Vysoké učení technické v Brně
Fakulta informačních technologií

POČÍTAČOVÉ KOMUNIKACE A SÍTĚ

2020/2021

Projekt IPK 2

Zadání DELTA – Scanner síťové dostupnosti

Tomáš Milostný (xmilos02)

Obsah

Obsah 2
Úvod 3
Popis implementace 4
ArgumentParser 4
SubnetParser 4
Subnet 5
NetworkScanner 5
IcmpPacket 5
Testování 6
Příklad s IPv4 6
Příklad s IPv6 8
Seznam literatury 9

Úvod

Projekt je implementovaný v jazyce **C# 9.0** nad platformou **.NET 5.0**. Využívá knihovny *System.CommandLine* (viz *ArgumentParser*) a *SharpPcap* (viz *NetworkScanner*).
Program skenuje zadané rozsahy IP adres pomocí protokolů ICMP/ICMPv6 a ARP. Oproti původnímu zadání nebyla implementována práce s protokolem NDP pro IPv6 subnety.

Popis implementace

Jádrem projektu je hlavní program v souboru `Program.cs` psaný ve stylu *Top-level statements* představeném v C# 9.0. Program začíná načítáním argumentů příkazové řádky pomocí třídy `ArgumentParser` a její statické metody `ParseArguments`, která vrací trojici zpracovaných argumentů:

- `NetworkInterface @interface` s informacemi o zadaném síťovém rozhraní.
- `int timeout` označující časový limit v milisekundách pro jednotlivé sockety vytvářené později.
- `Subnet[] subnets` – pole s objekty typu `Subnet` vytvořených ze všech sítí zadaných argumenty --**subnet**/**-s**.

ArgumentParser může vyhazovat výjimky, které jsou tedy ošetřeny v try-catch bloku, více o těchto výjimkách v sekci ArgumentParser. Výjimka pro chybějící parametr --**interface**/**-i** je ošetřena voláním funkce `PrintAllInterfaces`, která vypíše všechna dostupná rozhraní a ukončí program.

Pokud jsou všechny argumenty příkazové řádky v pořádku, pokračuje se voláním funkce `ListScanningRanges`, která vypíše všechny načtené subnety a počet IP adres v jejich rozsahu. Následuje funkce `ScanNetwork`, kde se pro jednotlivé subnety vytváří instance třídy `NetworkScanner` a zavolá se její metoda `ScanAsync`, která projde rozsah IP adres a zjistí dostupnost pomocí protokolů ICMP/ICMPv6 a ARP (více v sekci NetworkScanner).

ArgumentParser

Třída se statickou metodou `ParseArguments`, která pole `string[] args` pomocí knihovny `System.CommandLine` [1] převede na trojici `NetworkInterface, int, Subnet[]`. Zpracovává argumenty:

- --**interface**/**-i –** Název rozhraní parsovaný na objekt typu `NetworkInterface`. Chybějící nebo argument zadaný špatným jménem rozhraní vyhodí výjimku `InvalidOperationException`, kterou generuje Linq metoda `First` vyhledávající v seznamu dostupných síťových rozhraní
- --**wait**/**-w –** Celé číslo reprezentující časový limit v milisekundách pro sockety (výchozí hodnota 5000ms). Pokud je zadaná hodnota záporná, je vyhozena výjimka `IndexOutOfRangeException`.
- --**subnet**/**-s** – IPv4 nebo IPv6 adresa s maskou parsovaná na objekt typu `Subnet` třídou `SubnetParser`, může jich být zadáno více než 1
- --**help**/**-h**/**-?** – výpis nápovědy programu
- --**version** – přidáno knihovnou System.CommandLine, vypíše verzi programu

SubnetParser

Třída se statickou metodou `ParseSubnets`, která podle řetězců v poli `string[] subnets` vytvoří pole objektů typu `Subnet`. Tato metoda je navržena asynchronně, což jí umožňuje pracovávat parametry v poli paralelně privátní statickou metodou `ParseSubnet` a vliv počtu parsovaných subnetů je tedy minimální. Rovněž během vytváření objektu kontroluje validitu zadané IP adresy a velikosti masky sítě.

