Šablona pro odevzdávání výstupů z distančních cvičení předmětu MPC-PKT určená k editaci a odevzdání po vytvoření PDF verze

Vaše jméno	Alex Sporni
VUT ID	204633
Vypracovaný lab (označení)	**Lab2 (Srovnání vlastností IPv4 a IPv6 protokolů v NS3)**

---

1. číslovaný úkol z návodu

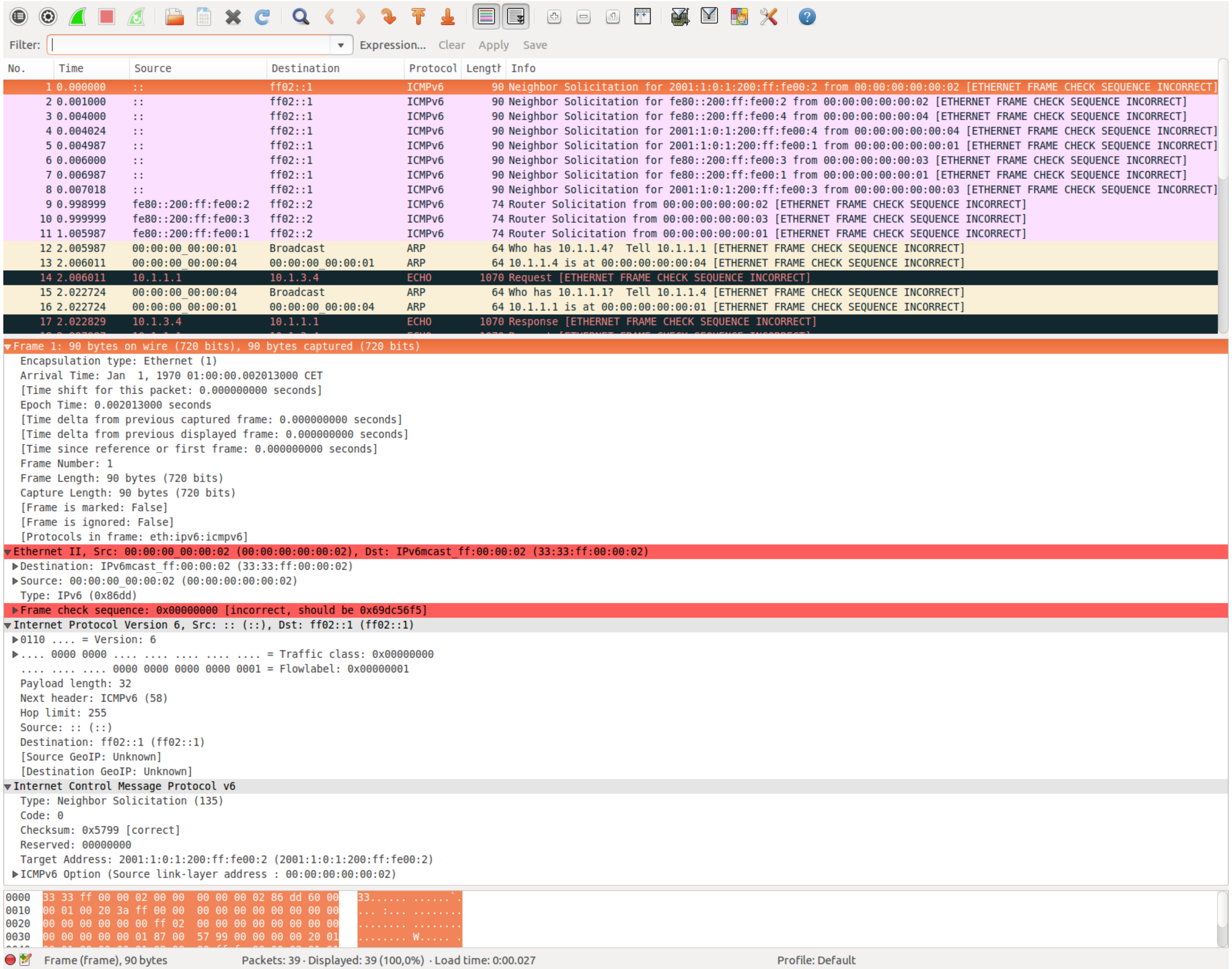
Zadání úkolu: **Otevřete si vytvořený soubor zachyceného provozu pro uzel n0 ve Wiresharku, obrázek vložte do protokolu a krátce popište, co se v simulaci děje. Popište stručně část s IPv4 i IPv6 provozem. Všimněte si i celkové doby odezvy u jednotlivých echo paketů.**

