VILNIUS TECH

Faculty of Electronics

Department of Computer Science

and Communications Technologies

Alexandre Sauvan

ISKfu-22*

Computer Networks

Report of laboratory work

Evaluate by Dr. Andžej Borel

Vilnius, 2025

Contents

List of Figures

List of Tables

Chapter 1

Part I: Internet Protocol (IP) addressing

1.1 Objectives

To become acquainted with the Open System Interconnection (OSI) model and Transfer Control Protocol and Internet Protocol (TCP/IP) stack. Understand the main principles of IP addressing and the division of a network into sub-networks.

1.2 OSI levels

There are 7 levels in the OSI model:

1. Physical
2. Data link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

1.2.1 Router (3 layers: physical, data link, network)

A **router** operates at three layers of the OSI model:

1. **Physical layer (Layer 1)**: It transmits electrical, optical, or radio signals through physical interfaces (Ethernet, Wi-Fi, etc.).

2. **Data link layer (Layer 2)**: In some cases, it handles MAC addresses (e.g., in NAT or VLANs).
3. **Network layer (Layer 3)**: It manages IP addresses, determines the best route for data packets, and performs routing.

Why Layer 3? A router needs to understand **IP addresses** to forward packets across different networks efficiently.

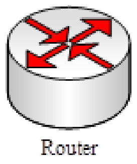


Figure 1.1: Schematic representation of a router

1.2.2 Switch (2 layers: physical, data link)

A **switch** typically operates at two layers:

1. **Physical layer (Layer 1)**: It transmits signals through Ethernet cables.
2. **Data link layer (Layer 2)**: It switches frames based on MAC addresses.

Why not Layer 3? A switch does not interpret IP addresses or perform routing. It simply forwards frames based on MAC addresses within the same local network.

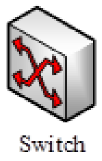


Figure 1.2: Schematic representation of a switch

1.2.3 Special case: layer 3 switch

Some advanced switches, called **Layer 3 switches**, can handle routing functions and operate at Layer 3, combining the features of a router and a switch.

1.3 Convert numbers

1.3.1 Binary to decimal conversion

To convert a **binary** number to **decimal**, we apply the weighted sum of powers of 2.

Method

Each bit is multiplied by 2^n, where n is its position starting from 0 (from right to left).

Example

Convert 1011_2 to decimal:

$$(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

$$= (1 \times 8) + (0 \times 4) + (1 \times 2) + (1 \times 1) = 8 + 0 + 2 + 1 = 11$$

Thus, $1011_2 = 11_{10}$.

1.3.2 Decimal to binary conversion

To convert a **decimal** number to **binary**, we successively divide by 2 and keep track of the remainders.

Method

1. Divide the number by 2.
2. Record the remainder (0 or 1).
3. Repeat with the quotient until reaching 0.
4. Read the remainders from bottom to top.

Example

Convert 13_{10} to binary.

Division	**Quotient**	**Remainder**
$13 \div 2$	6	**1**
$6 \div 2$	3	**0**
$3 \div 2$	1	**1**
$1 \div 2$	0	**1**

Reading the remainders from bottom to top gives 1101_2.
Thus, $13_{10} = 1101_2$.

1.4 IP address

1.4.1 IP address classes

Class	Address range	Mask
A	0.0.0.0 - 127.255.255.255	255.0.0.0 (/8)
B	128.0.0.0 - 191.255.255.255	255.255.0.0 (/16)
C	192.0.0.0 - 223.255.255.255	255.255.255.0 (/24)
D	224.0.0.0 - 239.255.255.255	None
E	240.0.0.0 - 255.255.255.255	None

1.4.2 Network and host numbers

Method

We can determine the network and host numbers of an IP address. In fact, if we have an IP address, we can determine the class in which it is. Then, we can find the **Subnet Mask**. Finally, we can determine the network part and the host part.

Example

- **IP address:** 173.233.252.206
- **Class:** Class B (since the first octet, 173, is between 128 and 191)
- **Subnet mask:** 255.255.0.0 (/16) for Class B

The first 16 bits (first two octets) represent the network part.

- **Network part:** `173.233.0.0`
- **Host part:** `252.206`

Thus, the network address is: **173.233.0.0** and the host address is: **252.206**.

1.4.3 Network example

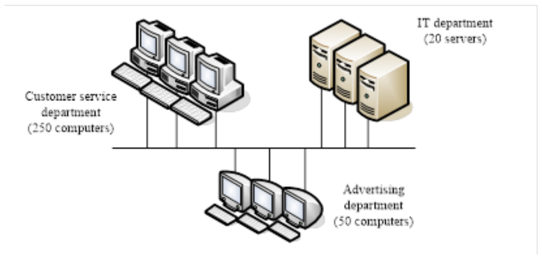


Figure 1.3: Network example

Network analysis

To determine the appropriate IP address class for the organization's network, we analyze the number of devices:

- **Customer service department**: 250 computers
- **Advertising department**: 50 computers
- **IT department**: 20 servers
- **Total devices**: 250 + 50 + 20 = **320 devices**

Choosing the IP address class

IP address classes are divided as follows:

- **Class A**: Supports 16,777,214 hosts (Range: 1.0.0.0 – 126.0.0.0)
- **Class B**: Supports 65,534 hosts (Range: 128.0.0.0 – 191.255.0.0)
- **Class C**: Supports 254 hosts (Range: 192.0.0.0 – 223.255.255.0)

Since the total number of devices exceeds the limit of **Class C (254 hosts)**, we need at least a **Class B** network.

