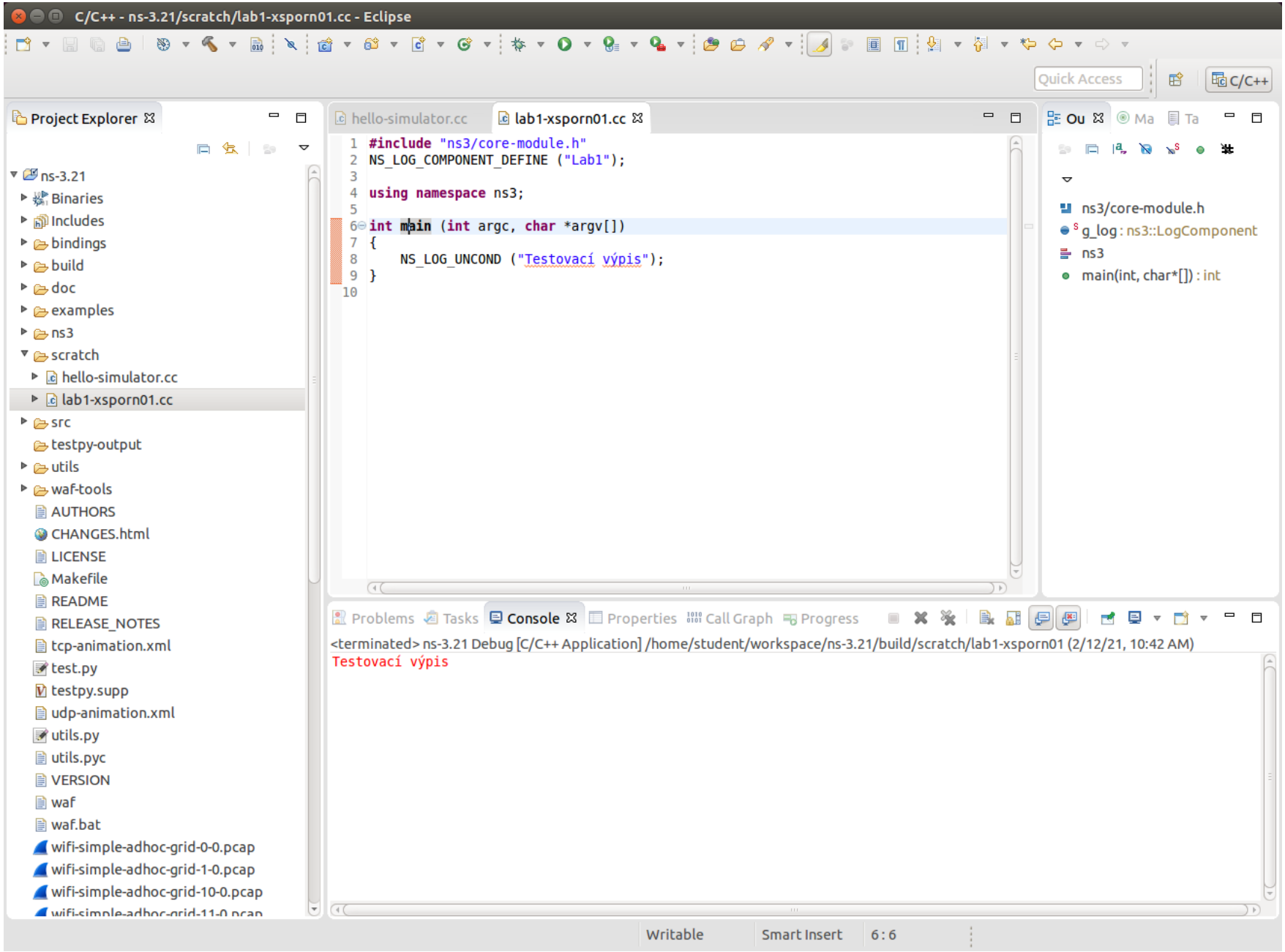
Šablona pro odevzdávání výstupů z distančních cvičení předmětu MPC-PKT určená k editaci a odevzdání po vytvoření PDF verze

Vaše jméno	Alex Sporni
VUT ID	204633
Vypracovaný lab (označení)	**Lab1 (Simulace topologie point-to-point a bus)**

