VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové komunikace a sítě – 2. projekt

Dokumentácia projektu
Varianta Zeta: Sniffer paketov

22. apríla 2022

Štefan Gajdošík - <xgajdo30>

Obsah

1 Úvod

Clieľom projektu bolo vytvoriť sieťový analyzátor, ktorý je schopný na určitom sieťovom rozhraní zachytávať a filtrovať pakety. Program je schopný zachytávať pakety protokolov ARP, TCP, UDP a ICMP. To všetko s podporou IPv4 aj IPv6. Zachytávanie paketov je taktiež možné filtrovať a to podľa: rozhrania, protokolu, alebo čísla portu. Existuje aj možnosť odchytávať určitý počet paketov (predvolená hodnota je jeden paket).

Paket je vypisovaný na štandardný výstup vo formáte: čas, zdrojová MAC adresa, cieľová MAC adresa, dĺžka rámca, zdrojová IP adresa (ak je), cieľová IP adresa (ak je), zdrojový port (ak je), cieľový port (ak je). Následne je vypísaný aj paket vo formáte podobnom výstupu programu `hexdump`.

2 Implementácia

2.1 Použitý jazyk

Pri práci na projekte som využil jazyk C++ a zdrojový kód som prekladal pomocou programu `g++`. Hlavnou motiváciou k zvoleniu tohto jazyka bola existencia `std::string` čo výrazne zjednodušilo vytváranie filtrovacieho reťazca.

2.2 Použité knižnice

Pri práci na projekte som využíval 3 typy knižníc:

C++ knižnice

- `<iostream>`
- `<ctime>`

Knižnice na odchytávanie paketov a ich štruktúry

- `<netinet/if_ether.h>`
- `<netinet/ip.h>`
- `<netinet/in.h>`
- `<net/ethernet.h>`
- `<pcap/pcap.h>`
- `<netinet/tcp.h>`
- `<netinet/udp.h>`

- <netinet/icmp6.h>
- <netinet/ip_icmp.h>
- <netinet/ip6.h>

Knižnice jazyka C

- <getopt.h>
- <string.c>

2.3 Funkcia `main()`

Na začiatku funkcie `main` parsujem argumenty pomocou knižnicovej funkcie `getopt_long` za využitia štruktúry `option` a reťazca `short_opts`. Následne skontrolujem či bolo zadané rozhranie na ktorom majú byť pakety odchytávané, a ak nie, všetky sú vypísané pomocou funkcie `get_and_print_all_interfaces()` a program je ukončený.

Ak je rozhranie zadané, skontrolujem, či je potrebné upraviť filtrovací reťazec (predvolená hodnota odchytáva pakety **UDP**[na všetkých portoch], **TCP**[na všetkých portoch], **ARP** aj **ICMP**) a ak áno, je zavolaná funkcia `create_filter()` ktorá vytvorí filtrovací reťazec podľa zadaných argumentov.

Po vytvorení filtrovacieho reťazca sú volané funkcie `pcap_open_live()` na otvorenie zvoleného sieťového rozhrania, `pcap_compile()` na parsovanie filtrovacieho reťazca `pcap_setfilter()` ktorá spracovaný filter aplikuje na rozhranie. Pokiaľ všetky 3 zmienené funkcie prebehnú úspešne, je volaná funkcia `pcap_loop()` na rozhraní, s parametrom `NUMBER_OF_PACKETS` ktorý určuje koľko paketov má byť odchytených a "call-back"funkciou `packet_parsing()`.