IPv6

- V IPv4 využívame ARP a Broadcast aby sme zistili, ktoré zariadenia sú pripojené do siete v **IPv6** tieto možnosti nemáme.
- V IPv6 boli tieto možnosti nahradené NDP (Neighbor Discovery protocol), ICMPv6 a Multicastom.
- V prvom rade sú od uzlov sieťou rozosielané pakety typu **Neighbor Solicitation (135)** (náhrada ARP), pomocou ktorých zariadenia zisťujú adresy susedov na linkovej vrstve.
- V ďalšom kroku sa uzli v sieti pomocou **ICMPv6 Router Solication (133)** snažia nájsť IPv6 routre a pošlú správu na **all routers Multicast adress ff02::2** adresu. Všetky routre obdržia správu, ktorá obsahuje hlavičku, v ktorej sa nachádza zdrojová IPv6 adresa zariadenia, ktoré sa na dané routre dotazovalo (napr. FE80::200:ff:FE00:1) a destinačná adresa **FF02::2**.
- Následne pakety typu **Neighbor Advertisment 136** sú odoslané uzlami ako reakcia na správu **Neighbor solicitation 135**

IPv4

- V sieti je nasadený aj ARP protokol. Zariadenia 10.1.1.1 rozošle po sieti správu typu Broadcast, v ktorej sa pýta, „ktoré zariadenie má IPv4 adresu 10.1.1.4 ? Po následnom zistení všetkých IP adries smerovačmi, sa zaháji komunikácia pomocou ECHO medzi zariadeniami **10.1.1.1** a **10.1.3.4**. Zariadenie 10.1.1.1 vyšle ECHO Request v čase t = 2,008023 s, následná odpoveď od serveru prichádza v čase t = 2,024 s. Následne prebehne ešte niekoľko requestov a responsov so sekundovými intervalmi.


