
# Počítačové komunikace a sítě (IPK)

2021/2022

## Projekt 2 - varianta ZETA:

## Sniffer paketů

Lucie Svobodová (xsvobo1x)

23. 4. 2022

# Obsah

# 1 Úvod

Monitorování přenosu dat v počítačové síti je jednou z nejdůležitějších činností v oblasti práce s počítačovými sítěmi. K takovému sledování toku dat sítí slouží mimo jiné programy nazývající se sniffery paketů. Umožňují zobrazování informací o zachycených paketech a tyto informace mohou být později využity k další analýze sítě. Cílem druhého projektu do předmětu Počítačové komunikace a sítě byla implementace takového jednoduchého snifferu, který zobrazuje základní informace o zachycených paketech protokolů ARP, TCP, UDP, ICMP a ICMPv6 a umožňuje jejich filtrování.

V následující kapitole jsou uvedeny základní informace, které bylo potřeba nastudovat před začátkem analýzy přenosu dat po síti, následuje kapitola věnovaná vlastní implementaci snifferu paketů a pár slov k jeho závěrečnému testování.

# 2 Teoretické podklady

Sniffer paketů, jinými názvy analyzátor paketů nebo síťový analyzátor, je program, který je určen k monitorování přenosu dat na počítačové síti. Je možné jej využívat k analýze těchto paketů a k identifikaci paketů chybných nebo podezřelých, což může pomoci udržovat efektivní a kontrolovatelný přenos dat. Pokud je sniffer nastaven na promiskuitní režim, je schopen zachytávat všechny pakety, které danou sítí prochází. Tyto programy mohou být mimo jiné součástí antivirového software, ale zároveň je možné je využít i k nekalým úmyslům, například zavedení snifferu do interní sítě a sledování přenášených dat uvnitř této sítě může být použito k síťovému útoku. Vzhledem k tomu, že sniffer paketů pouze pasivně sleduje provoz na síti, je poměrně těžké jeho přítomnost v dané síti odhalit. [1, 2, 3]

Sniffer paketů tedy monitoruje síťové přenosy. Jak ale vlastně taková síť funguje? V následujících podkapitolách jsou uvedeny základní informace potřebné k porozumění principu fungování packet snifferu.

## 2.1 Model ISO/OSI

Referenční model ISO/OSI je modelem používaným pro popis počítačové sítě. Skládá se ze sedmi vrstev, které spolu komunikují pomocí protokolů a zajišťují tak propojení celé sítě. Protokoly, se kterými pracuje v tomto projektu implementovaný packet sniffer, budou popsány v následujících podkapitolách.

Nejnižší vrstvou modelu ISO/OSI je vrstva fyzická, která zajišťuje přenos jednotlivých bitů mezi odesílatelem a příjemcem. Po ní následuje vrstva linková, která se stará o přenos bloků dat - rámců. Zároveň se na této vrstvě kontrolují kontrolní součty (checksum), díky kterým můžeme odhalit poškozené rámce. Mezi nejznámější protokoly linkové vrstvy patří např. Ethernet nebo WiFi. Zatímco linková vrstva zajišťuje přenos rámců mezi MAC adresou odesílatele a MAC adresou příjemce, další vrstva, síťová, zajišťuje přenos síťových paketů (datagramů) mezi jednotlivými uzly ležícími mezi uzlem odesílatele a příjemce. Na této vrstvě je nutné řešit směrování (routing), tedy volbu vhodné trasy mezi koncovými uzly. K nejdůležitějším protokolům síťové vrstvy patří Internet Protocol (IP). Mapování IP adresy na MAC adresu zajišťuje Address Resolution Protocol (ARP). Toto mapování je ukládáno do ARP tabulky, kterou má každý IP host. Čtvrtou vrstvou ISO/OSI modelu je vrstva transportní, která se zabývá end-to-end komunikací - komunikací pouze mezi odesílatelem a příjemcem. Pakety na transportní vrstvě se nazávají segmenty. Velké segmenty mohou být na této vrstvě děleny na menší a tyto poté opět skládány zpět. Protokolem transportní vrstvy je například Transmission Control Protocol (TCP) a User Datagram Protocol (UDP). Posledními třemi vrstvami ISO/OSI modelu jsou vrstvy relační, prezentační a aplikační. Tyto vrstvy jsou v internetovém zásobníku (Internet Stack) sdružovány do jedné, aplikační, vrstvy. Tato vrstva zajišťuje komunikaci mezi samotnými programy, přičemž se spoléhá na protokoly transportní vrstvy. Nejpoužívanějšími protokoly aplikační vrstvy jsou File Transfer Protocol (FTP), Domain Name System (DNS) nebo Hypertext Transfer Protocol (HTTP). [1, 4]

Každý paket se skládá ze dvou částí - hlavičky (header) a samotných přenášených dat (payload), což jsou ve většině případů zapouzdřená data z vyšších vrstev.

