Vysoké učení technické v Brně
Fakulta informačních technologií

IPK - Počítačové komunikace a sítě – 2. projekt

Varianta ZETA: Sniffer paketů

24. apríla 2022 Marián Keszi

Obsah

Zoznam obrázkov

1 Úvod

Našim cielom v projekte bolo napísať program, ktorého úlohou bude odchytávať pakety na sieti. Program mal byť schopný odchytávať konkrétne TCP, UDP, ICMPv4, ICMPv6 a ARP pakety. Programu zároveň môže užívateľ pri spustení zadať ako argumenty, ktoré pakety chce konkrétne odchytávať, zároveň mu môže zadať počet paketov, ktorý chce aby bol odchytený, na akom porte majú byť pakety odchytávané alebo na akom rozhraní majú byť pakety odchytávané.

2 Implementácia

Projekt sme implementovali v programovaciom jazyku C++, čo nám pomohlo minimálne aby sme sa nemuseli starať o prácu s pamäťou. Samotný program sa následne implementoval postupne po jednotlivých častiach, ktoré sú popísané samostatne nižšie.

2.1 Spracovanie vstupných argumentov

Na začiatku programu sme implementovali funkciu pre načítanie argumentov, ktoré nám užívateľ pri spustení programu zadefinoval. O spracovanie vstupných argumentov sa teda stará funkcia PARSEARGS(). Vo funkcii už následne ako prvé skontrolujeme, či nám na vstupe bol zadaný požadovaný počet argumentov a ak nebol, tak pomocou funkcie PRINTINTERFACES()(3.4) vytlačíme na výstup všetky dostupné rozhrania. Ak sme na vstupe dostali ale správny počet argumentov, tak skontrolujeme, či medzi argumentmi nebol zadaný buď argument `-h prípadne --help` alebo argument `-i prípadne --interface` a ak takýto argument zadaný bol, tak buď vypíšeme pomocnú správu, prípadne vypíšeme všetky dostupné rozhrania v prípade ak rozhranie v argumente nebolo zadané správne. Po vykonaní tejto kontroly následne skontrolujeme, či nám neboli zadané nejaké konkrétne druhý paketov, ktoré by sme mali odchytávať a ak boli, tak nastavíme globálne bool premenné na true s čím budeme ďalej v programe pracovať. Ďalej si skontrolujeme, či nám nebol zadaný konkrétny port, na ktorom by sme mali pakety odchytávať alebo prípadne konkrétny počet paket, ktorý by sme mali odchytiť(základne je počet paketov nastavený na počet 1). Na konci funkcie ešte skontrolujeme, či všetky prepínače na druhy paketov sú nastavené na `false` a ak sú, tak ich nastavíme všetky na `true` a teda budeme odchytávať všetky druhy paketov.

2.2 createFilter()

Po spracovaní argumentov následne voláme funkciu CREATEFILTER(), ktorej úlohou je vytvoriť nám filter na konkrétne druhy paketov, ktoré chceme odchytávať pre funkciu ďalej v programe. Funkcia funguje na jednoduchom princípe spájanie reťazcov, kde najprv skontrolujeme, či nám bol zadaný na vstupe nejaký konkrétny port a ak bol, tak do globálnej premennej pre filter pridávame reťazce vo formáte `([názov protokolu] port [číslo portu]) or` v prípade ak daný protokol podporuje porty a ak nepodporuje, tak je formát v tvare `([názov protokolu]) or`.
V prípade ak nám nebol na vstupe zadaný žiadny port, tak sa prepneme do vetvy druhej a využívame formát v tvare `([názov protokolu]) or`. Ako posledné vo funkcii na konci vykonáme odstránenie posledných troch znakov z celého reťazca filtra, ktoré tam necheme mať.

2.3 Zachytávanie paketov

Akonáhle sme si už vytvorili náš filter, tak vykonáme pár funkcií, ktoré sú potrebné pre zachytávanie paketov. Ako prvé sa vykoná funkcia `pcap_lookupnet()`, ktorá sa používa na určenie čísla siete IPv4 a masky súvisiacej so sieťovým zariadením(rozhraním).

Ako ďalšie vykonáme funkciu `pcap_open_live()`, ktorá sa používa na získanie popisovača zachytávania paketov na prezeranie paketov v sieti. V našom prípade si tento popisovač ukladáme ďalej do premennej `device`, ktorá je tvaru štruktúry `pcap_t`.

