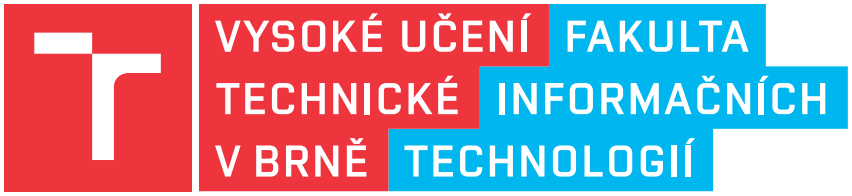


2. Projekt [IPK]
Snifer paketů (Varianta zeta)
Implementační dokumentace/manuál

11. dubna 2021 Jakub Komárek (xkomar33)

Obsah

1 Úvod

Cíl projektu byl síťový analyzátor, který bude schopný na určitém síťovém rozhraním zachytávat a filtrovat pakety.

Program podporuje zachytávání paketů na ethernetovém rozhraní. Je zde možnost selekce typu zachytávaných paketů podle typu protokolu, který daný paket využívá. Po zachycení se vypíše čas zachycení, cílová a zdrojová adresa spolu s portem (pokud paket obsahuje port). Následně se vypíše celý paket v přehledném formátování v hexadecimální reprezentaci a v ASSCI podobě.

2 Implementace

Pro implementaci jsem zvolil programovací jazyk `C++` s využitím knihovny `pcap` [8]. Tuto knihovnu jsem použil na „otevření" internetového rozehraní, filtrování a zachycení jednotlivých paketů. Poté jsem využil struktury z knihoven `netinet` [6] pro snadnější manipulaci s daty.

Základní znalosti o parsování paketů jsem nabyl z výukové stránky [5]. Program z těchto znalostí vychází.

2.1 Popis průběhu programu

Po spuštění programu se provede zpracování vstupních parametrů pomocí funkce `argParste`. Nesprávné parametry způsobí chybu. V případě nezadaného rozhraní se vypíší všechna dostupná rozhraní. V opačném případě se program pokusí rozhraní otevřít (opět v případě neúspěchu se vypíše chyba). Poté proběhne vytvoření filtru podle vstupních parametrů funkcí `filterGen` a jeho následná aplikace na otevřený popisovač. Poté začne proces zachytávání paketů pomocí funkce `sniffPacket` . V případě ukončení programu se odalokuje popisovač a program se ukončí.

2.2 Funkce sniffPacket

Tato funkce zpracovává jednotlivé pakety. Začne se získáním ukazatele na paket pomocí knihovní funkce `pcap_next`. Poté se naplní ethernetová hlavička (struktura `ether_header`). Ethernetová hlavička leží na začátku paketu->není potřeba dělat bytový posun. Vypíše se čas zachycení paketu funkcí `RFC3339`. Podle položky `ether_type` ve zmiňované struktuře se zvolí protokol síťové vrstvy (`IPv4, IPv6, ARP`), jiné protokoly této vrstvy nejsou podporovány.

2.2.1 ARP

V případě `ARP` protokolu se vypíše pouze `MAC` adresy odesílatele a příjemce. Tyto adresy jsou uloženy v hlavičce Ethernetu[1].

2.2.2 IPv4

V tomto případě se zpracuje `IP` hlavička paketu. Tato hlavička je umístěna pod ethernetovou hlavičkou, proto se musí k ukazateli na paket přičíst délka ethernetové hlavičky. Podle položky `_p` v této struktuře, která udává jaký transportní protokol daný paket používá (program podporuje `TCP,UDP,ICMP`) se rozhodne další manipulace s tímto paketem.

V případě `ICMP` protokolu se vypíše `IP` adresa odesílatele a příjemce. Porty v tomto protokolu nefigurují[2].