## 2.2 Ethernet Protocol

Jak již bylo zmíněno, Ethernet Protocol je protokolem linkové vrstvy. Zapouzdřuje protokoly síťové vrstvy přidáním hlavičky, která obsahuje MAC adresu odesílatele, MAC adresu příjemce, pole EtherType, které specifikuje protokol vyšší vrstvy, který daný rámec zapouzdřuje, a délku rámce. Hlavička Ethernet protokolu má délku 14 bytů. Dále rámec obsahuje část payload, která obsahuje samotná uživatelská data, a konečně část CRC (cyclic reduncancy check), díky kterému je možné rozpoznat chybně

přenesené rámce. [5, 6]

Hodnoty pole EtherType a jejich významy jsou vypsány v seznamu IEEE 802 Numbers (IANA) (viz [16]). Z něj tento projekt umožňuje analýzu paketů protokolů Address Resolution Protocol (ARP), Internet Protocol version 4 (IPv4) a Internet Protocol version 6 (IPv6).

### 2.3 Address Resolution Protocol (ARP)

Protokol ARP zajišťuje mapování MAC adres, které jsou využívány k identifikaci odesílatele a příjemce na linkové vrstvě, na IP adresy využívané na vrstvě síťové. Odesílatel, který potřebuje odeslat IP datagram na adresu ležící v dané podsíti, odesílá linkovým broadcastem ARP dotaz (ARP request) obsahující hledanou IP adresu. Všechna zařízení v dané podsíti si po přijetí tohoto dotazu uloží informace o odesílateli a zařízení identifikované hledanou IP adresou vyšle ARP odpověď (ARP reply), která obsahuje jeho IP i MAC adresu. Toto mapování si poté původní odesílatel uloží do ARP tabulky a může odeslat IP datagram. [1, 7]

ARP protokol ve své hlavičce mimo jiné tedy obsahuje MAC adresu a IP adresu odesílatele, MAC adresu a IP adresu příjemce a typ operace (např. 1 pro ARP request a 2 pro ARP reply) (pro kódy dalších operací viz [17]. [8]

### 2.4 Internet Protocol (IP)

Internet Protocol je protokol používaný na síťové vrstvě, který zapouzdřuje protokoly transportní vrstvy. Dnes jsou používány dvě verze IP - verze 4 (IPv4) a verze 6 (IPv6).

Délka hlavičky protokolu IPv4 nemá fixní velikost, ale musí mít nejméně 20 bytů. Hlavička protokolu IPv4 obsahuje především verzi protokolu (číslo 4), délku hlavičky, typ služby, délku celého IP datagramu (hlavička + payload), pole TTL (time-to-live), které určuje, za jak dlouho musí být datagram z routeru odstraněn, IP adresu odesílatele a příjemce, kontrolní součet a část options, které umožňuje rozšířit hlavičku a připojit tak další informace. Hlavička také obsahuje číslo protokolu, který má dále zpracovávat předávaná data (číslo 1 pro ICMP, 6 pro TCP, 17 pro UDP a 58 pro ICMPv6, další čísla viz [18]. Datagram samozřejmě obsahuje i část se samotnými daty, která jsou většinou segmentem transportní vrstvy (TCP, UDP), nebo data ICMP zprávy. [9, 10]

Hlavička protokolu IPv6 má fixní délku 40 bytů. Stejně jako hlavička IPv4 obsahuje číslo verze (6), dále velikost uživatelských dat, část next header, hop limit, kdy po dosažení hodnoty 0 musí být daný datagram odstraněn, IPv6 adresu odesílatele a příjemce a samotná data, který příjemce zpracovává podle protokolu specifikovaném v poli next header. Toto pole umožňuje rozšiřovat IPv6 protokol přidáváním dalších hlaviček, takzvaných extension headers. Tyto rozšiřující hlavičky opět obsahují část next header, která určuje kód další hlavičky - protokolu. Protokoly transportní vrstvy, které zapouzdřuje IPv6, jsou stejné jako u IPv4 (TCP, UDP). Stejně jako IPv4 může specifikovat protokol ICMP, IPv6 používá konkrétně protokol ICMPv6 (seznam protokolů viz [18].[1, 9, 11]

### 2.5 Internet Control Message Protocol (ICMP)

Jak už název napovídá, ICMP je používán pro informační zprávy a reportování chyb. Pokud například IP router není schopen najít cestu k příjemci, zašle ICMP zprávu, aby o této chybě odesílatel věděl. ICMP je také používán pro diagnostiku sítě, například nástrojem `ping`. [1]

ICMP pro IPv4 je identifikován protokolem IP číslem 1 ([18]). Hlavička ICMP obsahuje především typ zprávy (např. číslo 0 indikuje Echo Reply a číslo 8 Echo Request, viz [19]), dále obsahuje kontrolní

součet a samotná data. Hlavička má fixní délku 8 bytů. [12]

ICMP pro IPv6 se nazývá ICMPv6. V jeho hlavičce najdeme informaci o typu zprávy (informační nebo chybová zpráva), část kód (code) se používá jako další úroveň pro rozlišování zpráv a kontrolní součet (checksum) opět slouží k odhalení chyb přenosu. [13]