Následne vykonáme funkciu `pcap_datalink()`, ktorá slúži na skontrolovanie, či zariadenie, kotoré zachytávame je Ethernetové.

Ďalej ak zatiaľ všetko prebehlo vporiadku skontrolujeme, či máme vytvorený nejaký filter protocolov. Ak filter máme vytvorený, tak sa vykonaju funkcie `pcap_compile()`, kotrá sa používa na kompiláciu reťazca do filtrovacieho programu, ak kompilácia prebehne vporiadku vykoná sa druhá funkcia `pcap_setfilter()`, ktorá sa používa na špecifikáciu a nastavenie filtrovacieho programu.

Ak sa všetky funkcie doteraz vykonali bez problémov, tak vykonáme funkciu `pcap_loop()`, ktorá spracováva pakety zo živého vysielania pokiaľ neni spracovaný požadovaný počet paketov, ktorý je funkcii predaný v premennej. Pre spracovanie jednotlivých odchytených paketov je následne v tejto funkcii predávaná ako parameter ďalšia funkcia HANDLEPACKET(), ktorá sa už stará o spracovanie paketov a výpis požadovaných informácií.

2.4 Funkcia handlePacket()

V tejto funkcii na začiatku vypíšeme časovú stopu odchyteného paketu v časovom formáte `RFC3339`. K výpisu časovej stopy nám slúži pomocná funkcia `GetStrTimeStamp()`(3.5). Následne vytvoríme potrebné štruktúry pre IP hlavičky alebo pre jednotlivé pakety jednotlivých protokolov, pomocov ktorých budeme ďalej pristupovať k potrebným informáciám o odchytených paketoch.

Keď máme potrebné štruktúry vytvorené skontrolujeme typ paketu a to teda buď IPv4, IPv6 alebo ARP. Akonáhle zistíme o aký paket ide, napr. IPv4, tak sa dostaneme do vetvy pre IPv4 a tu pomocou prepínača zistíme na akom protokole bol jednotlivý paket zaslaný a to teda buď ICMPv4, TCP alebo UDP. V prípade iného protokolu len jednoducho skončíme spracovanie tohto paketu. V prípade ak teda zachytíme už niektorý z nami podporovaných protokolov, tak vypíšeme požadované informácie o paketoch a to zdrojovú a cielovú MAC adresu, dĺžku tohto paketu,zdrojovú a cielovú IP adresu, zdrojový a cielový port a na koniec vypíšeme obsah daného paketu. Výpis obsahu paketu vykonávame pomocou pomocnej funkcie `printPacketContent()`(3.7). Pri ostatných protokoloch je vypisovanie údajov na rovnakom princípe akurát výpis zdrojových a cielových portov je iba pri protokoloch TCP a UDP, ktoré tieto porty využívajú.

2.5 Koniec programu

Na konci programu sa už len uvolnia všetky prostriedky a program je korektne ukončený pomocov exit kódu 0, ktorý znamená, že celý program prebehol v poriadku a nenastal žiadny problém.

3 Pomocné funkcie

V programe sú používané rôzne pomocné funkcie na výpis rôznych dát a prípadne vykonanie konkrétnych akcií. Tieto funkcie sú samostatne popísané v nasledujúcej časti.

3.1 Funkcia signalHandler()

Táto funkcia nám slúži na kontrolu ak program obdrží signál na ukončenie, teda $CTRL + C$. Funkcia tento signál odchytí a postará sa o korektné ukončenie programu a uvolnenie prostriedkov.

3.2 Funkcia getInterfaces()

Táto funckia nám slúži k získaniu všetkých dotupných rozhraní. K získaniu rozhraní používame funkciu `pcap_findalldevs()`, ktorá vytvára zoznam sieťových zariadení. Funkcia následne vracia zoznam dostupných rozhraní, kde každé rozhranie je typu `pcap_if_t`.

3.3 Funkcia interfaceActive()

Táto funkcia sa stará o kontrolu, či rozhranie, ktoré sme dostali ako argument na vstupe sa nachádza v zozname dostupných rozhraní. Funkcia postupne prejde cez zoznam všetkých získaných rozhraní a porovná ich z požadovaným rozhraním. V prípade ak je rozranie v zozname, funkcia vráti honotu true pre kontrolnú bool premennú. Ináč je vypísaná chyba a program je ukončený s chybovým exit kódom 1.