Filter:
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
1 0.000000 :: ff02::1 ICMPv6 90 Neighbor Solicitation for 2001:1:0:1:200:ff:fe00:2 from 00:00:00:00:00:02 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
2 0.001000 :: ff02::1 ICMPv6 90 Neighbor Solicitation for fe80::200:ff:fe00:2 from 00:00:00:00:00:02 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
3 0.004000 :: ff02::1 ICMPv6 90 Neighbor Solicitation for fe80::200:ff:fe00:4 from 00:00:00:00:00:04 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
4 0.004024 :: ff02::1 ICMPv6 90 Neighbor Solicitation for 2001:1:0:1:200:ff:fe00:4 from 00:00:00:00:00:04 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5 0.004987 :: ff02::1 ICMPv6 90 Neighbor Solicitation for 2001:1:0:1:200:ff:fe00:1 from 00:00:00:00:00:01 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
6 0.006000 :: ff02::1 ICMPv6 90 Neighbor Solicitation for fe80::200:ff:fe00:3 from 00:00:00:00:00:03 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
7 0.006987 :: ff02::1 ICMPv6 90 Neighbor Solicitation for fe80::200:ff:fe00:1 from 00:00:00:00:00:01 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
8 0.007018 :: ff02::1 ICMPv6 90 Neighbor Solicitation for 2001:1:0:1:200:ff:fe00:3 from 00:00:00:00:00:03 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
9 0.998999 fe80::200:ff:fe00:2 ff02::2 ICMPv6 74 Router Solicitation from 00:00:00:00:00:02 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
10 0.999999 fe80::200:ff:fe00:3 ff02::2 ICMPv6 74 Router Solicitation from 00:00:00:00:00:03 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
11 1.005987 fe80::200:ff:fe00:1 ff02::2 ICMPv6 74 Router Solicitation from 00:00:00:00:00:01 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
12 2.005987 00:00:00_00:00:01 Broadcast ARP 64 Who has 10.1.1.4? Tell 10.1.1.1 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
13 2.006011 00:00:00_00:00:04 00:00:00_00:00:01 ARP 64 10.1.1.4 is at 00:00:00:00:00:04 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
14 2.006011 10.1.1.1 10.1.3.4 ECHO 1070 Request [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
15 2.022724 00:00:00_00:00:04 Broadcast ARP 64 Who has 10.1.1.1? Tell 10.1.1.4 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
16 2.022724 00:00:00_00:00:01 00:00:00_00:00:04 ARP 64 10.1.1.1 is at 00:00:00:00:00:01 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
17 2.022829 10.1.3.4 10.1.1.1 ECHO 1070 Response [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
Frame 1: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
Encapsulation type: Ethernet (1)
Arrival Time: Jan 1, 1970 01:00:00.002013000 CET
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 0.002013000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 90 bytes (720 bits)
Capture Length: 90 bytes (720 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ipv6:icmpv6]
Ethernet II, Src: 00:00:00_00:00:02 (00:00:00:00:00:02), Dst: IPv6mcast_ff:00:00:02 (33:33:ff:00:00:02)
Destination: IPv6mcast_ff:00:00:02 (33:33:ff:00:00:02)
Source: 00:00:00_00:00:02 (00:00:00:00:00:02)
Type: IPv6 (0x86dd)
Frame check sequence: 0x00000000 [incorrect, should be 0x69dc56f5]
Internet Protocol Version 6, Src: :: (::), Dst: ff02::1 (ff02::1)
0110 .... = Version: 6
.... 0000 0000 .... = Traffic class: 0x00000000
.... 0000 0000 0000 0000 0001 = Flowlabel: 0x00000001
Payload length: 32
Next header: ICMPv6 (58)
Hop limit: 255
Source: :: (::)
Destination: ff02::1 (ff02::1)
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
Internet Control Message Protocol v6
Type: Neighbor Solicitation (135)
Code: 0
Checksum: 0x5799 [correct]
Reserved: 00000000
Target Address: 2001:1:0:1:200:ff:fe00:2 (2001:1:0:1:200:ff:fe00:2)
ICMPv6 Option (Source link-layer address : 00:00:00:00:00:02)
0000 33 33 ff 00 00 02 00 00 00 00 00 02 86 dd 60 00 33...... ......`.
0010 00 01 00 20 3a ff 00 00 00 00 00 00 00 00 00 00 ... :... ....
0020 00 00 00 00 00 00 ff 02 00 00 00 00 00 00 00 00 ....
0030 00 00 00 00 00 01 87 00 57 99 00 00 00 00 20 01 .... W.....
Frame (frame), 90 bytes
Packets: 39 · Displayed: 39 (100,0%) · Load time: 0:00.027
Profile: Default

Řešení:

2. číslovaný úkol z návodu

Zadání úkolu: **Jak by vypadal řádek zdrojového kódu, který by do konzole vypsal směrovací tabulku na uzlu n0 taktéž v čase na konci simulace? Doložte i výpisem vytvořené směrovací tabulky z konzole po přidání řádku do vašeho projektu.**

Řešení: Za predpokladu že simulácia končí v čase t = 16 (čas ukončenia IPv6 klienta a serveru) tak príkaz bude vyzerať nasledovne:

```
routingHelper.PrintRoutingTableAt(Seconds(16.0), csmaNodesLAN1.Get(0), routingStream);
```

```
root@fekt: /home/student/ns-allinone-3.21/ns-3.21
At time 11.0078s client received 1024 bytes from 2001:1:0:3:200:ff:fe00:a port 7
Node: 3 Time: 16s Ipv6ListRouting table
  Priority: 0 Protocol: ns3::Ipv6StaticRouting
Node: 3 Time: 16s Ipv6StaticRouting table
Destination                     Next Hop                    Flag Met Ref Use If
::1/128                         ::                          UH   0    -   -   0
fe80::/64                       ::                          U    0    -   -   1
2001:1:0:1::/64                 ::                          U    0    -   -   1
fe80::/64                       ::                          U    0    -   -   2
2001:1:0:2::/64                 ::                          U    0    -   -   2
2001:1:0:3::/64                 ::                          U    0    -   -   2

Node: 4 Time: 16s Ipv6ListRouting table
  Priority: 0 Protocol: ns3::Ipv6StaticRouting
Node: 4 Time: 16s Ipv6StaticRouting table
Destination                     Next Hop                    Flag Met Ref Use If
::1/128                         ::                          UH   0    -   -   0
fe80::/64                       ::                          U    0    -   -   1
2001:1:0:2::/64                 ::                          U    0    -   -   1
fe80::/64                       ::                          U    0    -   -   2
2001:1:0:3::/64                 ::                          U    0    -   -   2
2001:1:0:1::/64                 ::                          U    0    -   -   1

Node: 0 Time: 16s Ipv6ListRouting table
  Priority: 0 Protocol: ns3::Ipv6StaticRouting
Node: 0 Time: 16s Ipv6StaticRouting table
Destination                     Next Hop                    Flag Met Ref Use If
::1/128                         ::                          UH   0    -   -   0
fe80::/64                       ::                          U    0    -   -   1
2001:1:0:1::/64                 ::                          U    0    -   -   1
::/0                            fe80::200:ff:fe00:4         UG   0    -   -   1

root@fekt:/home/student/ns-allinone-3.21/ns-3.21#
```