## 2.6 User Datagram Protocol (UDP)

UDP je protokolem transportní vrstvy používajícím IP jako síťový protokol. Počítačové aplikace rozlišuje pomocí portů (stejně jako TCP). UDP negarantuje využití kontrolního součtu (checksum), kvůli čemu není zaručena kontrola, že jsou data doručená v pořádku. Nepotvrzuje doručení UDP datagramů a data mohou být doručena ve špatném pořadí. Kvůli těmto problémům UDP poskytuje nezabezpečený přenos, nicméně i přes to je velmi často používaným protokolem, například při real-time přenosech dat (díky tomu, že se nemusí potvrzovat přijetí dat, není síť tolik zatěžována a přenos je rychlejší než při využití TCP). [1, 14]

Hlavička UDP paketu má velikost 8 bytů, obsahuje zdrojový a cílový port, délku dat včetně hlavičky a kontrolní součet (checksum), který ale nemusí být využíván. [14]

## 2.7 Transmission Control Protocol (TCP)

TCP je protokol transportní vrstvy, opět využívající jako síťový protokol IP. Narozdíl od UDP ale TCP před samotným odesíláním dat musí mezi klientem a serverem navázat spojení, což je realizováno pomocí tzv. three-way handshake. Díky tomu TCP garantuje spolehlivé doručování dat a správné pořadí doručení. K rozlišování aplikací TCP také používá porty, stejně jako UDP. [1]

TCP hlavička má typicky velikost 20 bytů, je ale možné ji rozšířit až na 60 bytů pomocí části options. Dále hlavička obsahuje zdrojový a cílový port, pořadové číslo (sequence number), které je použito k řazení paketů, potrvzovací číslo (acknowledgment number), které příjemce používá k žádosti o další TCP segment. Dalšími informacemi obsaženými v TCP hlavičce jsou například data offset, flags, options a kontrolní součet (checksum). [15]

# 3 Vlastní implementace

Packet sniffer byl implementován v jazyce `C++` s použitím knihovny `libpcap` (`https://www.tcpdump.org/`). Tato konzolová aplikace vypisuje informace o zachycených paketech na standardní výstup. Aplikace poskytuje nápovědu, která uživatele informuje o možném použití přepínačů, a kterou je možné vypsat pomocí přepínačů `-h` nebo `-help`. Pakety je možné filtrovat pomocí dalších přepínačů zadaných na příkazové řádce při spouštění aplikace. Informace o zadaných přepínačích jsou ukládány do struktury `opts`. Pokud uživatel nespecifikuje rozhraní, na kterém má být naslouchánо, program vypíše na standardní výstup všechna aktivní rozhraní a je ukončen.

Packet sniffer naslouchá na rozhraní, které je specifikováno argumentem příkazové řádky. Po úspěšném připojení na toto rozhraní je inicializována proměnná `pcap_t *pcap`, která představuje pcap handler. Po aktivaci tohoto handleru je zjištěn typ hlavičky linkové vrstvy (pro jednotlivé typy viz [20]). Pokud je tímto typem `Ethernet`, může být přistoupeno k dalšímu kroku. Jiné typy tímto packet snifferem podporovány nejsou.

Vzhledem k tomu, že v projektu jsou analyzovány pouze pakety protokolů TCP, UDP, ARP, ICMP a ICMPv6, je nejdříve potřeba pro zachytávání paketů vytvořit filtr. Ten je jako řetězec sestaven ve funkci `make_filter()` podle zadaných argumentů příkazové řádky. Pokud jsou specifikovány pouze některé protokoly, budou zachytávány pouze pakety daných protokolů. Pokud je v příkazové řádce zadán jako parametr port, pakety na daném rozhraní budou filtrovány podle tohoto portu (port může být jak v source části, tak v části destination). Pokud je v argumentech specifikován port, ale není dále uveden protokol TCP ani UDP, jsou zachytávány všechny pakety těchto protokolů na daném portu. Po sestavení filtru je filtr pomocí knihovních funkcí `pcap_compile()` a `pcap_setfilter()` nastaven na danou hodnotu.

Hlavní smyčku, ve které jsou pakety zachytávány, představuje knihovní funkce `pcap_loop()`. Funkce zachytí a zpracuje tolik paketů filtrovaných podle daného filtru, kolik uživatel uvedl v parametru `-n` (výchozí hodnota je 1). Zachycené pakety jsou zpracovávány ve funkci `process_frame()`. Tato funkce vypisuje základní informace o rámci (timestamp, MAC adresy odesílatele a příjemce, délku rámce). Dále podle seznamu etherTypes ([16]) určí protokol, kterým má být další obsah rámce zpracován.

V programu je často využíváno přetypovaní typu `u_char *` na některou ze struktur knihovny `netinet`. Vždy je potřeba pomocí určení velikostí jednotlivých hlaviček určit, na jaké pozici v řetězci obsahujícím celý rámec se další datagram nachází, poté je možné řetězec přetypovat na strukturu hlavičky daného protokolu (např. `tcphdr`, `ether_header`). Díky tomuto přetypování je následné vypisování dalších informací o paketu přehlednější.