3.4 Funkcia printInterfaces()

Funkcia postupne vypisuje jednotlivé získané rozhrania pokiaľ nedôjde na koniec zoznamu. Po ukončení výpisu funkcia uvolní všetky zdroje a korektne ukončí program.

3.5 Funkcia GetStrTimestamp

Funkcia, ktorá sa stará o výpis časovej stopy odchyteného paketu. Funkciazískava čas zo strojového časua vypisuje ho v požadovanom formáte RFC3339 aj spolu s milisekundami.

3.6 Function printIpv6()

Funckia, ktorá sa stará o výpis IPv6 adresy v správnom formáte.

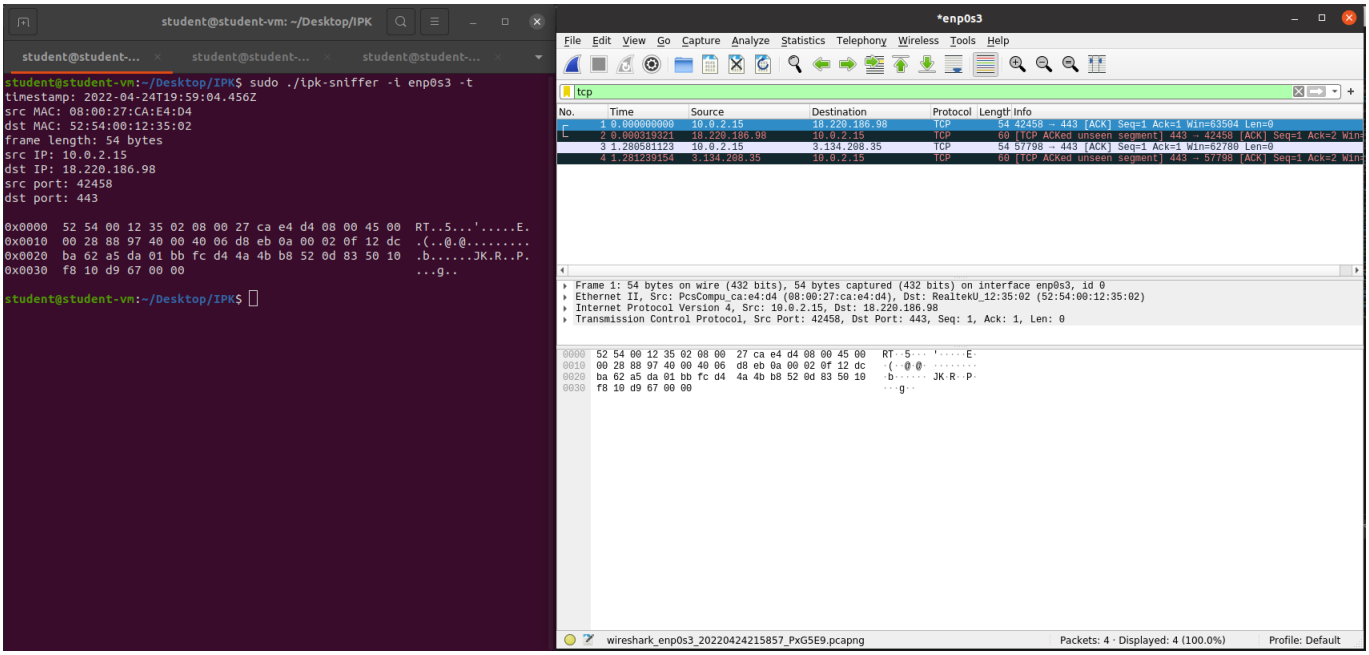
3.7 Funkcia printPacketContent

Funckia slúžiaca pre výpis celého obsahu paketu.