3. číslovaný úkol z návodu

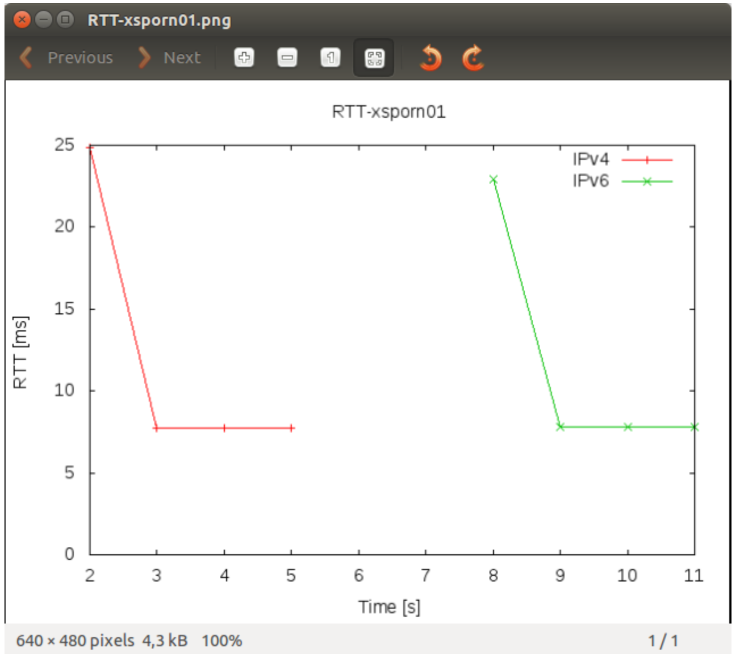
Zadání úkolu: **Zjistěte a popište do protokolu, co znamená směrovací cesta do IPv6 sítě ::/0 ve směrovací tabulce uzlu n0.**

Řešení: **::/0** predstavuje defaultnú IPv6 static route, používa sa na routovanie (smerovanie) do všetkých IPv6 sietí zo zariadenia **n0** (viac info tu: https://bit.ly/3k02ltC)

---

4. číslovaný úkol z návodu

Zadání úkolu: **Upravte kód, tak aby v konečném důsledku v Obr. 8 byla zobrazena doba odezvy i u prvního odesílaného paketu UDP echo aplikace přes IPv4 i IPv6 (v čase 2 s a 8 s). Bude nutné upravit i rozsah y osy grafu. Projekt znovu spusťte a následně pomocí příkazu gnuplot RTT-login.plt vytvořte nový obrázek RTT-login.png. Obrázek vložte do protokolu.**

Řešení: 

---

5. číslovaný úkol z návodu

Zadání úkolu: **Do protokolu uveďte také, proč se doba RTT u prvního paketu tak liší od ostatních paketů, co udává hodnota RTT a proč se tato hodnota liší u protokolů IPv4 a IPv6 (podívejte se na délku jednotlivých paketů).**

Řešení: Hodnota RTT prvého packetu je ovplyvnená fungovaním protokolu ARP (Address Resolution Protocol), resp. ICMPv6 protokolu v prípade IPv6 komunikácie. Hodnota RTT (round-trip time) predstavuje dobu, ktorá uplynie od vyslania signálu z jednej komunikujúcej stanice na druhú plus čas, ktorý je potrebný na prijatie tohto signálu na prvej stanici. Rozdielne hodnoty RTT pri IPv4 a IPv6 sú spôsobené rôznou dĺžkou jednotlivých packetov. V prípade IPv4 UDP echo request packetu (packet 22) 1052 bajtov a v prípade IPv6 UDP echo request packetu je to 1032 bajtov. IPv4 má dynamickú veľkosť hlavičky, môže sa meniť na rozdiel od IPv6 kde je staticky daná. Plus to môže byť ovplyvnené ARP Broadcastom pri IPv4 (náročnejšie na réžiu)