Rámce jsou dále zpracovávány v příslušných funkcích. Podle typu `etherType` je zavolána jedna z funkcí `process_ipv4()`, `process_ipv6()` nebo `process_arp()`. Ve funkci `process_ipv4()` je zpracováván IPv4 protokol. Jsou zde vypsány informace o IP adrese odesílatele a příjemce a následně je podle typu protokolu síťové vrstvy rozhodnuto, jak se bude daný paket zpracován. Zpracování protokolů TCP, UDP a ICMP je stejné jako u protokolu IPv6 (bude popsáno dále).

O paketech protokolu ARP, zpracovávaného ve funkci `process_arp()`, jsou vypisovány následující informace: opcode, sender MAC address, sender IP address, target MAC address, target IP address. Nakonec, stejně jako u zpracovávání všech protokolů, je vypsaný obsah celého rámce.

IPv6 pakety se zpracovávají ve funkci `process_ipv6()`. Stejně jako u IPv4 jsou vypsány IP adresy odesílatele a příjemce. IPv6 může navíc obsahovat rozšiřující hlavičky, které jsou procházeny v cyklu

a jejich kódy (list) jsou postupně vypisovány na výstup. Pokud je kód další hlavičky roven kódu jednoho z protokolů TCP, UDP nebo ICMPv6, je zavolána funce analyzující pakety příslušného protokolu.

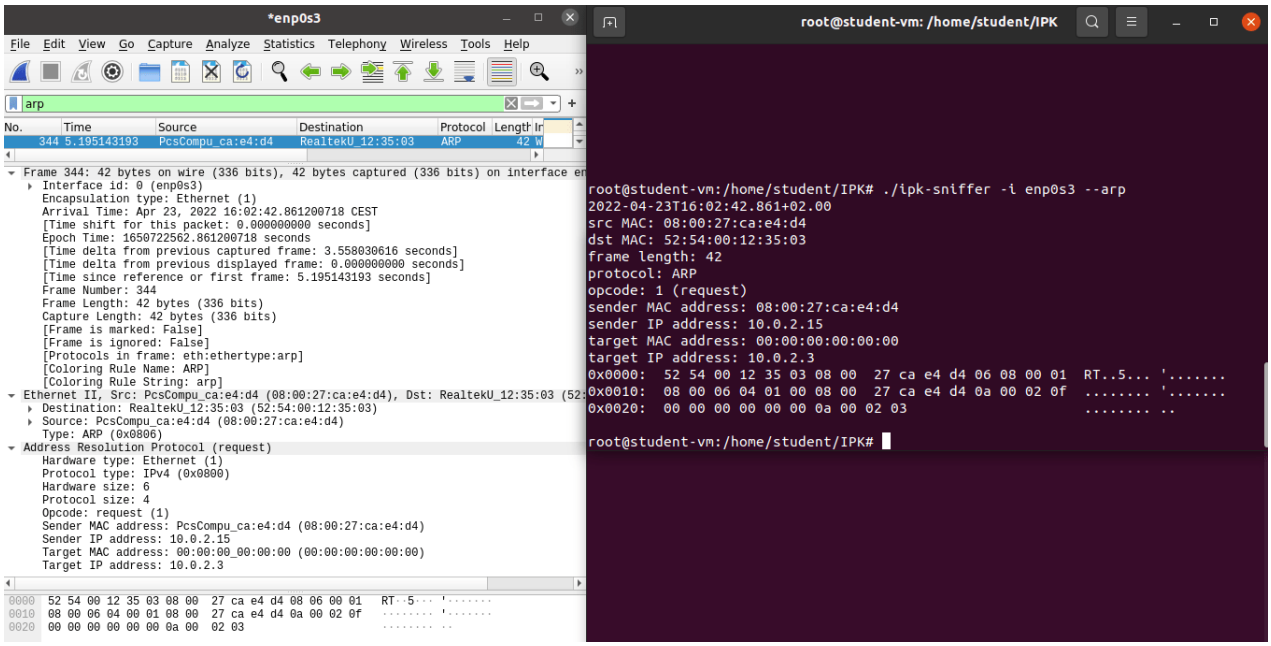
Informace o ICMP paketech jsou vypisovány ve funkci `process_icmp()`, která může být volána jak při zpracovávání IPv4, tak i IPv6. Vypsán je typ zprávy (type), kód zprávy (code), kontrolní součet (checksum) a nakonec data celého rámce.

U paketů TCP i UDP jsou vypsány informace o zdrojovém a cílovém portu (src port, dst port), kontrolním součtu (checksum) a nakonec je vypsát opět obsah celého rámce. Toto vypisování je realizováno ve funkci `process_tcp()` pro TCP a `process_udp()` pro UDP. Tyto funkce mohou být opět volány jak při zpracovávání paketů IPv4, tak při zpracovávání IPv6.