---

1. číslovaný úkol z návodu

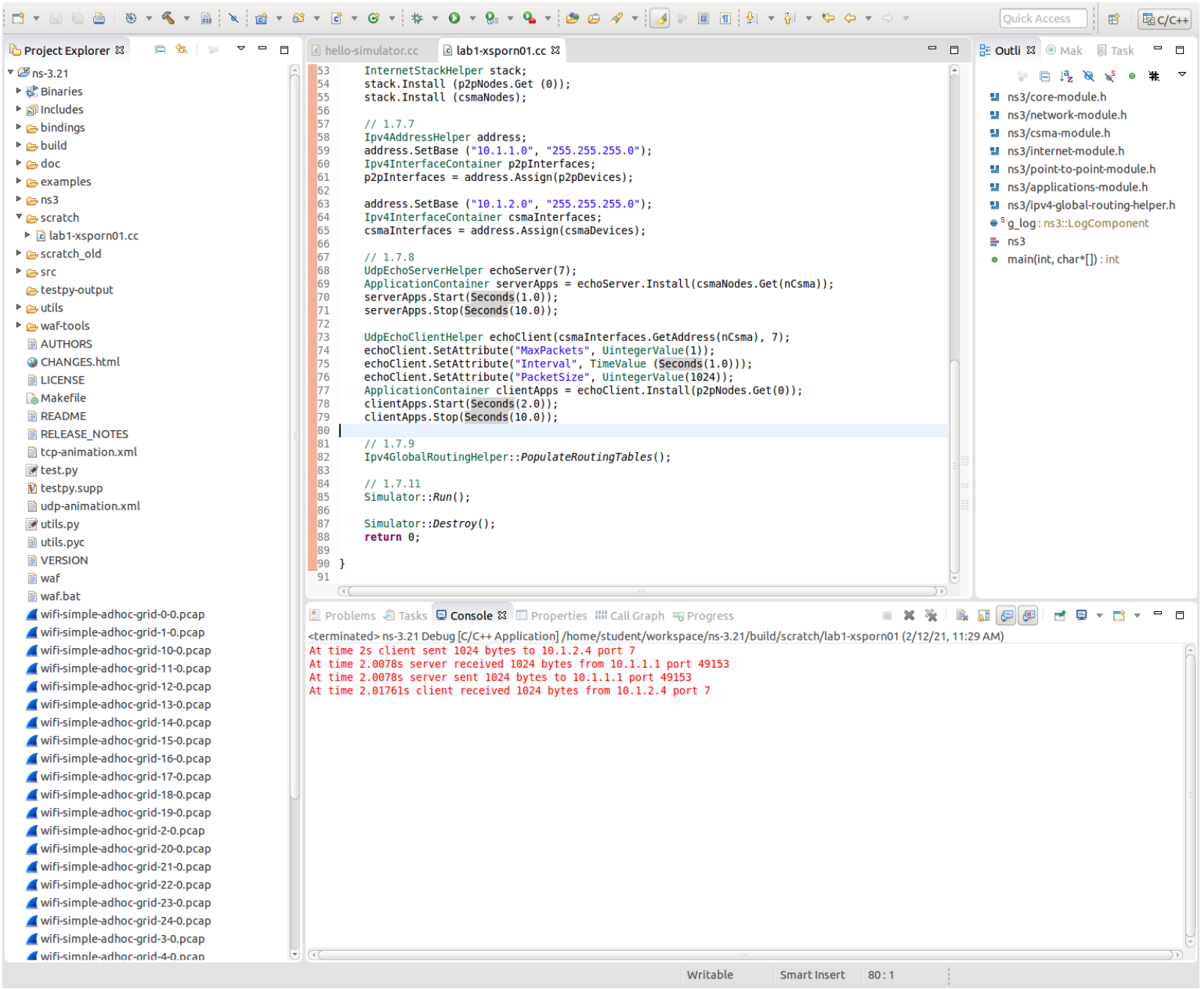
Zadání úkolu: **Zdokumentujte funkčnost překladu úvodního kódu pomoci printscreenu, kde bude zřetelně vidět celé prostředí Eclipse, včetně úspěšného testovacího výpisu.**


C/C++ - ns-3.21/scratch/lab1-xsporn01.cc - Eclipse
Quick Access
C/C++
Project Explorer
ns-3.21
Binaries
Includes
bindings
build
doc
examples
ns3
scratch
hello-simulator.cc
lab1-xsporn01.cc
src
testpy-output
utils
waf-tools
AUTHORS
CHANGES.html
LICENSE
Makefile
README
RELEASE_NOTES
tcp-animation.xml
test.py
testpy.supp
udp-animation.xml
utils.py
utils.pyc
VERSION
waf
waf.bat
wifi-simple-adhoc-grid-0-0.pcap
wifi-simple-adhoc-grid-1-0.pcap
wifi-simple-adhoc-grid-10-0.pcap
hello-simulator.cc
lab1-xsporn01.cc
1 #include "ns3/core-module.h"
2 NS_LOG_COMPONENT_DEFINE ("Lab1");
3
4 using namespace ns3;
5
6 int main (int argc, char *argv[])
7 {
8 NS_LOG_UNCOND ("Testovací výpis");
9 }
10
Ou
Ma
Ta
ns3/core-module.h
g_log : ns3::LogComponent
ns3
main(int, char*[]) : int
Problems
Tasks
Console
Properties
Call Graph
Progress
<terminated> ns-3.21 Debug [C/C++ Application] /home/student/workspace/ns-3.21/build/scratch/lab1-xsporn01 (2/12/21, 10:42 AM)
Testovací výpis
Writable
Smart Insert
6 : 6

Řešení:

2. číslovaný úkol z návodu

Zadání úkolu: **Zdokumentujte funkčnost simulace v této fázi pomocí printscreenu, kde bude zřetelně vidět celé prostředí Eclipse, včetně úspěšného výpisu o průběhu simulace (jako na obr. 1.5 v návodu) a tento printscreen vložte do protokolu.**