---

6. číslovaný úkol z návodu

Zadání úkolu: **V simulaci byla nastavena velikost dat zabalených do UDP u IPv4 i IPv6 přenosu na 1024 B. Program Wireshark vypisuje, že celková velikost poslaných dat s protokolem IPv4 byla 1070 B a s protokolem IPv6 1090 B. Vysvětlete rozdílné hodnoty a jak je možné se od hodnoty 1024 B k těmto hodnotám dospět.**

Řešení: Zadaná veľkosť dát je 1024 B. Zbytok tvoria záhlavia. V prípade IPv4 je to UDP 8 B, IPv4 je 20 B, Ethernet 14 + 4 B → 1024 + 8 + 20 + 14 + 4 = **1070 B**. V prípade IPv6 to
je 1024 + 8 + 40 + 14 + 4 = **1090 B**

---

7. číslovaný úkol z návodu

Zadání úkolu: **Z obrázků se záhlavími IPv4 a IPv6 protokolů (Obr. 5, Obr. 6) je vidět, že záhlaví IPv4 obsahuje položku Header length (velikost záhlaví), záhlaví IPv6 však žádnou podobnou položku neobsahuje. Proč v případě protokolu IPv6 nemusí tato informace být součástí záhlaví? Vysvětlete základní princip rozšíření záhlaví v případě protokolu IPv4 a IPv6.**

Řešení: V prípade IPv6 dochádza k zjednodušeniu hlavičky oproti IPv4 hlavičke. Veľkosť hlavičky pri IPv6 absentuje z dôvodu, že jej veľkosť je v prípade IPv6 **konštantná** podľa zdroja: https://ibm.co/3k2PAhz. V prípade IPv4 hlavičky môže dochádzať k zmene veľkosti. V prípade IPv6 sú dodatočné informácie posielané pridané ako extended header a nie sú zahrnuté do hlavičky.

---

8. číslovaný úkol z návodu

Zadání úkolu: **Na co slouží položky záhlaví Protocol (v případě IPv4) a Next header (v případě IPv6)? Popište opět v souvislosti s obsahem přenášených paketů v simulaci.**

Řešení:

- Protocol = Určuje, ktorému protokolu vyššej vrstvy sa majú dáta predať pri doručení viac informácii o číslach protokolov sú na zdroji.
- Next header = Špecifikuje typ ďalšej hlavičky. Toto políčko zvyčajne určuje protokol transportnej vrstvy, ktorý je používaný obsahom packetu.

