



**SLEZSKÁ
UNIVERZITA**
FILOZOFICKO-
PŘÍRODOVĚDECKÁ
FAKULTA V OPAVĚ

Šárka Vavrečková

Study materials for

Computer Network and Internet

Institute of Computer Science
Faculty of Philosophy and Science
Silesian University in Opava

Opava
March 21, 2024

Summary: This document is intended for students of the subject Computer Network and Internet at the Silesian University in Opava. It contains the material taught in the lectures, there is the second document for exercises.

Computer Network and Internet


RNDr. Šárka Vavrečková, Ph.D.

Institute of Computer Science
Faculty of Philosophy and Science
Silesian University in Opava
Bezručovo nám. 13, Opava

Typeset with L^AT_EX

Preface

What we can find in this document






 *Quick preview:* This document is intended for students of computer science at the Silesian University in Opava. It contains the material taught in the lectures of Computer Network and Internet, which deals with (as the name suggests) computer networks. This document contains mainly theory (how devices connected to the network work, how communication is formed, etc.), and it is supplemented by document for exercises.




We will mainly deal with local networks (Ethernet, Wi-fi) and related mechanisms including network services such as DNS. The last topic discussed is security, where we will learn about the principle of VPNs, network analyzers and the filtering process in a firewall.

Some areas are “additional” (marked with purple icons), these are not discussed and will not appear on the exam – their purpose is to motivate further independent study or experimentation or to help in the future to gain further information. If there is a purple icon in front of a chapter (section) title, it applies to everything in that chapter or section.

Marking

We use the following colourful icons:

-  *Quick Preview*, in which we find out what it’s going to be about.
-  *Keywords*.
-  *Study Objectives* for a chapter tell us what new things we will learn in the chapter.
-  New *terms* are marked with the blue symbol, seen here on the left. This icon (as well as the following) can be found at the beginning of the paragraph in which the new concept is introduced.
-  *Methods* and tools, ways of solving various situations that a computer administrator may find himself in, etc. are also marked with a blue icon.

-
-  Some parts of the text are marked with a purple icon, indicating that they are *optional sections* that are not discussed (mostly; students can request them or study them on their own if they wish). Their purpose is to voluntarily expand students' knowledge of advanced topics that usually don't get much time in class.
 -  The yellow icon indicates links where you can get *further information* about the topic. Most often, these icons are web links to sites where the authors go into more detail about the topic.
 -  Red is the icon for *warnings* and notes.

If the amount of text belonging to a certain icon is larger, the whole block is delimited by a space with icons at the beginning and at the end, for example to define a new concept:



Definition

In such an environment, we are defining a concept or explaining a relatively familiar but complex concept with multiple meanings or properties.



Similarly, the environment may look like a longer procedure or a longer note or more links to more information. Other environments may also be used:



Example

This is what the environment looks like with an example, usually of a procedure. The examples are usually annotated to make it clear how to solve them.



Tasks

Questions and tasks, suggestions for testing, which are recommended for practicing the material, are enclosed in this environment. If there are more than one task in the environment, they are numbered.



Contents

Preface	iii
1 Introduction to Computer Networks	1
1.1 How It Works	1
1.1.1 Network Service and Network Technology	1
1.1.2 Components of a Computer Network	2
1.1.3 Data Transmission	4
1.1.4 Topology	7
1.2 Active Network Equipment	10
1.3 What and How We Transfer	13
1.3.1 Data and Information	13
1.3.2 Protocols: how Electronic Strangers Talk to Each Other	14
1.3.3 Properties of Protocols	15
1.4 Standards and Standardization Institutions	16
1.5 The ISO/OSI Reference Model	19
1.5.1 Overview	19
1.5.2 Cooperation of Protocols	23
1.5.3 Protocol Stacks	25
1.6 The TCP/IP Network Model	26
1.7 Transmission Characteristics	29
1.7.1 Transmission Signal	29
1.7.2 Baseband and Broadband Transmission	30
1.7.3 Multiplexing	34
2 Local Networks – Ethernet	37
2.1 The Meaning of Ethernet	37
2.2 Communication in Ethernet	38
2.2.1 Device Types	38
2.2.2 Interconnection of Network Nodes	38
2.2.3 Collision Method	39
2.3 Ethernet at the Data-link Layer	40
2.3.1 L2 Addresses	40
2.3.2 L2 Sublayers	42

2.3.3	EtherType	43
2.3.4	Ethernet Frame Format	44
2.3.5	Collision and Broadcast Domain	47
2.4	Ethernet at the Physical Layer	48
2.4.1	Copper Cabling	49
2.4.2	Optical Cabling	52
2.4.3	Crossover	56
2.5	Working with Cables	58
2.5.1	Crimping the connector to a straight-through UTP cable	58
2.5.2	Testing	60
2.5.3	Crimping the Crossover UTP Cable	61
2.5.4	Installation of UTP cable into Outlet	62
2.5.5	Optical Terminations	64
2.5.6	Other Ethernet Connectors and Modules	65
2.6	Parameters	67
2.7	Course of Transmission	69
2.7.1	Half Duplex	70
2.7.2	Full Duplex	73
2.8	Ethernet-related Technologies	74
2.8.1	Autonegotiation	74
2.8.2	Power over Ethernet	74
2.9	Structured Cabling	75
2.9.1	Rack	75
2.9.2	Horizontal Cabling	76
2.9.3	Backbone Networks	78
2.9.4	Structured Cabling as a Whole	80
3	Additional topics on local area networks	81
3.1	VLAN	81
3.1.1	VLANs on a single switch	82
3.1.2	VLAN frames on the path through the trunk	84
3.1.3	Inter-VLAN Routing	86
3.2	Switches and Loops	90
3.2.1	The STP Protocol	90
3.2.2	Convergence of the Switched Network	92
3.2.3	The STP Protocol and its Variants	94
3.3	Overview of the IEEE 802 Standards	95
4	Network and Transport Layer	97
4.1	Network Layer and Logical Addresses	97
4.2	IPv4 Protocol	98
4.2.1	IPv4 Addresses	98
4.2.2	Special IPv4 Addresses	100
4.2.3	IPv4 Packets	100
4.2.4	TTL and Packet Lifetime	102


4.2.5	MTU and IPv4 Packet Fragmentation	103
4.3	IPv6 Protocol	105
4.3.1	IPv6 Addresses	106
4.3.2	Special IPv6 Addresses	107
4.3.3	IPv6 Packets	109
4.4	ICMP Protocol and Control Messages	111
4.4.1	Purpose of ICMP	111
4.4.2	ICMPv6	113
4.4.3	Reachability Testing	114
4.5	Transport Layer	117
4.5.1	Port Numbers	117
4.5.2	Transport Layer Protocols	118
4.5.3	TCP	119
4.5.4	TCP Connection	121
4.5.5	UDP	124
5	Network Addresses and Routing	126
5.1	Addresses in Frames and Packets	126
5.2	Neighbors Discovery	128
5.2.1	Neighbors Table	128
5.2.2	Protocols	129
5.3	IPv4 Address Range	130
5.3.1	Classes	130
5.3.2	Subnetting	132
5.3.3	VLSM	136
5.3.4	CIDR	137
5.4	How to Get an IP Address	139
5.4.1	How to Get IPv4 Address	139
5.4.2	How to Get IPv6 Address	139
5.5	NAT and Private Addresses	141
5.6	Routing	144
5.7	How Routing Works	144
5.8	Cooperation of Routing Protocols	145
5.8.1	Autonomous System	145
5.8.2	Distance Vector Routing Algorithm	146
5.8.3	Link-state Algorithm	147
5.8.4	BGP	148
6	Wireless Local Networks	149
6.1	Wireless Technologies	149
6.2	Wi-fi	150
6.2.1	Physical Layer	151
6.2.2	Signal and Antennas	155
6.2.3	Multiple Antennas	159
6.2.4	Access point	160

6.2.5	Wi-fi at L2 Layer	162
6.2.6	Collision Method	163
6.3	Security in Wireless Communication	164
6.3.1	AAA	164
6.3.2	Security Protocols	165
6.3.3	WPS	168
6.3.4	So how to Secure a Wireless Network	169
7	Application Protocols	170
7.1	DNS Service	170
7.1.1	Domains and Name Service	170
7.1.2	Zones and DNS Servers	172
7.1.3	Evaluation of DNS Requests	173
7.1.4	Host Table	175
7.1.5	DNS Protocol and DNS Packet	176
7.1.6	WHOIS Database	177
7.2	WWW Service and HTTP Protocol	178
7.2.1	Addresses	178
7.2.2	HTTP Communication	179
7.2.3	HTTP Information Codes	182
7.3	E-mail Service	183
7.3.1	Infrastructure	183
7.3.2	Protocols	183
7.3.3	Communication and Settings	184
7.3.4	MIME	186
7.4	File Services	187
7.4.1	FTP	187
7.4.2	Sharing Resources on a Local Network	189
7.5	IP Address Assignment and DHCP	189
7.6	Other Server Types	191
7.7	Remote Access and Remote Communication	192
7.7.1	Telnet	192
7.7.2	SSH	193
7.8	Network Management	195
7.9	Protocols and Ports Overview	195
8	Introduction to Network Security	197
8.1	Network Security	197
8.1.1	Types of Attacks	197
8.1.2	Security at Various ISO/OSI Layers	200
8.2	VPN	201
8.2.1	Principle of VPN	201
8.2.2	Tunnels in Linux	205
8.3	Network Analyzer	207
8.4	Firewall	208


8.4.1	Principle of Firewall	208
8.4.2	Filtering	209
8.4.3	IDS/IPS and Deep Packet Inspection	211
Recommended Resources		214


Chapter 1

Introduction to Computer Networks

 *Quick preview:* In this chapter we will explain the basic concepts related to computer networks, and in the following chapters we will build on these concepts. We will get acquainted with the terms used to describe data transmission, topologies, and common network hardware.

The last section of this chapter is devoted to the physical parameters of transmission paths. We will learn the basics of signal handling, modulation and multiplexing principles.

 *Keywords:* Network equipment, link, channel, circuit, ISP, PAN, LAN, MAN, WAN, duplex, simplex, datagram service, virtual circuit, transmission, best effort, topology, point-to-point link, backbone, data, information, repeater, hub, switch, bridge, router, gateway, protocol, standard, ETSI, ITU, ISO, IEEE, IEC, IETF, IANA, ANSI, TIA, EIA, protocol data unit (PDU), ISO/OSI reference model, protocol suite, protocol stack, TCP/IP, transmission signal, carrier signal, frequency, wavelength, amplitude, phase, baseband, broadband, modulation, multiplex.

 *Objectives:* After studying this chapter, you will have an overview of the basic concepts of computer networking and be prepared to study the topics discussed in the following chapters. You will learn to navigate the layers of the ISO/OSI reference model and the TCP/IP network model. You will understand the basics of signal transmission.

1.1 How It Works

1.1.1 Network Service and Network Technology

While service determines what is provided, technology determines how it works. Table 1.1 shows the relationship between some well-known network services and their associated technologies.



Remark

The predecessor of the Internet was ARPANET (Advanced Research Projects Agency Network) launched in 1969. It was a US Department of Defense project, an experimental network funded by DARPA (Defense Advanced Research Projects Agency). Originally, it was a networking of powerful computers that could be accessed remotely. Among European countries, Norway was

Service	Technology
internet access	xDSL, WiMAX, GPRS, EDGE, LTE, . . .
internet telephony	VoIP, VoLTE, . . .
e-mail	SMTP, POP3, IMAP, . . .
WWW	HTTP, HTTPS, . . .
files transmission	FTP, SFTP, . . .
remote management	rlogin, telnet, SSH, . . .


Table 1.1: Network services and network technologies

the first to join ARPANET (1973). ARPANET was discontinued in 1990, when its successor, the Internet, was fully operational.



1.1.2 Components of a Computer Network

In the following we will use the term computer network, but this is inaccurate, in fact it is far from just computers (however, the term is quite common).

 Intuitively, we certainly understand that a computer network is about connecting certain devices so that these devices can communicate with each other. What kind of devices do we want to connect? Certainly a computer, a laptop, a tablet, a smartphone, a server, possibly a smart TV or other “smart device” – anything is smart nowadays, there are even kitchen appliances that can be connected to a computer network. These types of nodes (at the beginning or end of a communication path) are collectively called *client devices*. Furthermore, a computer network includes the network elements through which communication takes place:

- *active network equipment* – actively influence communication, they provide routing, signal amplification, etc.,
- *passive equipment* – only passively transfer data, e.g. cables.

Definition (Computer network)

A computer network is a sum of devices – nodes – interconnected by *transmission paths*. A node is either a terminal equipment or an active network equipment.



In the definition we see two important concepts – node and transmission path. We’ve already covered the concept of a node, but what is a transmission path? It can be one particular *transmission medium*, such as a passive network element, such as a metallic cable, or it can be air (for Wi-fi). Or the transmission path can be a combination of several various transmission media.

Definition (Link, transmission channel, transmission circuit)


Link is a general (abstract) term for the transmission path used between nodes in a network.

Transmission channel is a unidirectional transmission path determined not only by the nodes it connects, but also by physical characteristics – transmission rate, bandwidth, noise level, etc.

Transmission circuit is a bidirectional transmission path, it can be a pair of transmission channels (one channel for each direction, simply multiple channels – at least one for each direction).




Transmission circuits as communication paths for data are usually created by an organization that leases them to its customers. Such a circuit is called a leased circuit.

 **Definition (Network Connectivity Provider, Internet Service Provider)**

Network Connectivity Provider is an organization that provides a network access service. An *ISP* (Internet Service Provider) is an organization providing access to the Internet (it may intermediate the service of another – superior – ISP).



 Transmission circuits exist in the following variants:

- *physical circuit* – designed directly by the physical transmission path, not interrupted by any intermediate node, not much used today (only for private secure paths),
- *virtual circuit* (virtual circuit) – logical circuit leading through various shared physical transmission paths, can be temporary, when re-created can lead through different physical intermediate nodes each time, multiple virtual circuits can lead in one physical transmission path, according to the duration we distinguish
 - *permanent virtual circuit* (PVC) – exists stably throughout the lease/ownership period, long term,
 - *switched virtual circuit* (SVC) – it is created dynamically when communication is needed, so it exists only for a short period of time.

Permanent virtual circuits are considerably more expensive, but on the other hand they bypass intermediate network nodes (which means faster communication) and are more resilient to the failures that come with circuit sharing. However, switched virtual circuits are more widely used in today's wide area networks because they are cheaper, more flexible and do not unnecessarily block the capacity of the transmission path when the circuit is not in use.

 **Remark**

Previously, circuits were switched manually (the switchboard operator had to connect two contacts with a wire), but today this operation is automated even in traditionalist telecommunications.



VPN (Virtual Private Network) lines can also be considered a very specific type of switched circuit.

 Depending on the size, the following types of computer networks can be distinguished:

- PAN (Personal Area Network) – a network in a small area (e.g. connecting devices via Bluetooth, IrDA, USB),
- LAN (Local Area Network) – spread over tens to hundreds of meters, usually within a single building (e.g. Wi-fi, Ethernet),
- MAN (Metropolitan Area Network) – on the scale of a city or a block of buildings (kilometres, tens of kilometres), used to interconnect different LANs (e.g. WiMAX), often Internet access networks,

- WAN (Wide Area Network) – large networks covering a region, state, continent, etc., used to interconnect smaller networks (e.g. different LANs and MANs), performs a similar role to MAN but at a higher level, the Internet is also a set of WANs.

Metallic cables or air (radio links – electromagnetic waves) can be used as the transmission medium in all types, in LANs and larger ones we can meet fiber optic cables (the more extensive the more likely), infrared transmission is used in PAN networks, in MAN networks we can meet laser links.

In this course we will mainly deal with LANs.

1.1.3 Data Transmission


In a network, it is not so much about the connection of nodes, but primarily about the possibility of data transfer between these nodes. The link (transmission path, connection) is only a means, the goal is *transmission*: transmission always proceeds using a link.

 We distinguish transmission

- *simplex* – communication is in one direction only,
- *full duplex* – communication is in both directions,
- *half duplex* – communication is in both directions, but not at the same time (nodes alternate in communication, they cannot transmit both in parallel at the same time).

A full-duplex link can be implemented by two or more simplex links.

Most of the time, of course, we want to communicate bi-directionally, but sometimes this is not technically possible; some transmission paths need to be reserved, because sharing would cause collisions. In wireless solutions, where the transmission path is air, sharing is solved by multiplexing: for example, by allocating different frequencies or by allocating transmission time.

 The transmission is further divided according to the form of the data transmitted into


- *stream-oriented* – a continuous stream of data is transmitted, which does not need to be defined or internally subdivided in any way, *packet-oriented* (block, . . .) – data is pre-divided (packed) into blocks of a certain size, provided with “logistic” information (addresses, content type, handling along the way, etc.) and sent as such, unpacked and reassembled at the destination.

Stream transmission is typical e.g. for voice transmission over a telecommunication line (after the connection is established, the audio signal is transmitted without any division) or in hardware, e.g. for HDMI (multimedia signal transmission). Packet transmission is, on the other hand, typical for computer networks, in the hardware domain for the DisplayPort interface.

The communication is in fact structured, several entities (protocols) cooperate to send data. The communication can be packet-oriented from the point of view of one protocol, from the point of view of another protocol it can be stream-oriented (the data stream can actually be a stream of packets, but the given protocol does not recognize these packets from its point of view).

The concept of a “transmission circuit” is defined a bit above. Stream-oriented transmission is closely related to circuits – it always proceeds in such a way that a circuit over which the transmission is supposed to be performed, or a connection is established over this circuit, and data can start to “flow”.

With packet transmission we do not have to (but can) bind to a specific circuit. The address or other identification of the destination is part of the packet, so if the data is split into multiple packets, each packet may take a different physical path. The important thing is that all packets arrive at the destination, and there is also a procedure for how to order them (packets can arrive in a different order than they were sent).

 From the above it follows that we are sending either a stream of data or individual packets. At the boundary between two data streams or two packets, *switching* is performed. We distinguish:

- *circuit switching* – the entire stream of data is transmitted over the same circuit, which must be assembled in advance; the data always arrives in the correct order,
- *packet switching* – it is not necessary to build a transmission circuit, packets contain the information needed to be directed to the correct destination, the packet goes the way that is considered optimal at the moment, at the destination it is necessary to order the packets correctly.




Remark

Circuits are similar to a railway network – the locomotive and its carriages run on one specific dedicated track, the carriages are not mixed along the way. Switching *packages* is like a road network – a large group of people can travel in multiple cars, with each car taking a different route (each driver has his/her own idea of the best route), and they may arrive at their destination in a different order than they left.

While telecommunications stay with “railways” (circuits), computer networks rely more on “roads” (packets).



Packet switching as a principle first appeared in the ARPANET in 1964 (Paul Baran), because the concept of circuit switching common in telecommunications was found in experiments to be unsuitable for sending data: reserving a circuit would make it impossible for anyone else to use that path, and overall the use of circuits would be inflexible. However, the term “packet” itself has been used since 1965 (Donald Davies).

 Depending on whether the connection is established before the actual transmission, we distinguish between transmission

- *connection-oriented* – first the connection is established, followed by the transfer, then the connection is terminated,
- *connectionless* – no connection is established immediately before the actual transfer.

What is the relationship between circuit/packet switching and connection-oriented/connectionless service?


- Circuit switching is always a connection-oriented service. Creating a transfer circuit is always establishing a connection, terminating a circuit after data transfer is terminating the connection.
- Packet switching can be done in two different ways:
 - *datagram service* – connectionless transmission method, the packet itself contains the recipient’s full address,

- virtual circuits – a connection-oriented service where the data is split into packets but the path for transmission is predetermined, switched virtual circuits (SVCs) are used.

When using virtual circuits the packets contain only short routing information for the active network elements. There is then an entry in a special table on the network elements indicating in which direction the packet with that particular information should be sent. So the recipient's full address doesn't actually need to be in the packet or on the intermediate network elements (there is only local information about which network interface the path leads to the recipient).

Typical examples are cells in an ATM network (they have headers only 5 B long), frames in a FrameRelay network, more recent examples are MPLS networks.

During transmission, the transmitted data may be damaged or lost (e.g. due to interference, crosstalk, weak signal). This is commonly encountered, for example, in Wi-fi: when the network becomes congested, the transmission slows down not only because the network elements can reduce the speed when detecting problems, but also because the corrupted packets have to be resent.

 Depending on the reliability, the transmission is divided into

- *reliable* – if a transmission error is detected (corrupted or missing packet), a retransmission is requested (e.g. a corrupted packet is resent),
- *unreliable* – a corrupted packet is simply discarded, a missing packet is ignored.



Remark

Note: unreliable service does not mean that the data may not arrive at the destination at all! It simply means that someone else will take care of the error handling. Networking technologies are typically divided into layers where each layer performs a specific task, and if one layer provides unreliable service, another layer can provide error handling.



Providing a reliable service (with error handling) is advantageous in terms of error occurrence, but on the other hand such a service is time consuming. Therefore, unreliable transmission is used wherever either the occasional lost packet does not matter much or another layer deals with errors. For certain types of traffic (e.g., audio transmission), it is usually not even necessary to retransmit lost packets: if something is missing here and there, the human ear will not notice it.




Definition (Best Effort Principle)

The *Best Effort* principle addresses service unreliability as follows:

- *best effort* to deliver the packet,
- *no guaranteed result* – if there are not enough resources, bandwidth, etc., delivery will not occur.




Maximum effort means that as long as there are enough resources, packets will be delivered, but when resources run out, packets will be dropped without distinction until new resources are found. This principle is not just about packet delivery, but about managing a limited set of resources in general.

 According to the number of recipients we distinguish communication

- *unicast* – the data sent is intended for only one receiver,
- *multicast* – the data is addressed to *group* of receivers (all belonging to a specific group), broadcast (omnidirectional communication) – data is addressed to all connected, *anycast* – data is addressed to one (any) receiver from a specific group.


1.1.4 Topology

The concept of topology is usually related to space and the relationships (arrangements) between objects in (physical or logical) space.


 **Definition (Physical and logical network topology)**

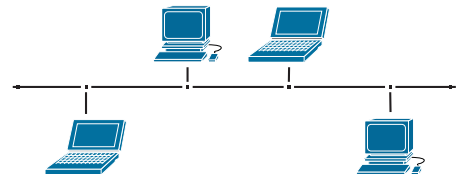
In computer networks we distinguish between physical and logical topology:

- the physical topology determines how the nodes of the network are interconnected and where they are specifically located,
- logical topology determines how these nodes communicate with each other, who “sees” whom directly, what is the hierarchy of communication (addresses).

With topology, the path length, throughput, or technology used is not important, rather the way the nodes are physically or logically connected. 

Nowadays, the concept of topology is not as important as it was in the past, but we do encounter topologies from time to time, so we will introduce a few of the most common ones.

 **The bus topology.** In terms of the physical bus topology, the transmission path is shared by all connected devices, no central element is needed. We also talk about a segment or backbone (this is the cable to which all devices are connected). A signal sent by a device propagates in both directions, it can be received by any other connected device. Usually both ends of the backbone need to be terminated with an *terminating resistor*, which absorbs the signals to prevent back propagation and thus interference with other communications. The devices are connected to the main cable using *T-bar connectors*, as we can see in Figure 1.1 on page 8.



The advantage is the simplicity and lower cable consumption, just cable between devices should be run as efficiently as possible. If the cable is damaged, the network is divided into two parts, and if the possible back propagation of the signal from the place of damage is temporarily solved, then it is possible to continue communication within the two segments thus created (but not between them).

The disadvantage is that the link load is too high – all devices share the same transmission path, and any signal sent is propagated along that path.

As a logical topology, this means that if one device is transmitting, the others must be “silent” because interference would occur.

¹From: <http://www.blbbytes.com/2014/06/networking-primer-part-6-2-media-access-control-csmacd-csmaca/>

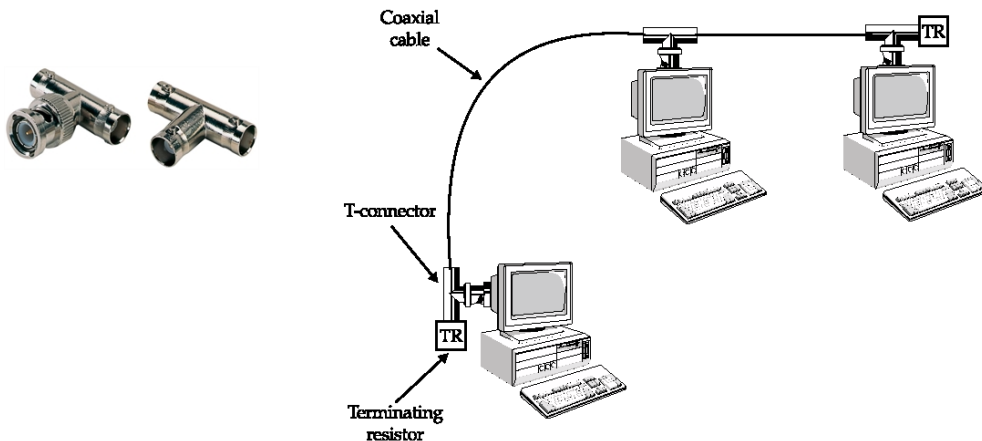
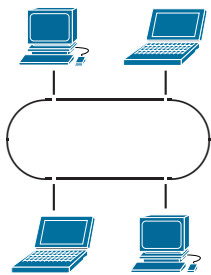


Figure 1.1: Two T-bar connectors and the Ethernet bus topology¹

Currently, this topology is not even used as a physical topology in computer networks, we meet it rather in cable TV, the related “cable internet” (DOCSIS standard). Previously it was used in Ethernet (over coaxial cable) and some hardware interfaces (SCSI as Daisy-Chain). As a logical topology it was used in older slower generations of Ethernet with hubs or operating at half duplex.



The ring topology. It is basically a variation of the bus topology with connected ends, the properties of this topology as a physical topology are similar to the properties of the bus, including the way the signal propagates, but as a logical topology it works a bit differently – for example, the principle of “token” as a special packet allowing transmission can be used, going around in a circle, thus uniquely determining which connected device can transmit at any given time.

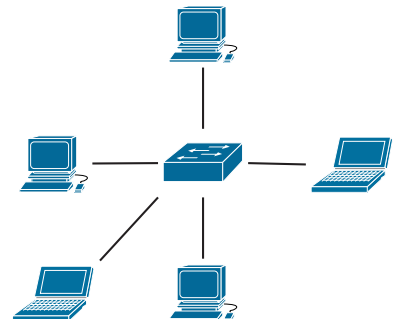
The advantages and disadvantages are similar to those of the bus.

In the past, this topology was used as the logical topology for Token Ring (local) and FDDI (WAN) networks.

The star topology. There is one central element in the network to which the other nodes of the network are connected. All communication takes place through this central element.

A star-type logical topology means that the individual lines are separated from each other and the devices do not interfere with each other.


The advantage is that if the link between a node and the central element is damaged, the network continues to function (only that one node cannot communicate). Another advantage is the possibility of separating the communication paths between each pair of nodes (the separation can be provided at the logical level by the central element).

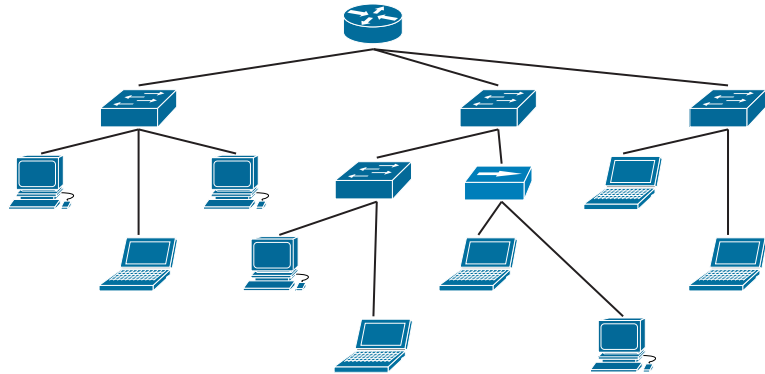


The disadvantage is that the central element is the “communication bottleneck” (i.e. any communication must go through it, it is the most heavily loaded), furthermore, damage or overload

of the central element means collapse of the whole network. The disadvantage is also the higher consumption of cabling (compared to the bus topology).

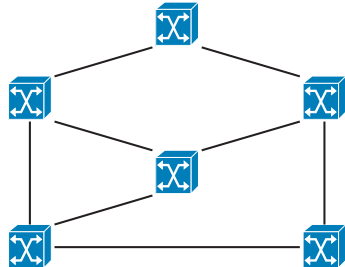
This topology is the basis of today's local area networks.