Project Explorer
ns-3.21
Binaries
Includes
bindings
build
doc
examples
ns3
scratch
lab1-xsporn01.cc
scratch_old
src
testpy-output
utils
waf-tools
AUTHORS
CHANGES.html
LICENSE
Makefile
README
RELEASE_NOTES
tcp-animation.xml
test.py
testpy.supp
udp-animation.xml
utils.py
utils.pyc
VERSION
waf
waf.bat
wifi-simple-adhoc-grid-0-0.pcap
wifi-simple-adhoc-grid-1-0.pcap
wifi-simple-adhoc-grid-10-0.pcap
wifi-simple-adhoc-grid-11-0.pcap
wifi-simple-adhoc-grid-12-0.pcap
wifi-simple-adhoc-grid-13-0.pcap
wifi-simple-adhoc-grid-14-0.pcap
wifi-simple-adhoc-grid-15-0.pcap
wifi-simple-adhoc-grid-16-0.pcap
wifi-simple-adhoc-grid-17-0.pcap
wifi-simple-adhoc-grid-18-0.pcap
wifi-simple-adhoc-grid-19-0.pcap
wifi-simple-adhoc-grid-2-0.pcap
wifi-simple-adhoc-grid-20-0.pcap
wifi-simple-adhoc-grid-21-0.pcap
wifi-simple-adhoc-grid-22-0.pcap
wifi-simple-adhoc-grid-23-0.pcap
wifi-simple-adhoc-grid-24-0.pcap
wifi-simple-adhoc-grid-3-0.pcap
wifi-simple-adhoc-grid-4-0.pcap
hello-simulator.cc
lab1-xsporn01.cc
53 InternetStackHelper stack;
54 stack.Install (p2pNodes.Get (0));
55 stack.Install (csmaNodes);
56
57 // 1.7.7
58 Ipv4AddressHelper address;
59 address.SetBase ("10.1.1.0", "255.255.255.0");
60 Ipv4InterfaceContainer p2pInterfaces;
61 p2pInterfaces = address.Assign(p2pDevices);
62
63 address.SetBase ("10.1.2.0", "255.255.255.0");
64 Ipv4InterfaceContainer csmaInterfaces;
65 csmaInterfaces = address.Assign(csmaDevices);
66
67 // 1.7.8
68 UdpEchoServerHelper echoServer(7);
69 ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(nCsma));
70 serverApps.Start(Seconds(1.0));
71 serverApps.Stop(Seconds(10.0));
72
73 UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(nCsma), 7);
74 echoClient.SetAttribute("MaxPackets", UintegerValue(1));
75 echoClient.SetAttribute("Interval", TimeValue (Seconds(1.0)));
76 echoClient.SetAttribute("PacketSize", UintegerValue(1024));
77 ApplicationContainer clientApps = echoClient.Install(p2pNodes.Get(0));
78 clientApps.Start(Seconds(2.0));
79 clientApps.Stop(Seconds(10.0));
80
81 // 1.7.9
82 Ipv4GlobalRoutingHelper::PopulateRoutingTables();
83
84 // 1.7.11
85 Simulator::Run();
86
87 Simulator::Destroy();
88 return 0;
89
90 }
91
Outli
Mak
Task
ns3/core-module.h
ns3/network-module.h
ns3/csma-module.h
ns3/internet-module.h
ns3/point-to-point-module.h
ns3/applications-module.h
ns3/ipv4-global-routing-helper.h
g_log : ns3::LogComponent
ns3
main(int, char*[]) : int
Problems
Tasks
Console
Properties
Call Graph
Progress
<terminated> ns-3.21 Debug [C/C++ Application] /home/student/workspace/ns-3.21/build/scratch/lab1-xsporn01 (2/12/21, 11:29 AM)
At time 2s client sent 1024 bytes to 10.1.2.4 port 7
At time 2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.01761s client received 1024 bytes from 10.1.2.4 port 7
Writable
Smart Insert
80 : 1

Řešení:

3. číslovaný úkol z návodu

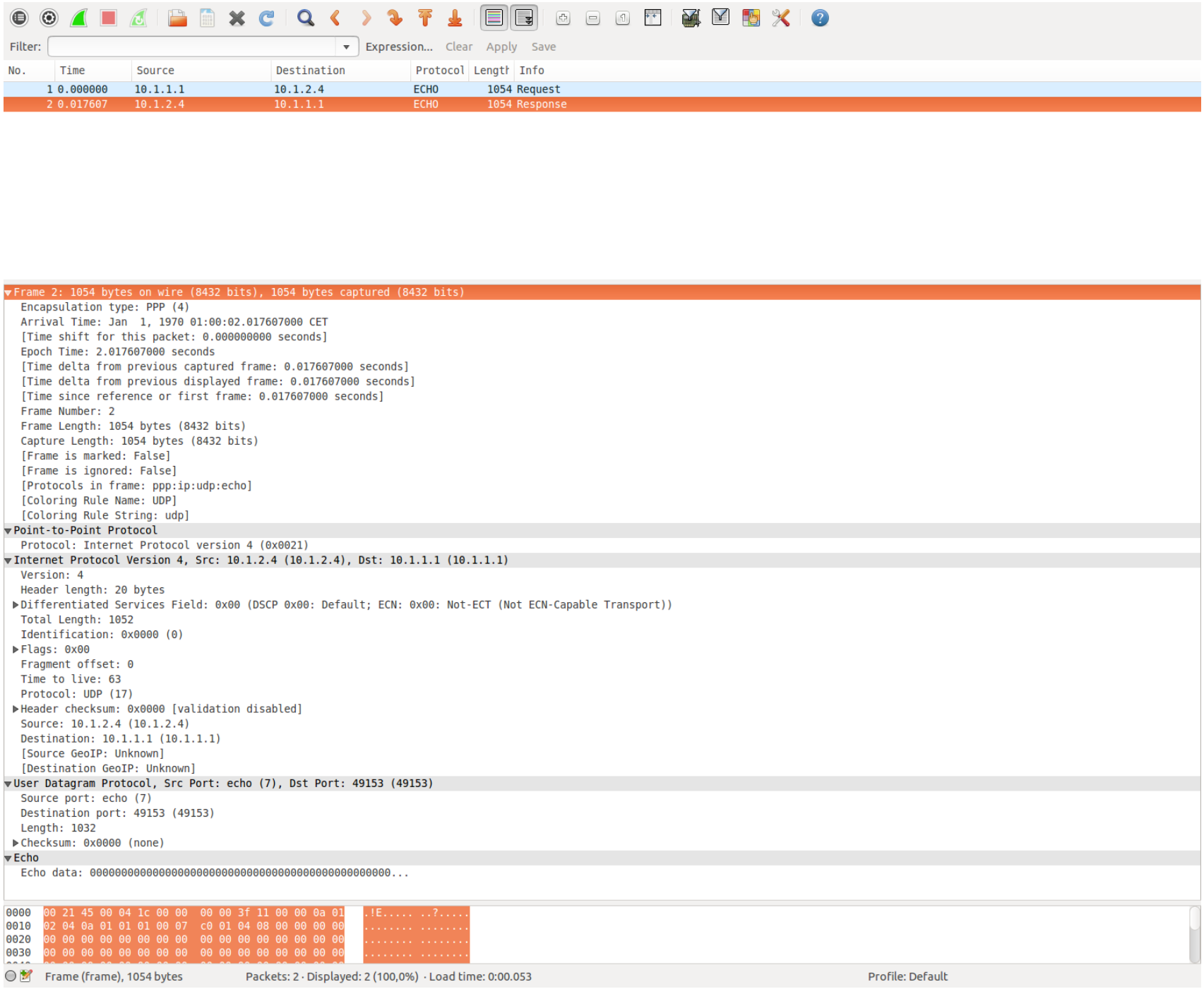
Zadání úkolu: **Všechny vámi vygenerované trasovací soubory si otevřete i v programu Wireshark. Do protokolu uveďte printscreeny u zachyceného provozu ze souborů lab1-0-0.pcap a lab1_prom-2-0.pcap. Zachycené průběhy komunikace vhodně popište. Neopomeňte zejména popis zachycené komunikace, zapouzdření paketů, účelu jednotlivých protokolů, časů, ve kterých byly zachyceny a jejich délky.**