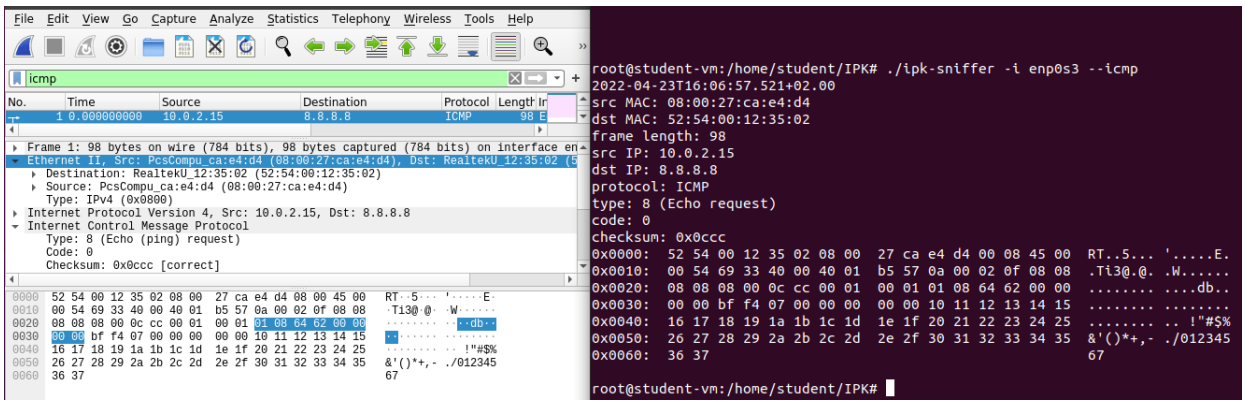
V projektu bylo ošetřeno zachycení signálu `SIGINT`, které je realizováno pomocí funkcí knihovny `csignal` (https://en.cppreference.com/w/cpp/header/csignal). Při zachycení signálu `SIGINT` je zavolána funkce `handle_signal()`. Ta nastaví globální proměnnou `sigint_received` na hodnotu 1, díky čemu je možné ukončit případně právě prováděný cyklus. Zároveň je zavolána funkce `pcap_breakloop()`, díky které je ihned ukončeno případné zachytávání paketů. Po zachycení signálu jsou uvolněny všechny alokované zdroje a program je ukončen s návratovým kódem číslo 1.

# 4 Testování

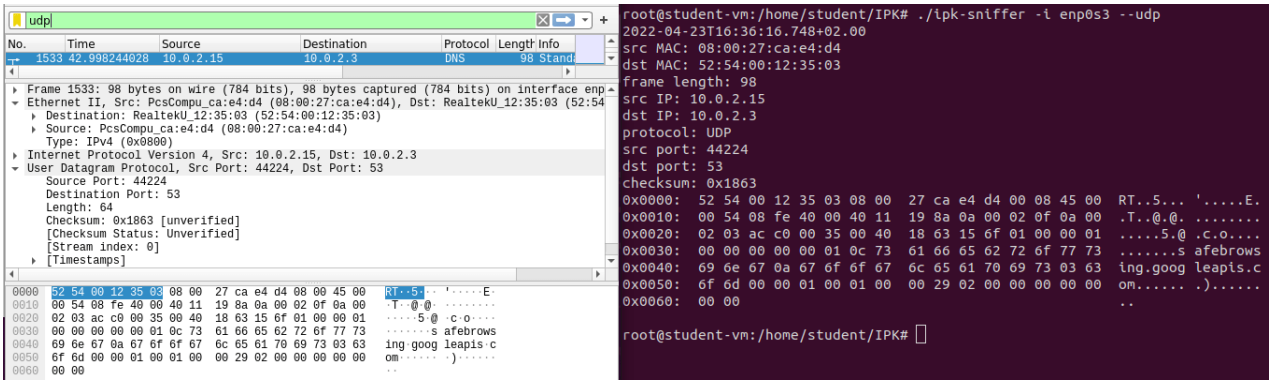
Testování implementovaného snifferu paketů bylo prováděno na referenčním Linux image specifikovaném v zadání. Výstupy snifferu byly porovnávány s výstupy programu Wireshark (`https://www.wireshark.org/`). V programu Wireshark jsou výstupy obsáhlé, byly tedy porovnávány pouze informace, které jsou vypisovány v packet snifferu implementovaném v tomto projektu. Na následujících snímcích obrazovky jsou zobrazeny výstupy programu Wireshark a výstupy packet snifferu pro stejné pakety.