Subnet

Třída držící informace o skenované síti. Udržuje IP adresu zadanou parametrem a masku podsítě, která je privátní metodou `CreateMask` vytvořena jako pole bytů na základě zadaného číselného parametru délky masky a verzi protokolu IP, tedy IPv4 nebo IPv6. Tato maska je následně privátní metodou `ApplyMask` aplikována na zadanou IP adresu bitovým maskováním, které vynuluje případné přebytečné bity nepatřící do adresy sítě. Umožňuje rovněž inkrementaci IP adresy operátorem `++` a veřejnou metodu `IsAtMaxIpAddress` pro zjištění, zda je aktuální adresa na poslední možné v rozsahu daném maskou.

NetworkScanner

Třída síťového skeneru. Konstruktor vyžaduje síťové rozhraní, časový limit pro sockety a skenovanou síť reprezentovanou objektem typu `Subnet`. Dle verze IP skenované sítě se z rozhraní uloží IP adres skenujícího zařízení a verze protokolu ICMP.

Samotné skenování je spuštěno voláním metody `ScanAsync`, která je navržena asynchronně, takže umožňuje daný rozsah IP adres oskenovat paralelně. Pro každou IP adresu z rozsahu subnetu je vytvořen task `EchoAsync`, který vrací čtveřici určující úspěšnost ICMP požadavku a ARP požadavku, cílovou IP adresu pro zpětnou identifikaci ve výpisu výsledků a MAC adresu získanou z požadavku ARP.

`EchoAsync` začíná vytvořením raw ICMP socketu, napojením zdrojové (`Bind`) a cílové (`ConnectAsync`) IP adresy, a odesláním (`SendAsync`) packetu vytvořeným strukturou `IcmpPacket`. Poté čeká na odpověď (`ReceiveAsync`), které je rovněž předán parametr `CancellationToken` s nastaveným časovým limitem.

Pokud nedošlo ke zrušení operace, je délka přijatých dat uložena do proměnné `icmpLength`, a pokud je větší než 0, je toto volání označeno za úspěch. Pro IPv4 je následně s pomocí knihovny `SharpPcap` vytvořen v metodě `GetArp` vytvořen a zachycen ARP [2] dotaz na MAC adresu zařízení, která je metodou vrácena, pokud nedošlo k chybě nebo vypršení časového limitu. V případě úspěšného získání MAC adresy, je ARP požadavek označen za úspěch a `EchoAsync` končí s výsledky, které jsou zobrazeny na standardní výstup.

IcmpPacket

Struktura ICMP packetu [3]. Obsahuje statickou metodu `EchoRequestAsBytes`, která tuto strukturu převede na pole bytů odeslané raw socketem. V této metodě je vytvořena struktura packetu, kde jsou vyplněny potřebné atributy.

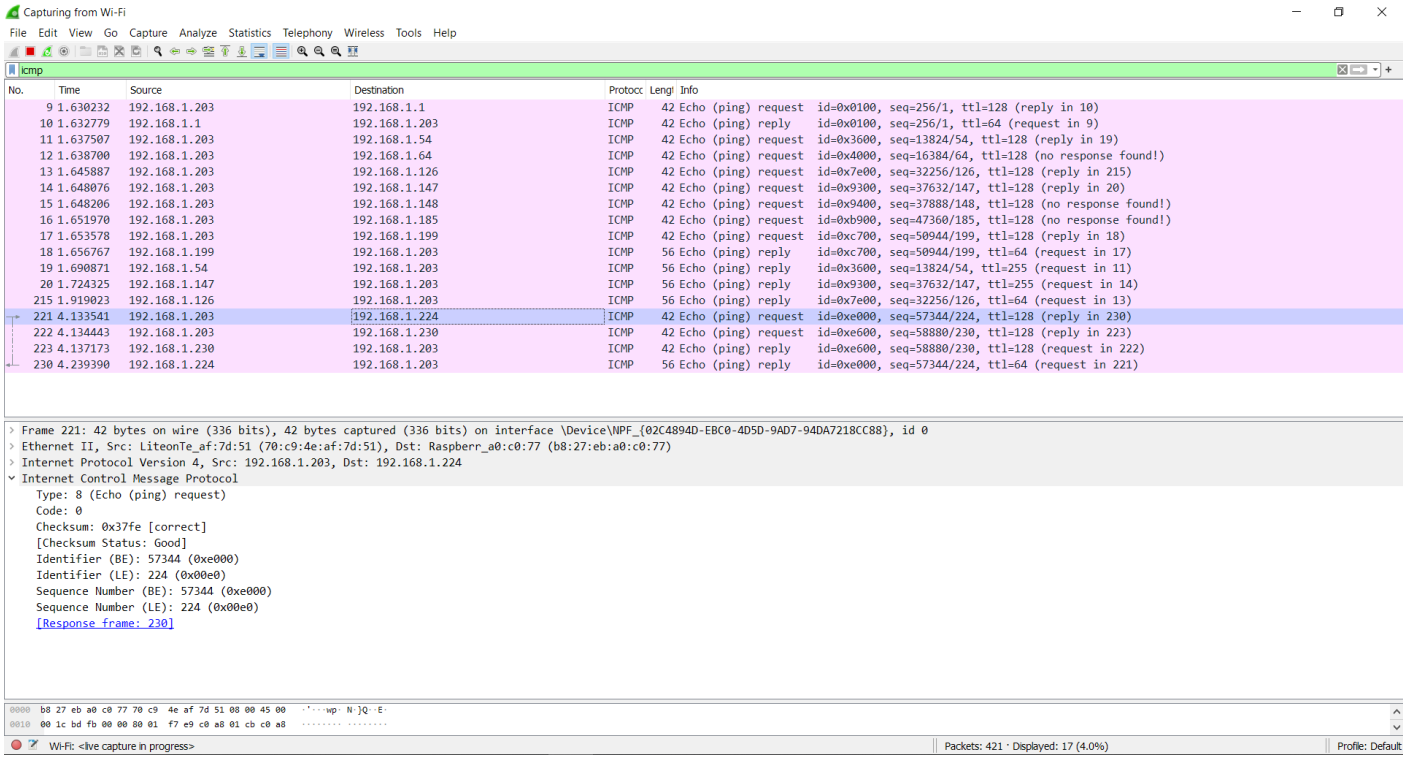
`TypeCode` je prvních 16 bitů značící typ echo požadavku lišící se podle verze protokolu ICMP [4] nebo ICMPv6 [5]. `Checksum` [6] je spočítána jako negovaný součet `TypeCode`, `Id`, `Sequence`.