lab1-0-0.pcap

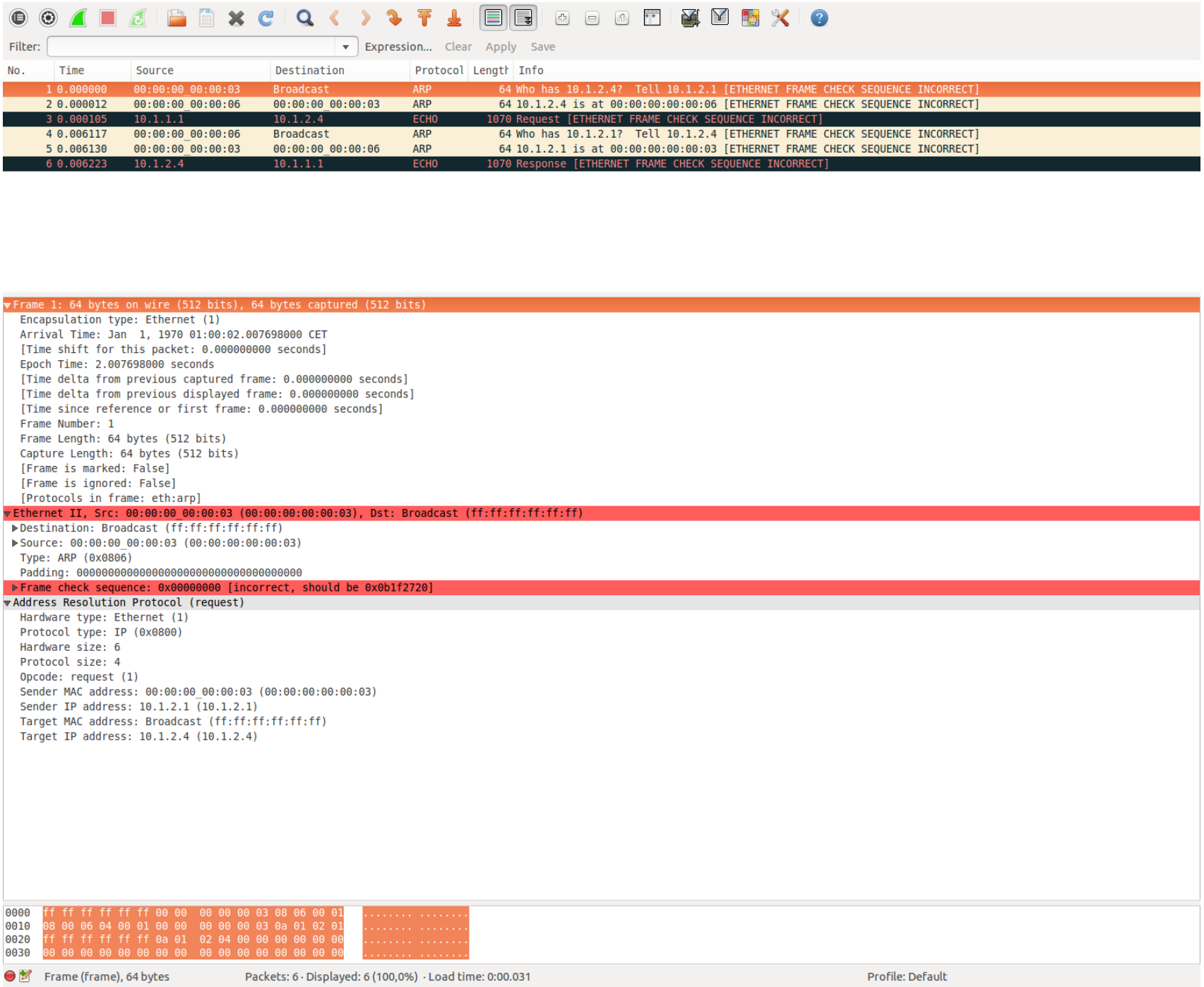
- V súbore lab1-0-0.pcap je zachytená sieťová komunikácia medzi zariadeniami **n0**, ktoré reprezentuje klienta a **n4**, ktoré reprezentuje server. Zariadenia majú pridelené IP adresy **10.1.1.1** a **10.1.2.4**.
- Typ komunikácie je **UDP** prevoz, ktorý zodpovedá funkcii **ICMP ping**, v našom prípade zasielame na server jeden paket. Server sa spustí v čase t = 1 s a klient v čase t = 2 s. Klient následne odošle na server paket o veľkosti **1054 bajtov**. Typ zapuzdrenia je **PPP** (Point-to-Point Protocol). V čase t = 2,0176 s prichádza na port serveru **7.** Následne server odosiela ICMP odpoveď na dynamicky pridelený port **49153** klientovi. Round-trip Time (RTT) dokopy trvalo 0,017607 s = **17,607 ms**.