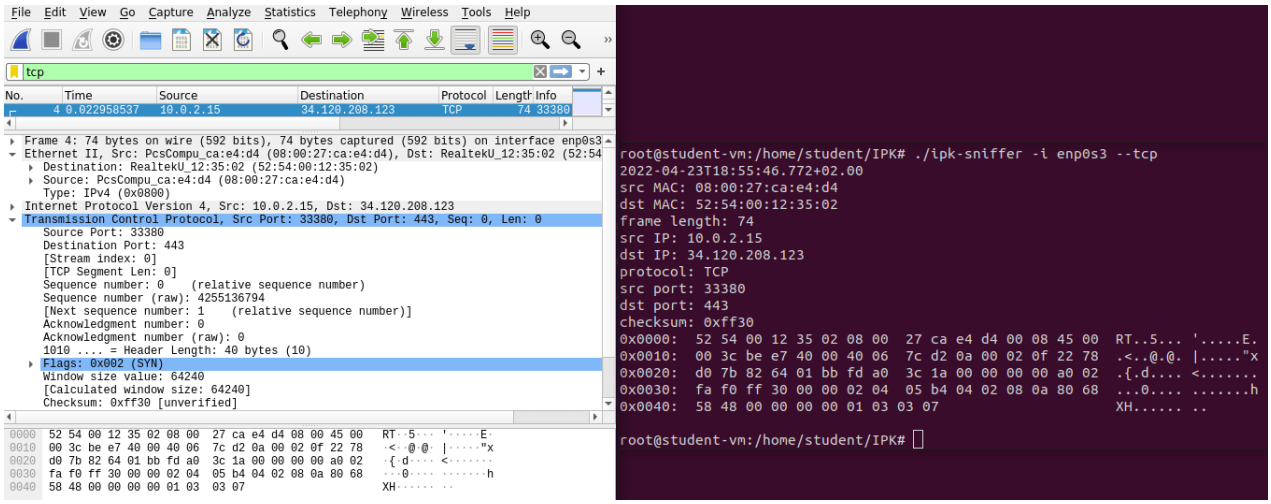

Obrázek 1: Ukázka zachycení ARP paketu. Vlevo výpis programu Wireshark, vpravo implementovaný packet sniffer.



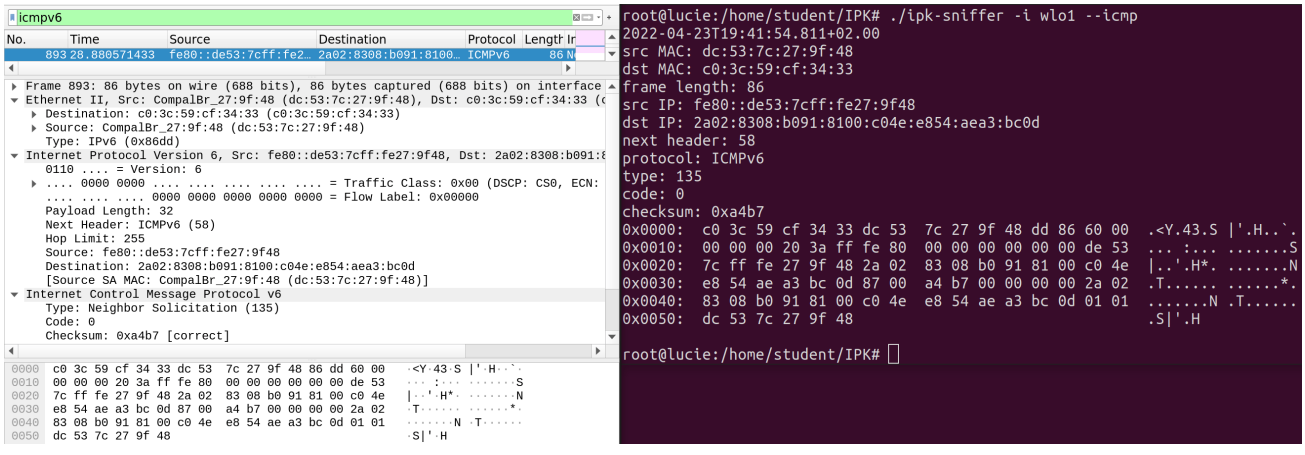

Obrázek 2: Ukázka zachycení ICMP paketu (IPv4). Vlevo Wireshark, vpravo implementovaný packet sniffer.