 **The tree topology.** It is a generalization of the star topology. The devices are divided into levels and the separation of traffic in the different parts of the network is assumed (as a physical topology). This does not mean that nodes in different parts cannot communicate with each other, they just do not interfere with each other (logical topology).



The advantage is easier network management and low risk of link congestion, the disadvantage is higher cable consumption (especially if the network is built spontaneously without a plan). If one of the internal nodes fails, its subtree will collapse.

The tree topology is commonly encountered in medium and large local area networks and is the basis of *structured cabling*. Ethernet is a typical representative.



 **The mesh topology.** Typical for this topology are *redundant links* (i.e. more than one path can lead between two nodes). The nodes in the network are more or less equivalent (only at the edges there may be other types of nodes that mediate the communication of internal nodes with the “rest of the world”). These networks are characterized not only by redundancy, but also by a large number of nodes, which makes pathfinding in the network more challenging.

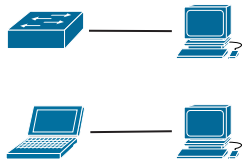
Full mesh topology means full interconnection of network nodes, i.e. everyone is connected to everyone. If the interconnection is not full, we talk about partial mesh.


This type of physical topology (partial mesh) is typical for current WANs (MPLS, etc.). Here, path reliability must be ensured, and traffic is on average large and variable, so redundancy is essential.

Full mesh as a logical topology is found where the individual network nodes must all know about each other and know the path to each network node. A typical example is a network of routers in the same domain using the same routing protocol.


We also encounter the mesh topology when we create more links in the switch network than necessary (some of them are redundant), but since in this case redundancy would mean that the network would be overwhelmed by repeatedly sent data units (which we will get to in Chapter 3 in the STP section), the redundant ports are disabled. Redundant paths usually do not work, but in the case of a failure of the “main” path, the backup path is automatically made operational quickly.


The advantage is high fault tolerance, the disadvantage is high cable consumption (in case of physical topology).



 **Point-to-point link.** Simple connection of two devices. As a physical topology, it is useful if we really only need to connect two devices, but as a logical topology it is currently much used – for example, if we have a logical star topology, we have as a central element a device that can separate communication with each connected device at the logical level: the different parts of this network are then just point-to-point links at the logical level (each end device “sees” only the central element).

It was mentioned above that the communication structure in the network is hierarchical. For higher levels of the hierarchy, no intermediate devices are visible (lower layers take care of the path determination), so at higher levels only the end nodes of the whole path are able to see each other. Here we can also speak of a logical point-to-point topology.


 **Backbone.** The backbone is the part of the network that connects its individual subnetworks. It is assumed that active network elements capable of separating traffic are connected directly to the backbone (they know whether to send the packet to “your” network or away to the backbone). In larger networks, the backbone is multi-tiered (each tier is hierarchically arranged). We have actually encountered the backbone in tree topology.

 **Remark**

The Internet also has a backbone. In the beginning it was formed by its predecessor ARPANET, in the second phase several networks of commercial ISPs became the backbone. Nowadays, the structure of the Internet is built on three levels:

- Tier 1 (Backbone Providers) – ISP organizations, mostly multinational in scope, that own the most important parts of the Internet backbone and control the routing of traffic on that backbone,
- Tier 2 (National Providers) – ISPs of national scope that operate their national portion of the Internet backbone, pay a Tier 1 ISP for access to the main backbone, and operate access points to that backbone (POPs – Point of Presence),
- Tier 3 (Local Providers) – there are (should be) several of these ISPs within a country, competing with each other, buying connectivity from their national Tier 2 provider and offering services (Internet access) to end customers (enterprises, general users, government institutions, etc.). They operate an access network for their customers, which is the third tier of the Internet backbone.




 **Additional information**

<http://www.slideshare.net/abdurrehmanabdurrehman391/iap-03-isp-hierarchy-types-of-delays>



1.2 Active Network Equipment

 Every device connected to the network must have at least one *network interface*, i.e. a component through which it can communicate with network (receive packets or send them). For a

regular computer, laptop, etc., this is simply a *network card*.

The network interface can be integrated on the motherboard of the device (this is quite common lately, especially for small devices) or it can be a dedicated card (for plugging into a PCI or PCIe slot) or external (connected mostly via USB) – network interface card (NIC).



Figure 1.2: Network interface cards (From: Heureka.cz)

In Figure 1.2 there are four different network cards for connecting a regular computer to a local network. The first card is dedicated to a PCIe slot and allows communication over Ethernet, the second is external with a USB interface and is also for Ethernet. The remaining two network cards allow communication over a Wi-fi wireless network, and also one is dedicated to PCIe and the other is external to USB.

The first two cards in Figure 1.2 have a physical *port* into which we insert the cable connector, because Ethernet is a network using cables. The other two cards have a port in the form of an antenna for wireless communication, in the second case integrated antenna. If the network card is integrated and needs physical ports, we find them on the back panel in the case of a traditional computer, or on the back or side in the case of laptops.

Servers have similar network cards, but usually with more than one port (either a backup connection, or parallel communication, or for extending the communication bandwidth – e.g. NIC Teaming), also we can encounter different plugs/interfaces than for regular computers.

Active network devices, i.e. devices that are not end devices (communication goes through them, not only from them or only to them), of course also need network interfaces. However, they usually have many more of them than regular computers or servers, because they work as communication brokers, intermediate devices connecting ones, tens, . . . of other devices. Because networking is their main activity, their network interfaces are typically integrated.

The following is an overview of common active network equipment. Each is also accompanied by information about which ISO/OSI layer it belongs to.



Repeater is simple active network device with two ports. The incoming signal is amplified (or regenerated) and sent on. Repeaters work at the physical layer of ISO/OSI (L1).

Nowadays, we encounter repeaters in wireless local area networks, but also, for example, HDMI, where they serve to increase the signal range.



Hub is simple active network device usually with more than two ports. Anything comes to a port, is only resent to all other ports (the signal is amplified or regenerated, it is similar to repeaters). Hubs work with signal only, at the physical layer of ISO/OSI (L1).

The advantage of hubs is high speed of signal processing, the disadvantage is that it unnecessarily floods the network (data is sent to all neighbors except for the sender). We usually do not meet hubs in common networks, they are used e.g. in some PAN networks or in USB communication.



Switch is a bit smarter active network equipment that keeps a table of device addresses (neighbors and other members of a network). It detects the recipient of an incoming packet (in the given network, usually not the final recipient), determines the port leading to the recipient according to the address table, and send the packet only to the found port. If the recipient's address is not present in the table or if it is a broadcast packet, it sends to all ports excluding the sender's port (flooding). The switch divides the network into segments – if nodes from the same segment communicate with each other, the such communication does not penetrate into another segment.

Switches work at the data-link layer (L2), but they can also include functionality of higher layers (= multilayer switch).

The advantage is that this device does not flood the network as much as hubs, and it is possible to implement basic management and security functions. The disadvantage is more computationally intensive operation (this is taken into account, a many functions are implemented in hardware). We commonly meet switches in LAN, MAN and WAN networks.



Bridge is a device that separates two network segments from each other, like a switch. Unlike a switch, it usually has fewer ports, its functionality is implemented in software and it is standardized (IEEE 802.1). In practice, we do not encounter bridges. Bridges work at the data-link layer (L2).



Router maintains a routing table – different from a switching table of switches; a routing table contains L3 addresses (e.g. IPv4 or IPv6 addresses) of networks and subnets, not the terminal equipment. The incoming packet is “dissected” a little more thoroughly than by a switch, finding out which network the recipient belongs to based on the destination address, verifying the outgoing port leading to the recipient's network. Broadcasts are dropped.

Whilst switches know only the own network and no other networks, routers don't know the interior of the own networks, they know only paths to networks as a whole. Switches separate various segments of one network, routers separate various networks and work at the network layer (L3).

The advantage is possibility of implementating advanced management and security functions and the ability to separate communication in different networks. The disadvantage is even more computationally intensive operation than switches, so typically routers have fewer ports (therefore less traffic), a more powerful processor, and more memory to handle higher computational load.



Gateway is intended for interconnection of two different types of networks, it serves as a connection point and “translator”. For example: if we have a VDSL router at home, then this device has a built-in gateway for communication between our local network (Wi-fi and Ethernet) and the ISP's VDSL network.

Gateways usually work at some higher layer than protocols to interconnect.



Figure 1.3: Examples of switches (D-Link and Cisco) and routers (Cisco and HP, back panels) (From: Heureka.cz)


1.3 What and How We Transfer

1.3.1 Data and Information


Humans usually count in decimal because they have ten fingers, but it is more natural for computers to count in binary because electronic components can be in one of two states – off/on, no current passing/ current passing. Also, the data we send over a computer network is actually in binary format. So far, we have used the term data for what is transmitted over the network, which is OK in most cases. However, a more specific term is information.

Definition (Information)

Information is data that reduces or removes uncertainty from the system. Information can be received, sent, stored and processed.

By *Quantity of information* we mean the difference between the state of uncertainty of the system *before* and *after* receiving the information. 

The extent to data being information is relative – it depends on the person making the decision, and how “new”, “useful” the data is to him/her.

 The unit of information is the *bit*, which can take one of two values – 0 or 1. The derived unit is the *Byte*, which usually takes up 8 bits. One Byte in some protocols can be only 7 bits, but usually it takes 8 bits.

Also, prefixes are used to specify multiplicative quantities – either SI (multiplicity 1000 = 10^3) or binary multiples (multiplicity $2^{10} = 1024$). Binary multiples are standardized as IEC 60027-2.

Table 1.2 shows the meaning of the suffixes used for multiplicity (in the case of the Byte unit). The difference between SI multiplicity and binary multiplicity is something to watch out for:


- 1 MB = 1 000 000 B
- 1 MiB = 1 048 576 B

This means that 1 MiB is 48 576 Bytes larger, a difference of almost 5%.

Name	Mark	Quantity in Bytes
Kibibyte	KiB	1 KiB = 2^{10} B = 1024 B
Mebibyte	MiB	1 MiB = 2^{10} KiB = 1024 KiB = 2^{20} B
Gibibyte	GiB	1 GiB = 2^{10} MiB = 1024 MiB = 2^{20} KiB = 2^{30} B
Tebibyte	TiB	1 TiB = 2^{10} GiB = 1024 GiB = 2^{20} MiB = 2^{30} KiB = 2^{40} B
etc.		

Table 1.2: Binary multiples

In computer science, one important principle is preferred – unambiguity. We need to express ourselves unambiguously, precisely, so that it is always clear what and which quantity we are talking about. Therefore, in addition to the term Byte, there is a more precise equivalent: octet.

 *Octet* is a unit of information quantity representing 8 bits. The meaning of this concept is reflected in its name – in Latin, “eight” is “octo”.

Thus, in most cases the terms Byte and octet are synonyms, but the term octet is always unambiguous, which is why it is used more in computer networking in particular.

1.3.2 Protocols: how Electronic Strangers Talk to Each Other

In a computer network, we can connect quite different devices – it’s not just that one computer is running Windows in a certain version, the next one is running Windows in another version, then there’s some Linux, a server with Solaris, MacOS, etc., on active network elements we often have Linux (really!), IOS (not the one from Apple, but from Cisco) or something completely different... Devices in a network differ in their hardware much more than software. How do such “strangers” actually manage to communicate with each other?

In the human world, we communicate in a language – Czech, English, Spanish, German, . . . , and we can converse with anyone who either speaks a language that we speak, or there is a willing translator who speaks one of our languages and one of the languages of our communication partner.

Language determines how things are named, how we are to greet at the beginning of a communication, how we are to pass on certain information to the other party, how the other party responds to this message, how we learn that the other party does not understand us, how we are to end the conversation and say goodbye.

In the world of electronics, instead of languages, there are *protocols*. A protocol specifies how communication is to begin, or how to agree on certain parameters of communication, how to pass certain types of information to the other party, how the other party is to acknowledge receiving or say that the transmission needs to be repeated, how communication is terminated, etc.

Definition (Protocol and its implementation)

A protocol is a convention according to which a certain type of (e.g. electronic) communication takes place. It defines the rules specifying syntax (how which signals are put together), semantics (meaning) and synchronisation of communication.

A protocol is just a set of rules that needs to be implemented (programmed). The implementation can be software, hardware or a combination of software and hardware.



For example, the HTTP protocol (among other things) determines how to communicate between a web browser and a WWW server. It is implemented (programmed) inside the operating system of the computer you are sitting at, and also on the other side – inside the operating system of the WWW server you are communicating with. Its implementation is usually in software.



Remark

There is one important rule for protocol design (in fact, it is followed elsewhere, such as UNIX): the protocol should be simple, short, and unambiguously implementable. It should not be too complex, because the greater the complexity, the greater the likelihood of errors. Therefore, no protocol is very universal – it can do one particular thing, and it can do it well. In English, this is expressed by the acronym KISS (Keep it Simple, Stupid).

In order to respect this principle, there are certain conventions for protocol cooperation: what a protocol cannot do, it passes on to another protocol for processing.



The principle of protocols is not just a matter of computer networks – protocols are used in telecommunications, consumer electronics, engineering, and even inside computers. Everyone has encountered USB, SATA,...


1.3.3 Properties of Protocols

Common protocols have usually the following important properties:

- they are well known and nearly everybody can implement them (except of proprietary ones),
- they are rather simple, not very complex,
- they can cooperate with other protocols.

Why is the first property important? Imagine a situation where a network hardware manufacturer comes up with a new device and decides that the device will communicate according to a new protocol, the specification of which it will not provide to anyone else. However, this device only understands devices that support the same protocol, so it cannot communicate with devices from other manufacturers. It might be advantageous for the manufacturer in question if it could persuade its potential customers to buy only from him, but the reality is different – customers would rather buy equipment that they could easily integrate into their network, in which they already probably have some devices from other manufacturers.


Therefore, a lot of protocols are either freely available (the specification is completely free to view, anyone can implement) or at least available in another way (for a fee, under a license).

 *Open protocol* is a protocol with freely available specification, on the contrary, a *proprietary protocol* is a protocol whose specification is not published anywhere and its creator either keeps it to himself or licenses it for a fee.

**Remark**

The prevalence of free specifications is evidenced, for example, by the fact that currently the most common routing protocol on routers is the open OSPF protocol. In contrast, IGRP (Cisco's proprietary routing protocol) was hardly used on devices from Cisco itself, and the specification of

its successor EIGRP was released by Cisco for other manufacturers (but the other manufacturers remain with OSPF, and Cisco implemented OSPF as well).

If a manufacturer supports some own proprietary protocol for a given function, most likely some open protocol for the same function is supported too, so the device could “have a talk” with devices from other manufacturers. 

Let’s focus now on the second and third feature – simplicity and the ability to interoperate with other protocols. With any product (hardware or software), the more complicated it is, the higher the probability of errors and the more difficult it is to create. In the technology field, development is moving forward very quickly, so the time aspect of creating or updating products is also important, besides security. Moreover – different types of devices have a certain set of features in common, they just differ in some ways. Therefore, it is common to create smaller pieces of code (programs, protocols, or in the case of hardware, pieces of components) and then compose them appropriately into the desired whole.

1.4 Standards and Standardization Institutions




Definition (Standard)

Standard (in the field of information technology) is a requirement to meet certain specific characteristics for a certain type of hardware, software, etc. It is a document describing requirements, specifications, procedures, and descriptions for a given product, process, or service. We distinguish standards

- *normative (de iure)* – their observance is required, usually determined by the relevant state institution or an international organization with bindings to state institutions,
- *recommendation (de facto)* – their observance is not required, but it is highly recommended and it is in the interest of manufactures to engage in wider infrastructure.

In the field of information technology we often simply say a “standard”.

Standardization is the proces of unifying the features and function of a given product or service by determining a standard. 

The following is an overview of the most important standardization organizations and institutions that issue standards related to computer networks, and technologies in general. Most of them are de facto standards (they are not obligatory, but they are usually observed).




ETSI (European Telecommunications Standard Institute) is although originally a European organization, actually its members are from all inhabited continents (states, major manufacturers of communication equipment, network service providers, research organizations, etc.). Proposals for new standards or changes to existing ones come from three sources – either agreed by at least four ETSI members, or an initiative is received from the European Commission (EC) or the European Free Trade Association (EFTA).

There are several types of ETSI standards: for example, EN (European Standard), ES (ETSI Standard), TS (ETSI Technical Specification) and others. The best known standards are related

to mobile networks (especially GSM, 3G, 4G networks), smart networks, machine-to-machine communication, ICT in healthcare, etc. Most of ETSI standards are freely available.

 <https://portal.etsi.org>

 **ITU** (International Telecommunications Union) is a global organization for standardization of information and communication technologies. It is a part of the UN (United Nations) and is based in Switzerland. It has three parts:

- ITU-R: for radiocommunications (originally CCIR),
- ITU-T: standardization of telecommunications (originally CCITT),
- ITU-D: development of telecommunications.

ITU-T is part of the ITU for the standardization of telecommunications, its activities are also related to computer networks. Voting members are the countries concerned, and any organization can be a non-voting member. Membership is paid.

ITU-T is further divided into *Study Groups* (SG), each with its own focus. For example, SG13 has recently released several standards on cloud computing, SG16 deals with multimedia (video transmission, images, sound, etc.), SG17 security, SG20 smart networks (Internet of Things, . . .). The standards created by ITU-T are open, freely available.


We meet the ITU-T standards quite commonly: for example, the H.323 protocol used in multimedia transmissions and Internet telephony is one of them, or G.992.1 and G.992.2 protocols for ADSL access networks, X.509 standard used in encryption, and others.

 <http://www.itu.int/en/ITU-T/Pages/default.aspx>

ITU-R is part of the ITU for radiocommunications, and therefore deals with standards for wireless transmissions. For example, standards for LTE Advanced (formally IMT Advanced), such as M.2012 – *Detailed specifications of the terrestrial radio interfaces of International Mobile Telecommunications Advanced (IMT-Advanced)*, are currently much discussed.

The standards beginning with the letter “M” are related to mobile radiocommunication networks, “RS” for wireless networks of sensors and their remote handling, “SM” for frequency spectrum management, etc.

 <http://www.itu.int/pub/R-REC>


 **ISO** (International Organization for Standardization) is an independent organization issuing standards for communication technologies. Its members are standardization institutions from various countries (each country has a representation there), while ISO is a member of the ITU. ISO is based in Switzerland.

ISO has a hierarchical structure. It is divided into 200 TCs (Technical Committee), each technical committee has its own scope. For example, ISO/TC 47 deals with chemistry, ISO/TC 20 with aircraft and space vehicles, ISO/TC 272 forensic sciences, ISO/TC 299 robotics, ISO/IEC JTC 1 with information technology.

Each TC is divided into subcommittees (SC) and each subcommittee is divided into working groups (WG). Proposals for standards or their change always go from below, from working groups, gradually reworked and “fight through” up until the approval of the standard. The working group creates an *Working Paper*, which is transported into the *concept* of the Committee Draft, followed by the *draft International Standard*, the last step is the *international standard*.


The most “popular” standard by ISO is the well-known reference model ISO/OSI (the standard ISO 7498 and multiple related standards). ISO also deals with sensor networks, remote access, UPnP, web services and other topics.

 http://www.iso.org/iso/home/standards_development/list_of_iso_technical_committees.htm

 **IEEE** (Institute of Electrical and Electronics Engineers, the English pronunciation is similar to “Eye-triple-E”, so it is [aj tripl i:]) is the world’s largest organization of experts in the fields of electronics, electrical engineering, informatics and related fields, so it is a professional association. It does not only deal with standards, but organizes various meetings, conferences, educational events, publishes professional journals and develops other activities. The IEEE is based in the USA. One of the active sections of the IEEE is the Czech one.

The IEEE Association is primarily behind a group of IEEE 802 standards for local and, in part, large-scale networks, such as IEEE 802.11 for wireless local area networks (Wi-Fi) or IEEE 802.3 for Ethernet. Access to the full text of the standards is paid.

 <http://www.ieee.org>

 **IETF** (The Internet Engineering Task Force) deals mainly with standards related to the Internet, including, for example, TCP/IP protocols. It works very closely with other organizations, such as The Internet Assigned Numbers Authority (IANA), the Internet Society (ISOC), the Internet Architecture Board (IAB), the World Wide Web Consortium (W3C), and others.

The standards issued by IETF usually begin with the abbreviation RFC (Request for Comments) followed by a number. If it is necessary to update some standard, the original text remains as the original RFC and a new document with a new RFC is created.

For example, for the above-mentioned open routing protocol OSPF, several documents (versions) were gradually created, of which they are currently up to date. RFC 2328 (OSPF version 2) and RFC 5340 (OSPF version 3).




Additional information

The official website of IETF is <https://www.ietf.org/>. All RFC documents are freely available, there are even a number of websites to access these documents:

- <https://tools.ietf.org/html/>
- <http://www.rfc-base.org/>
- <https://datatracker.ietf.org/doc/>
- <https://www.rfc-editor.org/>



 **TIA and EIA** (Telecommunication Industries Association, Electronic Industries Alliance) are the US organizations that focus primarily on the physical level of communication among devices, and they also collaborate a lot with ANSI.

TIA deals with standards for various types of cables and connectors, antenna connections, mobile networks, ICT in healthcare, smart networks. EIA is, for example, behind the old well-known SCART connector, and in networks we also encounter the RS-232 connector. There are standards created by the cooperation of these companies, such as TIA/EIA 568 used for copper cables for Ethernet.




Additional information

- <http://www.tiaonline.org/>
- https://www.eia.gov/about/eia_standards.cfm



1.5 The ISO/OSI Reference Model

1.5.1 Overview

 The *Protocol Data Unit* (PDU) is a sequence of data provided with metadata (information about data) related to a particular protocol. Protocols usually receive data, process it as needed (make structure, split, encrypt, compress, translate, specify the recipient's address, etc.) and add a *header* with the corresponding information (data length), encryption algorithm used, sender and recipient address, etc.). Some protocols also add a *trailer* containing data such as a checksum. Data transferred inside a protocol data unit is also called the *payload*.

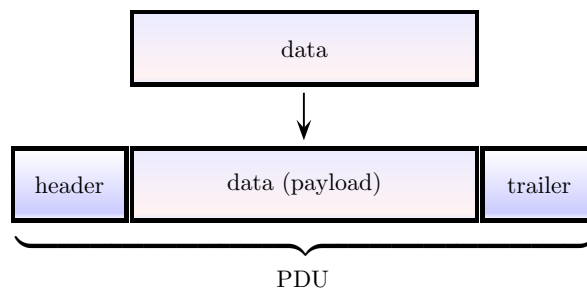



Figure 1.4: Protocol Data Unit (PDU)

 One of the most important standards in the field of computer networks is a group of standards describing the *OSI reference model* (Open Systems Interconnection) published by ISO, hence the name RM ISO/OSI. The original designation of the standard is ISO/IEC 7498-1, currently it is available as ITU-T X.200 on the web <http://www.itu.int/rec/T-REC-X.200-199407-1>.

OSI defines seven layers. Each layer performs a specific function in communication over a computer network and it is determined exactly what can happen on a given layer, so it is a *conceptual model* (ie it describes the design logic, the relationships between components).

The order of the layers and a brief description can be found in the figure 1.5. The layers shown in green (L1 to L3) depend on transmission media, the layer L4 (blue) is transitional, the layers shown in yellow (L5 to L7) are independent of transmission media.

The purpose of any conceptual design is to divide the whole into smaller parts that are easier to describe and to identify the relationships between these parts. In our case, these parts are layers and it is determined how the layers can be communicated between. In the previous sections we discussed that communication always follows certain protocols. The protocols in particular are organized into layers of the OSI model (usually a specific protocol belongs to one specific layer, depending on what its purpose is).

Next, we will discuss the individual layers. For each we need to know its purpose, the typical protocols that work on that layer, and the data units used.

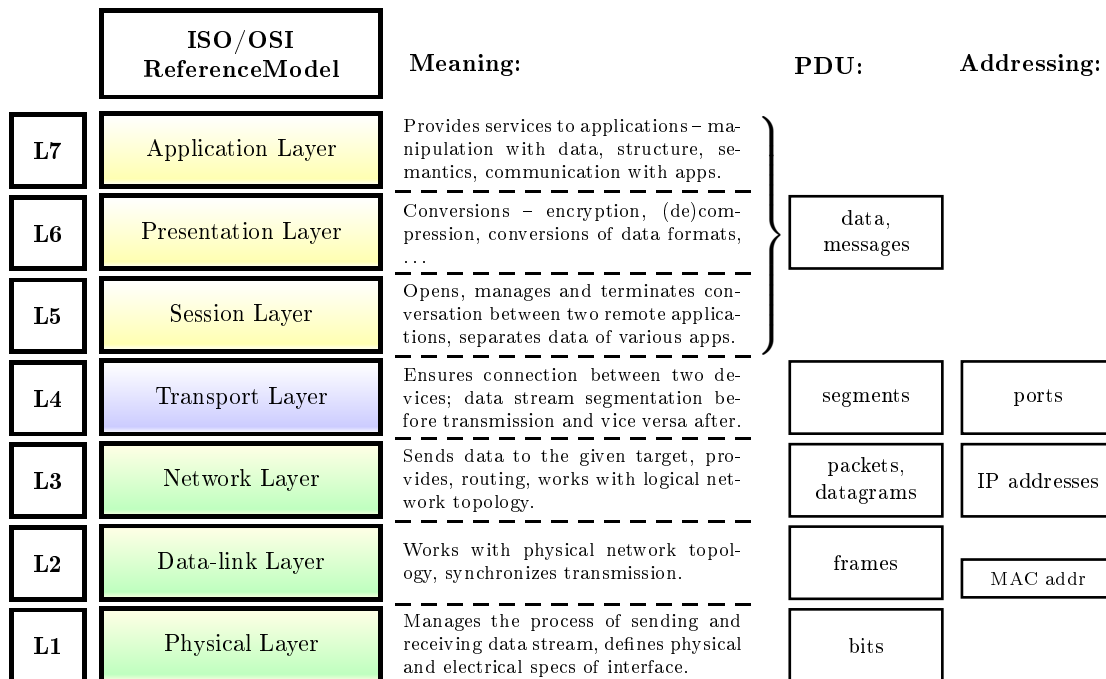




Figure 1.5: The ISO/OSI reference model

 **Physical Layer (L1).** The L1 layer is responsible for the actual physical transfer of data. It defines the transmission medium (cables, radio waves, etc.), how a bit with a value of 0 or 1 is represented (encoding), the network interface (ports, connectors), what exactly is used for which wire in the cable, and generally everything that is needed to convert the bit sequence into the form of the transmitted signal.

There are no data units on this layer, the layer only receives a stream of bits and converts them into the signal it sends, and vice versa.

The physical layer is implemented by all devices in the computer network, i.e. all devices that are equipped with a network interface.

There are also devices that have only a physical layer implemented: hubs (hubs) and repeaters (repeaters). In both cases, this is perfectly sufficient – the hub just receives a sequence of bits from one port and forwards it to all the other ports, the repeater takes the signal with the sequence of bits from one port, amplifies it and sends it to the other port, none of these devices needs “higher intelligence”. The advantage of these devices is simplicity and speed, the disadvantage is the inability to implement advanced features.

 **Data-link Layer (L2).** This layer determines the relationship between the incoming bit stream (as seen by the physical layer) and a particular node of the network. A device with the L2 layer implemented keeps track of the devices connected to the local network, keeping a table of the physical addresses of these devices (usually called a MAC table, CAM table, etc., depending on the specific protocol and the specific device). For each address in the table, we also have the port through which the device is reachable.


The link layer also provides functions that, although related to the transmission medium, require working with data units. For example, the baud rate is determined here, as it must also

be controlled according to the state in which the data units are received. If a data unit gets lost (which the physical layer wouldn't recognize), then obviously the whole mechanism "doesn't keep up" and the transfer rate needs to be reduced. The functions of the L2 layer therefore include detection of transmission errors.

The data units handled by protocols of this layer are usually called *frames*. In particular, each frame uniquely identifies the beginning and end of the data and contains (among other things) the physical address of the receiver and the physical address of the sender.

The L2 layer is implemented by those devices in the network that need to keep an overview of the nodes in the local network and work with the addresses of these nodes, it is not enough for them to know that "some bits" have arrived, so in fact almost all of them do (exceptions are e.g. hubs and repeaters).

Switches and bridges always implement the L1 and L2 layers, and there is always a protocol at the L2 layer that maintains a MAC address table. Of course, there are also switches that implement the L3 layer, but this is additional functionality.