IP address allocation

A **Class B network** typically uses a **default subnet mask of 255.255.0.0**, allowing a large number of hosts. We can allocate subnets as follows:

- **Customer service department (250 computers)**
  - Subnet: `172.16.1.0/24`
  - IP range: `172.16.1.1 - 172.16.1.254`
- **Advertising department (50 computers)**
  - Subnet: `172.16.2.0/26`
  - IP range: `172.16.2.1 - 172.16.2.62`
- **IT department (20 servers)**
  - Subnet: `172.16.3.0/27`
  - IP range: `172.16.3.1 - 172.16.3.30`

Conclusion

A **Class B IP addressing scheme** (e.g., `172.16.0.0/16` private range) is suitable for this organization, allowing efficient IP allocation and future expansion.

1.5 Reflections and feedback

1.5.1 Difficulties encountered

One of the main challenges was understanding how subnet masks work and how to choose the correct subnet size for different departments. Converting between binary and decimal also initially caused some confusion, especially when determining network and host portions of an IP address.

1.5.2 Solutions to the difficulties

To overcome these difficulties, I referred to additional documentation and practiced multiple examples. Visual aids, such as network diagrams and conversion tables, were also very helpful in making the concepts clearer.

1.5.3 Personal impressions

This part of the lab was very enriching because it provided a practical approach to IP addressing. It also emphasized the importance of subnetting for network scalability and management. I now feel much more confident in my ability to analyze and design IP schemes for different organizational needs.

Chapter 2

Part II: Internet Protocol (IP) addressing

2.1 Objectives

To become acquainted with the Open System Interconnection (OSI) model and Transfer Control Protocol and Internet Protocol (TCP/IP) stack. Understand the main principles of IP addressing and division of a network into subnets.

2.2 Bits allocated for subnets

Subnetting is a method used to divide a larger network into smaller, manageable subnetworks. This process helps optimize IP address allocation, improve security, and enhance network performance.

2.2.1 Concept of subnetting

In an IP network, an address is divided into two parts:

- **Network portion**: Identifies the network.
- **Host portion**: Identifies individual devices within the network.

By default, the number of bits allocated to the network portion depends on the IP address class. Subnetting extends the network portion by borrowing bits from the host portion.

Subnet masks

A subnet mask determines how many bits are used for the network and how many remain for hosts. It is expressed in dotted decimal notation or CIDR (Classless Inter-Domain Routing) notation.

For example, a subnet mask of:

$$255.255.255.0 \quad \text{(or /24 in CIDR notation)}$$

indicates that the first 24 bits belong to the network, while the remaining bits are for hosts.

Borrowing bits for subnetting

When subnetting, we extend the network portion by borrowing bits from the host portion:

- The more bits borrowed, the more subnets are created.
- The remaining bits determine the number of available hosts per subnet.

The number of subnets created is given by:

$$2^s$$

where s is the number of subnet bits.

The number of usable hosts per subnet is:

$$2^h - 2$$

where h is the remaining host bits, and the subtraction of 2 accounts for the network and broadcast addresses.

Example calculation

If an original network has a default subnet mask of /16 (16 network bits) and we extend it to /18:

- The extra $18 - 16 = 2$ bits are used for subnetting.
- The total number of subnets is $2^2 = 4$.
- The remaining $32 - 18 = 14$ bits are for hosts.
- The number of usable hosts per subnet is $2^{14} - 2 = 16,382$.

2.2.2 Conclusion

Subnetting is a crucial technique for efficient IP address management. By borrowing bits from the host portion, we can create multiple smaller networks, balancing the need for scalability and optimization.

2.3 Shortening an IPv6 address

2.3.1 Rules to shorten an IPv6 address

To convert an IPv6 address to its short form:

1. Remove leading zeros in each hextet.
2. Replace consecutive blocks of zeros with "::" (only once in the address).
3. Keep all non-zero hextets as they are.

2.3.2 Example

Full IPv6 address	Shortened IPv6 address
`2000:0015:abcd:efgh:0000:0000:0000:0001`	`2000:15:abcd:efgh::1`
`fe80:0000:0000:3145:0000:0000:0000:000f`	`fe80::3145:0:0:f`
`2001:0000:0000:0000:0000:0000:0000:0002`	`2001::2`

2.4 Creating link-local addresses (EUI-64)

2.4.1 Method to derive EUI-64 IPv6 link-local address

1. Split the MAC address into two 3-byte segments.
2. Insert "FF:FE" in the middle of the MAC address.
3. Flip the 7th bit of the first byte.
4. Append "fe80::" as the prefix.