lab1_prom-2.0.pcap

- V súbore lab1_prom-2.0.pcap je zachytená komunikácia celej siete na uzli **n1 (10.1.2.1)** vďaka **promiskuitnému režimu.**
    - **Frame 1**: komunikácia typu **ARP Broadcast** zdroj: **10.1.2.1** s požiadavkou na zistenie IP adresy **10.1.2.4**, veľkosť frame-u **je 64 bajtov**, typ zapuzdrenia je **Ethernet.**
    - **Frame 2**: zariadenie s MAC adresou **00:00:00:00:00:06** odpovedá zariadeniu **00:00:00:00:00:03** (n1 = 10.1.2.1), že je vlastníkom IP adresy **10.1.2.4**. veľkost frame-u je 64 bajtov a typ zapuzdrenia je Ethernet.
    - **Frame 3**: typ zapuzdrenia je Ethernet, jedná sa o **ICMP ECHO**, veľkost frame-u je 1070 bajtov. Zdroj: **10.1.1.1**, zdrojový port: **49153.** Destinácia: **00:00:00:00:00:06** port destinácie: **7**. **FCS** hlási checksum error. Čas odoslania je 105 mikrosekúnd.
    - **Frame 4**: rovnako ako v prípade frame-u 1 sa jedná o **ARP Broadcast (ff:ff:ff:ff:ff:ff)** na zistenie IP adresy zariadenia **00:00:00:00:00:03**. Zdroj ARP requestu je server **00:00:00:00:00:06**. Zapuzdrenie je vždy Ethernet, veľkosť frame-u je **64 bajtov**.
    - **Frame 5**: jedná sa o **ARP reply**, keď zariadenie **00:00:00:00:00:03** oznamuje zariadeniu **00:00:00:00:00:06**, že vlastní IP adresu **10.1.2.1**. Veľkosť frame-u je **64 bajtov**.
    - **Frame 6**: typ zapuzdrenia je Ethernet, jedná sa o **ICMP ECHO**, veľkost frame-u je 1070 bajtov. Zdroj: **10.1.1.4**, zdrojový port: **7** Cieľ je zariadenie 10.1.1.1, port destinácie je 49153. **FCS** hlási checksum error. Čas odoslania je 6,223 ms.


Filter:
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.1.1 10.1.2.4 ECHO 1054 Request
2 0.017607 10.1.2.4 10.1.1.1 ECHO 1054 Response
Frame 2: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
Encapsulation type: PPP (4)
Arrival Time: Jan 1, 1970 01:00:02.017607000 CET
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 2.017607000 seconds
[Time delta from previous captured frame: 0.017607000 seconds]
[Time delta from previous displayed frame: 0.017607000 seconds]
[Time since reference or first frame: 0.017607000 seconds]
Frame Number: 2
Frame Length: 1054 bytes (8432 bits)
Capture Length: 1054 bytes (8432 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: ppp:ip:udp:echo]
[Coloring Rule Name: UDP]
[Coloring Rule String: udp]
Point-to-Point Protocol
Protocol: Internet Protocol version 4 (0x0021)
Internet Protocol Version 4, Src: 10.1.2.4 (10.1.2.4), 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: 1052
Identification: 0x0000 (0)
Flags: 0x00
Fragment offset: 0
Time to live: 63
Protocol: UDP (17)
Header checksum: 0x0000 [validation disabled]
Source: 10.1.2.4 (10.1.2.4)
Destination: 10.1.1.1 (10.1.1.1)
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
User Datagram Protocol, Src Port: echo (7), Dst Port: 49153 (49153)
Source port: echo (7)
Destination port: 49153 (49153)
Length: 1032
Checksum: 0x0000 (none)
Echo
Echo data: 0000000000000000000000000000000000000000000000000000...
0000 00 21 45 00 04 1c 00 00 00 00 3f 11 00 00 0a 01 .!E..... ..?.....
0010 02 04 0a 01 01 01 00 07 c0 01 04 08 00 00 00 00 ....
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Frame (frame), 1054 bytes
Packets: 2 · Displayed: 2 (100,0%) · Load time: 0:00.053
Profile: Default

Řešení:


Filter:
Expression...
Clear
Apply
Save
No. Time Source Destination Protocol Length Info
1 0.000000 00:00:00_00:00:03 Broadcast ARP 64 Who has 10.1.2.4? Tell 10.1.2.1 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
2 0.000012 00:00:00_00:00:06 00:00:00_00:00:03 ARP 64 10.1.2.4 is at 00:00:00:00:00:06 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
3 0.000105 10.1.1.1 10.1.2.4 ECHO 1070 Request [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
4 0.006117 00:00:00_00:00:06 Broadcast ARP 64 Who has 10.1.2.1? Tell 10.1.2.4 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5 0.006130 00:00:00_00:00:03 00:00:00_00:00:06 ARP 64 10.1.2.1 is at 00:00:00:00:00:03 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
6 0.006223 10.1.2.4 10.1.1.1 ECHO 1070 Response [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
Encapsulation type: Ethernet (1)
Arrival Time: Jan 1, 1970 01:00:02.007698000 CET
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 2.007698000 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: 64 bytes (512 bits)
Capture Length: 64 bytes (512 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:arp]
Ethernet II, Src: 00:00:00 00:00:03 (00:00:00:00:00:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: 00:00:00_00:00:03 (00:00:00:00:00:03)
Type: ARP (0x0806)
Padding: 00000000000000000000000000000000000000
Frame check sequence: 0x00000000 [incorrect, should be 0x0b1f2720]
Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: 00:00:00_00:00:03 (00:00:00:00:00:03)
Sender IP address: 10.1.2.1 (10.1.2.1)
Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
Target IP address: 10.1.2.4 (10.1.2.4)
0000 ff ff ff ff ff ff 00 00 00 00 00 03 08 06 00 01 ....
0010 08 00 06 04 00 01 00 00 00 00 00 03 0a 01 02 01 ....
0020 ff ff ff ff ff ff 0a 01 02 04 00 00 00 00 00 00 ....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Frame (frame), 64 bytes
Packets: 6 · Displayed: 6 (100,0%) · Load time: 0:00.031
Profile: Default

---

4. číslovaný úkol z návodu

Zadání úkolu: **Vysvětlete princip metody přístupu ke sdílenému médiu (CSMA) a jeho variant.**