 **Network Layer (L3).** While link-layer protocols have an overview of the physical topology of the network, network-layer protocols work with the logical topology of the network and "see" beyond the local network. The role of network protocols is to determine the actual path (or path segment for which the device in question is responsible) and to *route* if necessary.

At the L2 layer we are concerned only with a local network (L2 devices do not even know that any other network exists), whereas at the L3 layer we are not interested in local networks and are concerned only with the relationships (boundaries, paths) between networks. The L3 device is on the boundary of networks, not inside them.


The data units at the network layer are referred to as *packets* or *datagrams* (depending on the specific protocol, and also on the specific information source). What is the difference between them?

- *Datagram* is a data unit sent as part of a connection-less service, see page 5.
- *Packet* is a data unit sent within a connection-oriented service, but is often used in a more general sense (simply as a data unit), including connectionless services.

The network layer is implemented on end devices and then on those active network elements that provide routing, work with the logical network topology, connect networks, which are routers. Thus, the router implements the L1 layer, partly the L2 layer (only what is necessary to connect to the surrounding layers) and the L3 layer.

There is also (at least one) address table on the network layer, but in this case it is the *routing table*. The routing table contains information about where to send a data unit belonging to a particular (sub)network – the address of the destination (sub)network, the gateway to another network, or L3 neighbor (i.e., determining the next step in the path), the network interface through which the data is to come out of this device, etc.

So when the router receives data to forward, it finds the logical address of the destination in the L3 header and sequentially looks through each row of the routing table – stopping at a row such that the destination address belongs to the (sub)network specified on that row. In the row, it can read the direction to send the data.

 **Transport Layer (L4).** This layer is a kind of transition between the transmission process-oriented layers (L1–L3) and the application-oriented and thus transmission process-independent layers (L5–L7). So upwards it needs a binding to a specific higher layer protocol (we call this binding, a number that is analogous to an address, *port*) and downwards it applies functions related to data transfer.

 **Remark**

The term “port” is used in computer networks in two very different senses:

- (physical) port as part of a network interface, as explained on page 11,
- port (specified by a number) in the transport layer data unit header indicating which higher layer protocol is currently being communicated with.


For example, the port number 80 indicates communication with the HTTP protocol on the server side.



The data unit of the transport layer is called *segment*, and if it is a connection-less service, the term *datagram* can also be used. The main task of transport layer protocols (other than port number work) is to *segment data* from the higher layer into small enough fragments that can be transported over the network. The data is divided into fragments of a given length, and a header with transport layer data is added to each fragment to create a segment.

It is possible to use a connection-oriented or connectionless service. In the case of the connection-oriented service, the L4 layer ensures the establishment of the connection (handshake), manages the progress of the connection, acknowledges the delivered segments, in case of data loss or corruption, ensures the retransmission of the segment and, if necessary, ensures the regulation of the connection parameters, and finally terminates the connection.

The transport layer is usually implemented only on end devices.


 **Session Layer (L5).** This layer separates data belonging to different applications. Each application communicating with the network through this layer establishes a *session* with an application on the second end device, and within this session, data is transferred. Hence, a session is a logical connection between two applications established (usually) for the purpose of exchanging data that can flow over the network.

 **Remark**


What is the difference between a connection at the transport layer and a session at the session layer? While a connection is established between two devices, a session is established between two applications on those devices. As mentioned above – the lower layer provides services to the upper layer, so if the session layer wants to establish a session for an application with an application on another system, it needs a connection provided by the transport layer.



The session layer, like all higher layers, is implemented on end devices.

 **Presentation Layer (L6).** The L6 layer is responsible for performing data conversions (none of the lower layers interfere with the data), e.g. text encoding modifications (ASCII, EBCDIC, etc.), compression and decompression, encryption and decryption, some multimedia processing tasks.

Why is this layer called the presentation layer? Because it has the task of presenting data to higher layers in a form that these layers understand.

 **Application Layer (L7).** Protocols communicating with applications work on this layer. For example, a web browser application uses (among other things) the HTTP application protocol. The function of this layer is to receive data in a given format from the application and pass it to lower layers when sending data; conversely, when receiving data, the data is received from lower layers and passed to the appropriate application. Thus, it mediates the binding to applications and determines the format of the data sent.




Remark

Application protocols are on the application layer, not applications!




1.5.2 Cooperation of Protocols

We know that protocols are not supposed to be very complex and therefore need to work with other protocols. The cooperation can take place either within a single layer or between neighbouring layers, with the lower layer providing services to the upper layer.

 *Entity* is an element at a particular layer in the ISO/OSI model that has a defined interface – a set of services that can be used by an entity from the upper layer. By entity we can imagine a (running) instance of a particular protocol or set of protocols.

The way of communication within one system (a chain of entities from neighboring layers) is called *vertical communication* in ISO/OSI. *Horizontal communication* in ISO/OSI is communication between two identical layers located on different machines, at the logical level. The use of both types of communication is indicated in Figure 1.6.

 The protocol is thus used when sending data (according to Figure 1.6 on the machine on the left)

- receives data from a higher layer,
- processes or breaks it into smaller blocks if necessary,
- determines relevant meta-information, such as addresses, data size, information about how to handle the data along the way, etc.,
- adds a header with meta-information and, if necessary, a trailer to the data, thus “packing” the data into a PDU (packet, frame, datagram, etc.),
- it passes the PDU to the lower layer.

If the data is received (according to Figure 1.6 on the machine on the right), then the protocol

- receives PDUs from the lower layer,
- separates the header and trailer (if there is a trailer) from the PDU,
- parses the meta-information in the header and trailer, processes it in the specified way,
- if the data has been split into multiple blocks before sending and it is a layer where blocks are being completed, it collects all the blocks sequentially and completes them,
- passes the “unpacked” data to the higher layer.

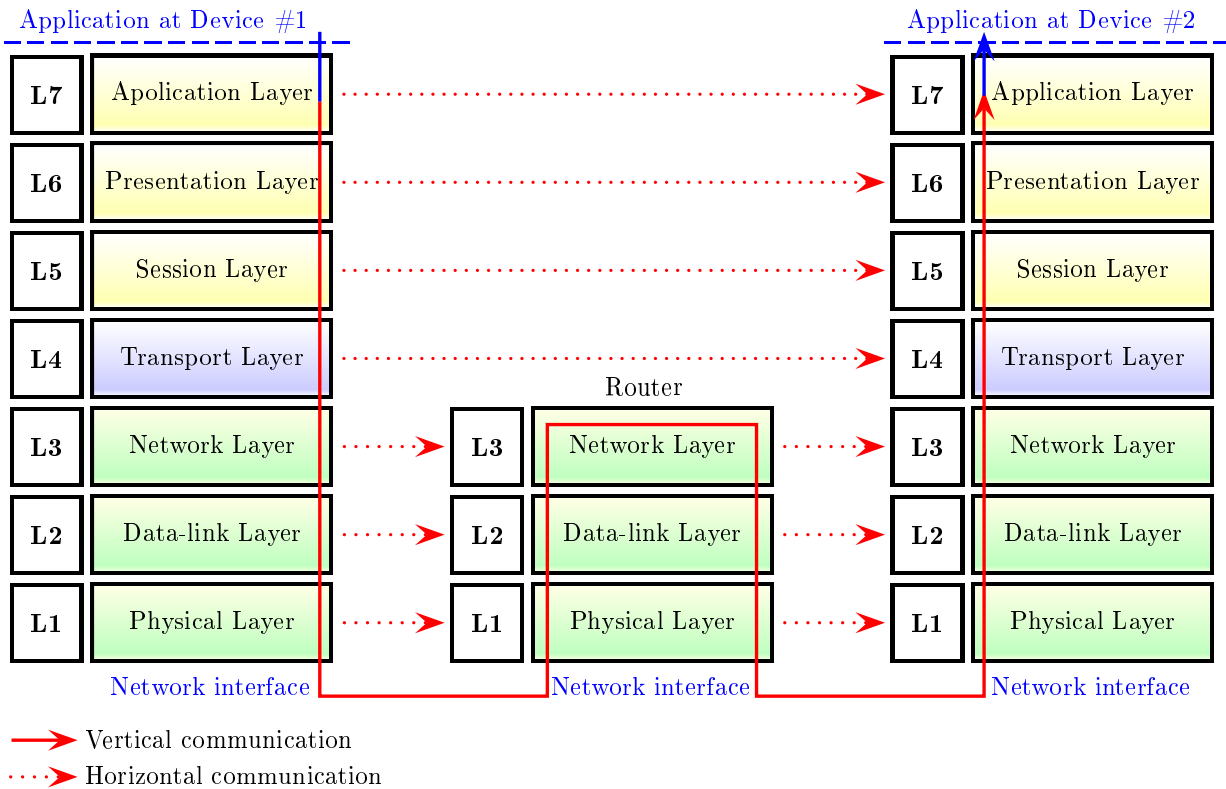


Figure 1.6: Horizontal and vertical communication in ISO/OSI

It is not necessary that all layers participate, some are not needed for a particular kind of communication.




 Thus, the *encapsulation* (packing) of data is performed when sending and *decapsulation* (unpacking) when receiving.

Figure 1.7 shows the encapsulation process. As each layer is traversed, a frame containing the headers of all previous layers involved is subsequently created.

 *Socket* is a combination of a network address (used at the L3 layer) and a port number (used at the L4 layer). If we write this pair “manually” (for example in the address bar of a web browser), we place a colon between the two items. A domain address can be used instead of a network (ie numeric) address.

 **Example**

The socket leading to `www.something.org` on port 8080 is represented by `www.something.org:8080`, i.e. we place a colon between the address and the port. For an IPv4 address, this will be similar, for example `193.84.214.5:8080`.

With an IPv6 address, this is more complicated because it contains colons itself, but the entry must be unique. Therefore, for example, when typing in the address field of a web browser, we place the IPv6 address in square brackets: `[2001:718:2201:208:5]:8080`.

Sockets can be found on layers L5–L7, they work as an interface between application protocols and the transport layer. For applications, they are available as Socket APIs (i.e. application

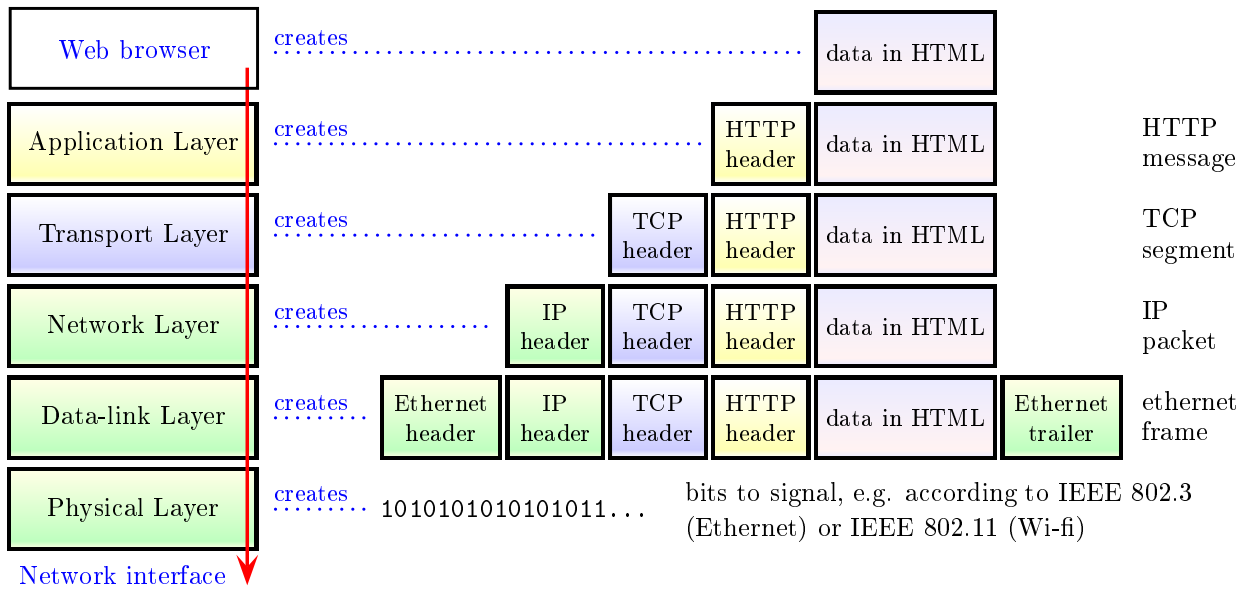




Figure 1.7: Encapsulation of PDU of the HTTP protocol


programming interfaces) in the form of libraries containing functions for working with sockets (creating a socket, accepting communication on the other side of the connection, listening, reading, writing to the socket). A special type of socket is *stream socket*, which guarantees the delivery of multiple blocks of sent data in the correct order, and thus only cooperates with reliable connection-oriented protocols (e.g. TCP) at the transport layer.

Sockets can be found on both UNIX(-like) systems (including Linux and MacOS X) and Windows (WinSock API).


 **Tasks**

Look at <https://www.binarytides.com/winsock-socket-programming-tutorial/>. This is a tutorial on programming sockets in Windows in the C/C++ language using WinSock. Find out which library is used to work with sockets, how the socket is created, how it connects to the server, how data is sent and received, and how the socket is closed. 


1.5.3 Protocol Stacks


 The *Protocol Suite* is a definition of a group of protocols that work together. The *Protocol Stack* is an implementation of a protocol suite. These two terms are often used interchangeably.


It does not necessarily have to be a protocol specification for all layers of the ISO/OSI model, only some layers can be specified, and cooperation with another (supplementary) protocol stack is assumed.

 The *TCP/IP Protocol Stack* (also called the *Internet Protocol Suite*) is a set of interoperable protocols that we need on devices connected to a large network (typically the Internet). In addition to the protocols contained directly in the name (TCP, IP), it includes others, most of them at the application layer. Protocols on layers L3–L7 are determined directly, for lower layers only the interface is specified.

This protocol stack is formalized as the *TCP/IP network model*, which will be discussed in more detail in the following text (including protocols).


 *IPX/SPX* is a competing TCP/IP protocol stack from Novell designed for the Novell Netware operating system – IPX runs on L3 instead of IP, SPX on the L4 layer instead of TCP. It was optimized for smaller networks, while TCP/IP is also designed for large networks. At present, we do not meet it.

 *LAN Protocol Sets* are Ethernet (IEEE 802.3), Wi-fi (IEEE 802.11) Token Ring (IEEE 802.5, no longer used), and others. Usually they implement only the L1 and L2 layer, above which the TCP/IP protocol stack is usually pushed.

 *WAN Protocol Sets* are ATM, Frame Relay, MPLS, and others. They also usually connect to TCP/IP in various ways.

For example, ATM slides under the network layer (L3), but inserts a special adaptation layer between it and its L2 implementation. Frame Relay implements the L2 layer, it assumes some suitable physical interface on L1, mostly according to EIA/TIA standards.

In contrast, MPLS is inserted between L2 and L3, i.e. the MPLS packet encapsulates a packet from the L3 layer (usually an IP packet) and is encapsulated in an L2 layer frame (for example an Ethernet frame). It can also run over ATM or Frame Relay, so technology from older devices can be used. It can also run on PPP and other protocols for first-mile networks.

 *Protocol sets for mobile networks* are, for example, sets for LTE, GPRS, CDMA, UMTS and others. They usually implement L1 and L2 layers and assume some specific protocols at higher layers, but it depends on the type of device (a terminal device will need a different set of protocols than a base station or other specialized devices in such mobile network) .

1.6 The TCP/IP Network Model

The ISO/OSI reference model is very complex and too theoretical. Gradually, several simplified variants were created. The most well-known of them is the TCP/IP network model, which is also called the DoD model (USA Department of Defense Model). Its components are also standardized by the IETF and available in RFC documents.

The TCP/IP network model is actually a formal description of the TCP/IP network stack, its connection to cooperating stacks, and the general possibility of involving other protocols. It consists of four layers, the relationship to the RM OSI layers is indicated in Figure 1.8.

The TCP/IP model has been designed with the following features:

- as decentralized as possible (no central management),
- as resistant as possible to various (even critical) operating conditions and as resistant as possible to transmission errors,
- terminal devices perform maximum of work (the core of the network has to be fast and flexible), network is managed in distributed way (every part manages itself),
- can connect networks with very different network architecture and technologies (to be as universal as possible).

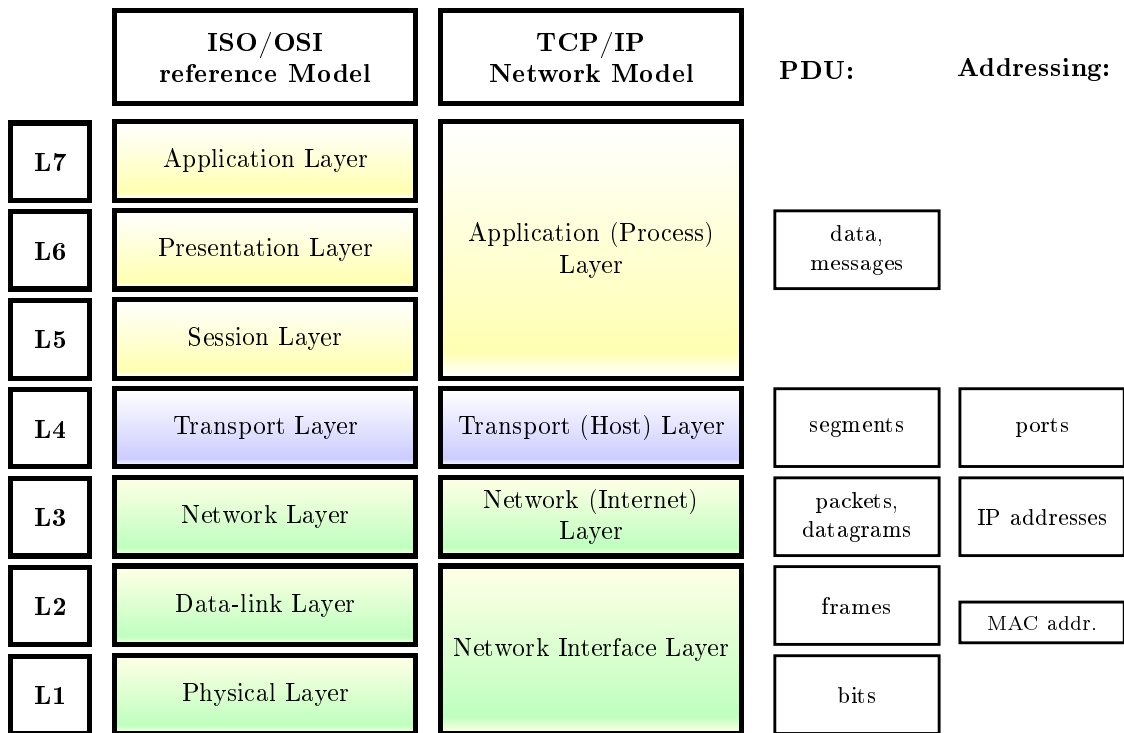


Figure 1.8: Comparison of RM ISO/OSI and TCP/IP

We will go through all the layers step by step and, unlike the reference model, we will focus mainly on the protocols working on these layers.

Network interface layer. This layer combines the functionality of the L1 and L2 layers of the reference model. No protocols are specified for it directly in TCP/IP, it is only determined how they should communicate with the network layer. It communicates with the hardware (network interface), or part of it can be hardware implemented.

Usually, protocol sets of local and wide area networks are connected to this layer (in other words – no protocols are listed directly in TCP/IP for this layer), for example

- IEEE 802.3 (Ethernet),
- IEEE 802.11 (Wi-fi),
- protocol sets of WAN networks or their parts, xDSL, mobile networks.

On the lower part of this layer (similar to L1) we can find simple devices working only with signals, of the hub or repeater type. In the higher part of this layer (similar to L2) there are more complex devices working with frames and connecting (or separating) individual network segments, i.e. switch and bridge.


If we pay attention only to the higher part of this layer, then the process of *frame switching* is performed here.

Network Layer. Also called “internet layer”, *internetworking* takes place at this layer, including *routing* between networks. The term “internet” (with a lowercase initial letter) generally refers to a network of networks, i.e. a network connecting not only individual nodes but smaller networks. A typical device in this layer is a router or L3 switch.


The main network layer protocols are two generations of IP (Internet Protocol – nowadays the both versions IPv4 and IPv6).

Routing protocols (OSPF, EIGRP, RIP, BGP and others) either work in this layer or there is some part of their functionality here.

Other network protocols include ICMP used for simple communication between devices implementing the network layer (this protocol is used, e.g., when the `ping` command is used to query a specific node in the network whether it is available).

 **Transport Layer.** We call it “host layer” too because it is implemented only on hosts on the network. A *host* is the term for the terminal device (computer, server, tablet, smart TV, etc.) that “hosts” data, applications, services; each host has its own name (hostname).

The most known transport protocols are TCP (for connection-oriented reliable service) and UDP (for connectionless unreliable best-effort service), but there are also other transport protocols for “special” purposes.

 **Application Layer.** Also “process layer”, because these protocols communicate with processes. It combines everything that is in RM ISO/OSI on the top three layers (L5–L7). At this layer we find a large number of protocols, most of which are communicated by applications that need to access the network. Examples of application protocols:

- The HTTP protocol is used, among other things, for communication between a web browser and a web server.
- The SMTP and IMAP protocols are used by mail clients or any applications capable of handling e-mails and also e-mail servers. SMTP allows to send e-mail, IMAP is used to access the mailbox.
- The DNS protocol helps with address translation (when a name address is entered in the address bar of a web browser, e.g. `www.google.com`, DNS translates it into a numeric IP address that is “understandable” to the active network elements).
- DHCP assigns us a dynamic IP address when we log into the network.



Remark

The terminology from ISO/OSI is usually used (for example, designation of layers, entities, etc.), but the overall layout of network-related activities and the implementation of procedures are usually according to TCP/IP.



Definition (Internetworking)

Internetworking is a mechanism for interconnecting networks i.e. providing communication among (inter) networks. This mechanism takes place at the L3 layer, on devices such as a router or switch with L3 layer functionality.





1.7 Transmission Characteristics


1.7.1 Transmission Signal


The *transmission medium* is used for data transmission. This can be a metallic cable (usually copper), an optical cable, air (radio link) or something else. Whatever transmission medium is used, the data is transmitted over it as *sequence of signals*.


The signal is transmitted through the transmission medium in the form of *wave* (oscillation of particles – electrons, photons, etc.). The signal can be generated, or the transmitted information is *modulated* into a *carrier signal* (i.e. the existing signal is modified). Figure 1.9 indicates some physical characteristics of the signal.

 *Amplitude* is the maximum deflection of the signal (in V or A).

 *Period* is the amount of time it takes an oscillating particle to go through a complete oscillation cycle.

 *Phase* is an immediate property of an oscillating particle determining the angle of its motion relative to the base, it determines the position of the particle in the current period. Any angle unit is used, such as the radian or degree. Because a particle usually oscillates along a sinusoid-like path, the phase changes during a period so that the particle is in the same phase at the beginning, middle, and end of the period.

 *Wavelength* is the distance the signal travels during the signal period, i.e. the distance between the two closest points in the same phase on the same side of the base. The wavelength will be of particular interest to us in fibre optic cables.

 *Frequency* is the inverse of the wavelength multiplied by the signal speed. The larger the wavelength, the smaller the frequency, and vice versa.

Frequency therefore takes into account the time, i.e. the time it takes the particle to move over a given distance. Wavelength takes into account only the distance itself.

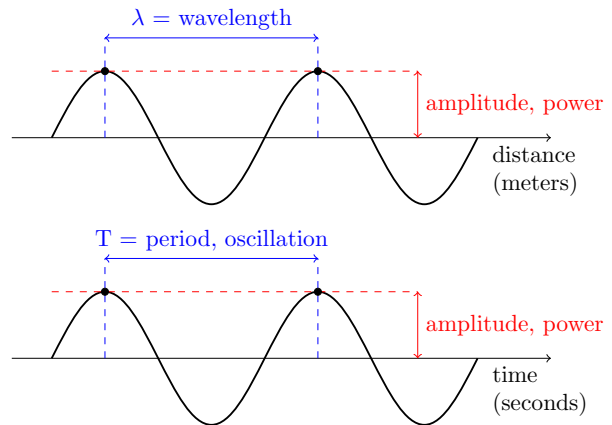


Figure 1.9: Physical characteristics of signal

 **Remark**

Let's remind ourselves that the unit of frequency is 1 Hz, and it is defined as the number of cyclic events occurring in 1 second, so it corresponds to 1 s^{-1} according to the SI system.

Figure 1.10 indicates the relationship between wavelength and frequency. The formula is

$$f = v \cdot \frac{1}{\lambda} \quad \text{resp.} \quad \lambda = v \cdot \frac{1}{f}$$

where f is frequency, v is the signal speed, and λ is wavelength.

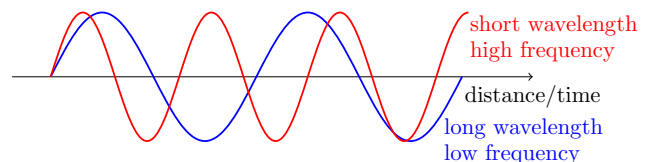


Figure 1.10: Relationship between wavelength and frequency



**Example**

Let's take an example about an audio signal. The speed of sound depends on altitude (i.e. air density) and temperature, so let's assume that in our case it is 330 mps. A sound with a frequency of 1000 Hz propagating at this speed will have a wavelength of $\lambda = 330 \cdot \frac{1}{1000} = 0.33\text{m} = 33\text{cm}$.

In an optical fibre designed for long-distance transmission (units of up to tens of kilometres), a 1310 nm optical signal is emitted by a laser. This is light, so the value of v is very large (almost 300 000 mps), whereas the wavelength of such a signal is very small ($1310\text{nm} = 1310 \cdot 10^{-9}\text{m}$). The formula implies that the frequency will be very high (over 200 THz).

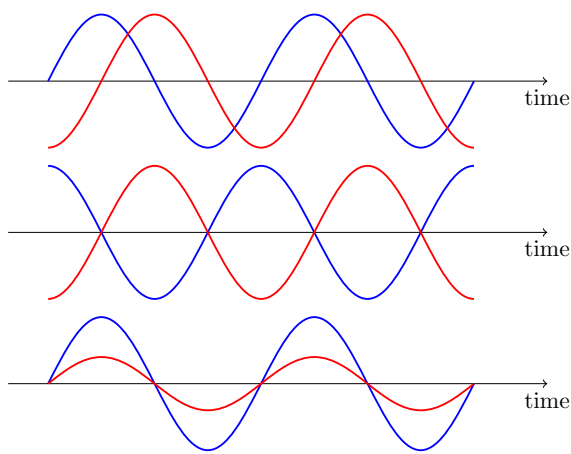


Figure 1.11: Phase shift examples

Let's go back to the phase shift. For simplicity, we can think of it as simply "shifting the wave to the right", more precisely it is a value in angular units (degree, radian). We always shift by less than a period (a period has a phase shift of 360° or 2π , so up to this value).

In Figure 1.11, the first picture shows the shift of size a quarter period ($\pi/2$), in the second case the shift is π . In the third picture, there are two waves with the same frequency but different amplitude: the phase changes at a different rate, but for the period as a whole it is equal (the difference is 0), so the given waves are "in phase".



Bandwidth is the difference between the lowest and highest frequency of the transmission signal, or the width of the interval of frequencies that can be transmitted over the transmission path. For example, for a Category 5e copper Ethernet cable (we'll learn about this later), the bandwidth is 100 MHz, and for Category 6 it is 200 MHz.

In general, we can say that the larger the bandwidth, the higher the transmission speed can be achieved (but it also depends on other parameters).

1.7.2 Baseband and Broadband Transmission


Transmission in a computer network is carried out either in the base band or in the broad band.



Baseband. The transmission takes place by directly encoding the sequence of ones and zeros to be transmitted into the frequency spectrum (emitting, i.e. directly creating a signal) and the resulting signal is transmitted through the communication channel. Usually, low frequencies are sufficient. Baseband is used e.g. for metallic local area networks (Ethernet on copper cable), but also for optical networks.

The disadvantages are limited range (smaller transmission distance) and problematic synchronization (if we encode the sequence of zeros and ones as is, then long sequences of zeros or long sequences of ones would be poorly decodable, it would be impossible to determine their length). The sending and receiving sides need to have properly aligned timers to synchronize the intervals between sections representing individual bits; but for long sequences of bits with the same value, this is not sufficient.

This problem is being avoided simply by changing the sequence of bits before encoding to a signal according to a certain key so that the sequence does not contain long sequences of the same digits, and of course so that the data can be returned to the original state after the transmission is completed. Each bit is simply replaced by a sequence of bits so that the values 0 and 1 are “alternated” often enough.