2.4.2 Example

MAC address	EUI-64 link-local address
`5e-29-07-ec-ae-b8`	`fe80::5c29:7ff:feec:aeb8`
`22-34-e5-34-e9-54`	`fe80::2034:e5ff:fe34:e954`
`c5-bb-12-79-45-af`	`fe80::c7bb:12ff:fe79:45af`

2.5 Special-purpose IPv6 addresses

- **Loopback ("::1/128")** – Used to refer to the local machine.
- **Link-local ("fe80::/10")** – Used for communication between nodes on the same link.
- **Multicast ("ff00::/8")** – Used to send packets to multiple destinations.
- **Unspecified ("::/128")** – Represents the absence of an address.

- **Documentation (“2001:db8::/32”)** – Reserved for examples.
- **Unique local (“fc00::/7”)** – Private IPv6 addressing.
- **6to4 (“2002::/16”)** – Used for automatic tunneling.

2.6 Reflections and feedback

2.6.1 Difficulties encountered

Understanding the EUI-64 process was initially confusing, especially the bit-flipping step and inserting "FF:FE" correctly in the MAC address. Also, distinguishing when and how to shorten IPv6 addresses using the "::" notation was a bit tricky.

2.6.2 Solutions to the difficulties

To address these issues, I relied on visual aids and breakdowns of the EUI-64 conversion process. I also practiced several IPv6 shortening examples until I understood the rules and their limitations. Revisiting the theory after doing examples made the process much clearer.

2.6.3 Personal impressions

This part of the lab made me appreciate the flexibility and robustness of IPv6 compared to IPv4. I understand better how modern networks manage addresses. The EUI-64 method was particularly interesting because it automates part of the address configuration process.

Chapter 3

Network packet monitoring and analysis tools

3.1 Objectives

Learn how to use network packet monitoring and analysis tools. Analyze the operating principles of key network layer (IP) and transport layer (TCP and UDP) protocols.

3.2 Wireshark network protocol analyzer

Wireshark is a free and open-source network protocol analyzer. It is widely used by network administrators, security analysts and developers to capture and inspect data packets traversing a network in real time.

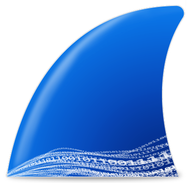


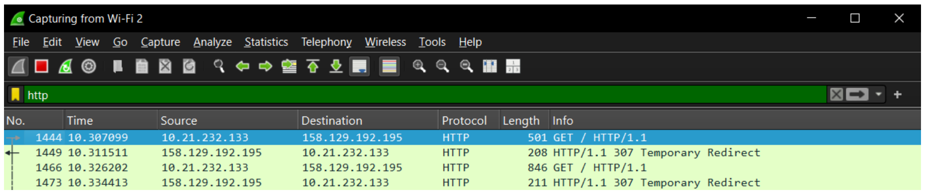
Figure 3.1: Wireshark logo

3.2.1 Search website IP address

Using this filter `dns.qry.name == www.vgtu.lt`, the IP address of `www.vgtu.lt` was identified as `158.129.192.2` by analyzing the DNS response packet.

3.2.2 HTTP filter

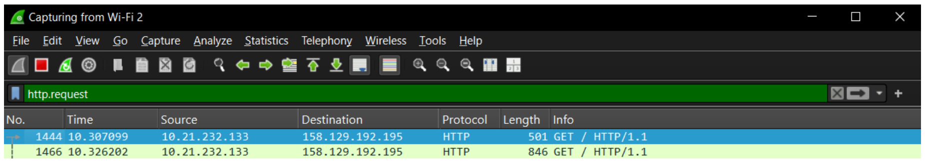
Applying the display filter `http`, only HTTP packets were shown.



```
No.   Time       Source           Destination      Protocol  Length Info
 1444 10.307099  10.21.232.133    158.129.192.195  HTTP       501 GET / HTTP/1.1
 1449 10.311511  158.129.192.195  10.21.232.133    HTTP       208 HTTP/1.1 307 Temporary Redirect
 1466 10.326202  10.21.232.133    158.129.192.195  HTTP       846 GET / HTTP/1.1
 1473 10.334413  158.129.192.195  10.21.232.133    HTTP       211 HTTP/1.1 307 Temporary Redirect
```

Figure 3.2: HTTP filter in Wireshark

The filter `http.request` displayed only request messages sent to the server.

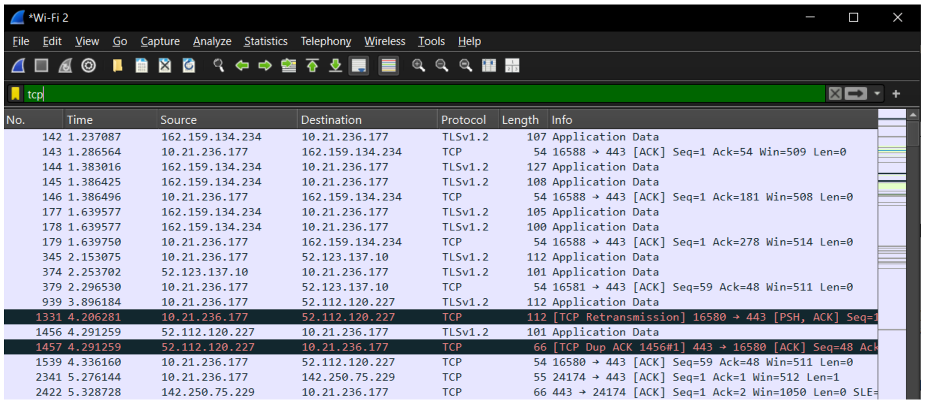


```
No.   Time       Source           Destination      Protocol  Length Info
 1444 10.307099  10.21.232.133    158.129.192.195  HTTP       501 GET / HTTP/1.1
 1466 10.326202  10.21.232.133    158.129.192.195  HTTP       846 GET / HTTP/1.1
```