Řešení: Siete Ethernet používajú protokol s názvom **CSMA/CD** (Carrier Sense Multiple Access with Collision Detection), ktorý zariadeniam pomáha rovnomerne zdieľať šírku pásma a pritom zabraňuje tomu, aby dve zariadenia na rovnakom sieťovom médiu vysielali súčasne. Protokol CSMA/CD vznikol kvôli riešeniu problému kolízií, ku ktorým dochádza, keď rôzne uzly (spravidla koncové zariadenia) vysielajú svoje pakety v rovnakom čase. Keď hostitel potrebuje vysielať v sieti, najskôr sa pokúsi na linke zachytiť digitálny signál. Pokiaľ je linka voľná a žiadny iný hostitel nevysiela, hostitel potom pokračuje vo vlastnom prenose. Vysielajúci hostitel neustále monitoruje linku, aby si overil, či nezačali vysielať žiadni iní hostitelia. Ak hostitel na linke detekuje iný signál, odošle rozšírený rušiaci signál, ktorý spôsobí, že všetky ostatné uzly prestanú odosielať dáta (môžeme to prirovnať k obsadzovaciemu tónu). Uzly na tento rušiaci signál reagujú tak, že nejakú dobu počkajú a potom sa pokúsia vysielať znovu. Čas, po ktorom môžu kolidujúce stanice opakovať svoje vysielanie, je určený algoritmami spätného časovača. Pokiaľ sa kolízie stále opakujú i po 15 pokusoch, príslušný uzol vo vysielaní už nepokračuje. [zdroj: Todd Lammle – CCNA výukový průvodce – computer press – ISBN 978-80-251-4602-6]

Existujú 4 rôzne varianty CSMA

1. CSMA – jedná sa o tzv. čisté CSMA, na zabránenie kolízii sa používa len počúvanie na nosnej. Vysielací uzol nezisťuje kolízie a prijímače nemôžu rozlíšiť medzi kolíziami a inými zdrojmi chýb rámcov.
2. CSMA/CA – využíva sa predovšetkým v bezdrôtových sieťach (Wi-Fi). Každý uzol musí informovať ostatné uzly o úmysle vysielať. Účastníci bezdrôtového prenosu nie sú schopný zároveň vysielať a prijímať.
3. CSMA/CD – využíva sa najmä v ethernetových half-duplex sieťach. Vysielacie uzly sú schopné detekovať výskyt kolízii, zastaviť vysielanie okamžite a počkať náhodnú dobu pred ďalším pokusom odoslanie.
4. CSMA/BD – je to CSMA s bitovou arbitrážou. Všetkým uzlom na prepojovacom vedení priradené identifikačné číslo či kód priority. Pri výskyte kolízie jeden z uzlov, ktorý sa pokúša vysielať súčasne dostane prioritu vysielať podľa identifikačného čísla či kódu priority. Používa sa v CAN komunikáciách. [zdroj: https://cs.wikipedia.org/wiki/CSMA]

---

5. číslovaný úkol z návodu

Zadání úkolu: **Vysvětlete pojem Promiskuitní režim (mód).**

Řešení: Je to označenie pre špeciálny režim sieťovej karty NIC alebo bezdrôtovej sieťovej karty WNIC, ktorý v počítačových sieťach umožňuje zachytávať i sieťovú komunikáciu, ktorá nie je priamo určená pre dané zariadenie alebo počítač. Využíva sa pri sieťovom útoku packet sniffing. Tento režim využívajú analyzátory paketov. [zdroj: https://bit.ly/3ai1voz]

---

6. číslovaný úkol z návodu

Zadání úkolu: **Vysvětlete, jak je možné, že při otevření trasovacích souborů ve Wiresharku pozná Wireshark, že se jedná o UDP Echo protokol.**

Řešení: Wireshark to rozozná podľa protokolového čísla **17**, ktoré je pridelené práve pre UDP. [zdroj: https://wiki.wireshark.org/User_Datagram_Protocol]

---

7. číslovaný úkol z návodu

Zadání úkolu: **Co vyjadřuje číselné označení 49153 na posledním řádku obrázku 1.9?**

Řešení: Číselné označenie **49153** označuje prvý voľný port z dynamického rozsahu UDP.

Dynamic, private or ephemeral ports [hide]

Port	TCP	UDP	Description
49152–65535	Yes	No	Certificate Management over CMS[373]

[zdroj: https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers]

---

8. číslovaný úkol z návodu

Zadání úkolu: **Zvyšte hodnotu „Delay“ point-to-point linky mezi n0 a n1 na dvojnásobnou hodnotu. Jaká je nyní celková doba odezvy přenosu tam a zpět (RTT) u UDP echo aplikace?**