 *Manchester encoding* is very simple. There are several variants, one of them operates as follows: the bits are encoded in the given direction of the signal’s oscillation – 0 is encoded as a top-down transition, 1 is encoded as a bottom-up transition (see Figure 1.12). The final signal has a rectangular waveform in baseband.

The result can be represented by a waveform or again as a sequence of ones and zeros, with the top-to-bottom transition written as 10 and the opposite transition as 01.

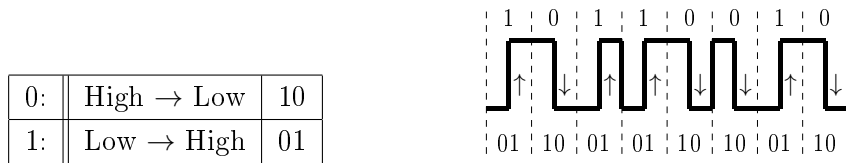


Figure 1.12: Manchester code of the octet $(10110010)_2$

The bit sequence 10111000 is encoded at Manchester encoding to 01 10 01 01 01 10 10 10.


As we can see, we only get the same symbols next to each other at places where the neighboring bits in the original sequence were different, resulting in at most two identical symbols following each other. This is the advantage of Manchester encoding, the disadvantage is the increase in the length of the data sent (the length is doubled compared to the original). At higher speeds there are other techniques such as data scrambling or self-correcting code (Gigabit Ethernet).

There are two basic variants of Manchester encoding:

- of the original G.E. Thomas, where the transitions are reversed from the above picture (1: transition down, 0: transition up),
- IEEE 802.3 for 10b Ethernet with transitions exactly as shown in the above picture.

The second variant is therefore used in 10Base-T, and also in one of the connection types of NFC.

We can also encounter a variant called differential Manchester, where the transitions are always only when the bits change (01 or 10). This variant is used e.g. in hard disks.

 *4B/5B encoding*: each quadruplet of bits (nibble) is replaced by five bits so that there are as many 1s as possible in the whole sequence (at least two per five), with no more than one zero at the beginning of the five and no more than two zeros at the end. The codes for nibbles are in the Table 1.3.

The sequence 10111000, which we have encoded into 16 characters in Manchester encoding, is processed here into 1011110010 of 10 characters. The resulting code is 25 percent longer than the original, which is much better than twice the length in Manchester.

4B/5B was originally designed for FDDI, in Ethernet we can meet it e.g. in 100Base-X (100Mb Ethernet over optics).


8B/6T encoding means that 8 bits of the original sequence are encoded with 6 signal changes (ternary symbols, i.e. states -,0,+). It was used on a twisted pair category 3 line with 4 pairs, the

4B	5B	4B	5B	4B	5B	4B	5B
0000	11110	0100	01010	1000	10010	1100	11010
0001	01001	0101	01011	1001	10011	1101	11011
0010	10100	0110	01110	1010	10110	1110	11100
0011	10101	0111	01111	1011	10111	1111	11101

Table 1.3: Codes for nibbles in the 4B/5B encoding

data is split into 3 pairs, always transmitted in one direction. The fourth pair was used to indicate collisions.

Other “slash” variants also consist in encoding a certain number of bits from the source into a certain number of bits/states in the target signal. For example, *8B/10B encoding* maps 8-bit sequences to 10-bit sequences, it is used e.g. in 1000Base-X (gigabit Ethernet over optics).

 *MLT-3 encoding* uses three voltage levels (previous encoding used usually two levels), namely -, base, +. The voltage change takes place only on signal 1, and on the “sinusoid”, but with a rectangular waveform.

This encoding is usually combined with some other encoding, for example the signal for MLT-3 can be preprocessed with 4B/5B encoding.

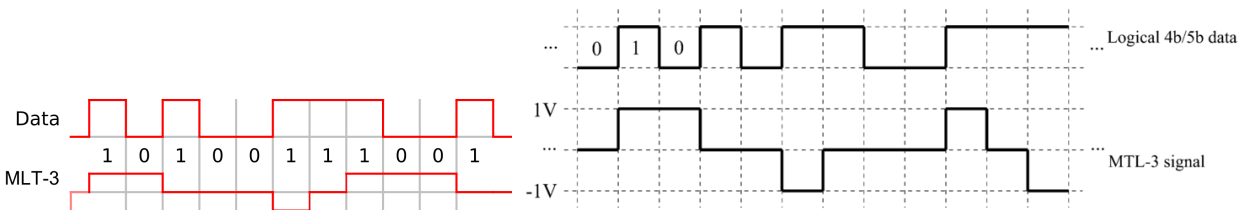



Figure 1.13: MLT-3 encoding (left: only MLT-3, right: combination with 4B/5B)²

Figure 1.13 on the left shows MLT-3 encoding used alone, on the right in combination with 4B/5B.


Compared to Manchester, the signal has only a quarter of the frequency, so it radiates less, generating much less interference to the surroundings.

For example, 100Base-TX (100Mb Ethernet over copper twisted pair cable) combines 4B/5B, NRZI (see below) and MLT-3.


 *PAM-5 encoding* uses four voltage levels for data and a the fifth one for the control bit. Data is encoded as follows:

- 00 → -1.0 V
- 01 → -0.5 V
- 10 → +0.5 V
- 11 → +1.0 V


PAM-5 is used for example in 1000Base-T (gigabit Ethernet over copper twisted pair cable).


 *NRZ encoding* (Non Return to Zero) uses two levels to represent signal, neither of which is 0 (base), for example +1,-1. There are several variations – unipolar (1 is represented by a positive voltage, 0 also positive voltage but smaller), bipolar (0 positive, 1 negative), NRZI, etc.

²Based on https://en.wikipedia.org/wiki/Line_code

 **NRZI** (*Non Return to Zero – Inverted*) means that 1 causes a signal change, 0 does not. It's similar to MLT-3, but only on two levels (positive and negative).

 **Remark**

If the 100Base-T, 1000Base-X, etc. abbreviations do not mean anything to you from the previous study, look up these abbreviations in the following chapter on Ethernet. 

 **Broadband.** This transfer is done by encoding a sequence of ones and zeros into a previously created signal (changing the existing signal). We call this translation *modulation*.

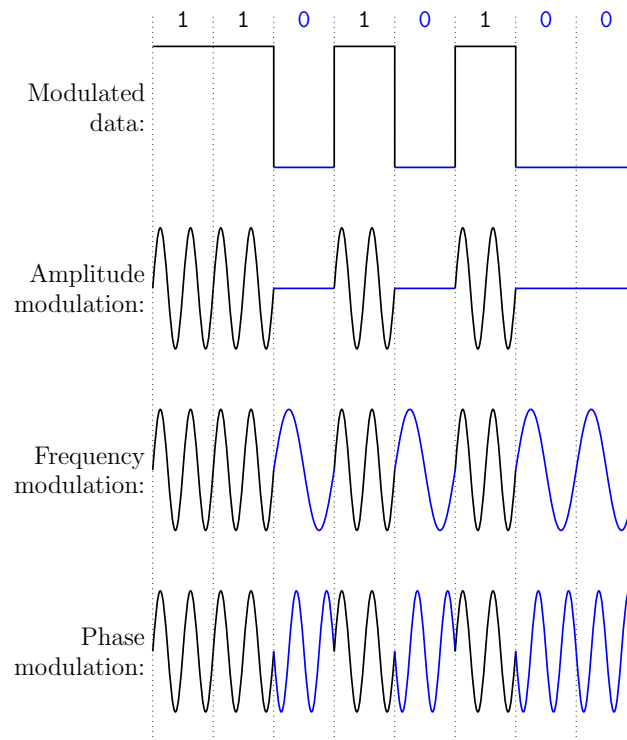



Figure 1.14: Example of amplitude, frequency and phase modulation of digital data


Modulation proceeds as follows:

- Determine the *carrier signal*, which is a suitable signal at the frequency at which the data is to be transmitted, a signal with a harmonic frequency is chosen, usually a sinusoid.
- Modify the parameters (amplitude, frequency, or phase) of this signal according to what data is to be transmitted – modulate the data to the signal.
- Merge and send as necessary.

Figure 1.14 on page 33 shows examples of three basic types of modulation – amplitude, frequency and phase. The horizontal axis is the variable t (time), the vertical is the result $s(t)$.

In reality, more than one of these three parameters often changes, and various modulations can change the same parameter with different intensities.

 **QAM** (*Quadrature Amplitude Modulation*) uses a combination of phase (PSK) and amplitude (ASK) modulation. It can modulate both analog and digital signals into the appropriate frequency range, resulting in an analog signal.

 There are several variants: 16-QAM, 64-QAM, 256-QAM. In each variant, amplitude modulation is applied to a certain number of carriers, while phase modulation is applied to the remaining carriers; the combination yields a certain number of states representing the modulated data or signal. For example, in the simplest 16-QAM variant, two subcarriers are used, and there are 36 combinations, but only 16 of them are used (the ones that are most easily distinguishable from each other).


QAM modulation in combination with OFDM multiplexing (see below) is nowadays used mainly in wireless and mobile networks, e.g. Wi-fi according to the IEEE 802.11n standard uses up to 64-QAM, whereas IEEE 802.11ac uses up to 256-QAM (in case of bad signal it “falls” to a lower variant). In fourth-generation mobile networks (LTE Advanced), 256-QAM is used.

Baseband	Broadband
signal is emitted	data is modulated into existing signal
the result is a digital signal with a rectangular waveform	the result is an analog signal with a sinusoidal waveform
shorter distance	longer distance
signal usually uses full bandwidth	signal is usually limited to a specified bandwidth

Table 1.4: Baseband vs. broadband


1.7.3 Multiplexing

Consider a single transmission channel with certain physical characteristics (frequency, etc.). Under normal circumstances, we could transmit only one signal through this transmission channel.

 *Multiplexing* is a technique that allows a transmission channel with sufficient bandwidth to be divided into multiple logical subchannels, seemingly separate, and to transmit different blocks of data in each. In real life, this means that multiple signals are combined into a single signal. Multiplex actually means “multiple access” (simultaneous access of multiple users to a transmission channel).

The component that performs multiplexing is called a multiplexer (multiplexor, MUX). The opposite operation to multiplexing is demultiplexing (DEMUX, DMX): on the sending side of the transmission channel, multiplexing (splitting into subchannels) is performed, on the receiving side, demultiplexing (removal from subchannels) is provided.

There are several common types of multiplexing:

 **Frequency Division Multiplexing** (FDM): each transmitted signal is allocated a portion of bandwidth (there must be spacing between the allocated bands to avoid interference), and this signal is mapped to this allocated portion. Each sub-channel is therefore allocated its own frequency interval.

FDM is only applicable to analogue signals, and another disadvantage is that it is more suitable for a “stable number” of users (or there is a cap on the number of users).

Frequency multiplexing is known e.g. from analogue radio, where different stations are assigned

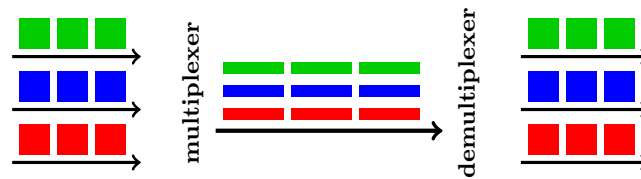


Figure 1.15: Frequency multiplexing

different frequencies, as well as in satellite transmissions and cable networks. It was also used on the analogue telephone network – each “telephonist” had a dedicated subchannel of 3.1 kHz.

Wavelength Division Multiplexing (WDM) is similar to frequency multiplexing, but for optical signal, where instead of frequencies we use division by wavelengths or colors of light (which we know is actually similar). Each signal gets assigned a range of wavelengths that are used in its transmission. As we know, wavelengths are easier to work with than frequencies for light, they are smaller numbers.

WDM is mostly used in optical cable transmission, i.e. mainly in optical WANs and also in FTTx technology (the optical fibre is routed as close to the customer as possible). The signal generated by laser or LED is modulated after multiplexing.

Time Division Multiplexing (TDM): the transmission channel is alternately allocated to different specific transmissions (users). The principle is outlined in Figure 1.16. The transmission channel is scaled into so-called *frames* (note that these are not the same frames as on the L2 layer, just a name match), in each frame there is one *slot* (socket) reserved for each sender, in which a packet can be placed (or left empty).

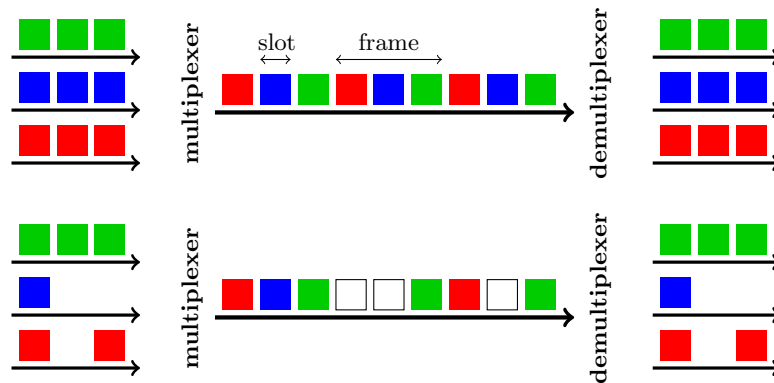


Figure 1.16: Time Division Multiplexing (top full load, bottom partial load)

The problem with “pure” TDM is that it is also suitable for a relatively constant number of users, moreover more or less communicating similarly (in terms of the number of packets sent). If a user sends significantly fewer packets, his or her slots are unused, as we can see in Figure 1.16 below. Another disadvantage of time multiplexing is the need for constant frame synchronization – both communicating parties must be clear when a frame starts and ends, and when which slot starts/ends and to whom it belongs. However, unlike FDM, it is also suitable for digital signal.

TDM is used e.g. in mobile networks when using basic GSM transmission.

Statistical multiplexing is similar to the time multiplex in that the transmission channel is divided into slots. However, in a statistical one, slots are not fixed to specific subchannels, but

are allocated as needed – a busier subchannel gets more slots than a less busy one. To make it clear which subchannel belongs to which slot, subchannel information (header) must be added to the transmitted data blocks.

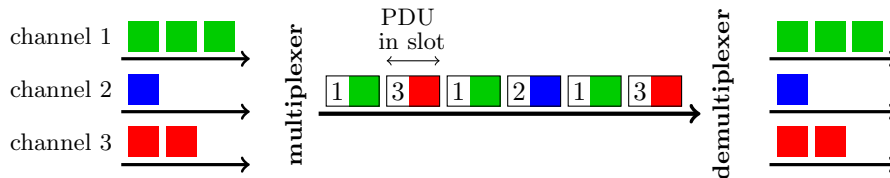


Figure 1.17: Statistical multiplexing

The advantage of statistical multiplexing is a slightly lower need for synchronization (no need to deal with slot frames), only the slots themselves are synchronized and the overall communication remains asynchronous. Another advantage is better utilization of the transmission channel. The disadvantage is the need to header the data and the associated overhead.

Statistical multiplexing is used in some WANs (data units are stacked into slots). It is important that the header added be as small as possible, but also be able to assist in routing over the circuit created over the WAN.

Code Division Multiplexing (CDM, also CDMA, Code Division Multiple Access, spread spectrum) is a digital method, whose creation was motivated primarily by the need for security (to make eavesdropping as difficult as possible). Each partial transmission is encoded (the encoding takes place at the end devices, not by the multiplexer), all parallel (encoded) transmissions are then merged by the multiplexer and transmitted over a shared channel to the end devices. The destination end device then decodes only what actually belongs to it with the help of a special code: without this code the content of the sub-channel cannot be accessed.

The problem with CDM is the need for complex synchronization and an upper limit on the number of subchannels (too many subchannels would interfere with each other and be impossible to decode). It is also necessary to ensure secure code negotiation for decoding. There are several different variants of CDM, mostly used in third generation and newer mobile networks.

Orthogonal Frequency-Division Multiplexing (OFDM) maps subchannels to different frequencies similar to FDM, but much more efficiently. Whereas FDM uses a single carrier for all subchannels, OFDM uses different subcarriers for each subchannel, and the individual subcarriers are orthogonal to each other, so they can overlap and still be separated at the receiver side. The data transmitted over the subchannel is modulated to the allocated subcarrier by some suitable modulation, usually some variant of QAM modulation is used.


OFDM (and its variants) is widely used today for modulation of digital data into analogue signals in computer networks (Wi-fi, WiMAX, LTE, xDSL, etc.), but also, for example, for digital TV transmission.

There are different variants of OFDM adapted to a specific application in a particular technology. For example, in mobile networks we often encounter OFDMA (OFDM Access), where subchannels of a single channel are spread over the entire spectrum and can be allocated to different clients.


Chapter 2


Local Networks – Ethernet

The term local area network (LAN) will be further understood as a collection of interconnected devices (host/end devices, active network equipment, etc.) that belong to the same network, usually within a single building or apartment, with a typical area of units up to hundreds of meters, occasionally more.

 *Quick preview:* Currently, the most common LAN technologies are Ethernet (as a LAN on copper or fiber optic cables) and Wi-fi (wireless). In this chapter we will focus on Ethernet, and Wi-fi will be discussed in a separate chapter.


We will deal with Ethernet at the L2 (addresses, frame format, etc.) and L1 (cables, etc.) layers. At the end of the chapter we will discuss Ethernet related technologies.

 *Keywords:* Ethernet, IEEE 802.3, DTE, DCE, Hub, Switch, access (collision) method, CSMA/CD, Backoff algorithm, MAC address, physical address, EUI-48, OUI, LLC, EtherType, SNAP, Ethernet II frame, collision domain, broadcast domain, UTP, STP, cable category, coaxial cable, twinax, fibre-optic cable, crimping, half duplex, full duplex, collision window, burst mode, autonegociation, PoE, structured cabling, patch panel, backbone

 *Objectives:* After studying this chapter, you will be able to work with UTP cables, understand the general operation of L1 and L2 layer networks, and specifically how data is transferred using Ethernet.

2.1 The Meaning of Ethernet

The original Ethernet specification was developed in cooperation between Xerox, Digital and Intel, published in 1976 and referred to as DIX Ethernet (based on the initial letters of the cooperating companies).

 Later, Ethernet was standardized as IEEE 802.3, but the name “Ethernet” does not appear in this standard at all and is incompatible with the original DIX Ethernet. Gradually, various variants – for different transmission media (metallic cables of various categories, fiber optic cables) and also increasing speeds – appeared.

**Definition (Ethernet, IEEE 802.3)**

Ethernet, or IEEE 802.3, is a standard describing a set of technologies for a local area network using cables (metallic or optical) where host devices share and compete for the same communications path.

The IEEE 802.3 standard describes the implementation for the physical and data-link layer (i.e., the entire network interface layer under TCP/IP), including the method of communication and collision handling.



2.2 Communication in Ethernet

2.2.1 Device Types

In the previous chapter, we used the terms host device and active network element. In addition, we will also encounter the terms DTE and DCE (not only for Ethernet).

**Definition (DTE, DCE)**

A *DTE* (Data Terminal Equipment) is a device on a network that is either a receiver or a sender of data (i.e. does not forward them). In Ethernet and other local area networks, these are usually terminal devices – computers, servers, routers, etc.

A *DCE* (Data Circuit-terminating Equipment) is a device on a network that forwards traffic (it is not the actual source or destination of the data), i.e. it forwards what it receives on one port to one or several other ports. In Ethernet, these are active network devices, usually switches.



Note that in the definition it is emphasized “In Ethernet...”. This is because in other types of networks, the terms DTE and DCE are understood as other specific types of devices. For example, in WANs (which are large networks typically connecting multiple smaller networks), DTEs are actually WAN edge devices, i.e. devices that “understand” the relevant WAN protocol, but their role is to mediate communication with the connected smaller network (from the WAN perspective, they perform a similar role to a computer or router in a local network, being “at the end of the path” through that particular network).

2.2.2 Interconnection of Network Nodes



Back in the 1990s, coaxial cable was used as a transmission medium (we will discuss transmission media later in the chapter) and the typical physical and logical topology was the *bus*, or it was possible to connect multiple buses by bridges or (later) switches. This means:

- All devices in the network are equal, none has priority.
- Devices connected to the same bus share the transmission medium, they are on the same network segment.
- If two buses are connected by a bridge, then only what is addressed to the other bus goes from one bus to the other across the bridge, otherwise this transmission will not pass through the bridge. The exception is multicast and broadcast traffic.

- Properly, only one device should transmit at a time. If this is not fulfilled, mutual interference (collision) will occur and both transmitted signals are damaged.


 Gradually, however, a big change took place – twisted-pair cabling (like telephone wiring) and fiber-optic cable began to be used as cabling, and the physical topology became a star or a tree. The logical topology remains the bus. Hubs and slow routers were originally used as active network equipment, but gradually moved almost exclusively to switches. Whereas a hub passes through anything that reaches it (i.e. it is only part of a network segment, it does not separate segments), a bridge and a switch send the signal only to where it is addressed, and thus separate segments.

Figure 2.1 shows the difference between a hub and a switch. As we can see, the hub does indeed send to all ports (except the one the frame came from), it does not use the destination address information (in fact it does not even get to it). In contrast, the switch checks the destination address, finds out which port the target is connected to, and sends the frame to that one port only.

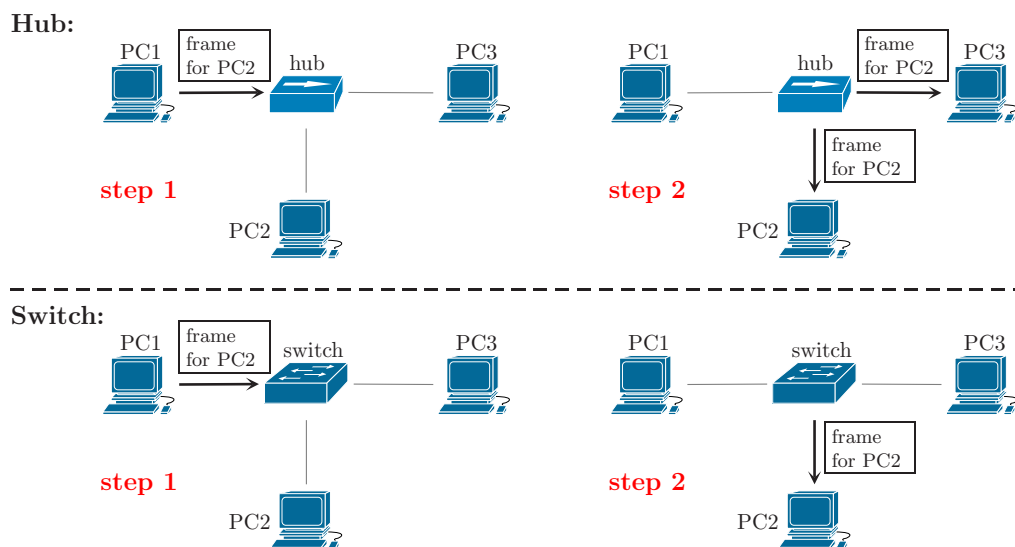



Figure 2.1: Comparison of hub and switch communication

2.2.3 Collision Method

The access (collision) method determines how it is decided whether or not a device can start transmitting and how to handle collisions. For Ethernet, the CSMA/CD access method is specified.

 **Definition (CSMA/CD)**

Ethernet uses the technology of multiple access to the transmission medium with carrier listening and collision detection – CSMA/CD:

- CS (Carrier Sense) – all nodes in the network constantly listen on the carrier to find out if another node transmits,
- MA (Multiple Access) – any node can transmit on the shared media if it finds out by listening that no other node transmits,
- CD (Collision Detection) – if a node does not catch to detect the transmission of another node before its own transmission (e.g. when the both nodes start transmitting at approximately

the same time), it has to be able to detect the resulting signal collision and respond to it accordingly.



The use of the collision method is important when communicating in half-duplex (or when we use hubs, but that is quite history).

What happens when a collision occurs:

- The transmitting node either wants to transmit and has detected that another node is transmitting, or it has detected a collision (it has detected that someone other is transmitting).
- If it has already transmitted, it stops transmitting (not immediately, it must allow the other transmitting node to detect the collision). It sends a *Jam signal*, which is a signal indicating a collision on the media.
- According to a special algorithm (Backoff algorithm), it determines the waiting time and after this time, it tries to transmit again.
- If the next attempt fails (either it detects that the link is not free before the transmission, or it detects a collision again), it returns to the previous point (waiting time and a new attempt).

The *Backoff algorithm* determines how long a node should wait to transmit when it detects that it is transmitting another node. The purpose is to ensure that in the case of multiple senders, each of them waits for a different time (determined by a randomly generated number), thus reducing the likelihood of another collision.



Remark

Note that while the physical topology is reflected at the L1 (physical) layer, the logical topology is reflected at the L2 (data-link) layer. The original Ethernet using coaxial cable was bus-oriented at both the physical and logical levels (physical due to the connection of devices to a shared transmission medium, logical due to the CSMA/CD access method). The newer ones are bus-oriented only at the logical level, and only when CSMA/CD is used (when communicating in half-duplex, both sides are on the same segment, i.e. sharing the medium).





2.3 Ethernet at the Data-link Layer

As a reminder – at the L2 (data-link) layer, frames are used as PDUs. When sending, a frame is created at this layer (a header is added to the beginning and a trailer to the end) and passed to the L1 layer; conversely, when receiving, the bit sequence received from L1 is split into header, data, and trailer, the header and trailer are analyzed, and the data is passed to the L3 layer.

2.3.1 L2 Addresses

MAC addresses (physical, hardware addresses) are used at the data-link layer. Each frame contains the MAC address of the destination device (the receiver) and the MAC address of the source device (the sender). This implies that the sender must know its own address and also the address of the destination (both of which it usually knows or obtains using some other protocol).

 In other words – a device operating at the L2 layer should have an overview of the *physical network topology*, i.e. the addresses of the devices it can communicate with, and which network interface (which path) the device is reachable through. Each device keeps this information in its MAC address table (it may be called a MAC table, CAM table, etc., depending on what the vendor calls it).

 *Hardware (MAC, physical, EUI-48) address* is actually the low-level address of the network interface. Every network interface has this address, even when it is not currently connected to the network. A network interface has its MAC address already assigned by its vendor, and this address should properly be globally unique.

But many rules have an exception, and this rule does too.

So we actually have two kinds of MAC addresses:

- the address assigned by the vendor (BIA – Burned-In-Address), which is stored in the ROM or flash memory of the network interface,
- local (temporary) MAC address that we have set ourselves.

In current networks, 48-bit MAC addresses (i.e. 6 octets) are used and are usually written in hexadecimal digits – the MAC address is written using 12 hexadecimal digits. Uniqueness is ensured as follows:

- The first half of the address is assigned to the vendor by the IEEE (large vendors have several, smaller vendors only need one), so the first half is vendor specific and no two vendors have the same. This number is called the *OUI* (Organizationally Unique Identifier).
- The second half is determined by the vendor itself, who ensures that these numbers are unique within the first half assigned to him.

In Table 2.1, the above described structure (two parts) is more clearly indicated.

24 bits	+	24 bits	=	48 bits
OUI = vendor ID	+	Interface ID	=	Entire address
Assigned by IEEE	+	Assigned by vendor	=	globally unique

Table 2.1: MAC address structure

The physical address is usually written by separating pairs of hexadecimal digits by a colon or a dash, or after two octets by a period, or without separators. For example:


50:E5:49:A2:80:61

50-E5-49-A2-80-61

50E5.49A2.8061

50E549A28061

MAC address change is not that common, but sometimes it is useful – e.g. when we need a MAC address in a specific form for the purpose of using a certain technology or application and there is no time or possibility to change the configuration, or in a virtualized environment (although unique addresses can be used there as well). Hackers use modified MAC addresses when attempting to penetrate a network protected by a blacklist or whitelist.

 Any locally valid address (i.e. not a BIA from vendor) should correctly have the *L/G bit* set. This bit is the second from the right in the first octet of the address (in common network implementations, such as Ethernet), as shown in the figure 2.2.

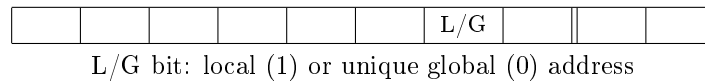




Figure 2.2: Location of the L/G (local/global) bit in MAC address

 If all the hexadecimal digits are set to *F*, it means that this address is not only locally valid (not globally unique), but also *broadcast*, i.e. it does not identify one specific device. It looks like this: **FF-FF-FF-FF-FF-FF-FF-FF**. Broadcast MAC address is used when a frame is sent to all devices on the local network.