Figure 3.3: HTTP Request filter in Wireshark

3.2.3 TCP filter

Applying `tcp` as filter displayed all TCP packets. However, TLS packets were also present.



```
No.   Time      Source           Destination      Protocol  Length Info
  142 1.237087  162.159.134.234  10.21.236.177    TLSv1.2    107 Application Data
  143 1.286564  10.21.236.177    162.159.134.234  TCP         54 16588 → 443 [ACK] Seq=1 Ack=54 Win=509 Len=0
  144 1.383016  162.159.134.234  10.21.236.177    TLSv1.2    127 Application Data
  145 1.386425  162.159.134.234  10.21.236.177    TLSv1.2    108 Application Data
  146 1.386496  10.21.236.177    162.159.134.234  TCP         54 16588 → 443 [ACK] Seq=1 Ack=181 Win=508 Len=0
  177 1.639577  162.159.134.234  10.21.236.177    TLSv1.2    105 Application Data
  178 1.639577  162.159.134.234  10.21.236.177    TLSv1.2    100 Application Data
  179 1.639750  10.21.236.177    162.159.134.234  TCP         54 16588 → 443 [ACK] Seq=1 Ack=278 Win=514 Len=0
  345 2.153075  10.21.236.177    52.123.137.10    TLSv1.2    112 Application Data
  374 2.253702  52.123.137.10    10.21.236.177    TLSv1.2    101 Application Data
  379 2.296530  10.21.236.177    52.123.137.10    TCP         54 16581 → 443 [ACK] Seq=59 Ack=48 Win=511 Len=0
  939 3.896184  10.21.236.177    52.112.120.227   TLSv1.2    112 Application Data
 1331 4.206281  10.21.236.177    52.112.120.227   TCP        112 [TCP Retransmission] 16580 → 443 [PSH, ACK] Seq=1
 1456 4.291259  52.112.120.227   10.21.236.177    TLSv1.2    101 Application Data
 1457 4.291259  52.112.120.227   10.21.236.177    TCP         66 [TCP Dup ACK 1456#1] 443 → 16580 [ACK] Seq=48 Ack
 1539 4.336160  10.21.236.177    52.112.120.227   TCP         54 16580 → 443 [ACK] Seq=59 Ack=48 Win=511 Len=0
 2341 5.276144  10.21.236.177    142.250.75.229   TCP         55 24174 → 443 [ACK] Seq=1 Ack=1 Win=512 Len=1
 2422 5.328728  142.250.75.229   10.21.236.177    TCP         66 443 → 24174 [ACK] Seq=1 Ack=2 Win=1050 Len=0 SLE=
```

Figure 3.4: TCP filter in Wireshark

This is because TLS operates on top of TCP: TLS packets are actually encapsulated in TCP segments. Wireshark displays them under the name of their application protocol for greater clarity, even though they are technically TCP packets.

3.2.4 IP filter

To filter packets sent from an IP address or to an IP address, there are two different explicit filter:

- `ip.src == [IP]` (source)
- `ip.dst == [IP]` (destination)

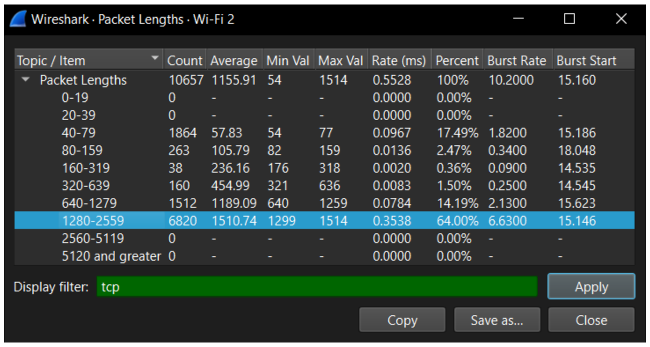
3.2.5 Other examples of filters

- To display packets sent to TCP port 80: `tcp.dstport == 80`.
- To search for a specific string `tcp contains "public"`.
- To show ARP and DNS packets: `arp || dns`.
- To exclude HTTP from TCP: `tcp && !http`.

3.2.6 Statistics

Wireshark offers various statistical tools accessible via the **Statistics** tab. Key options include.

- **Summary**: to consult total packets or average bandwidth.
- **Protocol Hierarchy**: to see number of ARP, UDP, TCP packets.
- **Conversations**: to know packets or bytes number sent between any IP.
- **Packet Lengths**: to see most and least common bytes number.



Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	10657	1155.91	54	1514	0.5528	100%	10.2000	15.160
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	1864	57.83	54	77	0.0967	17.49%	1.8200	15.186
80-159	263	105.79	82	159	0.0136	2.47%	0.3400	18.048
160-319	38	236.16	176	318	0.0020	0.36%	0.0900	14.535
320-639	160	454.99	321	636	0.0083	1.50%	0.2500	14.545
640-1279	1512	1189.09	640	1259	0.0784	14.19%	2.1300	15.623
1280-2559	6820	1510.74	1299	1514	0.3538	64.00%	6.6300	15.146
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Figure 3.5: TCP packet lengths in Wireshark

Please note that the statistics take into account the filter if it is applied! It's necessary to remove it to see all statistics.

3.3 Analysis results of the TCP protocol

3.3.1 Netcat listening on port 23

The command used:

```
netstat -an | find "23"
```

Output confirms Netcat is listening on port 23.

3.3.2 TCP 3-Way Handshake

TCP uses a *three-way handshake* mechanism to establish the connection. The TCP handshake occurs in three separate steps, as shown in figure 3.6.

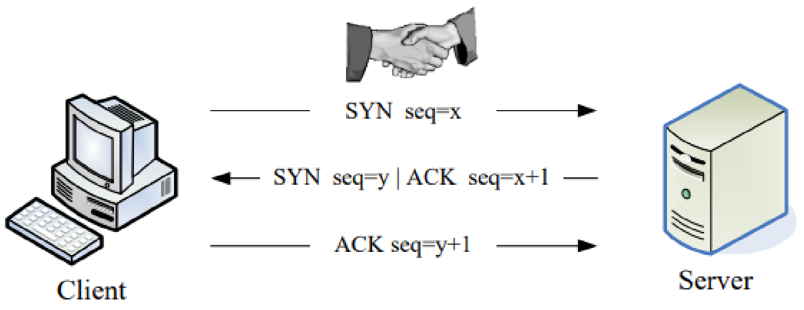


Figure 3.6: TCP three-way handshake

To summarize this, I drew up a table to analyze captured packets with an example port (here 1025).

Field	**1st packet**	**2nd packet**	**3rd packet**
Source IP address	Guest IP	Host IP	Guest IP
Destination IP address	Host IP	Guest IP	Host IP
Source port number	1025	23	1025
Destination port number	23	1025	23
Flags	SYN	SYN, ACK	ACK
Relative Sequence Number	0	0	1
Absolute Sequence Number	134567890	198765432	134567891
Relative Ack. Number	–	1	1
Absolute Ack. Number	–	134567891	198765433

Table 3.1: TCP 3-Way Handshake Analysis

Note: Ack. means Acknowledgment.

3.3.3 Host acknowledge

When SYN packet from Guest starts with sequence number x, the Host acknowledges with $x+1$ because the initial the acknowledgment number indicates the next expected byte (SYN counts as 1 byte).

3.3.4 TCP data transfer

Analysis of TCP segments when sending character strings shows how the protocol handles sequence and acknowledgment numbers. Each character sent is counted as a byte, and the acknowledgment number (ACK) returned by the receiver is the sum of the initial sequence number and the size (in bytes) of the data transmitted. This illustrates TCP's ability to track every byte exchanged, ensuring reliable and in-order delivery.

3.3.5 TCP connection termination process

The final four capture packets represent the TCP connection termination process. The following table shows the analysis of these packets.

Field	**1st (FIN)**	**2nd (ACK)**	**3rd (FIN)**	**4th (ACK)**
Source IP	[Guest]	[Host]	[Host]	[Guest]
Destination IP	[Host]	[Guest]	[Guest]	[Host]
Source Port	[Random]	23	23	[Random]
Destination Port	23	[Random]	[Random]	23
Flags	FIN, ACK	ACK	FIN, ACK	ACK
Rel. Seq. Num.	[e.g. 111]	[same]	[e.g. 200]	[same]
Rel. Ack. Num.	[e.g. 200]	[111+1]	[new]	[new+1]

Table 3.2: TCP Connection Termination

3.3.6 Closed port attempt

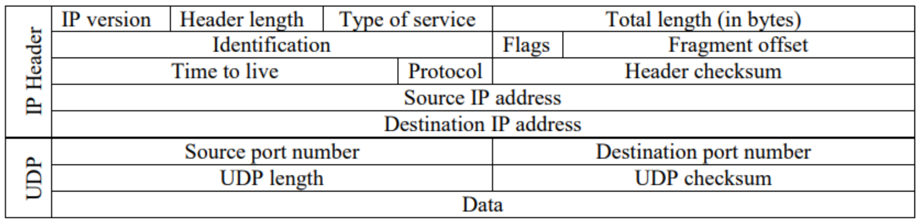
When the Telnet client attempts to initiate a TCP connection to a closed port on the Host, the server responds to the initial SYN segment with a TCP segment that has the **RST (Reset)** flag set. This response indicates that no service is listening on the specified port, and the connection is immediately refused.

Observing the captured packets in Wireshark, it can be seen that the Telnet client typically retries the connection multiple times, each followed by a RST response from the Host. This behavior illustrates how TCP handles connection attempts to unavailable services by resetting the connection promptly.

3.4 Analysis results of the UDP protocol

3.4.1 UDP packet structure

The IP and UDP header structure is displayed by the figure 3.7 below.