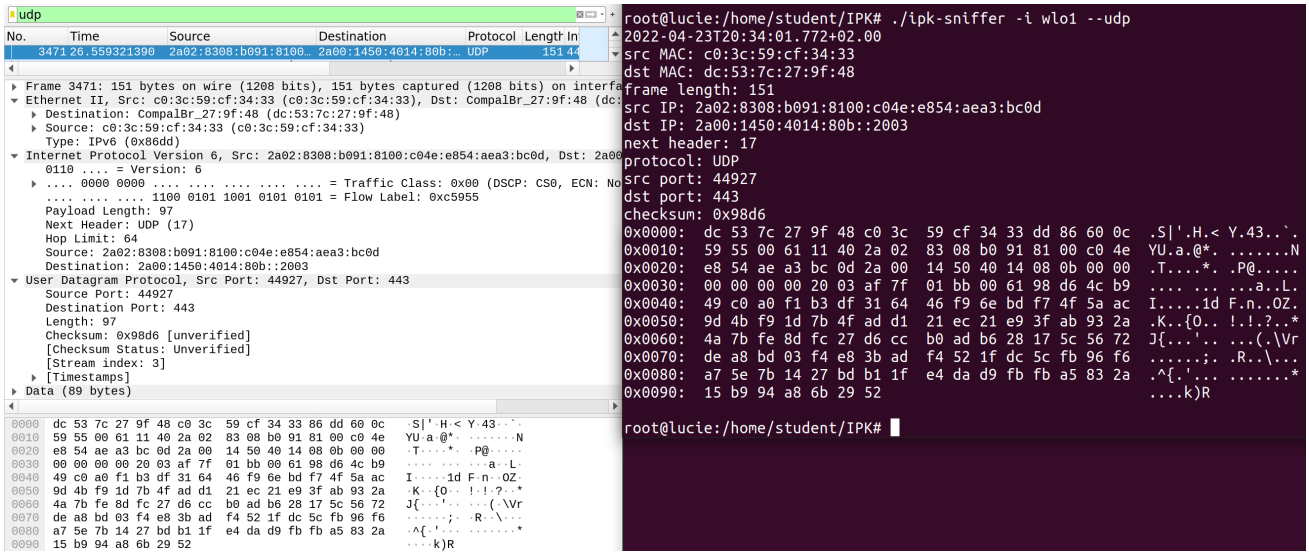

Obrázek 3: Ukázka zachycení UDP paketu (IPv4). Vlevo Wireshark, vpravo implementovaný packet sniffer.



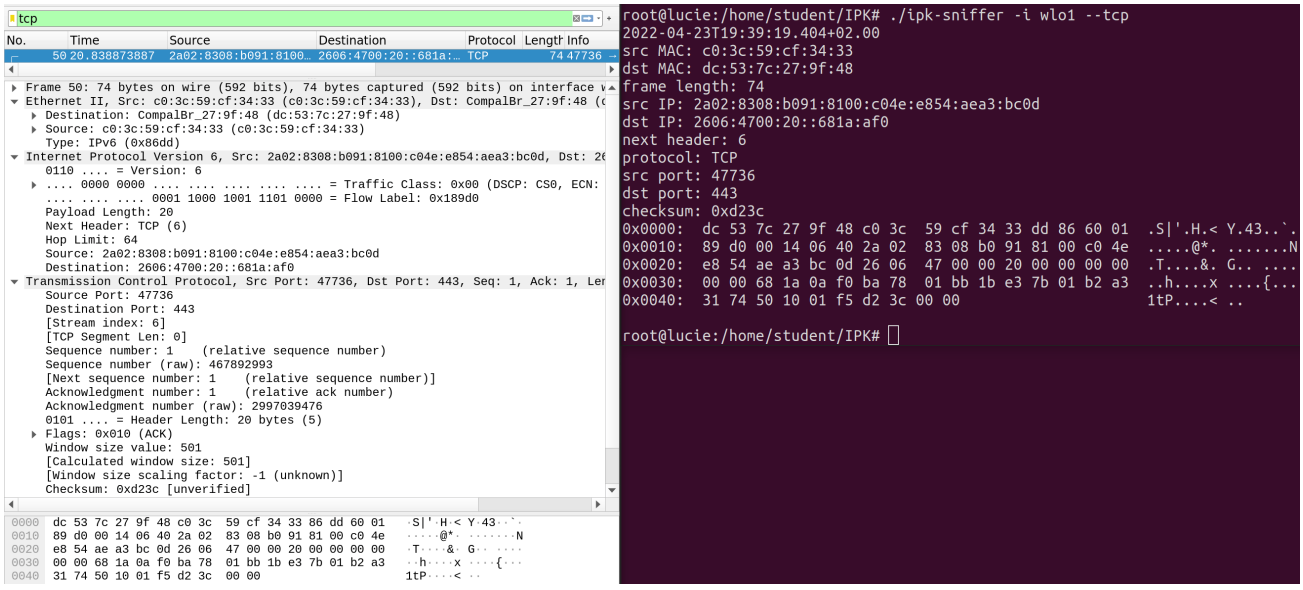

Obrázek 4: Ukázka zachycení TCP paketu (IPv4). Vlevo Wireshark, vpravo implementovaný packet sniffer.




Obrázek 5: Ukázka zachycení ICMP paketu (IPv6). Vlevo Wireshark, vpravo implementovaný packet sniffer.




Obrázek 6: Ukázka zachycení UDP paketu (IPv6). Vlevo Wireshark, vpravo implementovaný packet sniffer.




Obrázek 7: Ukázka zachycení TCP paketu (IPv6). Vlevo Wireshark, vpravo implementovaný packet sniffer.

Kromě výstupů jednotlivých paketů bylo testováno i zachytávání většího počtu paketů, ale vzhledem k tomu, že snímky obrazovky by v takovém případě byly příliš velké, nejsou sem vloženy. Dále byly testovány různé přepínače použité k filtrování paketů, a to jak filtry jednotlivých protokolů, tak i jejich kombinace. Dále bylo testováno filtrování paketů podle specifikovaného portu. Vypisované výstupy ve všech testovaných případech odpovídaly výstupům programu Wireshark.

Zároveň byla aplikace testována nástrojem `valgrind` (https://valgrind.org/), aby bylo zajištěno správné použití a uvolňování všech alokovaných zdrojů. Bylo testována práce s alokovanými zdroji i ve stavech programu, které vedly k předčasnému ukončení kvůli různým chybovým stavům. Všechny detekované chyby byly v průběhu implementace úspěšně odstraněny.

