika systémově specifických modulů pro zachytávání paketů v každé aplikaci. [3]

1.3 Nástroje pro monitorování sítě

Existující nástroje pro monitorování sítě poskytují zachytávání a vizualizaci dat. Níže jsou některé z nich uvedeny.

1.3.1 Wireshark

Wireshark je nástroj pro analýzu síťových protokolů, který umožňuje podrobné sledování a diagnostiku síťového provozu. Používá se k zachytávání a zobrazování paketů v reálném čase, což je užitečné pro analýzu komunikace na různých síťových vrstvách. Nabízí přehledné grafické rozhraní včetně širokých možností filtrování a analýzy. Pro svou funkčnost využívá knihovnu libpcap.

1.3.2 iftop

iftop je nástroj pro monitorování síťové aktivity v reálném čase, zaměřený na zobrazování šířky pásma, kterou jednotlivé spojení spotřebovává. Taktéž využívá knihovnu libcap, získává tak data přímo ze síťového rozhraní a

zobrazuje přehled aktuálně aktivních připojení spolu s informacemi o rychlosti přenosu, což ho činí užitečným pro rychlou kontrolu provozu a identifikaci spojení s největším využitím sítě.

1.3.3 isa-top

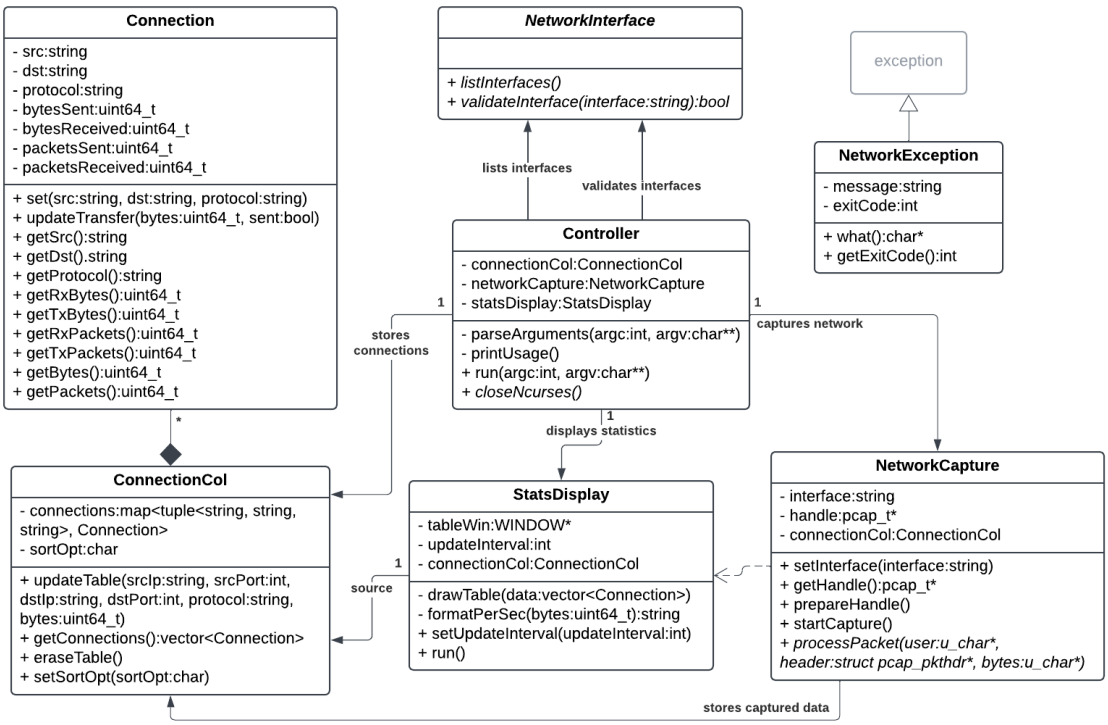
Aplikace isa-top, vytvořena v rámci předmětu ISA a popsána v dalších sekcích, se zaměřuje na zachytávání provozu na síťovém rozhraní pomocí knihovny libpcap a počítá přenosovou rychlost pro jednotlivé komunikující IP adresy. Program funguje jako konzolová aplikace a statistiky jsou zobrazeny a aktualizovány v rámci terminálu.

2 Návrh aplikace

Aplikace je navržena jako modulární systém pro zachytávání a zobrazování síťových statistik. Každá komponenta má svou definovanou zodpovědnost a jako celek spolupracují na zajištění základních funkcionalit aplikace, jako je sledování síťového provozu, ukládání a třídění informací o jednotlivých spojeních a zobrazování statistik uživateli.