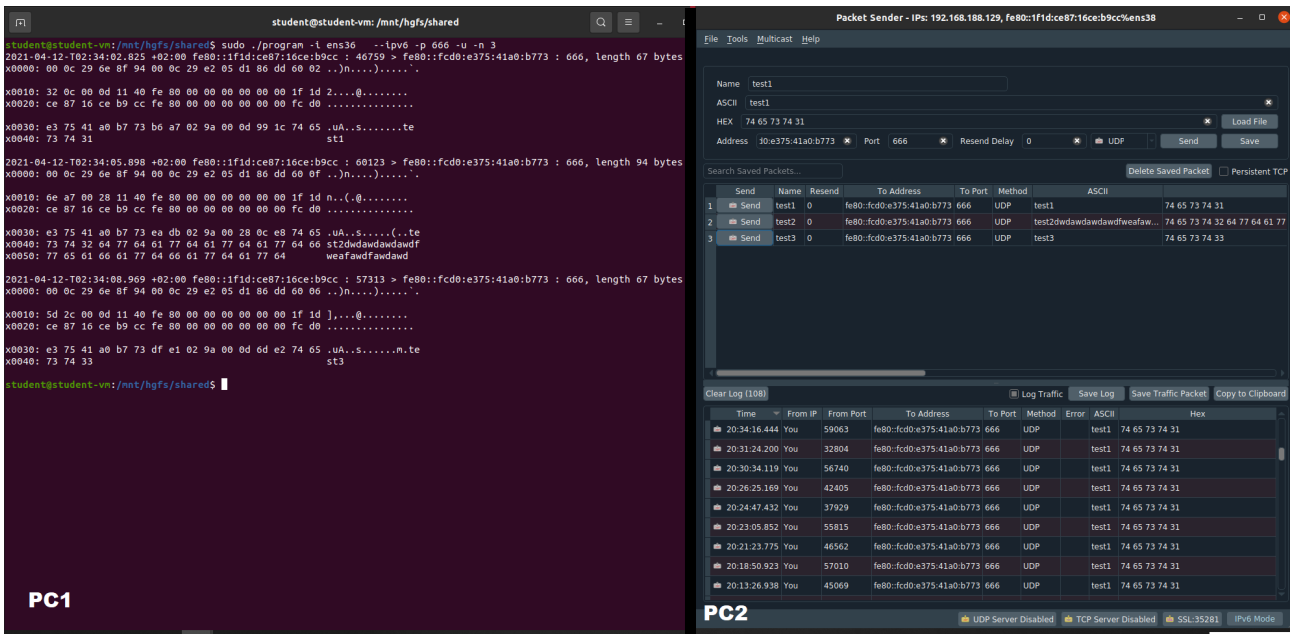
Pokud je zvolen `TCP/UDP` protokol, tak se podobným způsobem nahrají hlavičkové informace do odpovídajících struktur. Ve výpisu krom `IP` adres se zobrazí zdrojové a cílové porty daného paketu.

2.2.3 IPv6

Zpracování paketu je podobné jako v předchozím případě. Opět se vytvoří a naplní vhodná struktura pro hlavičku. Zde ale přichází menší komplikace v podobě variabilního počtu hlaviček[4]. Problém jsem vyřešil cyklem, který postupně projde všechny hlavičky a při tom vypočte bajtový offset. Pokud transportní protokol je typu `ICMP6`, je vypsána pouze dvojice adres(příjemcova a odesílatele). V případě `TCP/UDP` se postupuje stejným způsobem jako u `IPv4` s jediným rozdílem, že pro vyplnění hlavičky těchto protokolů se využije vypočtený offset ze zmiňovaného průchodu hlavičkami(`IPv4` hlavička má o své délce záznam ve své hlavičce[3]).
Po výpisu výše zmíněných adres a v určitých situací portů se připíše ještě celková délka paketu. Poté se přes dvojitý cyklus vypíše celý paket v přehledném formátu v `hexadecimální` a `ASSCI` podobě.

3 Testování

Testování probíhalo na virtuálním stroji pomocí programu `VMware`. Použil jsem doporučený testovací systém. Testoval jsem manuálně pomocí programu `Packet Sender` [7] dostupný v Ubuntu obchodu. Správnou funkčnost jsem ověřil porovnáním zadaných dat s výpisy programu. Implementoval jsem do kódu ladící výpisy, pro snadnější kontrolu vnitřních stavů programu (ladící mód se zapíná parametrem -d). Jelikož samotné parsování paketu se skládá ze dvou velkých sekcí (`IPv4 a IPv6`) přidal jsem možnost upravit filtrování dvěma parametry navíc (`-ipv6 a -ipv4`) pro snadnější testování.



Obrázek 1: Ukázka průběhu testování na 2 virtuálních počítačích

Protokoly, které nešly testovat tímto softwarem(ICMP, ICMP6) jsem otestoval individuálně přes program `ping`.

4 Zdroje

Odkazy

[1] *Address Resolution Protocol*. URL: https://cs.wikipedia.org/wiki/Address_Resolution_Protocol. (navštíveno: 11.04.2021).

[2] *Internet Control Message Protocol*. URL: https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol. (navštíveno: 11.04.2021).

[3] *Internet Protocol version 4*. URL: https://en.wikipedia.org/wiki/IPv4. (navštíveno: 11.04.2021).

[4] *Internet Protocol Version 6*. URL: https://en.wikipedia.org/wiki/IPv6. (navštíveno: 11.04.2021).

[5] Luis MartinGarcia. *Programming with pcap*. URL: https://www.tcpdump.org/pcap.html. (navštíveno: 11.04.2021).

[6] *netinetin.h*. URL: https://pubs.opengroup.org/onlinepubs/7908799/xns/netinetin.h.html. (navštíveno: 11.04.2021).

[7] *Packet Sender Documentation*. URL: https://packetsender.com/documentation. (navštíveno: 11.04.2021).

[8] *pcap*. URL: https://pcapplusplus.github.io/. (navštíveno: 11.04.2021).