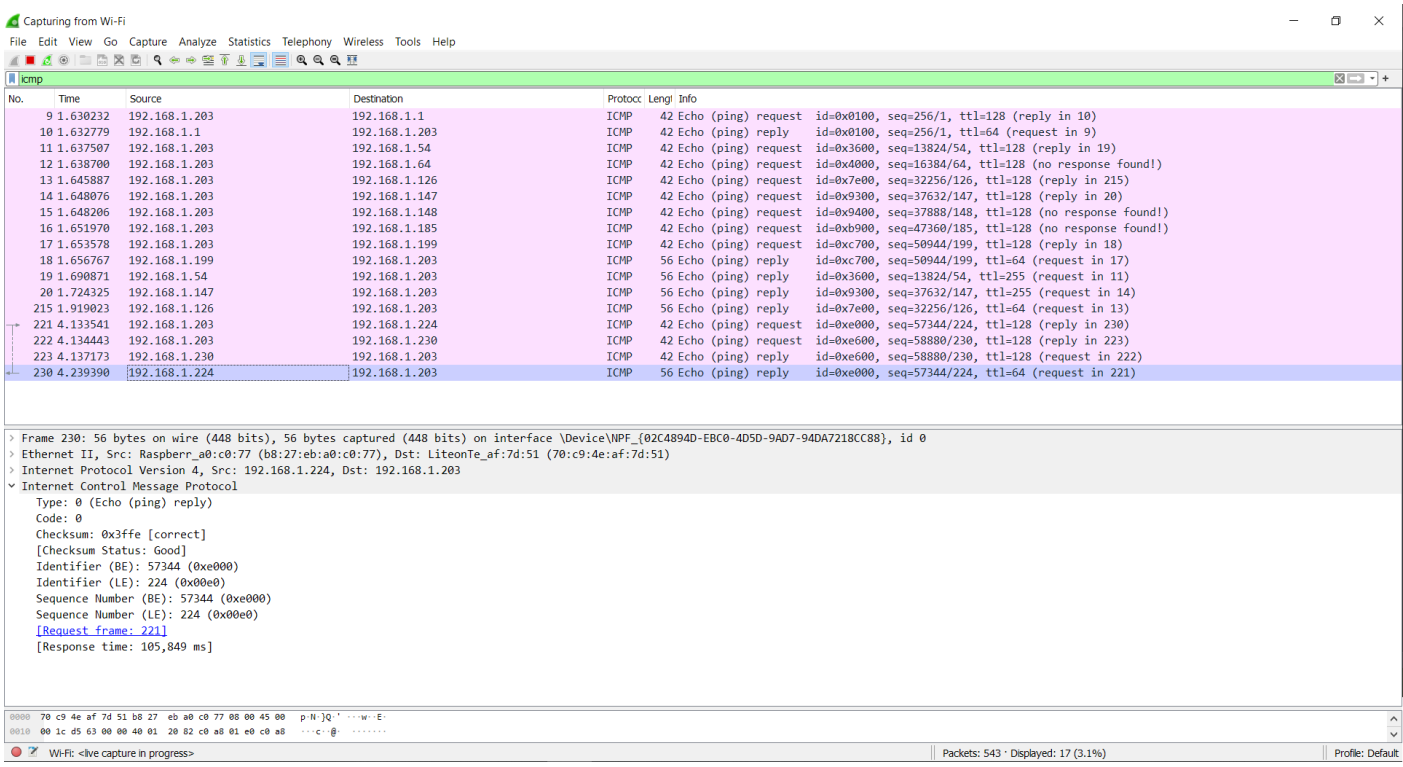
Poté je pomocí metod `System.Runtime.InteropServices.Marshal` pro práci s automaticky nespravovanou pamětí převedena struktura packetu na binární reprezentaci odeslanou v raw socketu.