 If the address is to identify a group of devices (but not necessarily all), it is called a *multicast* address. In Ethernet and some other networks that use MAC addresses, we recognize a multicast address by the rightmost bit in the first octet (the least significant bit of the first octet). For a multicast address, it is set to 1, which means that the first octet will be an odd number.

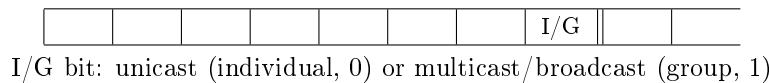



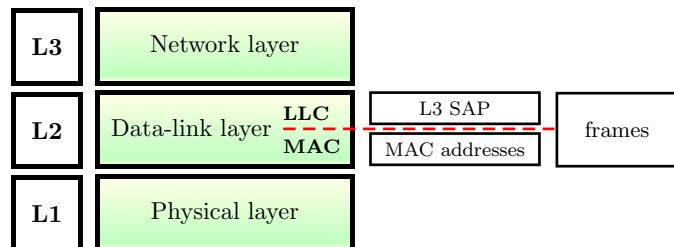
Figure 2.3: Location of the I/G (individual/group) bit in MAC address

2.3.2 L2 Sublayers

 The so-called *data-link protocols* (L2 layer protocols) work on the L2 (data-link) layer. Such a protocol either determines the operation of the entire L2 itself, or two protocols are used that can do the same thing together. The L2 layer is actually divided into two sublayers:

- LLC (Logical Link Control) – upper sublayer, provides cooperation with the L3 (network) layer, adds SAP information to the frame for cooperation with L3 and other control information, multiplexing is also performed here.
- MAC (Media Access Control) – the lower sublayer, provides cooperation with the L1 (physical) layer, adds the physical address of the receiver and sender to the frame, adapts the frame structure for the specified bit rate and to the beginning of the synchronization sequence of bits (to make it clear where exactly the frame starts), the CSMA/CD mechanism is implemented here.

The figure on the right indicates the relationship of the sublayers of the data-link layer to the neighboring layers. As mentioned above, in a protocol suite, either both sublayers can be “filled” by a single protocol, or two protocols are used – one for the LLC sublayer and one for the MAC sublayer.




Ethernet can be implemented at the L2 layer as follows (for L2 protocols):

1. The IEEE 802.2 protocol (called LLC) is used for the LLC sublayer, and data from the upper layer is encapsulated in the LLC frame. It is then encapsulated in a MAC frame according to the IEEE 802.3 protocol, creating a complete L2 layer frame.

2. The SNAP extension of LLC can be used instead of the original LLC protocol.
3. The data from the upper layer (L3) is encapsulated according to the Ethernet II protocol, which “manages” both the LLC and MAC sublayers.

2.3.3 EtherType

Before we focus on the structure of the frame, let us talk about one of the identification codes that determine which protocol we are actually communicating with, that is, what exactly is to be encapsulated in the frame.

 *EtherType* is an identifier specifying the type of data encapsulated in the frame. It usually determines the protocol that created the payload to be sent by L3. It is represented by four hexadecimal digits, so the maximum value is $(FFFF)_{16}$ (the full form is $0 \times FFFF$), in the decimal system 65 535. In fact, there is the lower limit as well – for the Ether type we use the numbers with the minimal value 0×0600 , in decimal 1536. Thus, the EtherType is from the interval $0 \times 0600 \dots 0 \times FFFF$.



Remark

In the header of a frame created in L2 there are two bytes usually used for the EtherType number. But this field can be used for other type of data as well – length of the frame. However, in ICT, everything has to be completely unique, so it has to be possible to distinguish whether the field contains EtherType or frame length.

The length of the SDU encapsulated in the Ethernet frame is (almost) always less than $0 \times 05DC$ (1500 in decimal), we round it up and get 0×0600 .

If it is necessary to transfer a very large frame (above the specified limit), then it is *jumbo frame* and there is a special value for this purpose in the given field.



There are a lot of EtherType values, some of them follow:

- 0×8870 : jumbo frames
- 0×8100 : VLAN frames according to 802.1Q
- 0×0800 : IPv4
- $0 \times 86DD$: IPv6
- 0×0806 : ARP
- 0×0835 : RARP
- 0×08137 , 0×08138 : IPX
- 0×8847 : MPLS unicast
- 0×8914 : FCoE Initialization Protocol
- $0 \times 814C$: SNMP



Remark

From the term “EtherType”, it might appear that this identifier is used only in Ethernet. Although it was the first to be used in Ethernet, we actually encounter it in other types of networks as well.




Additional information

<https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml>



2.3.4 Ethernet Frame Format

We will gradually go through all three possibilities. The format of the frame (and later the format of various other PDUs) will be represented in the form of a table, where the length of individual fields is indicated in the header.

 **LLC frame.** The structure of the LLC frame (according to the IEEE 802.2 protocol) is simple – a header with three fields is added to the SDU received from the parent layer.

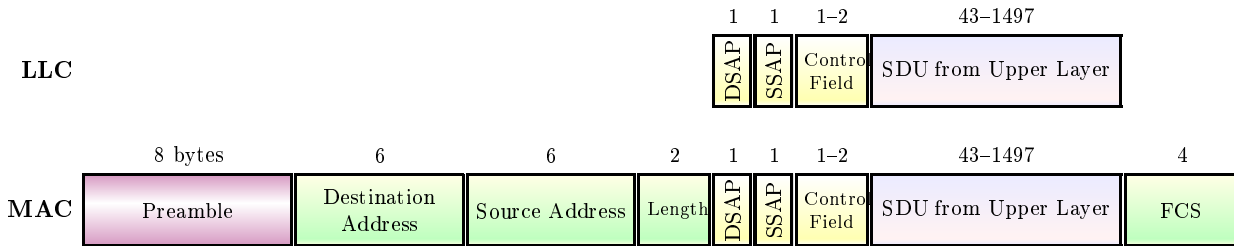


Figure 2.4: IEEE 802.3/802.2: LLC frame encapsulated in MAC frame according to IEEE 802.3


Figure 2.4 shows the format of the given frame type. The LLC header is yellow, the header and trailer of the MAC frame is green and purple). The individual fields in the LLC frame (yellow) have the following meaning:

- *SSAP, DSAP* (1 octet each) – access points (SAP) on the destination (DSAP) and source (SSAP) devices, which occupy 7 bits from each octet; the remaining (least significant, right) bit in each octet indicates:
 - I/G bit (individual/group) in the destination octet determining if the given SAP is group or individual,
 - C/R bit (command/response) in the source octet for the packet type (command or response),
- *control field* (mostly 1 octet) – determines the type of frame in terms of service, ordinal number, etc.,
- SDU from the upper layer encapsulated in the LLC frame.

The values for the DSAP and SSAP fields are standardized, some of them follow:

- | | |
|-------------------------------|------------------------------|
| • 0x06: DoD IP | • 0x98: Arpanet ARP |
| • 0x42: IEEE 802.1 Bridge STP | • 0xE0: Novell Netware (IPX) |


Some other values are used, for example, for LLC management (service frames, they have nothing to do with the upper layer), others are reserved for some protocols that are no longer used.

 **Example**

If an LLC frame header contains these numbers and other data:

0x42	0x42	0x03	data
------	------	------	------

This means that on both the source and destination devices, the L2 layer (its LLC sublayer) communicates with the Spanning Tree Protocol (STP). The number 0x42 is binary 1000010, so

the I/G bit and C/R bit have the value 0 (individual SAP, command). Inside (like “data”) is the STP protocol PDU. 



Additional information

- <https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tutorials/llc.pdf>
- <https://tools.ietf.org/html/rfc1700> (SAP numbers)



The MAC sublayer adds the following fields according to IEEE 802.3 (green and violet):

- *Preamble* (8 octets) identifies start of frame, it is a synchronization information. The first 7 octets of the preamble contain alternating ones and zeros, except for the last bit of the eighth octet – this is set to one instead of zero. The whole preamble therefore looks like this:

```
10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011
```


In the literature we also find another characteristic – a preamble on 7 octets with alternating ones and zeros, and then one octet, which violates this rule in its least significant bit. This octet is called SOF or SD (Start-of-frame Delimiter).

- *Destination Address* (DA) and *Source Address* (SA) are the MAC addresses of the communicating nodes in the network. The source address must always be unicast, the destination can be unicast, multicast or broadcast.
- *Length* of the encapsulated LLC frame (so the length of the data from the upper layer plus the LLC header).
- *FCS* (Frame Check Sequence) is a checksum that is calculated from address fields (DA, SA), length, LLC headers and higher layer data (i.e. the header but the preamble, which is always the same, data, and except FCS).




Remark

The FCS checksum (a CRC – Cyclic Redundancy Check) is important: it is calculated by both the sender and the receiver. When the destination device receives a frame, it calculates the FCS over the same fields as the sender did, and if it finds that its calculated value is different from the one found in the trailer, it considers the frame corrupted and discards it.

It is interesting that in this field the bits are stored in the opposite order than in other fields, and moreover the position of this field can be determined algorithmically (so technically it is not even necessary to have the frame length or payload length in some field). 

The LLC frame according to IEEE 802.2 usually encapsulates the data units of “operating” protocols on L2, such as the above-mentioned STP, or for example LLDP (Link Layer Discovery Protocol). LLDP is an open standard (i.e. various vendors can implement it and do so), through which devices communicate with each other at the L2 layer and share their parameters.

 **SNAP frame.** SubNetwork Access Protocol (SNAP) was created by extending the IEEE 802.2 LLC protocol, modifying and extending the LLC sublayer header. The SNAP frame looks like this:

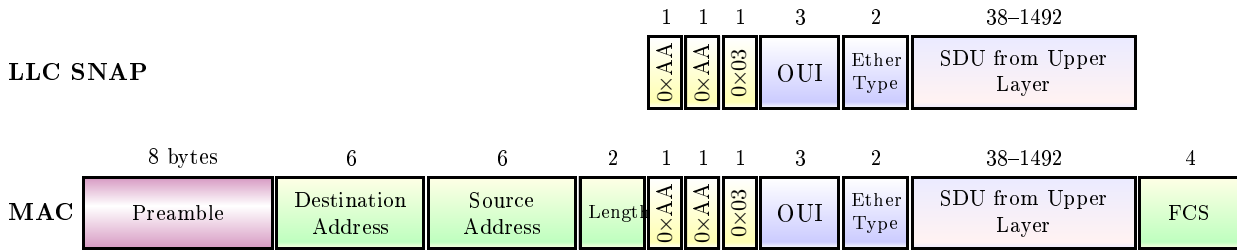


Figure 2.5: IEEE 802.3/SNAP: SNAP frame encapsulated in MAC frame according to IEEE 802.3

In Figure 2.5, the fields added by the SNAP extension are colored blue. As we can see, SNAP sets certain constant values in the LLC header – the code 0xAA is used as DSAP and SSAP (or the code 0xAB is also allowed) and the number 0x03 is saved into the control field.

The SNAP extension adds the following fields (blue in Figure 2.5):

- *OUI* (Organizationally Unique ID) is usually set to 0. If not, then this is the identifier of the manufacturer of the sending device. For example, Cisco has an OUI = 0x00 00 0C.
- *EtherType* (or more generally Type) specifies the protocol of the upper layer if OUI = 0. If not, then this field is purely directed by the organization whose OUI is in the previous field, for example for packet transmission according to proprietary network protocols.

In the previous pages there is the list of several commonly used values for the SSAP and DSAP fields in the LLC header, and the link to the complete list. The value 0xAA is one of the identifiers for the SNAP extension.

Remark

How do we know if an incoming frame is of the type IEEE 802.3/LLC or IEEE 802.3/SNAP? Several fields (MAC frame headers according to 802.3) are the same, the difference starts at the beginning of the nested LLC/SNAP frame. If in the “yellow part” (three octets after the MAC header) we find the value 0xAAAA03, then it is an SNAP frame. Otherwise, it is an LLC frame.

Ethernet II frame. Currently, this type of frame is used almost exclusively in Ethernet (for all data frames), it is the simplest and the most efficient.

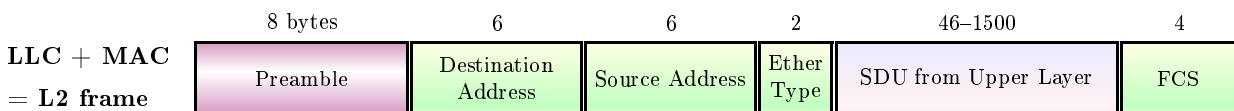



Figure 2.6: Ethernet II frame

In Figure 2.6 we can see that the structure of the frame is actually very similar to what the IEEE 802.3 protocol provided on the MAC sublayer in the previous cases. The preamble fields, both MAC addresses and checksum have the same meaning. The only difference is in the field in which the frame length was in the previous cases – within Ethernet II we find the value EtherType there.

Remark

How do we know that this is an Ethernet II frame and not any of the previous ones?

One of the previous sections describes the values and meaning of the EtherType field. Among other things, we read in one note that EtherType and frame length are stored in the same field, with a number greater than 00600 meaning EtherType, less than meaning frame length. So, while loading the first $8 + 6 + 6 = 20$ octets, the switch still can't distinguish between frame types, but according to the 21st and 22nd octet can already distinguish whether it is Ethernet II. 


**Additional information**

- <http://www.infocellar.com/networks/ethernet/frame.htm>
- http://www.wildpackets.com/resources/compendium/ethernet/frame_formats


**Tasks**

There are many sites on the web that offer files with captured network traffic for download. These files are great for practice, we can see typical traffic on them using certain specific protocols. Some of these sites are better avoided, but others are worth trying, for example:

- <https://wiki.wireshark.org/SampleCaptures>
- <http://packetlife.net/captures/>
- <http://www.netresec.com/?page=PcapFiles> (recommend after gaining more extensive knowledge in the field of computer networks, especially in eavesdropping on malicious software)
- <https://www.nostarch.com/packet2.htm> (the link “Download the capture files for this book” leads to the files with captured traffic)

Choose at least two of these addresses and go through the content of the websites. 

**Remark**

If the actual, minimum or maximum length of the frame is stated somewhere, the preamble is usually not included. If we go back to the drawings of the structure of each frame type and add up the numbers given above each field (the lengths of these fields) except for the preamble, we find that in all cases the minimum frame length is $46 + 18 = 64$ octets, the maximum length is $1500 + 18 = 1518$ octets. 

2.3.5 Collision and Broadcast Domain

**Definition (Collision and Broadcast Domain)**

A *Collision Domain* is a collection of devices on a network that can collide with each other in communication. Switch (and router) can filter traffic and only allows unicast frames on the port to which the destination device is connected, so it separates collision domains. Hub does not separate collision domains (instead, it connects them) because it forwards frames to all ports. A collision domain usually corresponds to a network segment.

Broadcast domain is the set of devices on a network that can receive each other's broadcast frames – if any one of these devices sends a broadcast frame, all the others will receive it. Because

the switch and hub forward broadcast frames to all ports (except the one from which the frame came), they do not separate broadcast domains. Conversely, the router does not forward (discards) broadcast frames, so it separates broadcast domains. A broadcast domain corresponds to the term network.



Simply said:

- If a device filters unicast frames (sends them only where they belong), it separates collision domains. If a device floods unicast frames, it does not separate collision domains.
- If a device withholds broadcast frames, it separates broadcast domains. If a device floods broadcast frames, it does not separate broadcast domains.

If we use a switch and half duplex (with the CSMA/CD collision method), then the collision domain includes these two interconnected devices. If we use a switch and a full duplex, there is no reason to talk about a collision domain, even these neighbors cannot collide (and CSMA/CD is not used).

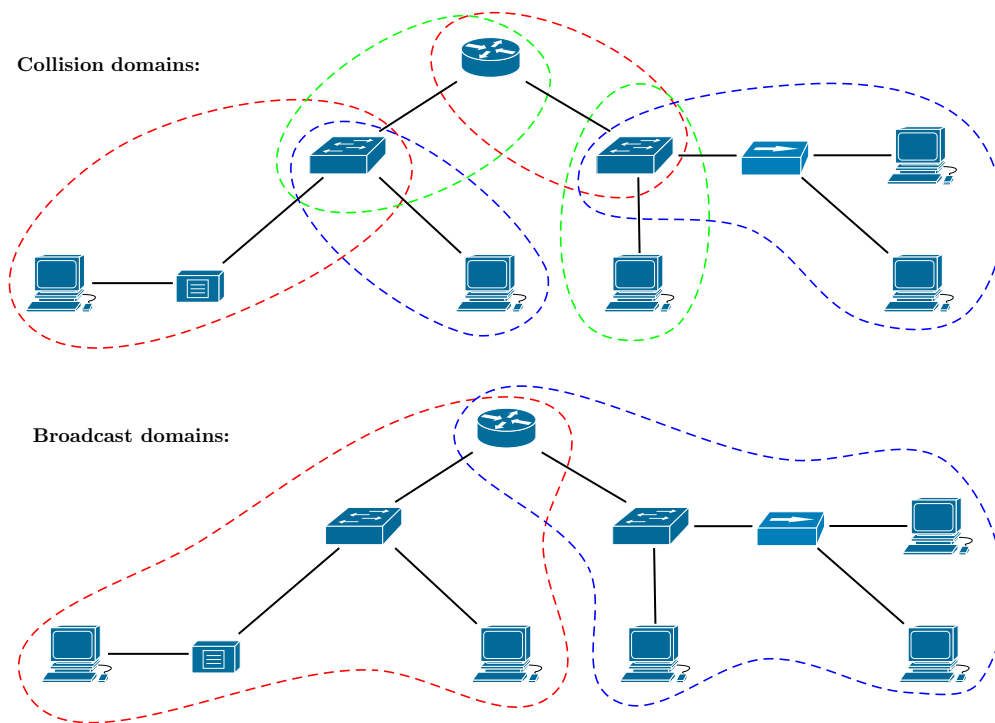


Figure 2.7: Collision and broadcast domains relative to various network equipment

2.4 Ethernet at the Physical Layer


The following operations are provided on the physical layer in Ethernet:

- cooperation with the L2 layer, taking the SDU from this layer (i.e. the frame), in the opposite direction passing the data to the L2 layer,
- coding, multiplexing,
- transmitting and receiving signal, synchronization,


- auto-negotiation – the network interface needs to negotiate with the network interface of the neighbor device all transmission parameters (speed, half or full duplex, etc.) in order to understand each other.

2.4.1 Coper Cabeling


Ethernet is quite variable in terms of cables. Metallic cables are usually unshielded twisted pair (UTP), but there are also standards for shielded twisted pair (STP) and older standards for coaxial cable. For longer distances (backbone networks), fiber optic cables can be used.


 **UTP (Unshielded Twisted Pair).** Unshielded twisted pair is a metallic cable derived from telephone cable. Properly, it is supposed to have four pairs of conductors (i.e. eight conductors in total), but we also encounter “stitched” cables containing only two pairs of conductors. These “weak” cables are usable but do not reach higher speeds (only up to 100 Mbps). The cables have only basic shielding (each conductor has colored plastic insulation and the whole cable is in a plastic wrap).

The wires are in pairs, the pair of wires is twisted into each other (hence the name, actually we should say “bundle of twisted pair wires”). The reason for twisting is to reduce reception of electromagnetic interference from the surroundings and also to reduce the emission to the surroundings (pairs in a bundle should not interfere with each other) – suppressing the “antenna effect”. The higher the speed, the higher the emission, so a higher quality cable is needed for higher speeds.

 A whole pair is always necessary for signal transmission (one conductor is not enough), because the signal is characterized as the *difference of electric potentials* of the two conductors in the pair. Standards define which pairs are used for which purpose (for full duplex, at least one pair is usually used for each direction).


In general, the higher the category, the higher the quality for longer distances. Table 2.2 on page 50 gives an overview of the categories. Currently the most common category is Cat.5e, with higher categories being promoted in new construction (Cat.6_A is recommended). Higher categories already require a different type of connector (and both cable and connectors are more expensive), so they are not yet very popular.

 **STP (Shielded Twisted Pair), FTP, etc.** Shielded twisted pair cable has added shielding in addition to the plastic protection, either pair shielding (each pair is additionally wrapped with aluminum foil) or whole cable shielding (all pairs together are wrapped with aluminum foil or metal mesh) or both. On shielded cables, the termination is fitted slightly differently due to the need to ground the cables. If the cable is not grounded, the metal shielding (especially in the form of metal braid) acts as an antenna and causes interference.

 In the abbreviations of cable names “F” is used for wrapping individual pairs or all pairs with aluminium foil, “S” is used for securing the bundle with a metal mesh. For example, U/FTP means that each pair has its own foil, F/UTP means that all pairs together are wrapped in foil, F/FTP means both (foil around the twin lines and the bundle), S/UTP is a cable with a metal mesh braid around the bundle, etc. for other combinations. Some manufacturers don’t stick to this designation, S-STP often hides the shielding with aluminum foil for individual pairs and the whole bundle.

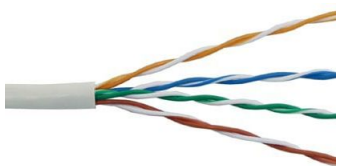
Category	Usage	Shielding
Cat.1, Cat.2	Not currently in use.	no
Cat.3	Mostly for telephone wiring. In computer networks for old 10Mb Ethernet.	no
Cat.4	For Token Ring networks. Currently it is practically not used.	no
Cat.5	For Fast Ethernet (100 Mbps). Some higher quality cables can also be used for Gigabit Ethernet, but it is not recommended.	no
Cat.5e	For Fast Ethernet and Gigabit Ethernet. Currently the most widely used cabling for LANs. The bandwidth is 100 MHz.	yes
Cat.6	For Gigabit Ethernet and higher. For 10G Ethernet, the distance is limited to cca 55 m. The bandwidth is 250 MHz.	yes
Cat.6A	Similar to Cat.6, but with a higher bandwidth (500 MHz) and at 10 Gb/s it can transmit up to 100 m. These cables are often available in shielded versions.	yes
Cat.7	For 10G Ethernet and higher, unlike Cat.6A it is fully shielded and offers higher bandwidth (600 MHz).	yes
Cat7A	Another improvement, even higher bandwidth (1000 MHz).	yes
Cat.8, 8.1, 8.2	For 40 Gbps Ethernet, bandwidth is up to 2000 MHz. Designed for data centers, for short distances between switch and server.	yes

Table 2.2: Categories of twisted pair cables

 Regarding shielding, in practice, we most often encounter the types listed in the table on the right (or the inaccurate marking STP appears), marking according to ISO/IEC 11801. There are other possible combinations of shielding (after a bit of research, you will certainly find out what the letters “F” and “S” mean in the marking in a particular place).

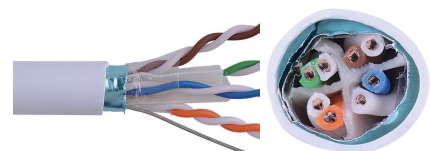
Marking	Shielding	
	pairs	bundle
UTP, U/UTP	—	—
U/FTP	foil	—
F/UTP	—	foil
S/FTP	foil	mesh
SF/UTP	—	foil + mesh

Twisted pair is used for baseband transmission. The connector used for the 6A category is an RJ-45 with eight pins, which is correctly referred to as 8p8c. The wire order in the connector follows the TIA/EIA 568-B or TIA/EIA 568-A standard up to 100 Mbps, but if the cable is to be used for higher speeds, the ANSI/TIA 568-C standard must be chosen.

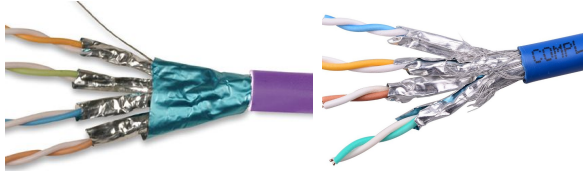


On the left, there is a UTP Cat.5e cable. There are 4 pairs (8 wires), each pair is twisted itself, which prevents mutual interference. Each conductor has insulation in a different color, while in a pair there is always one conductor in solid color and the other white with a (sometimes not very noticeable) stripe in that color.


On the right, we can see the F/UTP Cat.6 cable. The pairs have no special shielding, just the usual colored insulation, while the whole bundle of pairs is wrapped in aluminum foil as an additional shielding. Category 6 and higher cable




may have a center plastic separator inside with a “cross” profile that separates the four pairs from each other to further reduce interference, and also makes the cable stronger overall. The center separator can also be in Cat.5e, but not usually. The picture further to the right shows a cross-section of the entire cable, showing the location of the twisted pairs as well as the center separator and aluminum shielding.




On the left, we can see the F/FTP Cat.7 cable. Next to it is the S/FTP cable, where there is aluminum foil around the pairs, but we have metal mesh around the whole bundle.

 A twisted pair line does not need to have only two or four pairs. In telecommunications (but not at the customer’s side), cables with 25, 100 or other number of pairs are used, and the connectors correspond to this.

Cables marked as *outdoor* are resistant to temperature fluctuations, humidity, light, may have internal steel wire reinforcement (for resistance when hanging), or external metal cover against biting (rats and similar animals). Conventional cables are reinforced only with Kevlar fiber.

 There are also cables designed for special applications. For example, twisted pair cables labeled *plenum* are designed for ventilation, air conditioning and heating environments. They are modified to be more resistant to fire and not to produce dangerous gases in the event of a fire, which is reflected in their price. Slightly cheaper are the *riser* cables, which are more resistant to fire but not to produce dangerous gases.

 *Co-axial cable* is nowadays used either for connecting external antennas or for cable or satellite TV distribution used for network signal.



It is a coaxial cable (two conductors with a common axis), where one conductor is the centre wire, the other conductor is a metal mesh (as seen in the picture on the right), with a non-conductive insulation (dielectric) between them. The signal is determined by the difference in the electrical potentials of these conductors.


The typical physical topology for coax is a bus – a number of devices are connected to a single cable, and a “branching” must be created for each device from the main cable. Both ends of the main cable must be fitted with terminators that absorb the signal. If we did not do this, the signal would be reflected back at the end of the cable and interference would be generated. The bandwidth is greater than that of twisted-pair, so it is possible to use broadband transmission, but most of the time in computer networks, transmission is in the baseband.

The advantage is better shielding. However, a significant disadvantage is the overall poorer cable handling (coax is thick and not very flexible, and crimping the connectors is more difficult). Another typical characteristic (sometimes an advantage, sometimes a disadvantage) is that it is a typical *shared medium* – more than two devices can be connected and the signal always propagates in both directions (twisted pair, on the other hand, is a dedicated medium where sharing can only be implemented using hubs).

Nowadays, coax is not used for Ethernet, it is mainly found in TV distribution (connecting TVs to an antenna or satellite) with network signal support (the DOCSIS standard) or in wireless

solutions to connect an external antenna. While we use coax with an impedance (specific resistance) of 50Ω for computer networks including antennas, coax with an impedance of 75Ω is used in TV distribution. It is always necessary to use a cable with the impedance that is supposed to be there, otherwise the signal cannot be transmitted properly.

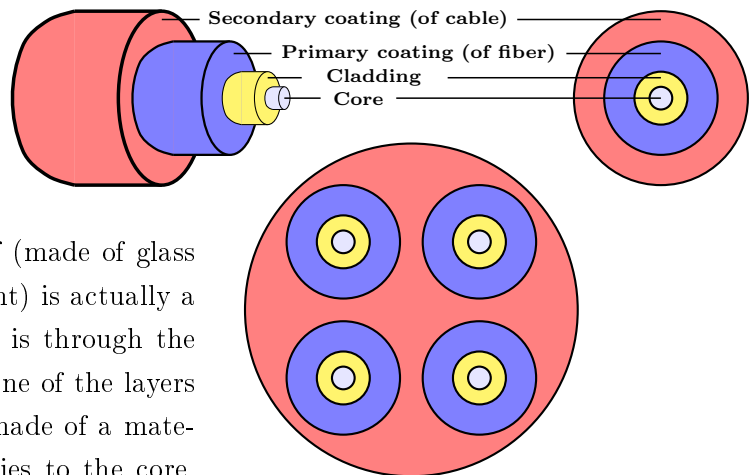


 *Twinax* (twin-axial cable) is similar to coaxial cable with the difference that the two conductors are in the form of copper wire (but are also insulated from each other), no metal braid is used. We can see twinax cable in some Ethernet specifications, usually for use in large data centers to connect a server to a switch because of its better latency.

2.4.2 Optical Cabling

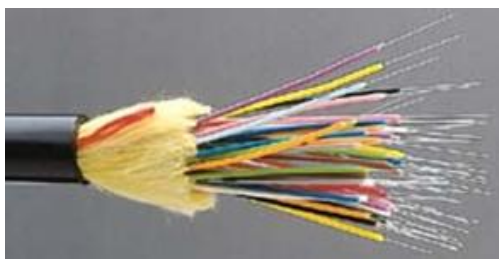
Optical cabling should be used especially where distances up to 100 m are not sufficient. These are distances of hundreds of meters or units or tens of kilometers (depending on the type and standard).