Hlavní logika aplikace je řízena třídou `Controller`, která komponenty propojuje. Ovládá a spravuje třídy `ConnectionCol`, `NetworkCapture` a `StatsDisplay`. `ConnectionCol` slouží jako kolekce spojení jednotlivých komunikujících IP adres a poskytuje metodu pro zanalyzování dat pro výstup. `NetworkCapture` se stará o zachytávání síťového provozu prostřednictvím vybraného síťového rozhraní. Třída `NetworkInterface` nabízí metody pro získání seznamu dostupných rozhraní a validaci výběru rozhraní, což zajišťuje, že aplikace operuje se správnými síťovými zdroji. `StatsDisplay` zajišťuje zobrazení výstupních dat pomocí knihovny `ncurses` a pravidelně aktualizuje tabulku na základě dat získaných z `ConnectionCol`. `NetworkException` pak zpracovává chyby v průběhu zachytávání a zobrazení dat.

Strukturu aplikace shrnuje UML diagram tříd:



Obrázek 1: UML Diagram tříd popisující strukturu nástroje isa-top

3 Popis implementace

Program je psán v jazyce C++ se standardem C++20. Začíná hlavní funkcí, která má za úkol zapnout výstup konzole nastavením knihovny `ncurses`. Dále vytvoří instanci třídy `Controller` a společně se vstupními argumenty programu jí předává vedení. V souboru `main.cpp` se ještě vyskytuje odchytávání a zpracování výjimek `NetworkException`, které mohou nastat z důvodu chyb ve zpracování vstupních argumentů nebo zachytávání dat, a funkce `signalHandler`, která obstarává hladké ukončení programu při signálu přerušení.

3.1 Třída `Controller`

Třída `Controller` nejprve zajišťuje korektní zpracování vstupních argumentů, při chybě nebo specifických parametrech pak poskytne uživateli návod k použití a ukončí program. Využívá abstraktní třídu `NetworkInterface`, která poskytuje statické metody pro zjištění a validaci aktivních rozhraní. Při správném užití pak třída `Controller` zavolá metodu `prepareHandle()` třídy `NetworkCapture`, která připraví zachytávání síťového provozu pomocí funkcí knihovny `libpcap`. [4]

3.2 Řízení dvou vláken

Třída `Controller` má poslední zodpovědnost, a to rozčlenit program do dvou vláken, kdy čeká na ukončení obou z nich aby mohl bezpečně ukončit program.

3.2.1 Vlákno zachytávání síťového provozu

Vlákno `captureThread()`, běžící v metodě `startCapture` třídy `NetworkCapture`, zachytává síťový provoz, dokud není pozastaveno signálem přerušení. Pro každý zachycený paket pak volá statickou metodu `processPacket()`, která z paketu extrahuje informace, jako jsou:

- Zdrojová IP adresa
- Zdrojový port
- Cílová IP adresa
- Cílový port
- Jméno transportního protokolu (podporované jsou TCP, UDP, ICMPv4, ICMPv6, IGMP)
- Množství přenesených dat v bytech (z IP hlavičky)

Tato data pak ukládá do tabulky spojení `ConnectionCol` za využití vzájemného vyloučení (`mutex`) k zajištění správností programu a předcházení současného přístupu obou vláken ke sdíleným datům, což by mohlo vést k nekonzistencím.

Aby tabulka spojení počítala velikost přenesených dat pro jednotlivé komunikující IP adresy, při obdržení dat mapuje klíč `srcIP:port, dstIP:port, protocol` na jednotlivá spojení k jejich identifikaci. Pakliže tento klíč neexistuje, vytvoří nový záznam, v opačném případě přičte přenesený objem dat již existujícímu spojení. Směr jednotlivých spojení v tabulce je pak identifikován prvním záznamem, tedy pro výstup se rozlišuje zdrojová a cílová IP adresa podle zápisu prvního záznamu spojení do tabulky.

3.2.2 Vlákno zobrazování statistik

Vlákno `displayThread`, běžící v metodě `run()` třídy `StatsDisplay`, zobrazuje zachycené statistiky o síťovém provozu na výstup terminálu. Cyklus pak probíhá tak, že se nejprve vymaže stávající tabulka, bezpečně se získají data z tabulky spojení `ConnectionCol`, která se po přečtení vymaže, aby získávala nová, aktuální data. Přečtená data jsou pak vykreslena na výstup pomocí funkcí knihovny `ncurses` v uživatelsky přívětivém formátu. Cyklus následně čeká po dobu specifikovanou proměnnou `updateInterval`. Její výchozí hodnota je 1 sekunda, avšak ji lze nastavit vstupním argumentem, a proces opakuje, dokud není pozastaven signálem přerušení. V případě delšího intervalu aktualizace program spí periodicky po 1 sekundě a kontroluje, zdali nebyl běh programu pozastaven, aby se zamezilo dlouhému čekání v případě signálu přerušení.