# 5 Bibliografie


**[1]** Kurose, James F., and Keith W. Ross. *Computer networking: a top-down approach featuring the Internet*. 7. vyd. Hoboken, New Jersey: Pearson, 2017. ISBN: 0-13-359414-9.

**[2]** Kaspersky Lab. *What is a Packet Sniffer?*. AO Kaspersky Lab [online]. [cit. 2022-04-22]. Dostupné z: `https://www.kaspersky.com/resource-center/definitions/what-is-a-packet-sniffer`.

**[3]** Paessler AG. *What is packet sniffing?*. Paessler AG [online]. [cit. 2022-04-22]. Dostupné z: `https://www.paessler.com/it-explained/packet-sniffing`.

**[4]** Peterka, J. *Referenční model ISO/OSI - sedm vrstev*. earchiv.cz [online]. [cit. 2022-04-22]. Dostupné z: `https://www.earchiv.cz/a92/a213c110.php3`.

**[5]** Elprocus. *What is an Ethernet Protocol : Working & Its Applications*. elprocus.com [online]. [cit. 2022-04-22]. Dostupné z: `https://www.elprocus.com/ethernet-protocol/`.

**[6]** Eastlake D. 3rd, and Abley J. *IANA Considerations and IETF Protocol and Documentation Usage for IEEE 802 Parameters*. Internet Engineering Task Force (IETF) [online]. Říjen 2013 [cit. 2022-04-22]. Dostupné z: `https://www.rfc-editor.org/rfc/rfc7042.html`.

**[7]** Wikipedia contributors. *Address Resolution Protocol*. Wikipedia, The Free Encyclopedia [online]. Wikipedia, The Free Encyclopedia, Revidováno 6. 4. 2022. [cit. 2022-04-22]. Dostupné z: `https://cs.wikipedia.org/wiki/Address_Resolution_Protocol`.

**[8]** Kozierok Ch. M. *ARP Message Format*. The TCP/IP Guide [online]. [cit. 2022-04-22]. Dostupné z: `http://www.tcpipguide.com/free/t_ARPMessageFormat.htm`.

**[9]** Cloudflare, Inc. *What is the Internet Protocol?*. Cloudflare, Inc [online]. [cit. 2022-04-22]. Dostupné z: `https://www.cloudflare.com/learning/network-layer/internet-protocol/`.

**[10]** GeeksForGeeks. *Introduction and IPv4 Datagram Header*. GeeksForGeeks [online]. Revidováno 28. 6. 2021 [cit. 2022-04-22]. Dostupné z: `https://www.geeksforgeeks.org/introduction-and-ipv4-datagram-header/`.

**[11]** GeeksForGeeks. *Internet Protocol version 6 (IPv6) Header*. GeeksForGeeks [online]. Revidováno 20. 10. 2021 [cit. 2022-04-22]. Dostupné z: `https://www.geeksforgeeks.org/internet-protocol-version-6-ipv6-header/`.

**[12]** Wikipedia contributors. *ICMP*. Wikipedia, The Free Encyclopedia [online]. Wikipedia, The Free Encyclopedia, Revidováno 13. 12. 2020. [cit. 2022-04-22]. Dostupné z: `https://cs.wikipedia.org/wiki/ICMP`.

**[13]** Wikipedia contributors. *ICMPv6*. Wikipedia, The Free Encyclopedia [online]. Wikipedia, The Free Encyclopedia, Revidováno 24. 3. 2018. [cit. 2022-04-22]. Dostupné z: `https://cs.wikipedia.org/wiki/ICMPv6`.

**[14]** GeeksForGeeks. *User Datagram Protocol (UDP)*. GeeksForGeeks [online]. Revidováno 26. 10. 2021 [cit. 2022-04-22]. Dostupné z: `https://www.geeksforgeeks.org/user-datagram-protocol-udp/`

**[15]** NetworkLessons.com contributors. *TCP Header*. NetworkLessons.com [online]. [cit. 2022-04-22]. Dostupné z: `https://networklessons.com/cisco/ccie-routing-switching-written/tcp-header`.

**[16]** *IEEE 802 Numbers*. Internet Assigned Numbers Authority [online]. Revidováno 22. 2. 2022 [cit. 2022-04-22]. Dostupné z: https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml#ieee-802-numbers-1.

**[17]** *Address Resolution Protocol (ARP) Parameters*. Internet Assigned Numbers Authority [online]. Revidováno 20. 7. 2016 [cit. 2022-04-22]. Dostupné z: https://www.iana.org/assignments/arp-parameters/arp-parameters.xhtml.

**[18]** *Protocol Numbers*. Internet Assigned Numbers Authority [online]. Revidováno 7. 4. 2021 [cit. 2022-04-22]. Dostupné z: https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml.

**[19]** *Internet Control Message Protocol (ICMP) Parameters*. Internet Assigned Numbers Authority [online]. Revidováno 25. 9. 2020 [cit. 2022-04-22]. Dostupné z: https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml.

**[20]** The Tcpdump Group. *LINK-LAYER HEADER TYPES*. The Tcpdump Group [online]. [cit. 2022-04-23]. Dostupné z: https://www.tcpdump.org/linktypes.html.