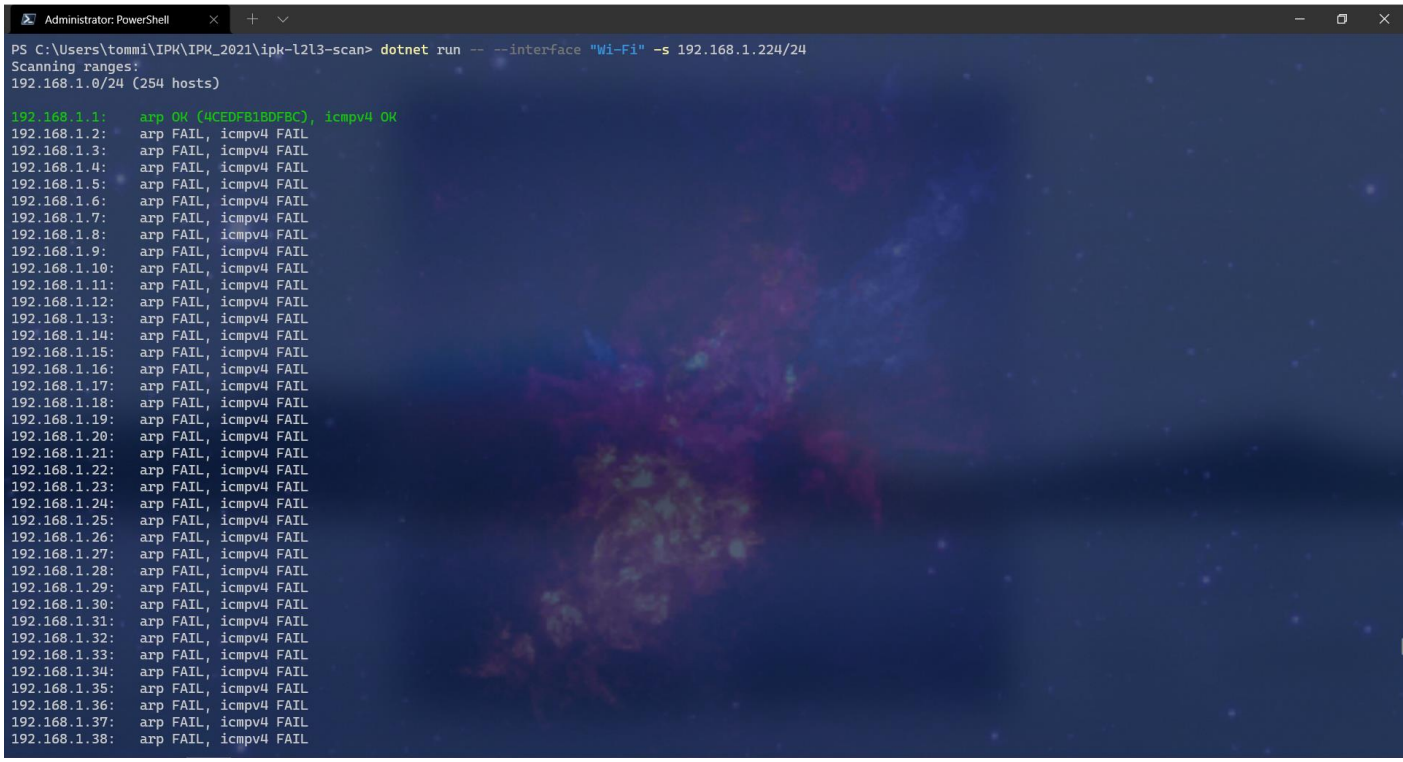
Testování

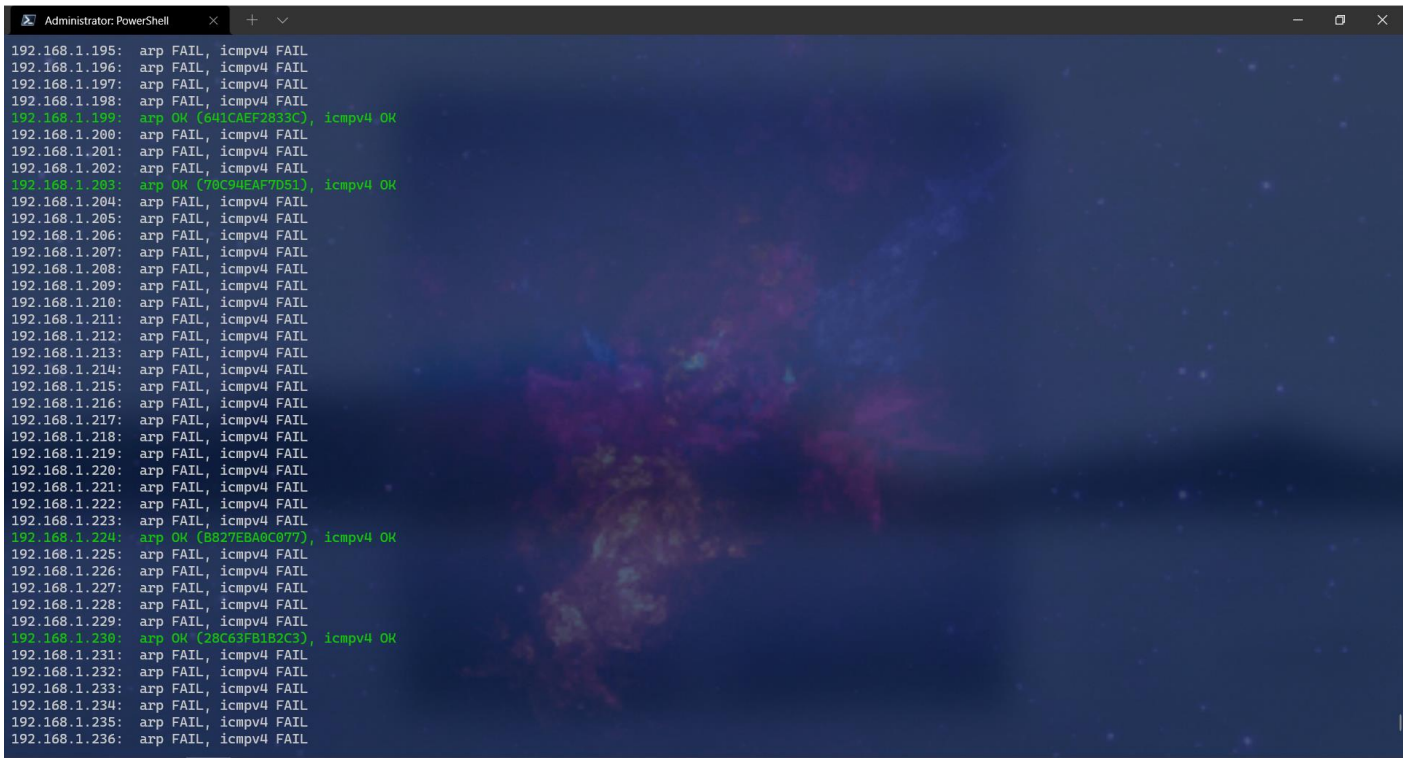
Příklad s IPv4

Spuštění programu: `dotnet run -- --interface "Wi-Fi" -s 192.168.1.224/24`
Možno vysledovat ICMP požadavky i odpovědi v aplikaci Wireshark.