4 Testovanie

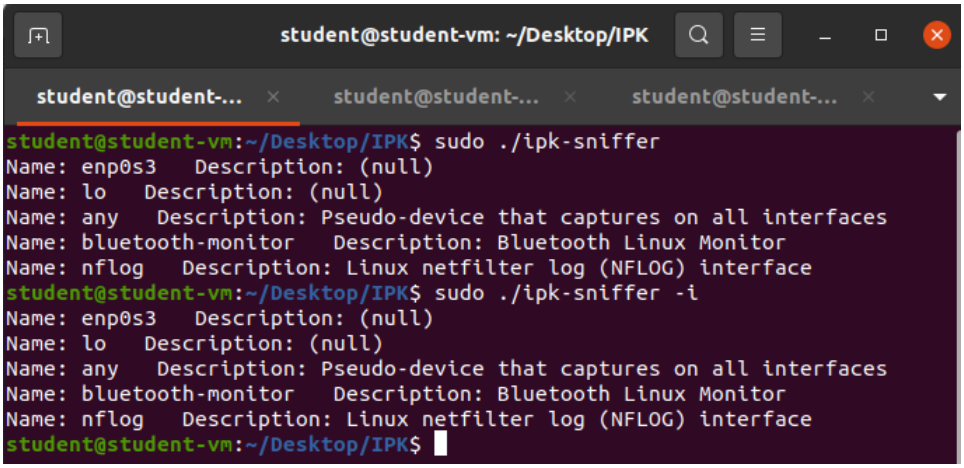
Testovanie funkčnosti programu sme vykonávali pomocou postupného odchytávania konkrétnych paketov a porovnávaním s naším výstupom a výstupom z programu WIRESHARK [1]. Bohužial sa mi však nepodarilo odtestovať funkčnosť odchytávania IPv6 paketov, nakoľko som nemal prístup k pripojeneie cez IPv6 adresu.