Třída `ConnectionCol` již analyzuje data a předává je metodou `getConnections()` k zobrazení na výstup. Nejprve je seřadí podle počtu přenesených bytů nebo paketů (výchozí je řazení dle bytů) a vrátí prvních 10 s největším počtem.

4 Návod na použití

Program je přeložitelný nástrojem `make` do výsledného spustilného souboru `isa-top`.

Po spuštení aplikace s parametrem `-h` či `--help` je uživateli vypsána následující zpráva návodu k použití:

```
Usage: ./isa-top -i int [-s b|p] [-t s]
  -i [interface]                  interface to capture network traffic on
  -s b|p          (default: b)    sort output by bytes/packets per second
  -t [seconds]    (default: 1s)   update time interval
  -i                              display active interfaces
  -h|--help                       display usage

  run $ man ./isa-top.1 for more information
```

Parametr `-i` bez dalšího argumentu zajistí výpis aktivních síťových rozhraní. Pro běh aplikace je třeba specifikovat jedno rozhraní s parametrem `-i`, na kterém bude probíhat získávání statistik. Parametr `-s` značí styl řazení výstupu s výchozí hodnotou dle bytů a možností specifikace řazení dle počtu paketů. Parametr `-t` pak specifikuje interval, ve kterém se budou statistiky aktualizovat, zadáván číslem v sekundách (výchozí 1s).

Při korektním užití pak uživatel vidí v terminálu strukturovanou tabulku zobrazující statistiky o síťovém provozu na daném síťovém rozhraní aktualizující se v daném intervalu, v opačném případě je informován o chybě včetně poskytnutí návodu na použití.

5 Programová dokumentace

Zdrojové kódy jsou přehledně strukturované a okomentované. Jednotlivé soubory pak obsahují `Doxygen` komentáře v hlavičkách souborů a u definicí tříd a metod. Programovou dokumentaci tak lze snadno vygenerovat pomocí příkazu `make doc`. Odkaz na html stránku lze pak najít zde: `doc/html/index.html` .

Program má navíc svou manuálovou stránku `isa-top.1`. Příkazem `man ./isa-top.1` lze zobrazit tento manuál na výstup terminálu. [5]

6 Testování aplikace

6.1 Kompilace

Při vývoji i ve finální verzi zdrojových kódů byly pro kompilaci použity příznaky `-Wall -Wextra -Werror -pedantic -O2`, které zajišťují korektnost a optimalizaci aplikace. Tento proces byl vyzkoušen jak na lokálním počítači, tak na referenčním stroji merlin.fit.vutbr.cz, bez chybového výstupu.

6.2 Práce s pamětí

Aplikace byla testována na referenčním systému pomocí programu `valgrind` pro zjištění korektní práce s pamětí.

Analýza potvrdila, že nedochází k žádným skutečným únikům paměti v kódu aplikace ani k chybám. Z výpisu jsem zjistil, že určitá paměť zůstává přístupná i po ukončení programu a po vypsání detailu se ukázalo, že ji zanechává knihovna `ncurses`. Tato paměť je však po ukončení procesu uvolněna operačním systémem a není tedy považována za problémový únik.

6.3 Spuštění programu

V této sekci jsou popsány příklady spuštění včetně ukázky a validace výpisu. Vzhledem k oprávněním na referenčním stroji není aplikace spustitelná, proto jsem ji testoval na svém lokálním stroji s operačním systémem typu Unix.

6.3.1 Výpis aktivních rozhraní

```
./isa-top -i

Active interfaces:
ap1
en0
awdl0
llw0
...
```

Výpis je zkrácen. Po ověření podobnými nástroji jako `ifconfig` nebo `ip a` se výpisy shodují.

6.3.2 Výchozí hodnoty a `ping`

V jednom terminálu jsem pustil aplikaci `isa-top`, zatímco v druhém je spuštěn program `ping` generující ICMP zprávy. Pro izolaci této komunikace byl použit filtr `"host 1.1.1.1"`

```
./isa-top -i en0
```

SRC IP:PORT	DST IP:PORT	PROTO	b/s RX	p/s	b/s TX	p/s
192.168.1.22	1.1.1.1	icmp	84.0	1.0	84.0	1.0

Program `ping` odesílá a obdrží každou vteřinu ICMP paket (ICMP echo request a ICMP echo reply) o velikosti 64B na IP adresu 1.1.1.1. Ve výchozím nastavení aplikace `isa-top` aktualizuje statistiky každou vteřinu, tedy lze pozorovat jeden odeslaný a jeden přijatý ICMP paket s velikostí 84B (64B dat + 20B IP hlavička).