IP Header	IP version	Header length	Type of service	Total length (in bytes)	
	Identification			Flags	Fragment offset
	Time to live		Protocol	Header checksum	
	Source IP address				
	Destination IP address				
UDP	Source port number			Destination port number	
	UDP length			UDP checksum	
	Data				

Figure 3.7: Structure of IP and UDP headers

Each field plays a specific role, as described in the table below.

Field	**Description**
IP Version	Indicates IP version, typically IPv4 (value: 4)
Header Length	Length of IP header, usually 20 bytes
Type of Service	Priority and service flags (e.g. 0x00 = default)
Total Length	Full length of IP packet (e.g. 58 bytes)
Identification	Packet ID for fragmentation (e.g. 0x1c46)
Flags	Control flags (e.g. 0x4000 = Don't Fragment)
Fragment Offset	Offset of fragment in data (0 if not fragmented)
Time To Live (TTL)	Max hops before packet is discarded (e.g. 64)
Protocol	Protocol type (17 = UDP)
Header Checksum	Error-checking value for IP header
Source IP Address	IP address of sender (Guest)
Destination IP Address	IP address of receiver (Host)
Source Port	Port used by sender (e.g. 54321)
Destination Port	Listening port on receiver side (e.g. 5555)
UDP Length	Length of UDP header and data (e.g. 38 bytes)
UDP Checksum	Error-checking value for UDP segment

Table 3.3: UDP Packet Header – Field Descriptions and Example Values

3.4.2 Response to UDP packet

The Host responded with an **ICMP Destination Unreachable** packet. A lack of response does not necessarily mean the packet was not received; UDP does not guarantee delivery.

3.5 Conclusions

This laboratory session provided practical experience in capturing, filtering, and analyzing network traffic using Wireshark. We explored the structure and behavior of the TCP and UDP protocols, including connection establishment and termination, and compared connection-oriented and connectionless communication.

Tools like Netcat and Telnet allowed us to simulate traffic in a controlled environment, enhancing our understanding of network interactions. This experiment emphasized the importance of protocol analyzers in both troubleshooting and learning contexts.

3.6 Reflections and feedback

3.6.1 Difficulties encountered

One challenge was filtering specific traffic types without accidentally excluding relevant packets. For example, when filtering TCP traffic, TLS packets appeared unexpected at first. Understanding that TLS is encapsulated in TCP segments clarified the confusion.

Another difficulty was interpreting some Wireshark fields, especially the sequence and acknowledgment numbers in TCP packets. Their absolute and relative values required close attention to analyze the handshake and termination phases correctly.

It was also initially unclear why some UDP packets received no response. Recognizing that UDP is connectionless helped understand this behavior.

3.6.2 Solutions implemented

I carefully reviewed Wireshark's documentation to correctly apply display filters. Using combined filters improved result precision.

To better understand TCP behavior, packet values were tabulated, and the relative/absolute values were compared side-by-side. This approach clarified the logic behind sequence numbers and TCP's reliability mechanisms.

For UDP behavior, I referred to the ICMP response and Wireshark protocol stack analysis to confirm delivery attempts and understand error responses.

3.6.3 Personal impressions

This lab was highly instructive. It helped me realize how much network protocol behavior can be observed and understood visually through packet captures. Tools like Wireshark give deep insight into underlying network operations that are usually invisible.

I particularly appreciated dissecting TCP's handshake and termination phases, as they are fundamental but often abstract in theory. Observing real packets helped bridge that gap.

Overall, this session improved both my technical skills and my confidence in using protocol analyzers effectively.

Chapter 4

Network routing using static routes and RIP

4.1 Objectives

Design a computer network which uses static routes and then enhance routing using RIP.

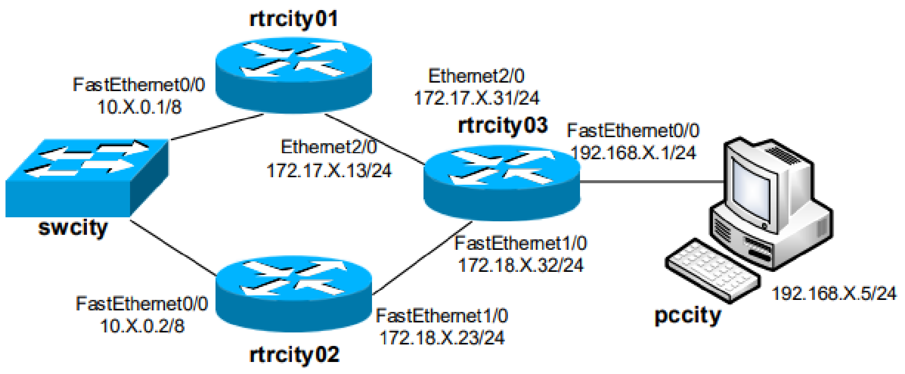


Figure 4.1: Network diagram

I chose city in Lithuania from the proposed list in the laboratory work to make the network asked with GNS3:

- `rtrcity01`: Vilnius
- `rtrcity02`: Šiauliai
- `rtrcity03`: Ukmergė
- `swcity`: Kaunas