4.1 TCP packet



Obr. 1: Testovanie odchytenia TCP paketu

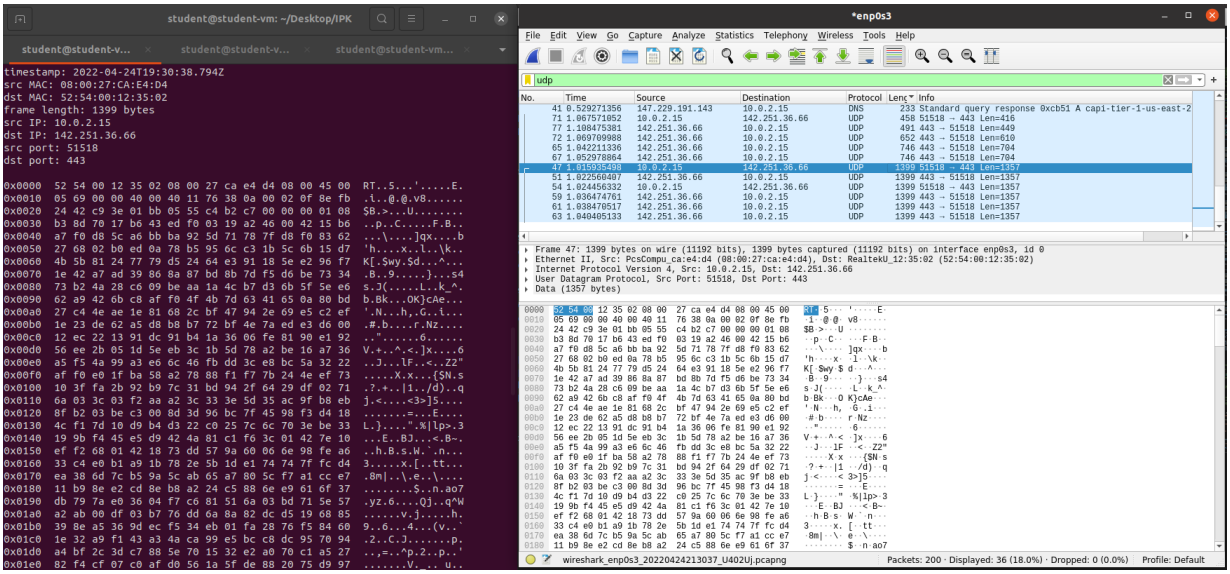
4.2 Výpis rozhraní



Obr. 2: Testovanie výpisu rozhraní

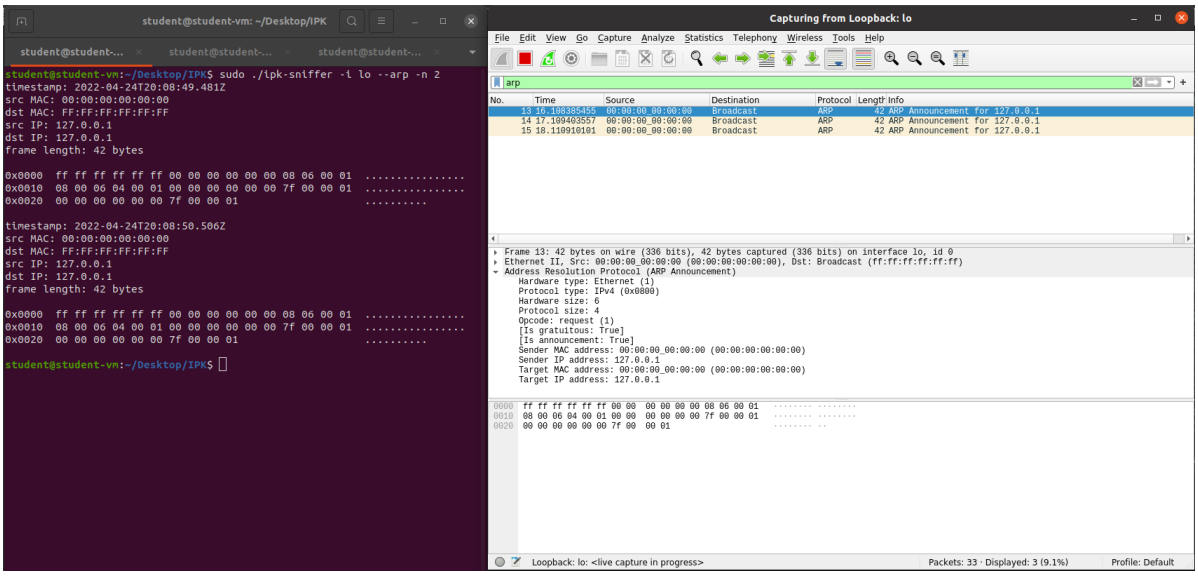
[1] Wireshark, program na odchytávanie paketov na sieti: https://www.wireshark.org/.

4.3 UDP paket



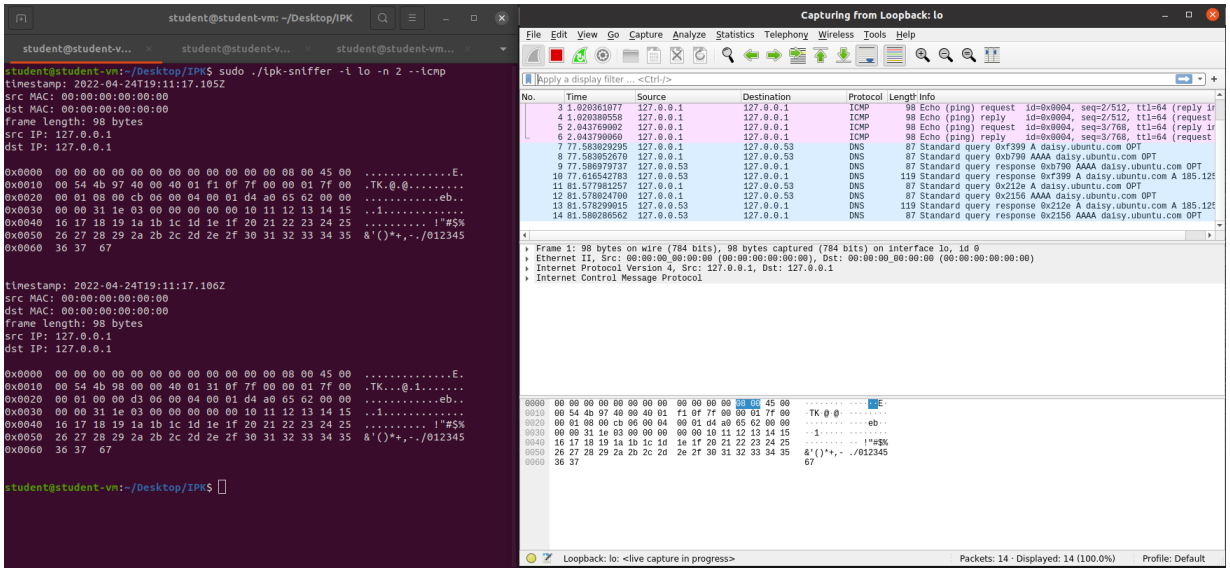
Obr. 3: Testovanie odchytenia UDP paketu