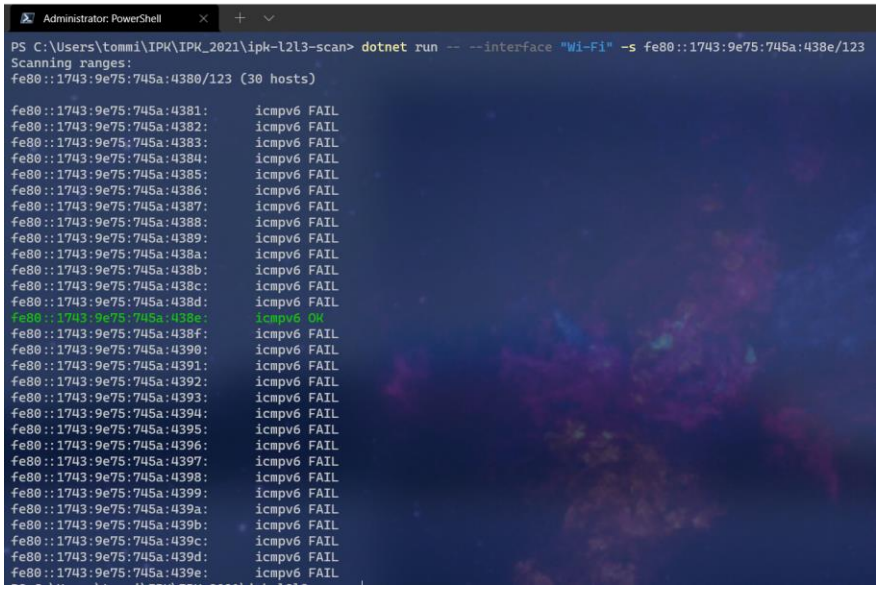


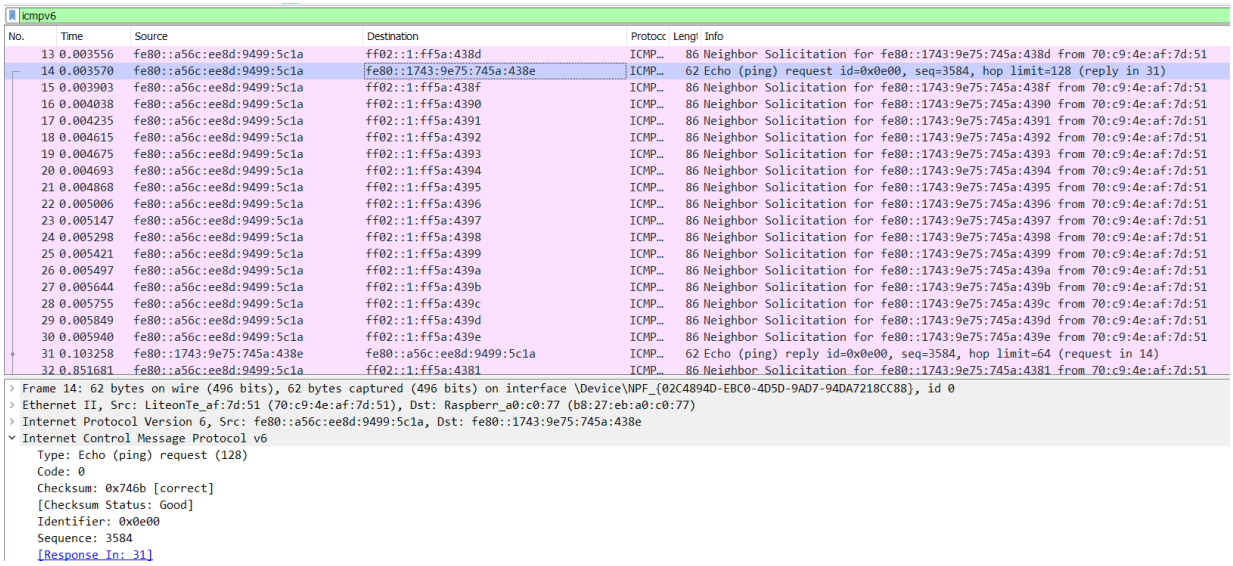


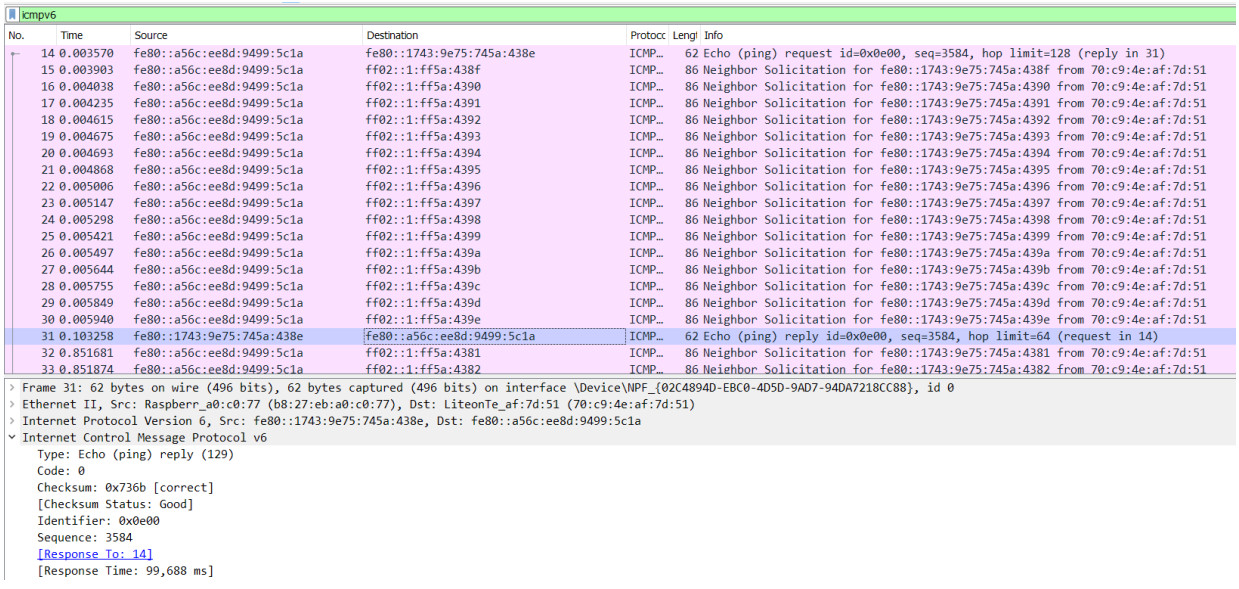

Administrator: PowerShell
PS C:\Users\tommi\IPK\IPK_2021\ipk-l2l3-scan> dotnet run -- --interface "Wi-Fi" -s 192.168.1.224/24
Scanning ranges:
192.168.1.0/24 (254 hosts)
192.168.1.1: arp OK (4CEDFB1BDFBC), icmpv4 OK
192.168.1.2: arp FAIL, icmpv4 FAIL
192.168.1.3: arp FAIL, icmpv4 FAIL
192.168.1.4: arp FAIL, icmpv4 FAIL
192.168.1.5: arp FAIL, icmpv4 FAIL
192.168.1.6: arp FAIL, icmpv4 FAIL
192.168.1.7: arp FAIL, icmpv4 FAIL
192.168.1.8: arp FAIL, icmpv4 FAIL
192.168.1.9: arp FAIL, icmpv4 FAIL
192.168.1.10: arp FAIL, icmpv4 FAIL
192.168.1.11: arp FAIL, icmpv4 FAIL
192.168.1.12: arp FAIL, icmpv4 FAIL
192.168.1.13: arp FAIL, icmpv4 FAIL
192.168.1.14: arp FAIL, icmpv4 FAIL
192.168.1.15: arp FAIL, icmpv4 FAIL