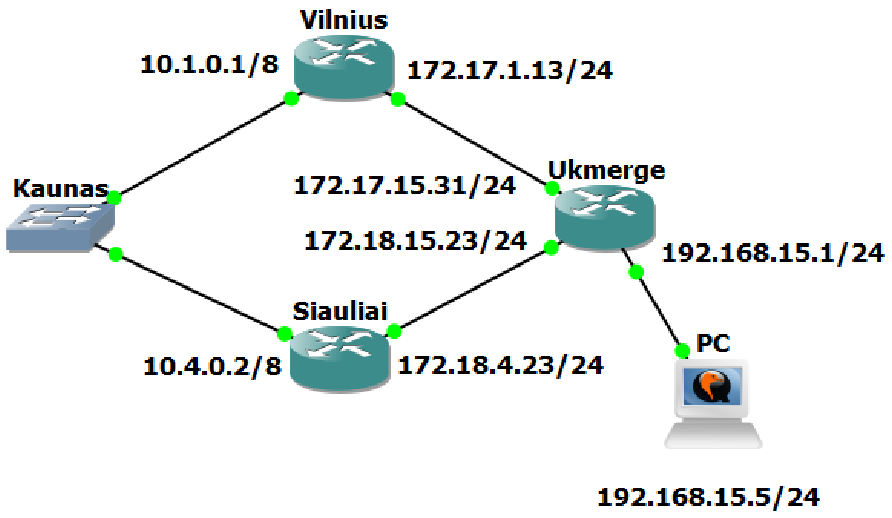


Figure 4.2: Lithuanian network from GNS3

4.2 Cisco Discovery Protocol (CDP)

Cisco Discovery Protocol (CDP) is a Layer 2 proprietary protocol developed by Cisco to allow Cisco devices to discover and share information with directly connected neighbors. It provides details such as device ID, interface, platform, and capabilities. CDP is mainly used for network discovery and troubleshooting within Cisco-based environments.

```
Vilnius#show cdp
Global CDP information:
        Sending CDP packets every 60 seconds
        Sending a holdtime value of 180 seconds
        Sending CDPv2 advertisements is  enabled
```

Listing 4.1: Settings of CDP on the router Vilnius

The global CDP settings on the Vilnius router are as follows:

- CDP packets are sent every **60 seconds**.
- The **holdtime** for CDP information is set to **180 seconds**, meaning CDP entries expire if no updates are received within this time.
- **CDP version 2 (CDPv2)** advertisements are **enabled**.

These settings confirm that CDP is actively running on the device and is configured to maintain neighbor information for up to three minutes.

```
Vilnius#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Src R. Bridge
```

```
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID    Local Intrfce     Holdtme    Capability  Platform  Port ID
Siauliai     Fas 0/0            179        R S I      3640      Fas 0/0
Ukmerge      Eth 2/0            159        R S I      3640      Eth 2/0
```

Listing 4.2: Configured neighbouring devices

The output of the `show cdp neighbors` command on the Vilnius router reveals the following topology:

- **Vilnius** is directly connected to two neighboring Cisco 3640 routers:
  - **Siauliai** via interface `FastEthernet 0/0`, connected to Siauliai's `FastEthernet 0/0`.
  - **Ukmerge** via interface `Ethernet 2/0`, connected to Ukmerge's `Ethernet 2/0`.
- Both neighbors support routing (R), switching (S), and IGMP (I) capabilities.
- CDP hold times are 179 and 159 seconds respectively, indicating active communication.

Only directly connected devices are shown, as expected from Cisco Discovery Protocol behavior.

4.3 Running configurations

During this task, two routing approaches were configured and compared on routers `Vilnius` and `Ukmerge`.

4.3.1 Static routing

Static routes were manually defined using the `ip route` command to direct traffic to specific networks. The following routes were present:

- `ip route 192.168.15.0 255.255.255.0 172.17.15.31` – defines a route to the local PC network via a next-hop address.
- `ip route 192.168.15.0 255.255.255.0 10.4.0.2` – alternative route added when the first becomes invalid.

These routes had to be manually removed during network restoration using the `no ip route` command.

4.3.2 RIP configuration

RIP (Routing Information Protocol) was enabled on routers `Vilnius` and `Ukmerge` with the following commands:

```
Vilnius (config)#router rip
Vilnius (config-router)#network 10.1.0.0
Vilnius (config-router)#network 172.17.1.0
```

Listing 4.3: Vilnius router

```
Ukmerge (config)#router rip
Ukmerge (config-router)#network 192.168.15.0
Ukmerge (config-router)#network 172.17.15.0
Ukmerge (config-router)#network 172.18.15.0
```

Listing 4.4: Ukmerge router

This configuration allowed the routers to automatically exchange routing information for directly connected networks.

4.3.3 Main differences

The key difference between static routing and RIP lies in how routing information is managed:

- **Static routing** requires manual configuration and updating of all route entries.
- **RIP** dynamically advertises routes to neighbors, adapting to changes automatically.

Thus, RIP offers scalability and automatic recovery in case of link failure, while static routing provides full control but requires more manual effort.

4.4 Routing table

In this section, we analyze the routing table of the routers `Vilnius`, `Siauliai`, and `Ukmerge` when static routes and RIP (Routing Information Protocol) are used.