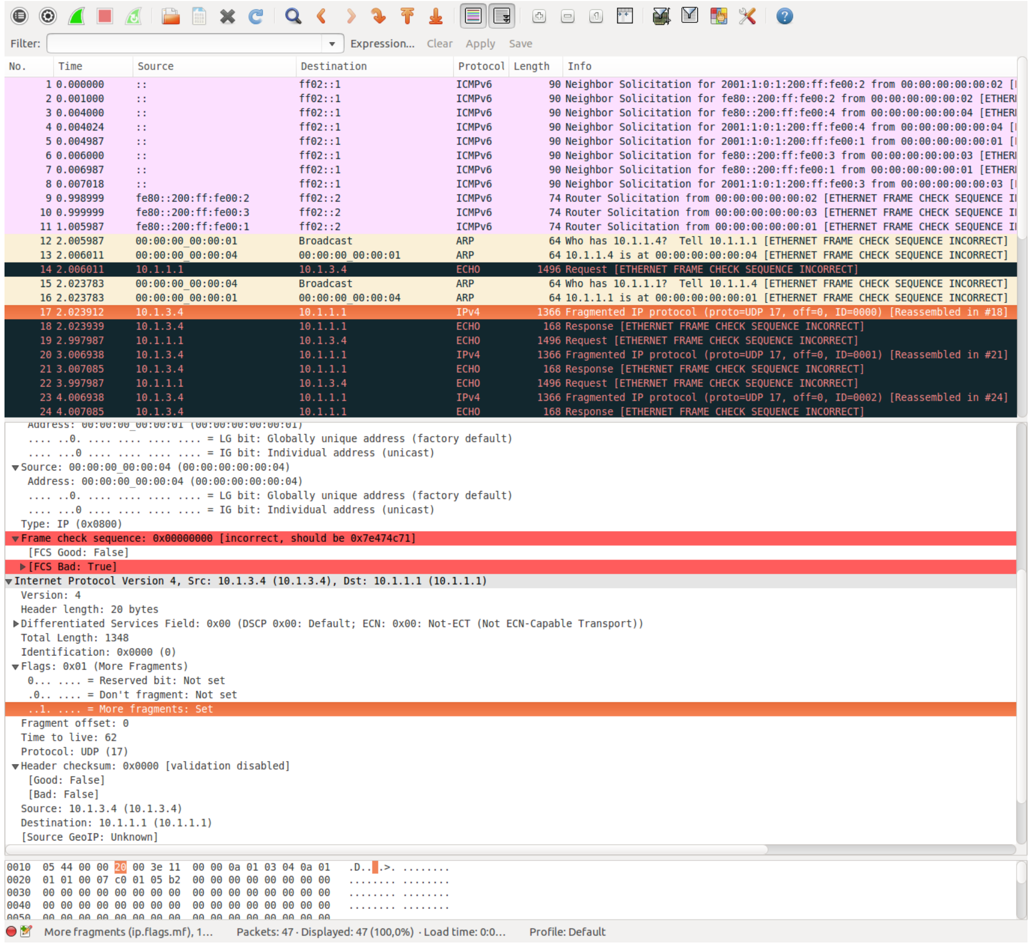
9. číslovaný úkol z návodu

Zadání úkolu: **Vytvořenou simulaci modifikujte tak, aby došlo k fragmentaci paketů na lince bod-bod – velikost datových částí paketů aplikací UDP echo zvyšte na 1450 B a velikost MTU linky point-to-point (bod-bod) snižte na 1350 B. Analyzujte provoz zachycený na uzlu n0 pomocí Wiresharku, dokumentujte pomocí printscreenů.**

Řešení: Bolo potrebné modifikovať nasledujúce riadky kódu:

```
pointToPoint.SetDeviceAttribute("Mtu", UintegerValue(1350));
echoClientIPv4.SetAttribute("PacketSize", UintegerValue(1450));
echoClientIPv6.SetAttribute("PacketSize", UintegerValue(1450));
```

Výsledkom sú fragmentované pakety, ktoré je možné vidieť na obrázku nižšie, fragmentáciu indikuje aj Flag **0x01** v IPv4 záhlaví. Rovnako fragmentácia nastáva aj v prípade IPv6



---

10. číslovaný úkol z návodu

Zadání úkolu: **Porovnejte v nastavení dle úkolu č. 9 chování IPv4 a IPv6 u prvního přenášeného paketu (v čase 2 s a 8 s). Popište zejména kde přesně v síti a proč dochází k (ne)fragmentaci v případě IPv4 a IPv6 UDP echo přenosu těchto paketů a odpovědí na ně ze strany serveru.**

Řešení: Fragmentácia packetu pri IPv4 môže nastať prakticky na ktoromkoľvek zariadení po ceste. Pri IPv6 musí fragmentácia nastať iba u odosielateľa. Pri frame 33 môžeme vidieť ICMPv6 správu (Frame Too Big), ktorá oznámi odosielateľovi, že router musel daný packet zahodiť, pretože neprešiel cez MTU. Nasledujúce packety v poradí by už mal byť automaticky fragmentované. K fragmentácii dochádza z dôvodu veľkosti dát. Veľkosť dát je väčšia ako maximálna MTU na P2P linke medzi R1 a R2.

---

11. číslovaný úkol z návodu

Zadání úkolu: **Porovnejte v nastavení dle úkolu č. 9 chování IPv4 a IPv6 u druhého přenášeného paketu (v čase 3 s a 9 s). Popište zejména kde přesně v síti a proč dochází k (ne)fragmentaci v případě IPv4 a IPv6. Jak se tato situace liší od prvních paketů daných aplikací (Úkol č. 10)?**

Řešení: V prípade IPv4 je chovanie úplne rovnaké ako v prípade packetu prenášaného v čase t = 2 s. V prípade IPv6 platí, že fragmentácia musí prevádzať iba odosielateľ. Fragmentácia následne prebehne pri requeste aj pri response, znovu je zahodený prvý packet a odosielateľ je oznámený, že musí fragmentovať nasledujúce packety.