Optical cable consists of several optical fibers surrounded by protective layers. The figure on the right outlines a simplified structure of a fiber optic cable containing one fiber and below four fibers.



The fiber itself (made of glass or glass-like plastic with silicon content) is actually a core surrounded by a coating, and it is through the core that the light beam travels and one of the layers serves as a reflective “mirror” – it is made of a material that has similar physical properties to the core, but a significantly different refractive index. In addition,

it increases the strength of the fiber and thus its durability. This is followed by primary protection of the fiber. Secondary protection can be e.g. Kevlar based yarn, some suitable plastic, gel, etc. The cable jacket is then combined with other protections, depending on the requirements for securing the cable against mechanical damage, water, chemical damage, UV radiation, rodents, etc. The most complex protection is intended of course for underwater cables.




 Now let’s look at how optical fibers actually carry the signal. An optical signal of a certain wavelength is generated at one end of the cable and must be fed to the other end. This is not exactly easy, because whenever the ray hits the border between the core and its cladding, it will be reflected at (almost) the same angle as when

it arrived; the light source will of course generate more than one photon, and it is possible for different photons to be sent out at slightly different angles. These angles may then change further along the way by multiple reflections, and the larger the angle, the longer the photon takes to travel.


What does this imply? The longer the path, the more likely it is that the photon bunches belonging to the individual bits sent will be “mixed” and it will not be possible to correctly detect the boundaries between the bits or the correct values of these bits at the destination. There are two solutions to this problem: either maximize the accuracy, or reduce the effective range (maximum cable distance).

To maximize accuracy, we can first use a high quality light source (laser instead of LEDs) and at the same time reduce the diameter of the fiber core as much as possible, thus reducing and refining the angle of reflection, or minimizing the amount of reflections (in the ideal laboratory case, photons will still fly straight without reflections).

 There are two types of optical fiber:

- *singlemode fiber* (SM) has a very small core diameter and uses a high quality light source, is more expensive but has a greater effective distance (units to tens of kilometres),
- *multimode fiber* (MM) has a larger core diameter, less demanding on the quality of the light source, is cheaper, and has a rather smaller effective distance (hundreds of metres).

The wavelength ranges of the transmitted signal also differ; single-mode fibers use higher wavelengths.

 In multimode fibers, signal reflections occur at the interface between the core and its cladding, we distinguish two versions according to the refractive index of the cladding:

- *step index fiber* where the index of refraction is the same across the whole cladding (and therefore the angle of reflection is still the same),
- *graded index fiber* where the refractive index of the cladding decreases gradually from the core. We don’t see them much nowadays.

Figure 2.8 shows the signal propagation in both types of multimode fibre and singlemode fibre. The signal is most optimally converted in singlemode fibre, followed by multimode fibre with gradient refractive index change.

A fibre optic cable does not necessarily have to contain only optical fibers. Metallic wires can also be carried through it (for example for powering, this would not be possible through the optics itself), or various reinforcements affecting the strength of the cable.

 **Remark**

Let’s compare the properties of metallic and optical cables:

- signal representation is different,
- this is related to the fact that when using optics, a transformation between optical and electrical signals must be performed on both sides,
- interference – with metallic cables there can be electromagnetic interference between individual pairs and from the outside of the cable, with optical cable this is not a threat, interference can be caused only by UV radiation (the wavelength corresponds to the transmitted signal), but this can be solved by optical isolation,
- distance – metallic about 100 m, optical hundreds of meters to tens of kilometers (i.e. optical cables have less attenuation),

¹From: <https://reggle.wordpress.com/2013/02/03/osi-layer-1-part-ii-fiber/>

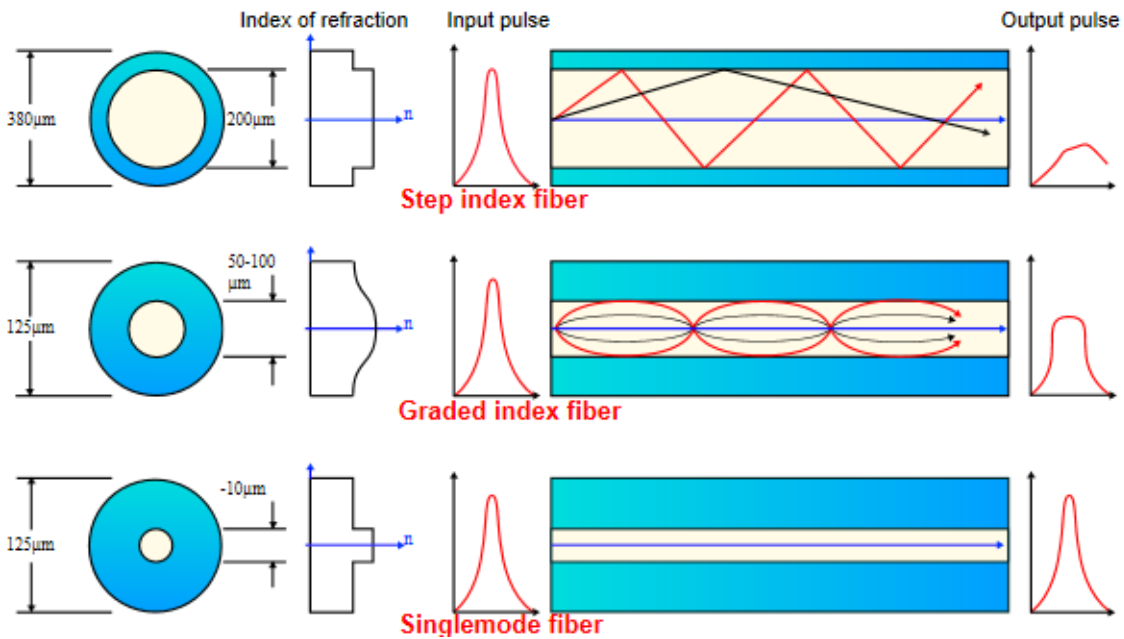




Figure 2.8: Various types of optical fibres¹

- powering – it is (not always) possible to power via metallic cable, this is not possible with the optics itself,
- price – metallic cables are considerably cheaper and the same applies to connectors and other equipment,
- physical properties – metallic cables are more flexible but easier to damage, on the other hand damage is easier to repair,
- crimping (attaching endings) is easier with metallic cables, etc.



 For the purpose of illustration, we will provide some specific data:

- Gigabit Ethernet:
 - 1000Base-SX uses an LED or laser with a wavelength of 850 nm (770–860 nm) as a light source, multifiber cables; if we use cables with a core diameter of 50 μm , the effective distance is 220 m, with a core diameter of 62.5 μm it is 550 m,
 - 1000Base-LX uses a 1300 nm (1270–1355 nm) laser, singlemode cables with a core diameter of 10 μm for distances in units of kilometers (but manufacturers usually guarantee up to 20 km), or multimode cables with a core diameter of 50 or 62.5 μm for shorter distances.
- 10Gbit Ethernet:
 - 10GBase-SR – laser 850 nm, multimode fiber, distance of tens of metres up to 400 m depending on the quality of the fibre (see below),
 - 10GBase-LR – laser 1310 nm, singlemode fiber, distance 10 km (in practice up to 25 km),
 - 10GBase-LRM – laser 1310 nm, multimode fiber, distance 220 m,
 - 10GBase-ER – laser 1550 nm, singlemode fiber, distance 40 km.

 The fiber is characterized by two numerical data: x/y , where the first data is the diameter of the core and the second is the diameter of the core cladding in micrometers. For example, 50/125 is a fiber with a core diameter of 50/*micrometers* and a cladding diameter of 125/*micrometers*.


 According to ISO 11801, the following designations are used for multimode optical fibers:

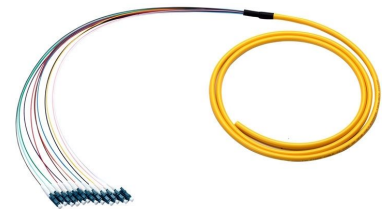
- OM1 – 62.5/125 fiber, usually for LED signal sources, max. speed 100 Mbps,
- OM2 – 50/125 fiber, typically for LED signal sources, up to 1 Gbps (with 82 m distance limitation, 10 Gbps),
- OM3 – 50/125 fiber, typically for laser signal sources, up to 10 Gb/s (higher speeds with significant distance limitation),
- OM4 – 50/125 fiber, laser source, up to 40 or 100 Gb/s (up to 150 m distance),
- OM5 – fiber again 50/125, laser source, is designed for transmissions with multiple subchannels (WDM – similar to frequency division multiplexing, but on wavelengths).


OM1 and OM2 category cables are orange (this means the outer jacket of patch cables, sometimes even pigtailed – see below), OM3 and OM4 are light blue (aqua), OM5 is lime green.

Singlemode fibers also have their categories, but less so. OS1 and OS2 are singlemode 9/125 fibers, the source is a laser. The difference between them is, e.g., the range (OS2 has a slightly higher range for a given speed). Single-mode cables tend to be yellow.

If there are more than one fibre in the cable, the individual fibres are also distinguished by colours, each colour corresponding to a number. For example, blue indicates fibre number 1, orange fibre number 2, etc. This is important because each of the fibres has to fit together and there must be no confusion when they are installed and connected to the infrastructure.

 A *Pigtail* is a short piece of cable (for example 1.5 m) terminated on one side with a connector. Pigtails are mainly used in cable types where the connector is more difficult to fit than the signal wires in the jumper, for example in fiber optic cables.



 Fiber optic cables running underground (backbone, but also those leading to the destination – last mile) are nowadays usually blown into the appropriate guiding elements, tubes. Special means are used for this: blowers with compressors. The cable is forced into the tube with a stream of compressed air.



Remark

You don't need to know all the above information by memory, it is enough to know that there are various types of optical fibers (when you see for example MM OM4, so that it is clear that it is one of the categories of multi-mode fibers and "rather better", and that they also differ in colors. It is good to know that yellow indicates single-mode fiber.



Additional information

- <https://community.fs.com/blog/advantages-and-disadvantages-of-multimode-fiber.html>
- <https://www.thefoa.org/tech/ColCodes.htm>

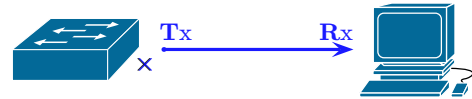



2.4.3 Crossover

The device actually performs two basic operations on its network interface:

- Tx (Transmit) – sends off (submits) something,
- Rx (Receive) – receives (listens, and when he finds out that a neighbor is transmitting, he receives the transmission).

For a twisted pair line, there are pairs of wires reserved for the Tx operation and pairs of wires reserved for the Rx operation, or this assignment is determined dynamically (but at any time this assignment is unambiguous). In other words – a network interface consists of two parts – Tx and Rx, and it matters which of these parts is connected to which pair in the cable. The sending device sends a signal to the Tx part of its network interface, and the receiving device receives that signal on the Rx part of its network interface, and the two parts must be physically connected by the same pair of wires.



 **Remark**

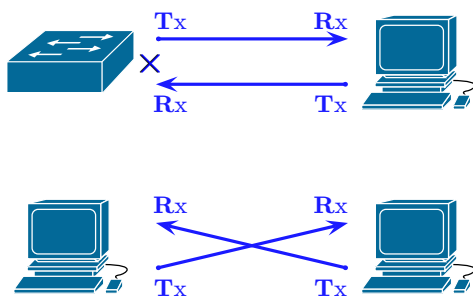
It follows that somewhere the transmission path must be *crossed*. If two devices are connected by a twisted pair line, then (at least one) pair of wires leading from the Tx of one device must be connected behind the Rx of the other device and vice versa.



There are three ways to ensure crossover:

- directly in the cable: the wire is connected in one connector to Tx (transmit end), in the other connector to Rx (receive end), for all wires,
- switches perform crossover on the interface (MDIX) by default,
- switches cross the path at logical level (L2), i.e. what they receive on Tx they process it and send to Rx.


The second option is controllable. We can enable or disable it, or allow the switch to set this feature itself depending on whether a cable with crossover is connected to the interface according to the first option.



In the picture on the left we see two solutions for direct connection of two devices – in the first case the crossover is provided by one of the devices (usually active network equipment, here switch), and the port providing the crossover is marked with a cross. In the second case, the crossover is provided directly in the cable. In Figure 2.9 we can see the cases when there are intermediate devices on the path,

on the ports of these DCEs the crossover is also provided.

The crossover inside the switch at the logical level is not shown here, but of course it exists.

 **Example**

Calculate the crossings on the path between the end devices in both drawn cases.

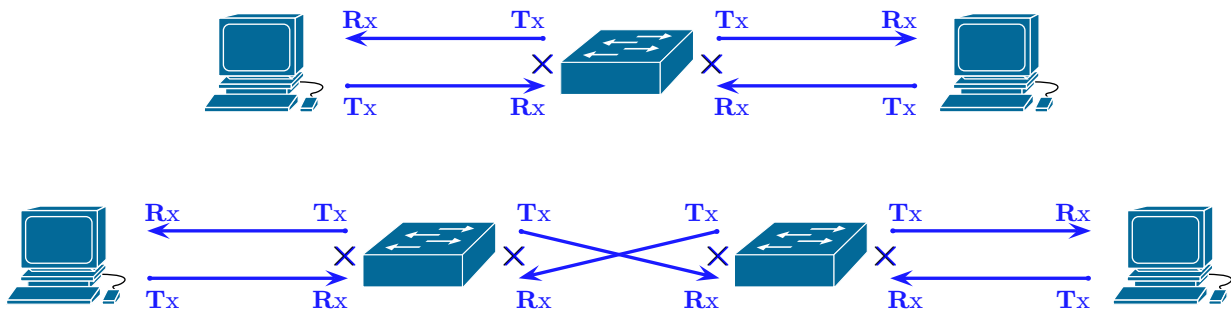


Figure 2.9: The principle of crossover on twisted pair

- On the first row we have (1) one crossover on the switch side at the left port, (2) one crossover inside the switch, (3) one crossover on the switch side at the right port ⇒ for a total of three crossovers.
- On the second row there are a total of (1) four crossovers on the port side of both switches, (2) one crossover provided by cable, (3) two crossovers inside the switches ⇒ for a total of seven crossovers.

Note that odd numbers came out – the number of crossovers on a path between two DTEs (via any DCE devices on the path) must always be an odd number. The same is true for a direct connection between two devices.



There are two types of twisted pair cables:

- *straight through cable* – does not contain any crossover, it is usually used to connect DTE and DCE (host–switch, switch–router) – from the Ethernet point of view the router is actually a DTE, i.e. an end device,
- *crossover cable* – contains a crossover, used to connect devices at the same level (DTE–DTE, DCE–DCE, e.g. host–host, switch–switch) and also to connect a host to a router.

Interconnection of:	Switch	Host	
Switch	×	—	— straight-through cable
Host	—	×	× crossover cable

Table 2.3: When to use a straight-through and crossover cable

In fact, the same cable is used in both cases; whether it is a straight-through or crossover cable is determined by the fitting of connectors. A straight-through cable has the connectors on both ends in the same way (the conductors are in the same order at both ends), whereas a crossover cable has the conductors at one end arranged differently from the other.

Nowadays (with 100Mb Ethernet and faster) we have a slightly simpler situation – newer network interfaces can usually automatically detect if they have or should provide internal crossover, so in most cases we can just use a straight-through cable. So the Ethernet interface can be:


- MDI (Medium Dependent Interface) – is on the end devices (from the Ethernet point of view), i.e. DTE, does not have internal crossover,

- MDIX (Medium Dependent Interface with crossover) – tends to be on switches, has internal crossovers.


In addition, switches usually support the *Auto-MDIX* function on each interface, which (if enabled, which it usually is) can detect by the connected cable whether the internal crossover of the interface should be enabled or disabled. If this feature is on, then we can safely use a straight-through cable even where we should theoretically use a crossover, but if it is off, we have to stick to the standard and use the “correct” cable.

2.5 Working with Cables


We’ve been introduced to cables, but every cable has to be terminated somehow. At both ends of the cable there is either a connector (patch cord type cable) or it is fitted into sockets in wall (when installed in wall), for some types of cables also pigtail type is used (for fiber optics or twinax). In some cases we need to connect two ends of the cable (for example when we have two short cables and we need one long cable), then we can use a special *coupler*.

 The standards distinguish between *telecommunications connector* and *telecommunications outlet* terminations. In the following we will only deal with UTP cables.

2.5.1 Crimping the connector to a straight-through UTP cable

 The process of putting a connector on a cable is called *crimping*. On a UTP Cat.5e cable we usually crimp an 8p8c connector (that means 8 positions and 8 contacts), which is (not exactly) also called RJ-45.

When crimping, we must, among other things, arrange the wires from the cable correctly (place them in the cable in the correct order). Currently, the TIA/EIA-568-B standard from 2001 is used to determine the order of the wires, although there is a newer version of ANSI/TIA-568-C from 2014.

 The standard specifies two *sequences* – T568A and T568B (be careful not to confuse the letters with the version of the standard), with the straight cable using the T568B sequence at both ends, whereas the crossover cable uses one of these sequences at each end. Note that this is not just a reversal of the order!

Procedure (Preparation for crimping the 8p8c connector)

We will need the following:

- a sufficiently long UTP cable, preferably category 5e, we should take into account that part of the cable will have to be removed for the alignment of the conductors and part of it will be inserted into the connector,
- terminals (connectors), there may be plastic covers for the connectors,
- crimping pliers (they are necessary for fixing the pins – contacts – to the wires of the cable), they must be for the 8p8c connector type,
- may be useful as a trimming tool for removing plastic insulation from the cable and wires, but is often included with the crimping pliers, or a sharp knife can be used,
- tester to show if all the wires are connected correctly.

Everything you need is available in the classroom.

Crimping pliers come in different qualities and also in different price ranges. Usually, more expensive pliers are of higher quality, with lower quality pliers there is a slightly higher risk of failing to correctly connect a pin to the wire. Alternatively, they may have a lower lifetime.



The pins are in the connector in a line, their numbers can be seen in Figure 2.10 on the left (the connector is rotated so that we are looking at the pins). On the right is an indication of how the pins belong together in pairs. Each pair of pins (as shown) has one pair of wires attached to it, with one of the two wires in solid color and the other in the same color but with a white stripe.

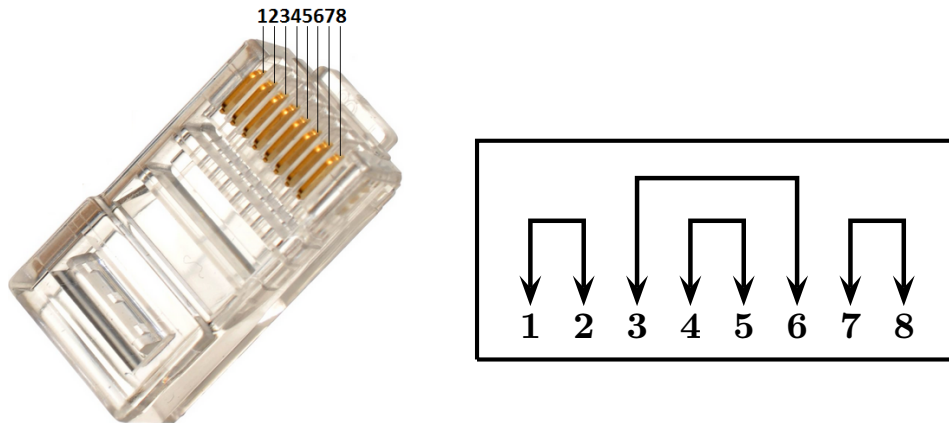
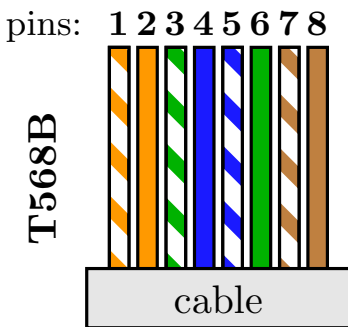


Figure 2.10: Pin order and pin pairing in the connector



The wires in the connector have coloured insulation, and it is by these colours that we know what order they should be in (colour code). The order according to T568B specified in the TIA/EIA-568-B standard is indicated in the figure on the left. In the vast majority of cases, this is the order we choose for the connectors at both ends of the cable.

Note that the pairing of wires determined by the color code corresponds to what is indicated in Figure 2.10 (above) on the right – the orange pair first (pins 1 and 2), the brown pair at the end (pins 7 and 8), the pair for pins 3 and 6 is green, and the pair for pins 4 and 5 (right in the middle) is blue.

Remark

How to remember the order of colours? First of all, remember that the sequence alternates between semi-coloured (with white) and full-coloured conductors. The second clue is the way the pins are paired (which will be helpful for remembering where the green and blue wires specifically should be). Some people also find it helpful that the most prominent color (blue) is in the middle and the least prominent colors (orange and brown), which also differ only in color saturation, are on the edges.

Mnemonic device: there is a river in the middle, green vegetation around it (it has enough water), then dry land (orange or brown).



Procedure (Crimping 8p8c connector on UTP cable)

Now that we know what we'll need, we'll focus on the procedure. The summary is as follows:

- if we want to put on the cover (which is optional for training purposes), we have to do it now (pay attention to the direction – be aware of what it is supposed to cover and how),
- fit the connector to the end of the cable, so that we know how much insulation to remove (about half a centimeter, fit the connector) – the wires should extend to the end of the connector through the pins, but at the same time the cable insulation should extend inside the connector,

if you haven't practiced it yet, it's better to leave the ends longer and unplug the overhang once it's straightened out,

- remove the outer insulation of the cable as measured in the previous step, we must not break the insulation of the wires themselves (otherwise we have to remove the whole connector and start again),
- untwist the pairs up to the place where the insulation is removed, line up the wires according to T568B (the line-up should “work” from the place where we cut the insulation), straighten as thoroughly as possible,
- put the wires close to each other, i.e. as they will be in the connector, and if their upper edges together do not form a plane or are too long, align them by cutting off what overhangs (usually this can be done with a blade directly on the crimping pliers),
- then carefully insert the connector – be careful here, this step is the most challenging, the wires must be exactly where they should be (in the right order and inserted all the way to the end of the connector), the connector must be correctly oriented (look for gold-plated pins),
- put on the crimping pliers and crimp it tightly,
- then if you've used a cap, now slide it onto the connector.

Caution – after using crimping pliers, the connector cannot be pulled out. If we subsequently discover that there is a mistake, we have no choice but to clamp the connector and start from the beginning.



2.5.2 Testing

We can “roughly” verify the correctness of crimping by sight, but if we want the transfer to be really smooth, we should use a tester. Testers can detect not only bad wire order, but also, for example, contact misclamping – then the pin on the connector doesn't reach the metal of the wire and electrons don't get to where they are supposed to go.

Different testers are different in complexity and functionality. In any case, they have (at least) two RJ-45 plugs, into which we plug both ends of the tested cable. This also accounts for the

differences in the price of these products. Some testers have only one plug, with one end of the cable plugged in here, and the other plugged into the working device (switch).



Figure 2.11: Ethernet cable testers (From: heureka.cz)

In Figure 2.11 we can see several available UTP cable testers. Their price is in quite a wide range. The cheapest ones have plugs for RJ-45 connectors (or also RJ-11 phone connectors) and after connection simply by blinking two rows of eight (or any other number, depending on the connector and cable) lights in sequence will show which wire is connected to which – with a straight-through cable the lights in both rows must blink in the same order. More expensive models have either built-in support for other connector types (USB, coax, etc.), a display with a clearer indication of wire connections, or other test-related features. Kits containing crimping pliers, tester and other components can also be purchased.

2.5.3 Crimping the Crossover UTP Cable

When crimping a crossed UTP cable, we must ensure that the Tx pair of one end is connected to the Rx pair of the other end and vice versa.


 If the cable is to be used only for Fast Ethernet (up to 100 Mbps), only two pairs of wires are used, more precisely pins 1, 2, 3 and 6, the remaining pins either remain empty (when using a cable with two pairs instead of four) or are simply not used (or are used for power). This means that we will only cross the pairs using the above mentioned pins and leave the other two pairs straight.

Figure 2.12 shows the wire sequence prescribed by the TIA/EIA-568-B standard (color code), which is valid for a maximum transfer rate of 100 Mbps. In the direction from the left, the order at one and the other end of the cable is indicated – that is, one end is crimped according to the T568B order and the other according to the T568A order. Then on the right we have a diagram of the wire connections relative to the two ends.

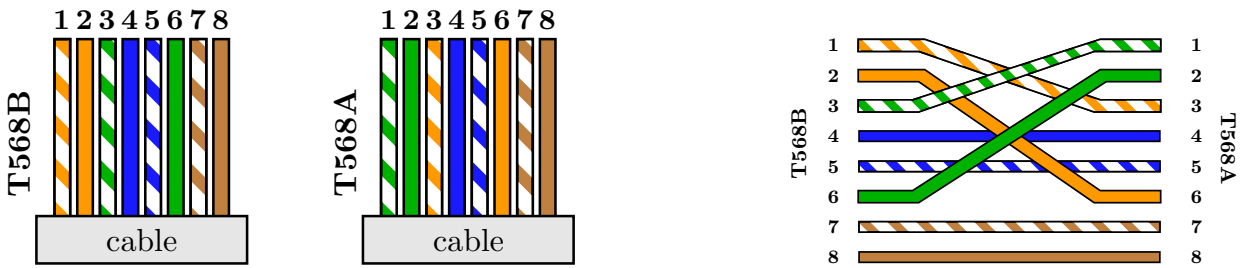



Figure 2.12: Colour code for crossover cable – Fast Ethernet

 The crossover UTP cable for Gigabit Ethernet must already be crimped according to a newer version of the standard – ANSI/TIA-568-C, which prescribes the crossover of both pairs. Here, however, the condition of alternating solid-color and white-color conductors is already violated.

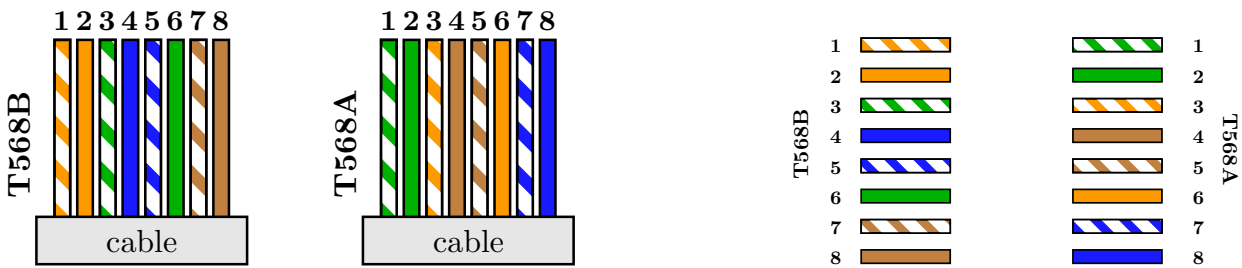



Figure 2.13: Colour code for crossover cable – Gigabit Ethernet


 **Additional information**

Diagrams for various colour codes: <http://www.flukenetworks.com/content/color-codes-dsx-cableanalyzer>

 **Remark**


Fortunately, crossover cables on faster interfaces usually do not have to be dealt with, as we found out earlier (switches usually have an auto-MDIX feature). In some cases, we may still need a crossover cable.

2.5.4 Installation of UTP cable into Outlet

 The socket in the wall into which we plug the RJ-45 connectors is referred to as *telecommunications outlet/connector* according to the standard, which means that the socket actually includes the equivalent of the RJ-45 connector itself in a patch cord, i.e. an internal connector into which the ends of the wires of the cable led (usually through the wall) to the socket must be inserted.

According to the ANSI/TIA-568-C standard, each workplace should have at least a pair of sockets (plugs), one of which is for a four-pair twisted pair cable of category 5e minimum, whether shielded or unshielded. The second one may be the same or intended for fiber optic cable. The older TIA/EIA-568-B standard required a minimum of Category 3, which of course is no longer sufficient today.

The outlets are either flush-mounted (they are used when the cable runs through the wall and the outlet is also to be recessed) or surface-mounted (the advantage of these is that we do not have to cut into the wall during installation, the cable is in the bar). The principle is actually very similar to how it works for power outlets, but the regulations are a bit more lightweight – for example, the cables don't have to be in conduits (we only have low voltage and current in the cables).