4.4 ARP paket

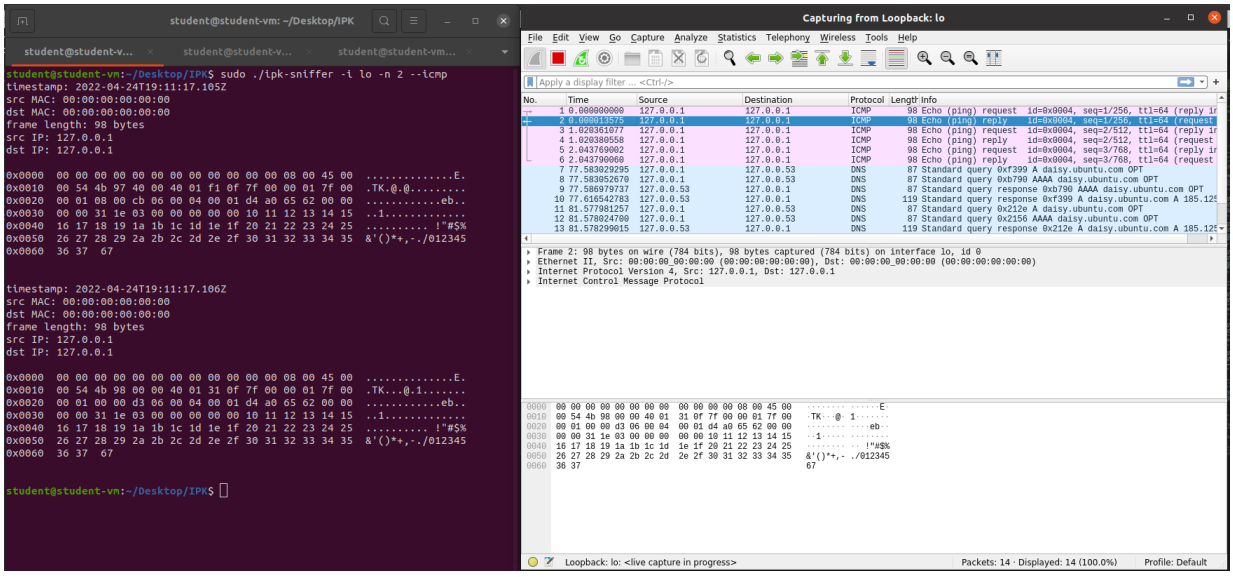


Obr. 4: Testovanie odchytenia ARP paketu

4.5 ICMP paket



Obr. 5: Testovanie odchytenia ICMP paketu



Obr. 6: Testovanie odchytenia ICMP paketu

5 Literatúra

- TCPDUMP/LIBPCAP public repository: http://www.tcpdump.org/
- Internet Assigned Numbers Authority: https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml
- Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/EtherType
- Hotexamples: https://cpp.hotexamples.com/examples/-/-/pcap$_f indalldevs/cpp-pcap_f indalldevs-function-examples.html$
- Stackoverflow: https://stackoverflow.com/questions/43724974/casting-pcap-if-t-to-pcap-if-t-in-c
- Stackoverflow: https://stackoverflow.com/questions/54325137/c-rfc3339-timestamp-with-milliseconds-using-stdchrono
- Tutorialspoint: https://www.tutorialspoint.com/how-do-i-catch-a-ctrlplusc-event-in-cplusplus
- Superglobalmegacorp: https://unix.superglobalmegacorp.com/Net2/newsrc/netinet/if$_e ther.h.html$
- Stackoverflow: https://stackoverflow.com/questions/5177879/display-the-contents-of-the-packet-in-c
- Programcreek: https://www.programcreek.com/cpp/?CodeExample=hex+dump
- Stackoverflow: https://stackoverflow.com/questions/3727421/expand-an-ipv6-address-so-i-can-print-it-to-stdout
- Geeksforgeeks: https://www.geeksforgeeks.org/internet-protocol-version-6-ipv6-header/
- Geeksforgeeks: https://www.geeksforgeeks.org/introduction-and-ipv4-datagram-header/