6.3.3 Zachycení komunikace vyhledávání ve webovém prohlížeči

Bez filtru jsem zachytával komunikace s intervalem obnovy statistik 10 vteřin, zatímco jsem provedl vyhledávání ve webovém prohlížeči Google Chrome (s implicitním řazením statistik dle počtu bytů)

```
./isa-top -i en0 -t 10
```

SRC IP:PORT	DST IP:PORT	PROTO	b/s RX	p/s RX	b/s TX	p/s TX
192.168.0.235:59758	142.251.36.68:443	udp	138.2k	153.2	8.2k	49.1
192.168.0.235:60720	23.1.106.35:443	tcp	16.0k	12.1	658.1	7.3
192.168.0.235:60719	23.1.106.35:443	tcp	12.9k	9.8	562.4	6.0
192.168.0.235:52370	142.251.36.106:443	udp	9.5k	8.8	721.9	5.1
192.168.0.235:60717	216.239.34.157:443	tcp	9.0k	7.9	546.9	6.2
192.168.0.235:60718	216.239.34.157:443	tcp	9.0k	8.0	448.1	4.3
192.168.0.235:59743	142.251.37.110:443	udp	6.7k	8.6	2.7k	7.6
192.168.0.235:49694	142.251.37.110:443	udp	3.8k	7.4	1.1k	5.9
192.168.0.235:49924	142.251.36.97:443	udp	3.6k	3.5	462.9	2.3
192.168.0.235:49639	142.251.37.97:443	udp	1.1k	3.7	1.5k	3.1

Z výpisu je patrné, že komunikace probíhaly především mezi mou lokální IP adresou a IP adresami společnosti Google a to na port 443 značící komunikační protokol HTTPS.

6.4 Testovací script `trafficGen.py`

Pro účely testování a ověření validního výstupu jsem vytvořil krátký skript v jazyce Python, který pomocí knihovny `scapy` [1] generuje nadefinovanou síťovou komunikaci. Funkce `generate_traffic()` vytvoří ethernetový rámec, kterému přidá IP hlavičku (IPv4 nebo IPv6) se zdrojovou a cílovou IP adresou, dále transportní protokol (TCP nebo UDP) se zdrojovým a cílovým portem a přidá daný počet bytů do paketu. Následně se pak daný počet těchto paketů odešle.

Parametry funkce `generate_traffic()` tvoří:

- Název rozhraní
- Verze IP
- Název transportního protokolu
- Zdrojová IP adresa
- Zdrojový port
- Cílová IP adresa
- Cílový port
- Počet paketů
- Velikost přidaných dat v bytech

Skript generoval nadefinovaný provoz a nástroj `isa-top` pak zachytával vygenerovaný síťový provoz s nastavením řazení dle počtu paketů. Pro účely tohoto testování jsem ve třídě `NetworkCapture` nastavil filtr na IP adresy generujících komunikací, abych tyto statistiky odizoloval.

6.4.1 Příklad spuštění

```
./isa-top -i en0 -s p -t 10
```

`trafficGen.py`:

```
generate_traffic("en0", 4, "UDP", "1.1.1.1", 1, "2.2.2.2", 2, 4000, 100)
generate_traffic("en0", 4, "UDP", "2.2.2.2", 2, "1.1.1.1", 1, 3000, 100)
generate_traffic("en0", 4, "TCP", "2.2.2.2", 2, "1.1.1.1", 1, 2000, 100)
generate_traffic("en0", 4, "TCP", "1.1.1.1", 1, "2.2.2.2", 2, 1000, 100)
```

Z výstupu pak lze ověřit, zdali nástroj `isa-top` korektně vypisuje statistiky. Jak bylo zmíněno, spojení jsou identifikované parametry transportní a síťové vrstvy a to vždy dle první komunikace. Hodnoty pak odpovídají skutečnosti. Počet přijatých paketů prvního spojení je 300, což odpovídá 3000 přijatým paketům za 10 vteřin. Počet přijatých bytů prvním spojením je 384 KB za 10 vteřin. Jeden UPD paket má 100B payload + 20B IP hlavička + 8B UDP hlavička = 128 bytů. Bylo jich odesláno 3000 za 10 vteřin, to značí 38 400B za vteřinu, což odpovídá výstupní hodnotě.

Reference

[1] Biondi, P.: *Scapy's documentation*. 2024.
URL `https://scapy.readthedocs.io/en/latest/`

[2] Etechblog: Co je monitorování sítě a proč je důležité? 2023.
URL `https://etechblog.cz/co-je-monitorovani-site-a-proc-je-dulezite`

[3] Group, T. T.: *libpcap README*. 2023.
URL `https://github.com/the-tcpdump-group/libpcap/blob/master/README.md`

[4] The Tcpdump Group: *Libpcap documentation (pcap(3pcap))*. 2024.
URL `https://www.tcpdump.org/manpages/pcap.3pcap.html`

[5] Wirzenius, L.: Writing manual pages. 2019.
URL `https://liw.fi/manpages/`