4.4.1 Routing table with static routes

When static routes were configured on `Vilnius` and `Siauliai`, the routing tables displayed explicit routes to specific networks. These routes were manually configured using the `ip route` command, as described earlier.

For example, on `Vilnius`, the routing table included:

- Direct route to the `192.168.X.0/24` network via next-hop `172.17.X.31`.
- Another direct route to the `192.168.15.0/24` network via next-hop `10.4.0.2`.

On `Siauliai`, similar static routes were configured, and both routers had their respective default routes pointing to the appropriate gateways.

4.4.2 Routing table with RIP

After enabling RIP on `Vilnius` and `Ukmerge`, the routing table dynamically populated with entries for remote networks. The routers exchanged routing information, and RIP automatically adapted to changes in the network.

On `Vilnius`, the RIP routing table displayed:

- Networks learned from RIP advertisements, such as `192.168.15.0/24`, `172.17.15.0/24`, and others.
- RIP-specific metrics indicating the cost to reach these networks (usually in hops).

A similar process occurred on `Ukmerge`, where the routing table reflected dynamic updates based on RIP updates from `Vilnius`.

4.4.3 Key differences between static routes and RIP

The main differences observed in the routing table between static routing and RIP are:

- **Static Routes:** Explicit routes to specific destinations with manual configuration. They do not change unless manually updated.
- **RIP:** Dynamically updated routing entries based on RIP advertisements. Routes can change automatically based on network topology.

In RIP, routes typically have a metric (hop count) associated with them, which does not exist with static routes unless manually configured. Additionally, RIP allows for easier scalability as it dynamically adjusts to network changes.

4.5 RIP routing steps

4.5.1 RIP routing process

RIP (Routing Information Protocol) was enabled on the routers `Vilnius`, `Siauliai`, and `Ukmerge`. RIP uses periodic updates to share routing information with directly connected neighbors.

By enabling debugging with `debug ip rip`, the following behaviors were observed:

- RIP version 1 was used by default.
- Routing updates were sent every 30 seconds.
- The broadcast address `255.255.255.255` was used to send updates.
- RIP version 1 does not support authentication or subnet information.
- Each route advertised carried a metric representing the hop count.

4.5.2 Switching to RIP version 2

RIP version 2 was then enabled on all routers with the following configuration:

```
Vilnius#conf t
Vilnius(config)#router rip
Vilnius (config-router)#version 2
Vilnius (config-router)#no auto-summary
```

RIP v2 brought several improvements:

- Multicast address `224.0.0.9` was used to send updates instead of broadcast.
- Support for Classless Inter-Domain Routing (CIDR) and subnet masks.
- Route summarization was disabled to ensure all subnet routes were correctly propagated.

4.5.3 Observations and differences

After switching to version 2 and reviewing both the routing tables and protocol information using `show ip route` and `show ip protocols`, the following was noted:

- RIP version 2 was now clearly indicated in protocol outputs.
- The routing table contained more specific subnet routes due to disabled summarization.
- The routers were using multicast to exchange routes, reducing unnecessary traffic.
- Debug output confirmed RIP v2 behavior: periodic updates sent every 30 seconds, received via multicast.

4.6 General conclusions

This project demonstrated the configuration and comparison of two fundamental routing methods: static routing and dynamic routing using RIP. By setting up a Lithuania-themed network topology using GNS3, we were able to simulate real-world routing behavior on several Cisco routers.

Initially, static routing enabled precise control of network paths, but required manual configuration and lacked adaptability. Any change to the topology required manual intervention, which could quickly become unmanageable in large networks.

By enabling Routing Information Protocol (RIP), we introduced a dynamic routing mechanism that simplified network management. RIP automatically exchanges route information between routers, adapting quickly to topology changes without the need for manual updates. The transition from RIP version 1 to RIP version 2 has further improved routing by supporting subnet information and reducing broadcast traffic through multicast communication.

In conclusion, while static routes are suitable for restricted or controlled environments, RIP offers scalability, flexibility and greater fault tolerance for expanding or more complex networks.

4.7 Reflections and feedback

4.7.1 Difficulties encountered

During the task, one difficulty was ensuring proper route propagation when switching from static routing to RIP. Initially, some routes were not correctly advertised between routers, especially when using RIP version 1, due to the lack of subnet mask support.

Another challenge was the manual removal of static routes before RIP could function correctly, as conflicting routes sometimes caused inconsistent behavior in the routing table.

4.7.2 Solutions implemented

To resolve the RIP advertisement issue, I switched to RIP version 2 and disabled automatic summarization using the `no auto-summary` command. This allowed the routers to correctly propagate subnet information.

To avoid conflicts between static and dynamic routes, I carefully removed all static routes before enabling RIP. Additionally, I used the `debug ip rip` and `show ip protocols` commands to verify and troubleshoot routing behavior.

4.7.3 Personal impressions

This task helped me better understand the differences between static and dynamic routing. I appreciated the practical demonstration of how RIP simplifies network management by dynamically adapting to topology changes. It was also interesting to see the importance of choosing the correct RIP version, as version 1 quickly showed its limitations.

Overall, this lab was very useful for reinforcing routing concepts and practicing real-world configurations in a simulated environment.