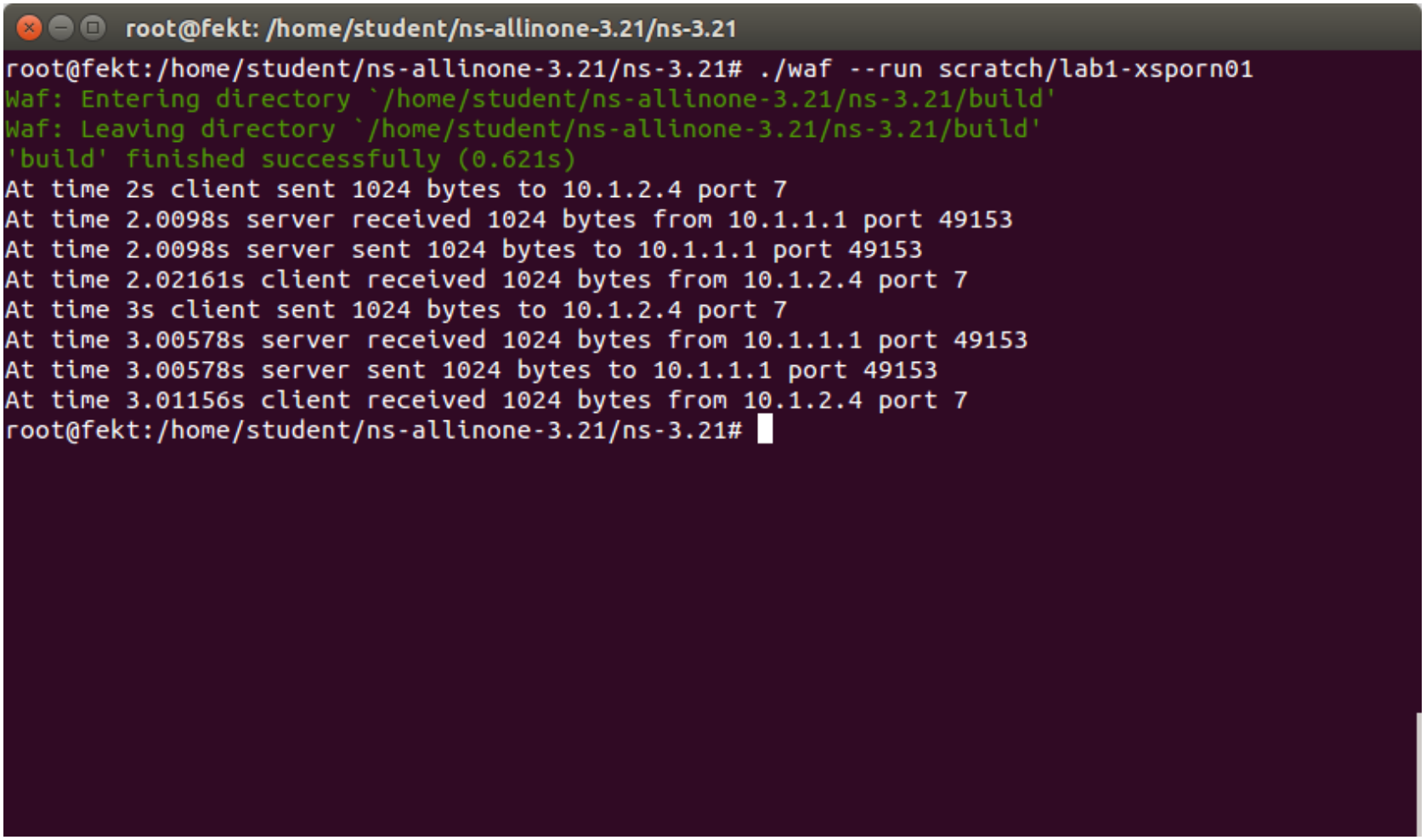
Řešení: Pôvodný RTT **0,01761 s** sa navýšil na **0.02161 s**

---

9. číslovaný úkol z návodu

Zadání úkolu: **Změňte parametr MaxPackets na 2, čímž dojde ke zvýšení počtu odeslaných echo paketů a i odpovědí na 2. První paket se odešle jako dříve, druhý o sekundu později. Jaká je celková doba odezvy (RTT) u druhého paketu (UDP echo pingu) ve srovnání s prvním UDP pingem v této situaci? Proč tomu tak je? Odpověď zjistíte ze souborů pcap (zejména lab1_prom-2-0.pcap).**

Řešení: Z konzolového výpisu je možné vidieť, že pri druhom pakete je RTT nižšie. Dôvod je ten, že neprebieha komunikácia cez protokol ARP, čiže už nie je potrebné zisťovať znovu cieľovú IP adresu.



---

10. číslovaný úkol z návodu

Zadání úkolu: **Změňte parametr PacketSize echo zprávy na 18 (bajtů). Jak změní hodnoty RTT prvního a druhého paketu? Jaký je důvod této změny? V programu Wireshark si prohlédněte libovolný rámec ECHO protokolu (použijte soubor lab1_prom-2-0.pcap), jeho zapouzdření a poznačte si celkovou délku tohoto rámce.**

Řešení: Po zmene parametra **PacketSize** na 18 B sa RTT znížilo. Dôvodom je spomínaná absencia ARP requestu, ale aj nižší objem prenášaných dát. Dĺžka rámca predstavuje 64 B, typ zapuzdrenia je Ethernet.

1	0.000000	00:00:00_00:00:03	Broadcast	ARP	64 Who has 10.1.2.4? Tell 10.1.2.1 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
2	0.000013	00:00:00_00:00:06	00:00:00_00:00:03	ARP	64 10.1.2.4 is at 00:00:00:00:00:06 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
3	0.000025	10.1.1.1	10.1.2.4	ECHO	64 Request [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
4	0.006037	00:00:00_00:00:06	Broadcast	ARP	64 Who has 10.1.2.1? Tell 10.1.2.4 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5	0.006050	00:00:00_00:00:03	00:00:00_00:00:06	ARP	64 10.1.2.1 is at 00:00:00:00:00:03 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
6	0.006062	10.1.2.4	10.1.1.1	ECHO	64 Response [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
7	0.996000	10.1.1.1	10.1.2.4	ECHO	64 Request [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
8	0.996012	10.1.2.4	10.1.1.1	ECHO	64 Response [ETHERNET FRAME CHECK SEQUENCE INCORRECT]

```
▼ Frame 4: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Jan  1, 1970 01:00:02.014125000 CET
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 2.014125000 seconds
    [Time delta from previous captured frame: 0.006012000 seconds]
    [Time delta from previous displayed frame: 0.006012000 seconds]
    [Time since reference or first frame: 0.006037000 seconds]
    Frame Number: 4
    Frame Length: 64 bytes (512 bits)
    Capture Length: 64 bytes (512 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:arp]
    [Coloring Rule Name: ARP]
    [Coloring Rule String: arp]
▼ Ethernet II, Src: 00:00:00_00:00:06 (00:00:00:00:00:06), Dst: Broadc
  ▶ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  ▶ Source: 00:00:00_00:00:06 (00:00:00:00:00:06)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000
  ▶ Frame check sequence: 0x00000000 [incorrect, should be 0xe7635718]
▶ Address Resolution Protocol (request)
```