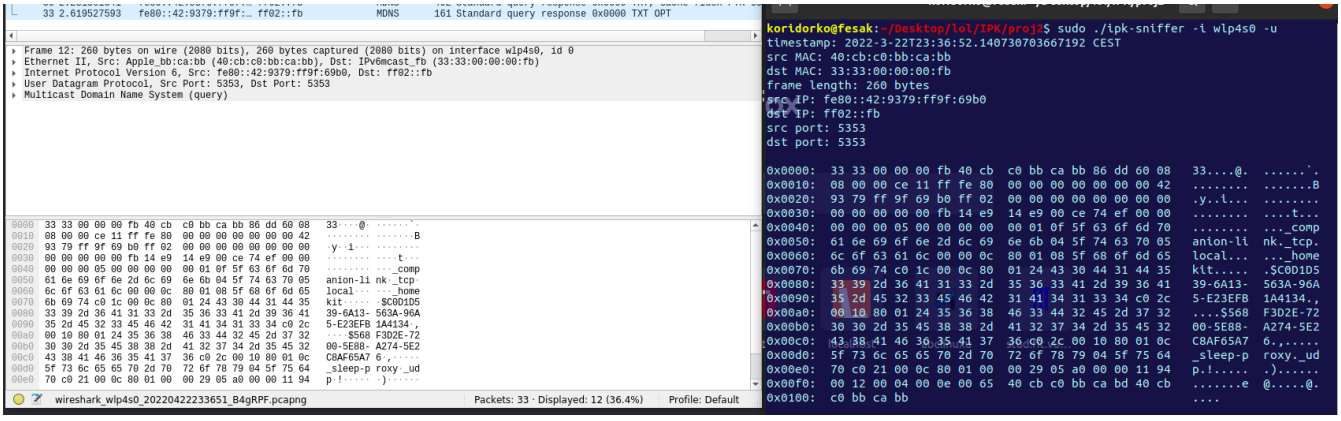
2.4 Funkcia `packet_parsing()`

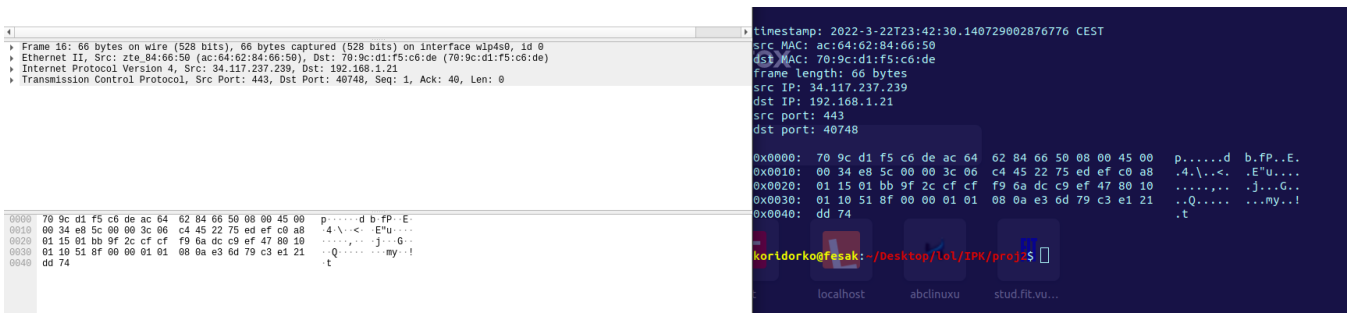
Fukncia `packet_parsing()` na začiatku deklaruje štruktúry pre každý typ paketu, ktorý je program schopný spracovávať a následne zavolá funkciu `get_time()` ktorá preformátuje `timestamp` paketu z formátu `Epoch-time` do formátu `RFC3339` a vytlačí ho na štandardný výstup. Potom sa začne vykonávať 2-úrovňový `switch:case` blok kódu, ktorý zisťuje na prvej úrovni či je paket typu **ARP** alebo **IPv4** alebo **IPv6**. Ak sa jedná o paket typu **ARP**, sú vytlačené MAC aresy, dĺžka rámca, a jeho obsah a program je buď ukončný alebo odchytáva ďalšie pakety.

Ak sa jedná o paket typu **IPv4** alebo **IPv6**, je následne zistené aký konkrétne (**UDP**, **TCP**, **ICMP**) a sú vytlačené údaje o nich a ich obsahy.

3 Testovanie a porovnanie s referenčným programom

Na testovanie som používal príkazy `curl` prípadne `curl -6`, a výsledky som porovnával s referenčným programom `wireshark`, pretože poskytuje funkcionalitu ktorá mi umožnila otestovať každý aspekt projektu. Prikladám aj 2 obrázky zobrazujúce správne odchytenie paketu.





4 Zdroje použité pri projekte

Literatúra

[1] Address Resolution Protocol. [vid. 16.4.2022]. Dostupné z: <https://cs.wikipedia.org/wiki/Address_Resolution_Protocol>

[2] Assigned Internet Protocol Numbers. [vid. 16.4.2022]. Dostupné z: <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>

[3] inet_ntop(3) — Linux manual page. [vid. 16.4.2022]. Dostupné z: <https://man7.org/linux/man-pages/man3/inet_ntop.3.html?fbclid=IwAR092i5b10QlQbid1_wbS1WM97TCUfNNO2MLPB1vrVyZqu4TPGRUU_-7t6w>

[4] IP aresa. [vid. 16.4.2022]. Dostupné z: <https://cs.wikipedia.org/wiki/IP_adresa>

[5] IPv4. [vid. 16.4.2022]. Dostupné z: <https://cs.wikipedia.org/wiki/IPv4>

[6] IPv6. [vid. 16.4.2022]. Dostupné z: <https://cs.wikipedia.org/wiki/IPv6>

[7] A Recommendation for IPv6 Address Text Representation. [vid. 16.4.2022]. Dostupné z: <https://datatracker.ietf.org/doc/html/rfc5952>

[8] Transmission Control Protocol. [vid. 16.4.2022]. Dostupné z: <https://cs.wikipedia.org/wiki/Transmission_Control_Protocol>

[9] User Datagram Protocol. [vid. 16.4.2022]. Dostupné z: <https://cs.wikipedia.org/wiki/User_Datagram_Protocol>

[10] of California, U.: Source to netinet/if_ether.h. [vid. 16.4.2022]. Dostupné z: <https://unix.superglobalmegacorp.com/NetBSD-0.8/newsrc/netinet/if_ether.h.html>

[11] Carstens, T.: Programing with PCAP. [vid. 16.4.2022]. Dostupné z: <https://www.tcpdump.org/pcap.html>

[12] GNU: Parsing Long Options with getopt_long. [vid. 16.4.2022]. Dostupné z: <https://www.gnu.org/software/libc/manual/html_node/Getopt-Long-Options.html>

[13] NanoDano: Using libpcap in C. [vid. 16.4.2022]. Dostupné z: <https://www.devdungeon.com/content/using-libpcap-c>

[14] Programiz: C++ gmtime(). [vid. 16.4.2022]. Dostupné z: <https://www.programiz.com/cpp-programming/library-function/ctime/gmtime>