192.168.1.16: arp FAIL, icmpv4 FAIL
192.168.1.17: arp FAIL, icmpv4 FAIL
192.168.1.18: arp FAIL, icmpv4 FAIL
192.168.1.19: arp FAIL, icmpv4 FAIL
192.168.1.20: arp FAIL, icmpv4 FAIL
192.168.1.21: arp FAIL, icmpv4 FAIL
192.168.1.22: arp FAIL, icmpv4 FAIL
192.168.1.23: arp FAIL, icmpv4 FAIL
192.168.1.24: arp FAIL, icmpv4 FAIL
192.168.1.25: arp FAIL, icmpv4 FAIL
192.168.1.26: arp FAIL, icmpv4 FAIL
192.168.1.27: arp FAIL, icmpv4 FAIL
192.168.1.28: arp FAIL, icmpv4 FAIL
192.168.1.29: arp FAIL, icmpv4 FAIL
192.168.1.30: arp FAIL, icmpv4 FAIL
192.168.1.31: arp FAIL, icmpv4 FAIL
192.168.1.32: arp FAIL, icmpv4 FAIL
192.168.1.33: arp FAIL, icmpv4 FAIL
192.168.1.34: arp FAIL, icmpv4 FAIL
192.168.1.35: arp FAIL, icmpv4 FAIL
192.168.1.36: arp FAIL, icmpv4 FAIL
192.168.1.37: arp FAIL, icmpv4 FAIL
192.168.1.38: arp FAIL, icmpv4 FAIL