 The outlet is often modular, consisting of several parts:

- *internal connector* with a terminal block into which the ends of the wires are cut, usually the slots are marked according to a colour scheme, we can also see the marking “keystone”,
- a cover (frame) with a socket to which the internal connector is attached after the wires are cut.

The specific layout and design varies according to the manufacturer, today we often see so-called toolless (or tool-free) solutions where the cable fitting is very simple. The terminal block is usually marked either 110 (also LSA) or krone.

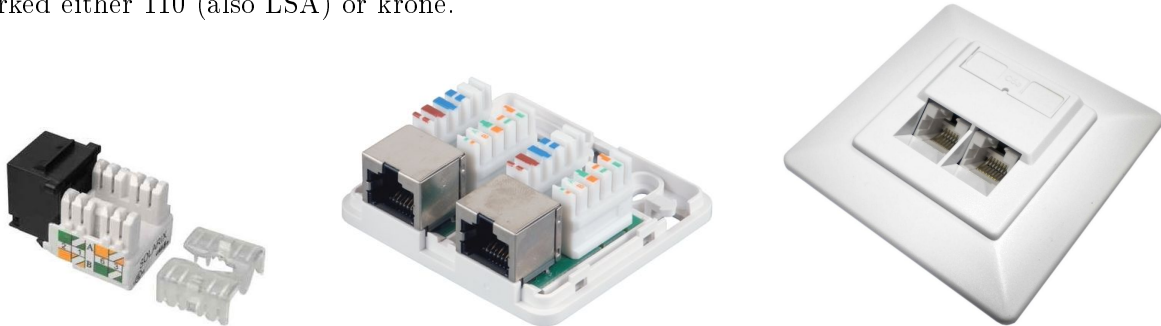


Figure 2.14: Single and double plug socket terminal block and outlet (From: solarix.cz)



Figure 2.15: Patch panel, inside terminal block, outside UTP RJ-45 (From: bscm.cz)

 **Procedure (Inserting the UTP cable into an outlet or patch panel)**

The procedure and tools needed vary slightly depending on the specific socket or patch panel, but we usually suffice with the following:

- outlet including internal connector or patch panel,
- cable (usually led through the wall),
- cutting pliers or other tool to cut off the excess piece of wire,

- notching tool (notching knife, boxer, impact tool, punch down tool), for *110* or *krone* type terminals (the type should match), but not needed for some outlets.

Be careful not to bend the cable too much, you could damage the wires. It should have some slack (also due to thermal expansion), some of it may remain coiled in the outlet.



Figure 2.16: Notching tools (From: aspa.cz, alza.cz, tsbohemia.cz)

The goal is to get the wires of the cable to the right places with the terminal block and then galvanically connect them to the pins in the connector. Procedure:

- Prepare the internal connector and check the color marking (usually we use the one according to T568B). Both are usually marked on the terminal block, we should not confuse them.
- Remove the outer insulation from a piece of cable (max. 13 mm) and pull or insert this piece into the connector (depending on the design). Carefully separate the pairs from each other.
- Untwist the pairs at least partially and separate them into slots according to the colour code.
- If the socket is not toolless, then we must insert the wires into the terminal block with a notching tool. In any case, remove the overhanging parts of the wires (with wire cutters).

For toolless sockets it may be necessary to clip a special cover on the connector, depending on the socket design, see picture 2.14 on the left).

- Now just plug the connector into the cover and mount it on the wall.



Additional information

Fitting the STP Cat.6A (F/FTP) connector: <https://www.youtube.com/watch?v=DbMTF8UmEoE>



2.5.5 Optical Terminations

Optical connectors are somewhat more demanding to fit. It should be remembered that the optical fiber must fit exactly to the other optical fiber to which it connects via the termination (i.e. the fibers must be exactly aligned against each other), any roughness or small shift would result in significant attenuation.

It is also necessary to keep the fiber termination clean (any dirt would again cause a significant signal attenuation), so if the termination is not inserted into the plug (in the device, module, etc.), it should be covered with a cap.

For fibre optic cables we usually encounter the following terminations:

- ST (Straight Tip) – rather history, bayonet connector remotely reminiscent of BNC connectors for coaxial, nowadays it is practically not encountered. It was designed for both single-mode and multi-mode fiber, rather within local area networks.

- FC (Ferrule Connector, Fiber Channel) – nowadays also rather history. Visually ST connector, but it is screw-on, it was mostly used on single-mode threads.
- SC (Standard Connector, Square Connector) – usable for single-mode and multi-mode cables, but nowadays it is more common on multi-mode lines (up to 100 Gb/s), gradually being displaced by the smaller LC connector. However, it is still used a lot in hybrid networks for cable TV (hybrid here means simultaneously with coax).
- LC (Lucent Connector) – currently the most used, compared to SC it is about half the size. It is often found in duplex design (LC pair, one termination for each direction).
- MPO (Multi-fiber Push On) – “multiconnector” for multiple fibers (can be two, but usually 12, 24, 48 or 72 fibers).

LC is the (most used) representative of the so-called Small-Form-Factor connectors (SFF), because it is smaller than the older “common” size connectors. Its advantage is not only its small size (so both fibers needed for duplex communication fit in the same space as RJ-45), but also that it is handled similarly to RJ-45, it snaps into the plug, including a lock against pulling out.

Similarly to fiber optic cables, fiber optic connectors have their own color codes. The connectors for OM3 and OM4 multi-mode cables are the same colour as the cables, i.e. light blue (aqua), similarly OM5 cables and connectors are one colour (lime green), while connectors for (yellow) single-mode cables are dark blue. Multi-mode OM2 cables have black terminals.



Figure 2.17: Fiber optic connectors, from left FC, ST, SC and LC (always in pairs)²



Additional information

- <https://www.showmecables.com/blog/post/types-of-fiber-optic-connectors-%E2%80%93-simplex-duplex-lc-st-sc-and-more>



2.5.6 Other Ethernet Connectors and Modules

If we are crimping a shielded cable, we need terminations with a metal ring and sidewalls. The metal is used to connect to ground. In Figure 2.19, the first two (sub)pictures from the left are this case (the first one is for a category 5e cable, the second one is for a category 6 cable). In the second case it is a self-cutting connector, so we don’t need crimping pliers.

²From: <https://www.showmecables.com/>

³From: <https://dimiks.store/mpo-mtp-trunk-cable-patch-cord/>

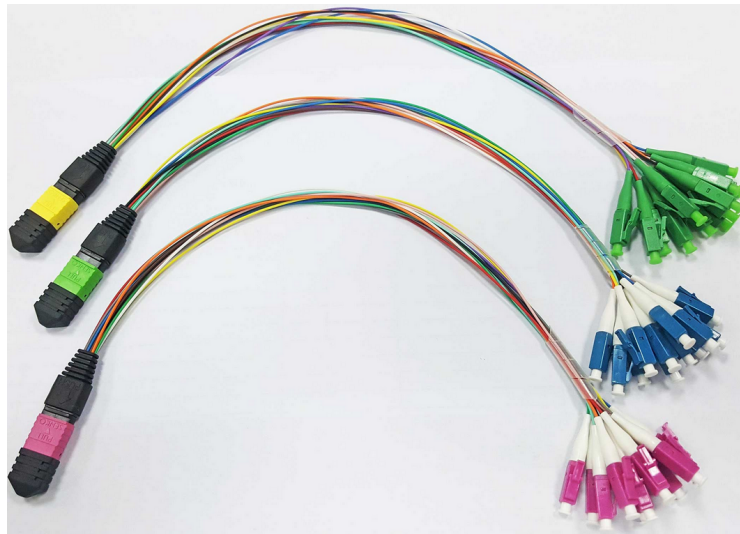



Figure 2.18: MPO-type connectors (left) led into LC connector sets³




Figure 2.19: Cat.5e shielded cable connector, Cat.6 self-cutting shielded cable connector, Cat.6 self-cutting UTP cable connector, Cat.5e female keystone mini, Cat.8.1 connectors
(From: bscom.cz)

In Figure 2.19 we can see a self-cutting connector with an extension for category 6 cable, followed by a female (i.e. the “opposite side” – regular cable connectors are of male type) of keystone type, i.e. we use a crimping tool to connect the cable just like with a socket. The last ones are the category 8.1 cable connectors.

 For cables designed for higher speeds, we can meet SFP modules (today often newer SFP+, QSFP+ and other variants). These modules are actually a removable transceiver (transmitter/receiver), which is the part of the interface that is otherwise inside the device and provides the functionality of the L1 layer portion.

Thanks to the fact that the hardware dependent part is “pulled” away from the device, it is possible to use an SFP module for metallic or fiber optic cable, or even another one, in the same socket (with some limitations). The module can either be directly on the cable, or it can have a plug for RJ-45 or some optical connector on the outside.

 Various types of modules (according to various standards) also differ in speed, for example SFP+ allows communication at speeds up to 10 Gb/s, SFP28 up to 25 Gb/s, QSFP+ up to 40 Gb/s.

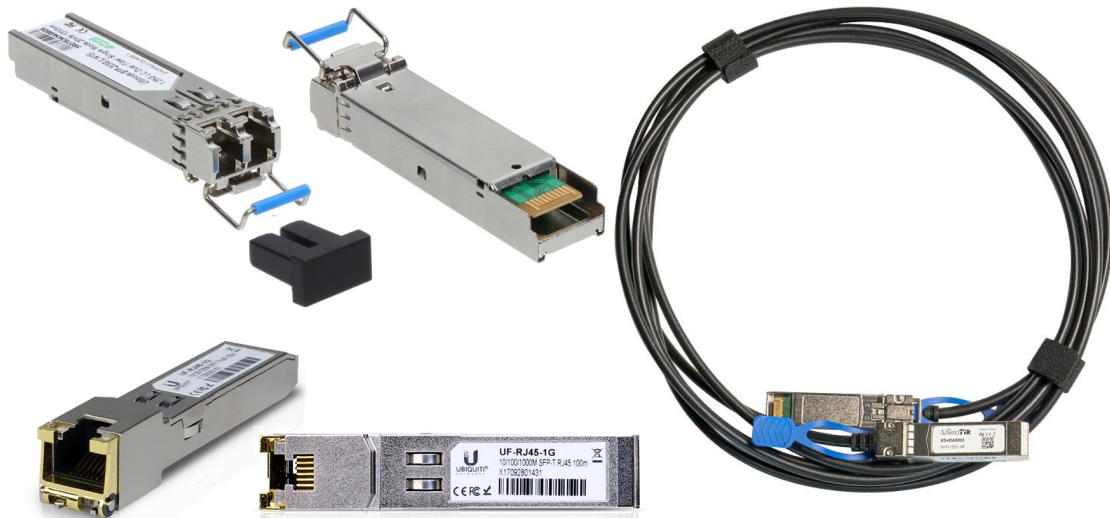



Figure 2.20: Top left optical SFP module (singlemode with full duplex, LC connector for each direction), next to the back side of the same module; below SFP module for 1G/s RJ-45 metallic cable and bottom side, right cable with SFP/SFP+/SFP28 compatible modules

From: abctech.cz

2.6 Parameters


 The vast majority of Ethernet types use baseband transmission. Other important parameters are the speed and the type of transmission medium (cable), both of which affect the encoding and multiplexing process used. At the physical layer, the following appears in the designation:

- speed: 10, 100, 1000, etc. – in Mbps, higher speeds are written with the letter G, e.g. 10G, i.e. in Gbps,
- baseband or broadband transmission,
- cable type – usually “T” means twisted pair, “F” means fiber optic, or similar.

So let’s start with history, but skip the oldest standards. We won’t cover all of them, for each speed we’ll just pick out some typical representatives.

 First *10Mb Ethernet*. All representatives allow 10 Mbps, other parameters:

- 10Base-T: baseband, the cable is UTP,
- 10Base-F: baseband, fiber-optic cable,
- 10Broad-36: broadband, coaxial cable with the impedance of 75Ω ; did not become widely spread, but became the basis for the later Ethernet used in cable TV networks (nowadays, the DOCSIS standard is used for data transmission in cable TV networks).

 We continue with *Fast Ethernet*. For all representatives the speed of 100 Mbps and baseband is used, compared to the slower predecessor the option of auto-negotiation (network interfaces can automatically agree on the parameters of the transfer) has been added. Other parameters:

- 100Base-TX: UTP cable at least Cat.5, using only two pairs (one for transmitting and one for receiving); essentially a successor to 10Base-T, it was possible to combine network cards of these types in the network,

- 100Base-FX: uses fiber optics, actually two – one for each direction. It is not compatible with 10Base-F because it uses a light signal of a different wavelength.



Gigabit Ethernet enables 1 Gbps, baseband. Other parameters:

- 1000Base-T: for UTP cable at least category 5 (minimum 5e recommended), all four pairs are used (so no weak cable with two pairs can be used), direction for each pair is determined adaptively – none is reserved for a specific direction,
- 1000Base-TX: this is a not-so-successful attempt to modify the 1000Base-T standard so that two pairs are reserved for each direction and a Category 6 cable is required; while 1000Base-T is an IEEE standard (IEEE 802.3ab), 1000Base-TX comes from TIA (TIA/EIA-854); commercially unsuccessful – cannot be used on legacy Category 5 or 5e wiring,
- 1000Base-X: a summary of standards for (mostly) optical fibers
 - 1000Base-SX (“S” for short) uses multimode optical fibers, typical range is in the hundreds of meters,
 - 1000Base-LX (“L” as long) uses optical fibers either single-mode (range in units of kilometers) or multi-mode (hundreds of meters),
 - 1000Base-CX – does not use fiber optics, but shielded twisted pair; typical range is up to 25 meters, used in data centers to connect servers to switches (for example, in some IBM products).



Remark

Very often we encounter confusion of the name – instead of 1000Base-T it is 1000Base-TX. This is usually because the “earlier generation” (i.e. Fast Ethernet) had the suffix TX. For network interfaces supporting various speeds, the designation tends to be 100/1000Base-TX, even though the latter speed is actually the “T” standard.



10 Gigabit Ethernet (10GE, 10GigE) enables 10 Gbps, baseband transmission. The CSMA/CD access method is completely eliminated, only communication in full-duplex is used.

- 10GBase-R is a group of standards for optical fibers:
 - 10GBase-SR (short range) for multimode fiber, range up to 62 m,
 - 10GBase-LR (long range) for single-mode fiber, range about 10 km,
 - 10GBase-LRM (long reach multimode) for multimode fiber, range up to 220 m,
 - 10GBase-ER (extended range) for single-mode fiber, range 40 km,
 differ not only in the cables but also in the wavelengths at which the signal is transmitted,
- 10GBase-CX4 uses a twin-ax cable, the range is only 15 meters, but on the other hand it has very good latency and the price is quite reasonable; it is used in data centers to connect servers to DCE,
- 10GBase-T is the successor of 1000Base-T, it uses twisted pair at least Cat.6; however, it is recommended to use a higher category, because it only has a maximum range of 55 m on Cat.6,
- 10GBase-W is already going beyond LANs – it adds a new WIS sublayer on top of the L1 layer, which allows communication with the WAN; it works similarly to 10GBase-R, but

encapsulates the Ethernet frame in a WIS frame, in which the data passes through the connected WAN.





Remark

So where are the standards? For example, 100Base-TX and 100base-FX (i.e. twisted pair and fiber optics for Fast Ethernet) are standardized in IEEE 802.3u, while 1000Base-T (twisted pair for Gigabit Ethernet) is in IEEE 802.3ab and Gigabit Ethernet on fiber optics is in IEEE 802.3z.

The IEEE 801.3 standard and its appendices (letters at the end) define the L1 layer and the lower part of the L2 layer (MAC sublayer), while in practice it is used rather what applies to the L1 layer (we explained earlier that on L2 we encounter rather Ethernet II frames).



 *2.5 and 5Gigabit Ethernet* were published as standards just in 2016, i.e. after “faster” siblings. These are twisted pair communications, formally 2.5GBase-T and 5GBase-T, the IEEE 802.3bz standard. The physical layer has been taken from 10GBase-T and adapted to “faster” cabling. 2.5G-Base-T can use unshielded twisted pair Cat.5e, 5GBase-T uses Cat.5e (less than 100 m distance) or Cat.6.

 *40 Gigabit Ethernet and 100 Gigabit Ethernet* (40GigE, 100GigE) prescribes fiber optic cables with a laser as a light source (at a distance of about 100 m for multimode fibers, or kilometers for single-mode fibers), a twinax cable at a distance of a few meters, and a twisted pair Cat.8 cable can be used at a distance of up to 30 m for 40GigE speed.



Remark

How is it possible to increase the speed by an order of magnitude when basically the same cables are still used (although better from generation to generation)? The increase was achieved by a combination of several methods, for example:

- by using all four pairs of twisted pair instead of just two,
- by using better quality cabling with better shielding, higher density twist and higher bandwidth,
- using better data-to-signal coding (less data is transmitted in real).



2.7 Course of Transmission

In the previous section, we learned that bidirectional transmission between two devices can take place in one of the following ways:

- in half duplex, where both devices can transmit, but alternately (only one device can transmit at a time), i.e. they share the same communication channel,
- in full duplex, where both devices can transmit at the same time, because (at least) one communication channel is reserved for each direction.

2.7.1 Half Duplex

Half-duplex mode can be used up to a maximum speed of 1 Gbps. Since the devices on the segment share a communication channel, the CSMA/CD collision method described above is used, including the Backoff algorithm for collision detection.



Definition (Collision Window, Slot Time)

The Collision Window is the amount of time a device must listen on the carrier in order to pick up another device's transmission and detect a collision hazard.



As far as the collision window is concerned, the following relationships apply:

- the largest possible distance between devices in the same collision domain (the larger the distance, the longer the device has to listen because the signal takes “long”), the larger the maximum allowed distance, and the larger the collision window must be,
- minimum frame length (the device must learn during the frame transmission that a collision has occurred in order to cancel the transmission, wait according to the Backoff algorithm and transmit again) – the smaller the minimum frame length, the smaller the collision window and the smaller the collision domain must be,
- if the collision window is small, the collision domain must also be small (short cables).

This implies that there is a very close relationship between the collision window, the size of the collision domain (the maximum distance between devices in the network), and the minimum frame length.



Remark

Consider this situation: there are two devices in the collision domain – Z_a and Z_b . The signal takes time t to get from one device to the other. At time t_a , device Z_a starts transmitting a frame whose length is such that the frame takes d_a to transmit.

If device Z_b starts transmitting before the signal from device Z_a reaches it, a collision will occur. It is now necessary for device Z_a to learn that a collision has occurred in time to cancel the transmission and retransmit after waiting. So it must know before the transmission ends, i.e. at the latest at time $t_a + d_a$. Device Z_b detects the collision at time $t_a + t$ at the earliest, and if it reacts immediately and sends a jam signal, this signal will reach device Z_a at time $t_a + 2 \cdot t$ at the earliest. So we have an inequation:

$$\begin{aligned} t_a + d_a &> t_a + 2 \cdot t \\ d_a &> 2 \cdot t \end{aligned}$$

This means that the minimum time it takes to transmit one frame must be greater than the time it takes for the signal to travel there and back between the two furthest devices on the segment (in the collision domain). Therefore, if we allow too large a collision domain (= we want too long cables), we must specify a higher minimum frame length.




So here we balance between two requirements:

- We want to cover as much space as possible, so long cables and a large collision domain are desirable.

- We don't want to overload the line unnecessarily in case we need to send also small frames (if we need to send less data than the minimum length limit, we have to add “padding” – data without meaning), so it is more efficient to have a low limit for the minimum length.

There is a third factor – the transmission speed. If we only increase the transfer rate, we have to increase the minimum frame length limit or decrease the domain, because the increase in speed will affect the transmission time.

Let's focus on different speeds – 10 Mbps, 100 Mbps, 1000 Mbps (CSMA/CD is no longer used for the faster ones).

 For 10 Mbps, the minimum frame length without preamble is set to 512 bits = 64 octets (see the Ethernet II frame structure on page 46), so it takes 512 μs to transmit 512 bits (that's the time d_a). The maximum time it can take to travel between the two farthest devices in the collision domain is half of this value. The distance in meters then depends on the transmission medium chosen (metallic and optical cables propagate the signal at different speeds).

If the minimum length of a MAC frame without a preamble is determined to be 64 octets, then what is encapsulated inside should be at least $64 - 18 = 46$ octets long (subtract the length of the header and trailer of the MAC frame).

There may be a situation where less than 46 octets of data should be transferred within a frame. The IEEE 802.3 standard, in cooperation with LLC and SNAP, handles this with a “pad” added after the data, where the pad length can be derived from the value in the *Length* field: if this field contains a number < 46 , then:

$$\begin{aligned} \text{DSAP} + \text{SSAP} + \text{Control field} + \text{Data} + \text{Pad} &= 46 \\ \text{Length} + \text{Pad} &= 46 \\ \text{Pad} &= 46 - \text{Length} \end{aligned}$$

For the case indicated in Figure 2.21, the value $\text{Pad} = 46 - (28 + 1 + 1 + 1) = 15$. This is important to show which octet of the frame the checksum field starts on. Note that the *Length* field also includes the length of the LLC header (shown in yellow).

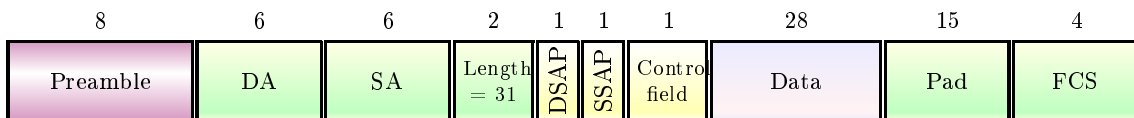




Figure 2.21: Very short LLC frame inside MAC frame according IEEE 802.3

Ethernet II frames that are too short also contain padding. This is found, for example, in ARP frames encapsulated in Ethernet II frames. It has a certain structure (a random number and a calculated hash that makes it detectable). Wireshark tuto vycpávku často ukazuje jako sekvenci nul.

 When upgrading to a higher speed, i.e. to 100 Mbps, it was decided to keep the minimum frame length (i.e. the same applies for the frame format as in the previous text) and the size of the collision window, i.e. it was necessary to reduce the size of the collision domain between DCEs (approximately 10×).

On a metallic cable this meant reducing the collision domain from 2000 m to 200 m, which

means that the distance between the DTE and the DCE is max. 100 m (this remained) and between two DCEs max. 200 m.

 When switching to Gigabit Ethernet (1 Gb/s) this solution was not possible (20 m as the maximum possible distance would be really small), so the minimum frame length without preamble counting was increased to 512 octets (4096 bits). Since smaller amounts of data are often transmitted, it is necessary to add an additional “padding” (Carrier Extension, a non-data extension) after the checksum for frames below the limit. The location is indicated in Figure 2.22 (the field on the far right).

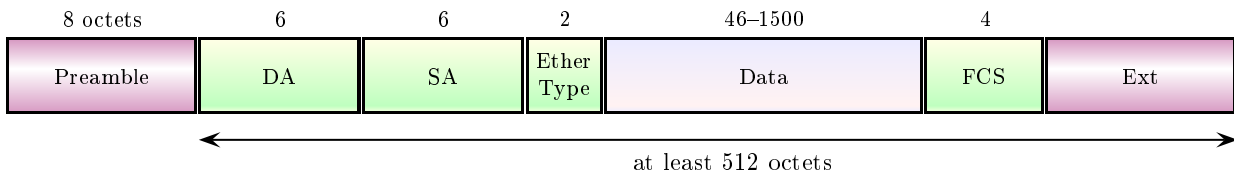



Figure 2.22: Carrier Extension inside a Gigabit Ethernet frame

The Carrier Extension is added on send after the checksum is calculated on the MAC sublayer, and is removed on the MAC sublayer of the destination system when the frame is received before the checksum is checked. The composition of symbols inside this suffix is proprietary – it must be distinguishable from the checksum field.

 Patent details are available at <https://patents.google.com/patent/US5940401>.

If a large number of frames with a small amount of encapsulated data were sent very frequently, the link would obviously be unnecessarily overloaded. Such a case is handled differently:

 *Burst mode* is intended for sending sequences of short frames. It is only used for transmissions at speeds from 1 Gb/s upwards, i.e. from Gigabit Ethernet. “Short” frames are sent in a burst that has the following structure:

- the first frame in the cluster has the structure described above, including the above mentioned Carrier Extension if it is shorter than 512 octets,
- followed by *IFG* (InterFrame Gap) – a special sequence of bits filling the space between frames, used for recognizable separation of frames in the cluster, has a similar role as the preamble at the beginning of the frame,
- further frames in the sequence separated by IFG sequences; if they are shorter than 512 octets, it is not necessary to add the Carrier Extension.

The total length of the burst should not exceed approximately 5.4 times the maximum possible frame length (8192 octets to be precise), but if this limit is exceeded while the last frame in the burst is being sent, the frame may still be sent and the burst terminated afterwards.

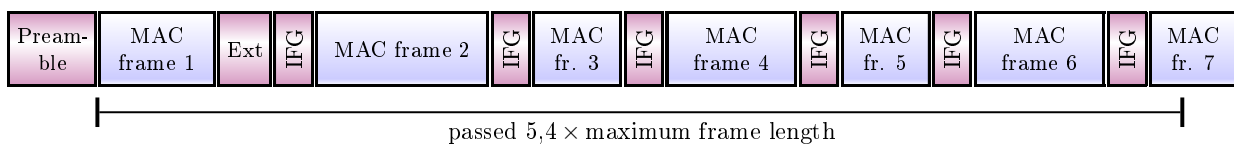


Figure 2.23: Frames in the burst mode

**Remark**

This means that during burst mode transmission, we need to monitor whether we are approaching the limit of 5.4 times the maximum frame length. If we reach it, then the frame we are currently sending is the last frame of the burst. This is important because while we are sending a burst of frames we are actually blocking the link and preventing other devices in the segment (in the same collision domain) from sending, so the blocking cannot last too long.



Calculations regarding minimum frame length, collision domain size, etc. will not apply to the exam...

2.7.2 Full Duplex

In a full duplex, the situation is much simpler. At the logical level, we are basically dealing only with point-to-point connections and all connections are actually dedicated, the device can transmit at any time. The CSMA/CD access method is not used and the link parameters are not limited by the properties of the collision domain and collision window, only by the physical properties of the transmission path (e.g. attenuation).

The transmission can practically still be done in burst mode as described for half duplex. Transmission and receiving take a different path, but may be on a single common cable (for example, in many physical layer standards, for UTP with four pairs of twisted pair, one pair is always dedicated to receive and the other pair to transmit, or the direction is determined adaptively). For fibre optic cables, there is either one cable for each direction, or more likely there is one or more fibres in the cable separately for different directions.



The only problem (which can occur even with half duplex, but is typical for full duplex) occurs when the receiver cannot receive and process incoming frames – the processor is not up to speed or there is not enough memory available. This problem is solved by a combination of the following methods:

- the device has a large enough *buffer*, it stores there what could not be processed yet,
- The sender needs to be slowed down, told to wait a certain amount of time before transmitting.

The first option is quite clear, but may not be sufficient. The second option is to use a *pause frame*, a special frame that is sent to the “hardworking” sender; this frame mainly contains information about how long to pause the transmission. The pause frame with time value 0 would mean “you can start sending immediately”.

**Remark**

Since 10G Ethernet only full duplex is used, half duplex is not supported. Collisions cannot occur in principle, so there is no point in using any collision method (thus CSMA/CD is not used in this case).




2.8 Ethernet-related Technologies

2.8.1 Autonegotiation

In order to be able to operate the transmission over the Ethernet network, it is necessary that the network interfaces of the communicating devices “understand”, i.e. agree on the following parameters:

- transmission speed,
- a specific standard for a given speed if there are multiple options for the transmission medium used (for example, to make it clear how particular UTP pairs will be used),
- full or half duplex.

It is also necessary to regularly check the link operability.


 If we connect two devices with a cable and make the corresponding network interfaces operational, then these network interfaces first agree on the above parameters – *autonegotiation* is performed (i.e. automatic agreement on parameters) and then they periodically echo when no frames are transmitted (to make it clear that the device is still connected and the link is working). Autonegotiation is performed at the physical layer, and is described directly in the IEEE 802.3 standard and amendments.


The first attempts at autonegotiation (but not yet in the full sense of the word) were with Ethernet 10Base-T, when the network interface sent pulses at regular intervals to determine whether the link was functional. This signal was called *NLP* (Normal Link Pulse). If no frame or NLP signal was received from the device for a certain period of time, the link was considered to be down.


Nowadays, in the era of higher speeds, the *FLP* (Fast Link Pulse) signal is sent, where a series of the original NLP signals are intermixed with additional information (as more parameters were gradually added and could be announced), and while originally this was an optional functionality, since Gigabit Ethernet, autonegotiation is mandatory for metallic cables.