---

12. číslovaný úkol z návodu

Zadání úkolu: **Vysvětlete stručně princip fragmentace v případě obou verzí protokolů. K čemu slouží příznakové bity (Flags) v záhlaví IPv4 a proč podobné pole neexistuje v záhlaví IPv6?**

Řešení: Pomocou fragmentácie je možné poslať pakety, ktoré majú väčšiu veľkosť ako je kapacita linky (MTU). Ak sa v sieti nachádza viacero routrov, medzi ktorými sú linku s rôznou hodnotou MTU napr. medzi R1 a R2 by bolo MTU 1300, medzi R2 a R3 by MTU bolo 1250 a medzi linkou R1 a R3 by bolo 1400, tak sa maximálna kapacita linky berie ako najnižšia v danej sieti, v tomto prípade by to bolo 1250. V prípade IPv4 má na starosti fragmentáciu Router za ktorým sa nachádza linka s menšou kapacitou ako je nastavená veľkosť paketu. Flagy sa obsahujú informácie za pomocou ktorých je možné daný fragmentovaný packet na druhej strane linky u užívateľa poskladať. V prípade IPv6 sa nepoužívajú príznakové bity z dôvodu, že fragmentácia nastáva už o odosielateľa. Následne u príjemcu prebehne rekonštrukcia packetu. V prípade, že veľkosť packetu je vyššia ako bod cez ktorý prechádza, tak vráti informáciu, že packet je príliš veľký – Packet Too big, je to možné vidieť na packete 33.

---

13. číslovaný úkol z návodu

Zadání úkolu: **Popište, jak vypadají relevantní položky záhlaví obou IP protokolů (Next header, Offset, More fragments) v souvislosti s fragmentací.**

Řešení: Daný flag nám indikuje či dochádza alebo nedochádza k fragmentácii

V prípade IPv4:

```
▼Internet Protocol Version 4, Src: 10.1.3.4 (10.1.3.4), Dst: 10.1.1.1 (10.1.1.1)
   Version: 4
   Header length: 20 bytes
  ▶Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
   Total Length: 1348
   Identification: 0x0000 (0)
  ▼Flags: 0x01 (More Fragments)
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
   Fragment offset: 0
   Time to live: 62
   Protocol: UDP (17)
```

IPv6:

```
  Source: 2001:1:0:1:200:ff:fe00:1 (2001:1:0:1:200:ff:fe00:1)
  [Source SA MAC: 00:00:00_00:00:01 (00:00:00:00:00:01)]
  Destination: 2001:1:0:3:200:ff:fe00:a (2001:1:0:3:200:ff:fe00:a)
  [Destination SA MAC: 00:00:00_00:00:0a (00:00:00:00:00:0a)]
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
▼ Fragmentation Header
   Next header: UDP (17)
   Reserved octet: 0x0000
   0000 0000 0000 0... = Offset: 0 (0x0000)
   .... .... .... .00. = Reserved bits: 0 (0x0000)
   .... .... .... ...1 = More Fragment: Yes
   Identification: 0xa68f4557
  Reassembled IPv6 in frame: 35
```

---

14. číslovaný úkol z návodu

Zadání úkolu: **Změňte velikost datové části paketů na 1700 B (tj. více než 1500 B s IP záhlavím, které umí přenést Ethernet) a velikost MTU na lince bod-bod vraťte na 1500 B. Opět analyzujte provoz v simulaci, měli byste dostat odlišné výsledky. V čem se tato situace liší oproti situaci z úkolů č. 9 až 11? Vysvětlete důvod rozdílných výsledků oproti předchozímu případu.**

Řešení: Fragmentácia nastáva hneď u odosielateľa. Technológia ethernet dokáže preniesť maximálne 1500 B avšak veľkosť paketu je nastavená väčšia.

---

15. číslovaný úkol z návodu

Zadání úkolu: **Kde dochází vždy k poskládání fragmentovaného paketu IPv4 či IPv6 a proč?**

Řešení: K poskladaniu fragmentovaného packetu dochádza až u užívateľa, z dôvodu aby sa zaistilo, že všetky packety dosiahnu cieľ.