Administrator: PowerShell
192.168.1.195: arp FAIL, icmpv4 FAIL
192.168.1.196: arp FAIL, icmpv4 FAIL
192.168.1.197: arp FAIL, icmpv4 FAIL
192.168.1.198: arp FAIL, icmpv4 FAIL
192.168.1.199: arp OK (641CAEF2833C), icmpv4 OK
192.168.1.200: arp FAIL, icmpv4 FAIL
192.168.1.201: arp FAIL, icmpv4 FAIL
192.168.1.202: arp FAIL, icmpv4 FAIL
192.168.1.203: arp OK (70C94EAF7D51), icmpv4 OK
192.168.1.204: arp FAIL, icmpv4 FAIL
192.168.1.205: arp FAIL, icmpv4 FAIL
192.168.1.206: arp FAIL, icmpv4 FAIL
192.168.1.207: arp FAIL, icmpv4 FAIL
192.168.1.208: arp FAIL, icmpv4 FAIL
192.168.1.209: arp FAIL, icmpv4 FAIL
192.168.1.210: arp FAIL, icmpv4 FAIL
192.168.1.211: arp FAIL, icmpv4 FAIL
192.168.1.212: arp FAIL, icmpv4 FAIL
192.168.1.213: arp FAIL, icmpv4 FAIL
192.168.1.214: arp FAIL, icmpv4 FAIL
192.168.1.215: arp FAIL, icmpv4 FAIL
192.168.1.216: arp FAIL, icmpv4 FAIL
192.168.1.217: arp FAIL, icmpv4 FAIL
192.168.1.218: arp FAIL, icmpv4 FAIL
192.168.1.219: arp FAIL, icmpv4 FAIL
192.168.1.220: arp FAIL, icmpv4 FAIL
192.168.1.221: arp FAIL, icmpv4 FAIL
192.168.1.222: arp FAIL, icmpv4 FAIL
192.168.1.223: arp FAIL, icmpv4 FAIL
192.168.1.224: arp OK (B827EBA0C077), icmpv4 OK
192.168.1.225: arp FAIL, icmpv4 FAIL
192.168.1.226: arp FAIL, icmpv4 FAIL
192.168.1.227: arp FAIL, icmpv4 FAIL
192.168.1.228: arp FAIL, icmpv4 FAIL
192.168.1.229: arp FAIL, icmpv4 FAIL
192.168.1.230: arp OK (28C63FB1B2C3), icmpv4 OK
192.168.1.231: arp FAIL, icmpv4 FAIL
192.168.1.232: arp FAIL, icmpv4 FAIL
192.168.1.233: arp FAIL, icmpv4 FAIL
192.168.1.234: arp FAIL, icmpv4 FAIL
192.168.1.235: arp FAIL, icmpv4 FAIL
192.168.1.236: arp FAIL, icmpv4 FAIL

Příklad s IPv6







Seznam literatury

[1] Parse the Command Line with System.CommandLine [online]. [cit. 2021-4-25]. Dostupné z: https://docs.microsoft.com/en-us/archive/msdn-magazine/2019/march/net-parse-the-command-line-with-system-commandline

[2] SharpPcap.ARP [online]. [cit. 2021-4-25]. Dostupné z: http://sharppcap.sourceforge.net/htmldocs/SharpPcap/ARP.html#M:SharpPcap.ARP.Resolve(System.Net.IPAddress)

[3] Raw Sockets and ICMP [online]. [cit. 2021-4-25]. Dostupné z: https://courses.cs.vt.edu/cs4254/fall04/slides/raw_6.pdf

[4] Internet Control Message Protocol (ICMP) Parameters [online]. [cit. 2021-4-25]. Dostupné z: https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml#icmp-parameters-codes-8

[5] Internet Control Message Protocol version 6 (ICMPv6) Parameters [online]. [cit. 2021-4-25]. Dostupné z: https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml#icmpv6-parameters-codes-6

[6] ICMP echo checksum [online]. [cit. 2021-4-25]. Dostupné z: https://stackoverflow.com/a/20247802