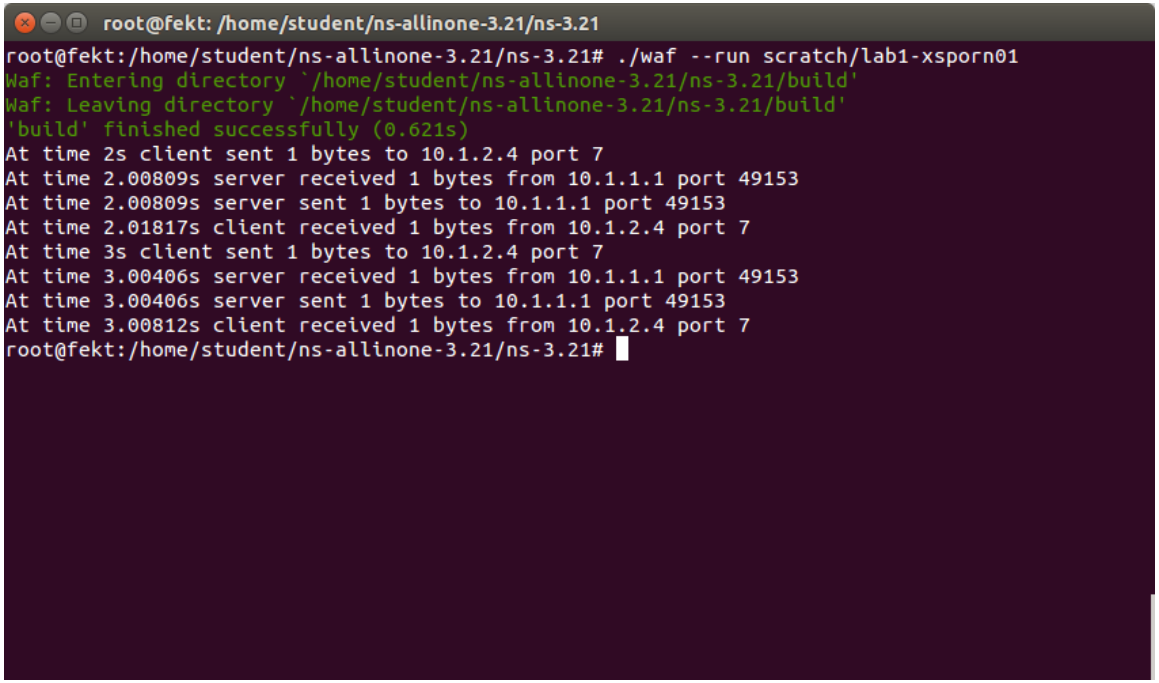
```
root@fekt: /home/student/ns-allinone-3.21/ns-3.21
root@fekt:/home/student/ns-allinone-3.21/ns-3.21# ./waf --run scratch/lab1-xsporn01
Waf: Entering directory `/home/student/ns-allinone-3.21/ns-3.21/build'
Waf: Leaving directory `/home/student/ns-allinone-3.21/ns-3.21/build'
'build' finished successfully (0.621s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 7
At time 2.0098s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.0098s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.02161s client received 1024 bytes from 10.1.2.4 port 7
At time 3s client sent 1024 bytes to 10.1.2.4 port 7
At time 3.00578s server received 1024 bytes from 10.1.1.1 port 49153
At time 3.00578s server sent 1024 bytes to 10.1.1.1 port 49153
At time 3.01156s client received 1024 bytes from 10.1.2.4 port 7
root@fekt:/home/student/ns-allinone-3.21/ns-3.21# ./waf --run scratch/lab1-xsporn01
Waf: Entering directory `/home/student/ns-allinone-3.21/ns-3.21/build'
Waf: Leaving directory `/home/student/ns-allinone-3.21/ns-3.21/build'
'build' finished successfully (0.637s)
At time 2s client sent 18 bytes to 10.1.2.4 port 7
At time 2.00811s server received 18 bytes from 10.1.1.1 port 49153
At time 2.00811s server sent 18 bytes to 10.1.1.1 port 49153
At time 2.01823s client received 18 bytes from 10.1.2.4 port 7
At time 3s client sent 18 bytes to 10.1.2.4 port 7
At time 3.00409s server received 18 bytes from 10.1.1.1 port 49153
At time 3.00409s server sent 18 bytes to 10.1.1.1 port 49153
At time 3.00818s client received 18 bytes from 10.1.2.4 port 7
root@fekt:/home/student/ns-allinone-3.21/ns-3.21#
```

---

11. číslovaný úkol z návodu

Zadání úkolu: **Změňte parametr PacketSize echo zprávy na 1 (bajt) a opět si prohlédněte rámce ECHO protokolu (použijte opět soubor lab1_prom-2-0.pcap) a taktéž si poznačte délku libovolného ze zachycených rámců. Jak se hodnota změnila? Jaký to má důvod?**

Řešení: Dĺžka zachyteného ECHO rámca predstavuje rovnakú hodnotu ako v úlohe č. 10 a to je 64 B, jedná sa o minimálnu veľkosť rámca podľa zdroja: https://searchnetworking.techtarget.com/answer/What-are-the-minimum-and-maximum-sizes-of-an-ICMP-packet
Ako je možné vidieť na obrázku nižšie, tak hodnota RTT sa po zmene PacketSize na 1 B zmenila len minimálne a to rádovo o pár milisekúnd čo nepredstavuje žiadny zásadný rozdiel.



---

12. číslovaný úkol z návodu

Zadání úkolu: **Upravte zdrojový kód tak, aby probíhalo zachytávání paketů i na serveru (uzel n4). Doplnění zdrojového kódu uveďte do protokolu.**

Řešení: `csma.EnablePcap("lab1_node", csmaDevices.Get(3)); // daný riadok kódu som doplnil na riadok 87. Funkčnosť tohto kódu potvrdzuje aj zachytený lab1_node-4.0.pcap v zložke ns-3.21, ktorý je možné vidieť nižšie.`