The whole mechanism is backward compatible, and in real life the highest possible parameters are used for the link from those that are met by both communicating sides. For example, if one device can handle 100 Mbps half-duplex and the other can handle 1 Gbps full-duplex, the 100 Mbps and half-duplex will be used.

2.8.2 Power over Ethernet


 *PoE* (Power over Ethernet) is a standard describing the possibility of powering smaller devices via a network (metallic) cable. The advantage is that we do not need to bring a power cable to the device in question, data and power are transmitted over a single Ethernet cable. Of course, such a powered device has to get by with lower power consumption, typically e.g. IP phones, IP cameras and some Wi-fi access points connected by twisted pair cable to a device that can act as a PoE source (which is usually a switch).

 The first PoE standard IEEE 802.3af is from 2003. Soon a new version of IEEE 802.3at came and we are currently seeing another version IEEE 802.3bt.

 PoE distinguishes between two parties: one side is the target of the power (PD, Powered Device) and the other side provides the power (PSE, Power Sourcing Equipment), which is usually a switch.

A switch serving as a PoE source must have a sufficiently dimensioned power supply (which is the component that transforms AC voltage from the power grid to DC voltage, just like a regular computer), and even then it usually cannot power all the devices on its ports, the remaining connected devices must have their own power supply.

According to older standards, in a twisted pair cable (with 4 pairs of wires), two pairs were used for data and the remaining two pairs were used for PoE power. But as we know, from 1 Gbps speed upwards, all four pairs are used for data. Therefore, according to the new IEEE 802.3bt standard, these four pairs are shared in the case of PoE, i.e. both data and power are transmitted over the four pairs.

 We distinguish *power classes* 0–4 (older standards) or 0–8 (new standard) according to the current and maximum power required by the device, with class 0 being for non-PoE devices. The higher the power, the higher the class.

Similarly to how the connection parameters are set, the speed and duplex are negotiated in autonegotiation, while in PoE the class is determined when connecting. First, a detection process is performed where a current is applied to the connected device at a low voltage that should not harm the device if it is not PoE-capable (corresponding to class 0). If the device responds, detection continues by determining the power class.



Additional information

https://www.arubanetworks.com/techdocs/AOS-CX/10.11/HTML/monitoring_6200/Content/Chp_PoE/pove-eth-oveview.htm



2.9 Structured Cabling


2.9.1 Rack

Where to put switches, routers, servers and similar devices? If they are high-performance devices for the corporate sphere, we place them in a rack, which is an enclosure that provides the devices with mounting, controlled cooling, wiring and connectivity. There are different types of racks – wall-mounted, rack-mounted, fully enclosed or open frame.

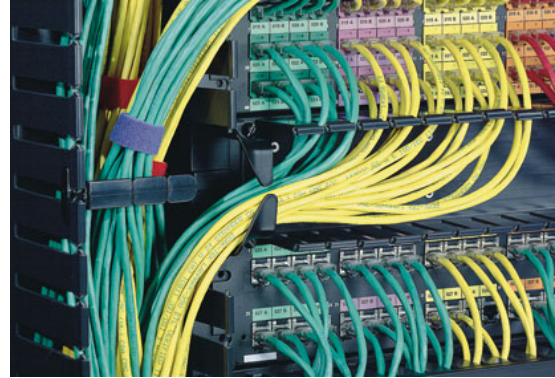
The dimensions are standardised, or there are several possible dimensions (width and depth). Inner widths are always given – 10", 19", 21", 23", 19" being the most common. The internal depth can also vary, with 600 or 800 mm being common. If the rack is deeper than necessary, it doesn't matter, usually the internal mounting mechanism can be adjusted (the opposite is of course already a problem).

The height of the equipment placed in the cabinet is given in standardized units called U (unit). The relationship $1\text{ U} = 1,75'' = 4,4\text{ cm}$ applies. LAN switches are usually 1U in height, with greater heights being more common for other devices. For example, a 2U device is almost 9 cm high.

The rack height is also specified in these units, for example, a rack with a height of 18U can accommodate various devices with heights that add up to 18U. However, it is not advisable to overcrowd the rack, especially when there are powerful devices inside that require good cooling.

 **Tasks**


1. Take a look at the following two pictures. Something is wrong in one of them. What is it?

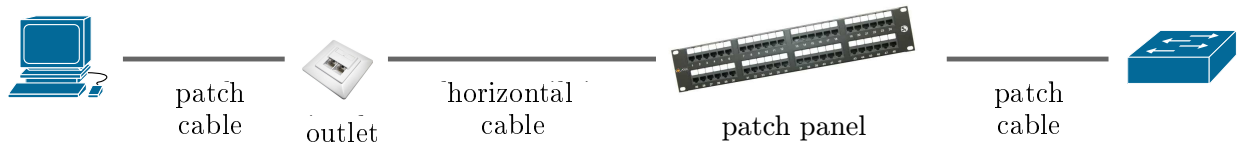



(hint: it's not the colours, although colours can help a lot with clarity)




2.9.2 Horizontal Cabling

 *Horizontal cabling* are named according to the fact that they run more or less horizontally; this is the name given to cabling connecting equipment within a single floor of a building. The usual structure of horizontal cabling is as follows:



-  As for the length of the individual parts of the path, the following applies:
- The entire physical transmission path from the host on the left side to the switch (or other active network equipment from the L2 layer above) on the right side must be no more than 100 m long.
 - The physical length of the horizontal cable (from the outlet to the patch panel) is a maximum of 90 m.
 - The length of the patch cable between the host and the outlet shall be a maximum of 20 m.
 - The length of other patch cables (including between patch panel and switch) is a maximum of 5 m.

If we sum the values from the second point on the list onwards, we get more than 100 m, but the limit is of course valid. So if we're approaching the upper limit at any point, we have to reduce elsewhere. In practice, we may see longer paths: if all the components of the path are of sufficient quality, this may work, although it is not recommended.

 **Cabling.** We've learned quite a lot about cables, the only thing left is the difference between patch cords and cables used in the wall for horizontal cabling. In most cases, twisted pair is used for both, but for each of these two cases the cable needs slightly different properties:

- *Stranded type cable* – for patch cables. The conductive core of the conductors is a bundle of stranded thin copper wires wrapped with insulation, it has a higher electrical resistance. The cable is more flexible, easier to bend, but there is a greater risk of damage or breakage of the conductors.
- *Solid type cable* – for horizontal cabling (inside the wall). The conductive core of the conductors is made of solid thicker wire. The cable is harder to bend, but more durable.



Remark

If you see the CCA (Copper Clad Aluminium) abbreviation in the specification or directly on the cable, it is better to avoid such a cable. The conductors in this cable are made of aluminum and coated with a thin layer of copper. Although they are cheaper (by quite a bit), they have poorer conductivity and cannot be installed at the distances that are common for all-copper cables.

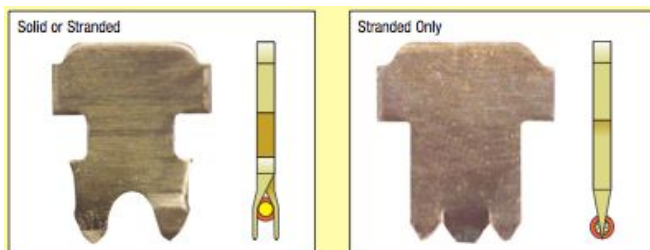


Example

The cable type and other data can be found directly printed on the outer insulation of the cable. For example:

“...4P 24AWG UTP Patch Cable ISO/IEC 11801 and TIA/EIA 568 ... verified CAT5e...”

which means that it is a 4-pair cable with 24AWG conductors (i.e. rather weaker – the smaller the number before AWG, the larger the diameter), unshielded patch cable (for connection to a host or switch) meeting the given standards, category 5e.



In fact, these two types of cables are also crimped a bit differently, respectively there are 8p8c connectors of the stranded type and solid type. They differ in how the pin in the connector is connected to the wire in the cable.

From: <https://www.brucetambling.com>


In the *solid* type connector, the pin knives “embrace” the conductor, whereby the insulation of the conductor is broken by the knife blades. In a *stranded* connector, the pin knife cuts into the conductor, which is not a problem because the conductor in such a cable is actually a tangle of thin copper wires. The most common blades in the solid type and stranded type connectors are shown in the picture on the left.



Remark

It follows that if we try to attach a stranded-type connector to a solid-type cable, the blades at the pins will either break or bend. Either the contact will not work at all or it will stop working after a certain period of time. The reverse is not usually a problem (a solid-type connector to a stranded-type cable). To avoid confusion, it is always better to have the same type of connector as the cable.



 **Patch panel.** As seen above, an important part of the horizontal wiring is the patch panel. We can think of it as a forward set of ports for a “hidden” or harder to reach switch, and we also usually mount the patch panel in a rack, typical height is 1U.




If we need to change the wiring for a specific horizontal cable leading from a specific socket, we do not need to make a change on the switch, just rearrange the cable on the patch panel. From: <http://www.excel-networking.com>

As we can see in the picture, there are usually ports on the front side of the patch panel for inserting the 8p8c connectors of horizontal cables, while on the other side we can find terminals similar to those in the outlets. So the patch cable leading from the switch does not go through the connector, but we insert the wires into the sockets on the terminal block at the specific patch panel port. This is an advantage because it takes less time to clip a large number of wires into the terminal block than it does to crimp the connectors.

There are also patch panels that allow the use of connectors on both sides. Another option is unmounted patch panels, which require modules to be supplied into which connectors or wires are fitted (one module is therefore a connection of an RJ-45 outlet on one side and a terminal block on the other side). The advantage of these is that we can only fit as many positions as we really need, and the remaining space can be used for airflow during cooling. Alternatively, we can combine different types of modules.


2.9.3 Backbone Networks


So far, we have solved the horizontal cabling – within one floor of the building. But that’s not enough, we need to connect the floors of the whole building and the individual buildings on the campus.

 There should be a well-secured area on each floor of the building (ideally a dedicated room if possible) where the horizontal cabling from all outlets is routed. If this is a separate room, we call it the *telecommunications room* or *network room* of the floor. It’s where

- *floor distributor* (FD) – the rack where the horizontal cables from the whole floor go,
- other resources serving the needs of the staff on that floor (for example, a local server).

Horizontal cabling more or less maintains the star topology, with the FD at the center of the star.

 *Building backbone* connects all floors of the building, respectively all FDs. If we cut off the horizontal cabling from the FDs, then the building backbone also has essentially a star-type physical topology, with *building distributor* (BD) at the center. If we add the horizontal cabling, we get a tree-type topology (with two floors for now), with BD at the root of the tree.

 The term *campus* refers to a group of buildings that belong together in a certain way (for example, a campus belonging to one company) and therefore need to be networked together. The *Campus distributor* (CD) is a rack covering the network in the whole campus, the individual BDs

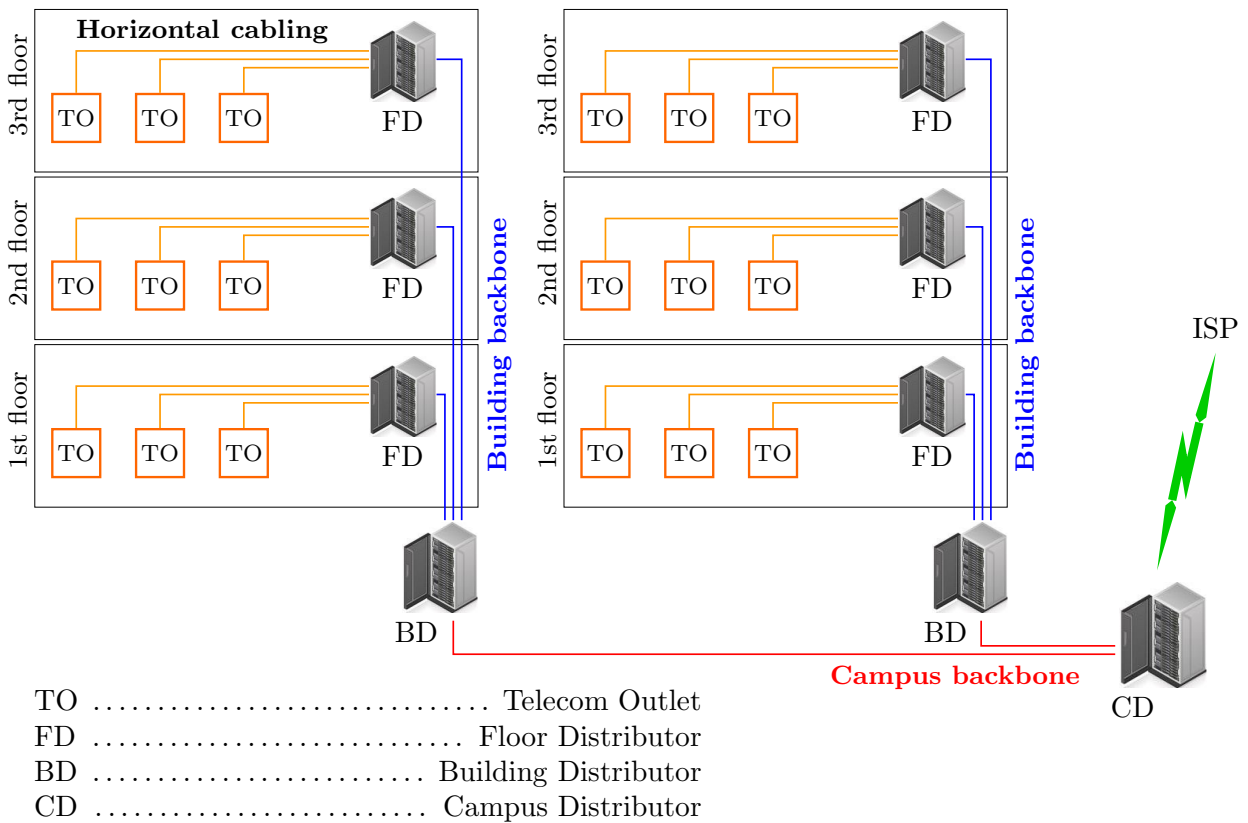


Figure 2.24: Schematic diagram of structured cabling

are connected to the CD by *Campus backbone*. It is through the CD that all communication with the Internet Service Provider (ISP) for the company goes.

While metallic cables (twisted pair) are usually used for horizontal cabling, backbone cabling is more likely to use optics (the higher the level, the more likely). Especially for the campus backbone, fibre optics is also suitable because we can lead the fibre optic cable over much greater distances, we are not limited to just 100 metres for the entire line.

It is also important to note that there are certain requirements for active network devices at the top level – in a tree topology there is a relatively large data flow through the root device, so this device is expected to have high performance, the least amount of filtering delays (active network devices deeper in the tree can do this), and also failover security – a redundant (secondary, spare) device that will take over the role of the primary device in case of failure.

Remark


If it is a smaller area, then some parts can be grouped together. For example, if we have to group only one building, it is not necessary to distinguish between BD and CD, or for a house with several outlets per floor and several floors, there is not necessarily a rack for each floor.

2.9.4 Structured Cabling as a Whole

The concept of “structured cabling” is far from just about how individual devices should be connected. It is a complex system involving


- requirements and recommendations for the layering of the entire network,
- passive equipment (not just cables, but also for example outlets and patch panels), cabling for specific parts of the network,
- requirements for wiring, redundant power supply (in case of failure) – UPS, overvoltage protection at different levels,
- mechanical security (e.g. against intrusion by unauthorised persons),
- fire and smoke protection, fire regulations, etc.


The network may include a branch telephone exchange (PBX, for the internal telephone network) and a connection to the nearest telephone exchange, through which it is also possible to access the Internet.


 Among the international de-facto standards, ANSI/EIA-568-C deals with structured cabling (it is definitely not just about colour codes for cable conductors), while the safety aspects of structured cabling are part of the ISO 27000 group of standards.

Chapter 3

Additional topics on local area networks

 *Quick preview:* In this chapter, you will find topics that complement the Ethernet issues. We will discuss the principle of VLANs (virtual local area networks), look at solving the problem of redundant physical paths in a switch network and the STP protocol, and give a brief overview of the IEEE 802 family of standards for reference.

 *Keywords:* VLAN, access port, trunk, native VLAN, management VLAN, IEEE 802.1Q, STP, IEEE 802.1D, convergence, designated port, root port, blocked port, RSTP, MSTP.

 *Objectives:* After studying this chapter, you will know what a VLAN is, how to partition a network at other than the physical layer, and how to provide communication within a VLAN and between different VLANs. You will also learn how to solve the problem of redundant paths in a switch network.

3.1 VLAN


A VLAN (Virtual LAN) is a logical grouping of certain resources (such as computers) located on one or more regular local area networks. Think of it this way: we have a large network that we divide into subnets. Each subnet at layer L3 corresponds to exactly one VLAN at layer L2. Typically, each machine belongs to just one VLAN and communication between different VLANs is limited. In other words – the VLAN determines who is allowed to communicate freely with whom. Remember the difference between physical and logical topology. Physical = how devices are connected, logical = how they communicate with each other.

Why do something like that? For example, for the following reasons:

- Security – We have the ability to restrict access to a specific resource (such as a server with confidential company information) to only someone (members of one specific VLAN).
- The restriction on communication between VLANs also applies to broadcast transmissions, so we have a tool operating (mostly) at the L2 layer that can provide separation of broadcast domains – it wouldn't be possible to do this except with VLANs.

Both could be done differently – by adding routers to the network. But routers have their draw-

backs: they are more expensive and much slower than switches, and routers actually work “outside Ethernet” – at the L3 layer.

 We use the following terms:

- *data* VLAN – VLANs for normal traffic, we assign devices (actually ports) to them,
- *default* – if nothing else is configured on the port, there is a default VLAN (for most manufacturers this is VLAN number 1),
- *native* – if something is transmitted between two switches that has no VLAN set, it will fall into “channel” native VLAN (e.g. STP frames are also transmitted over native VLANs); if no native VLAN is set manually in the configuration, VLAN 1 will be used as native,
- *management* – is for frame management (SSH, Syslog, SNMP,...), the default setting is VLAN 1, but it is definitely a good idea to change it.

 The specific port on the switch is

- *access* – there is one device directly behind it, it is set to belong to a single VLAN,
- *trunk* – behind it is another switch (potentially a number of different devices belonging to different VLANs), the neighboring switch also has the corresponding port set as a trunk port.

Over the access port we transmit only frames belonging to the VLAN set on it, over the the trunk port it is possible to transmit frames belonging to several different VLANs (the list of “allowed” VLANs is set, the list must be the same on both ends of the trunk). If we do not set a list of allowed VLANs, all VLANs are allowed.

3.1.1 VLANs on a single switch

The principle of VLANs is simple – we divide the devices on the network into groups, assign an identifier (VLAN number) to each group, and ensure that communications between these groups are separated. Figure 3.1 shows what this might look like (somewhat simplified).

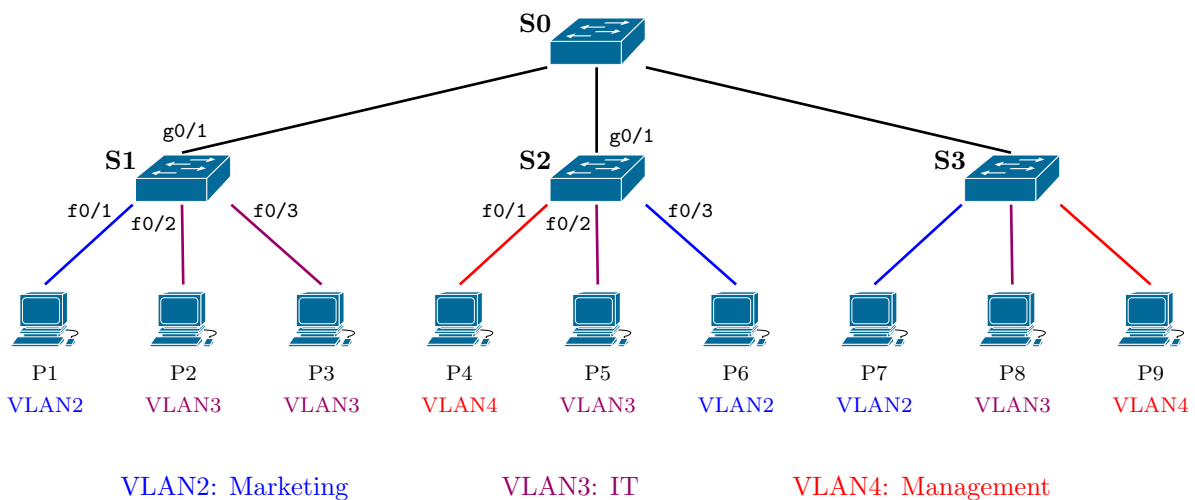



Figure 3.1: Example of VLAN usage

<i>MAC address</i>	<i>Port</i>	<i>VLAN</i>
P1-addr	f0/1	2
P2-addr	f0/2	3
P3-addr	f0/3	3
...		
P5-addr	g0/1	3
P6-addr	g0/1	3
...		

Table 3.1: Example of MAC address table with VLANs (Switch S1)

Each port on the switch specifies to which VLAN the device accessible through that port belongs. So in our case, on switch S1 the first port is assigned to VLAN2 and the other two ports to VLAN3. The port leading up to switch S0 will carry frames from different VLANs.

 We need a MAC address table on each switch. Since switching is only supposed to work between devices belonging to the same VLAN, each MAC address and port must have VLAN number information added to it. The switching will then work as follows:

- The switch receives a frame on a given port, and that port belongs to a specific VLAN.
- In the MAC address table, it focuses only on entries belonging to the same VLAN, it does not check others (more accurately: they'll go to the trunk).
- Among these “filtered” entries, it finds the one that matches the address of the target device.
- It locates the port in the found record and switches (sends) the frame to this port.

Simplistically, we can imagine that there is a separate MAC address table for each VLAN on the switch (in reality this is not the case, because then we would have nowhere to include those devices that do not belong to the regular “communication” VLANs).

As long as the communication stays within a single switch, switching frames is quite simple. For example, if computer P2 sends a frame to computer P3, switch S1 receives the frame on the “violet” port (VLAN3) and therefore follows the rows of the table for VLAN3, which directly indicates on which port the destination (computer P3) is available. So in real life it works the same as without virtualization.

As for broadcast frames, they are propagated only within the VLAN in which they were sent.



Remark

Be mindful that the end devices are completely unaware of any virtualization. When they send a frame, they only know which MAC address it should go to, but they don't know the VLAN number, this number is not even part of the frame being sent (yet). Only the switch can determine the VLAN, because the switch keeps a record of the VLAN to which the port (and the device connected to it) belongs for each port.

The MAC address of the destination can be found, but only for devices in the same (sub)network, so even in this way the sender can only directly access the addresses of devices in the same VLAN.




3.1.2 VLAN frames on the path through the trunk

We also need a mechanism that allows us to properly switch frames that are to go from source to destination through more than one switch. As long as we stayed within a single switch, this was simple – the switch determined the VLAN by the port the frame came from, and thus the specific table rows to look up the address and port of the destination device.

However, if the destination is connected to another switch, then the frame must first be sent to that switch, which then ensures that it is forwarded to the destination. But how do you pass the information about which VLAN the source belongs to to the “that other” switch so that the next switch knows which lines to look at?

Let’s focus again on Figure 3.1. Suppose computer P1 sends a frame to computer P6. Somehow this must work, because both computers belong to VLAN2.


Switch S1 receives the frame from computer P1, finds out from the table that it should forward it to switch S0, but it must *add VLAN number information*. Switch S0 receives the frame including this additional information and forwards both to switch S2. Only when switch S2 realizes that it can forward the frame directly to the port with the assigned VLAN number, and thus use the additional VLAN number information (to determine the address table), but will only send the frame to the destination port without the additional VLAN information.

 The link that we use to send frames belonging to different VLANs is called *trunk*. It is usually between two switches or between a switch and a router, rarely between a switch and a server. Hence, switches S1–S3 have both types of ports – access and trunk.

In Figure 3.1, the links leading to the access ports are shown in color (according to the VLAN colors), while the trunk ports are shown in black. Only the trunk links carry additional VLAN information.

Now for the additional information about VLANs. We’ve already established that when communicating within a single switch (i.e., only through access ports), this is not needed (in fact, if a frame with VLAN information comes in through an access port, it will be discarded because someone is probably trying to “sneak” it into a VLAN it doesn’t belong in), but when communicating through a trunk port, this information is necessary because the next switch along the path cannot know which access port (and therefore VLAN) the frame originally came from.

There are two approaches – the first one changes the frame header (adds VLAN information), whereas the second approach does not change the frame and instead packs it into a special frame with VLAN information.

 The first method is according to the *IEEE 802.1Q* standard. According to this standard, the VLAN information is inserted directly into the header of the transmitted frame, thus

- would make it clear that this is a VLAN frame, and
- no original entries are lost from the header.

So we keep all the original fields, but add two fields with VLAN information. In Figure 3.2 we can see where these two fields are inserted – right after the addresses.

The content of the new fields is as follows:

- EtherType value for VLAN (switch recognizes by this that the delivered frame contains VLAN information). The value for VLAN is 0×8100 .

- VLAN tag, which includes
 - IEEE 802.1p priority (3 bits), usually 0 (this means Best Effort – no priority, “nothing guarantees but promises to do its best to deliver”),
 - indication of the canonical form of the address (1 bit), usually 0,
 - VLAN number (12 bits).

Because this changes the frame header, the checksum needs to be recalculated, and therefore the contents of the last field of the entire frame (FCS) change.

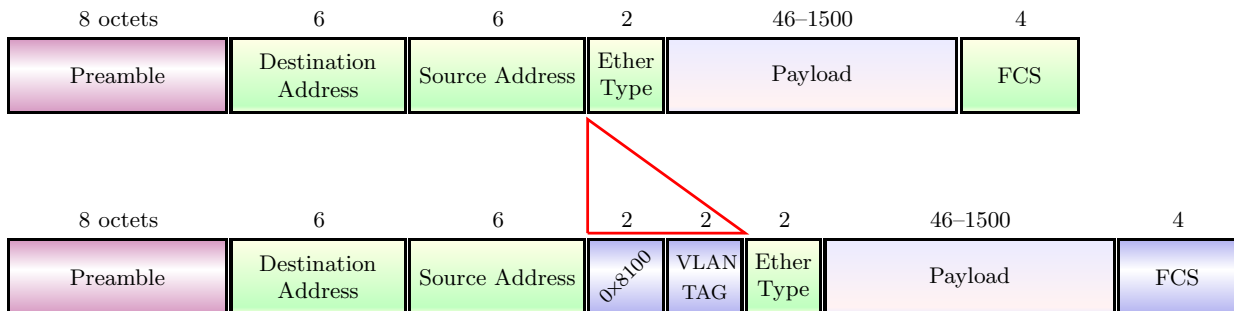


Figure 3.2: VLAN frame according to IEEE 802.1Q


If a switch (for example switch S2 as shown in 3.1) receives a frame on a trunk port, it checks the EtherType field. If it finds the number 0x8100 there, it is clear that it is a VLAN frame, and the next field will contain the identifier of the VLAN to which the sender belongs.


 The switch is supposed to transfer frames between ports according to the MAC address table.

- The switch receives a frame (without a VLAN ID) on the access port, and determines the VLAN by port. If it determines from the table (by filtering the rows by VLAN) that the destination will not be another access port, it adds both VLAN-related fields to the header, calculates a checksum, and sends the completed VLAN frame to the trunk port.
- If the switch receives a frame on a trunk port and determines that the destination is available on an access port (it has determined the VLAN from the frame header), it removes both VLAN fields from the header, calculates a checksum, and sends it to the appropriate access port.
- If we have switches in a tree hierarchy, then the top levels usually work only in native mode (trunks only), i.e. they do not check VLAN IDs – so that the backbone does not delay traffic unnecessarily. So such a switch receives the frame, reads the destination address, and simply switches to the appropriate port.

 **Remark**

The terms *access port* and *trunk port* belong to Cisco terminology. Other manufacturers have different terminology (but otherwise it works exactly the same, according to IEEE 802.1Q), for example on HP devices we talk about *untagged member* and *tagged member*, or *multiVLAN interface*.

The access port (or untagged member) connects a device that doesn't understand the VLAN (regular computer, server, etc.), while the trunk port (or tagged member) connects a device that understands the VLAN (typically a switch) and can work with VLAN frames. 

 The second approach to implementing VLAN frames is to not interfere with the original frame, but to encapsulate it in a special frame according to a specific protocol – adding a new header containing, among other things, the VLAN identification. Previously, Cisco devices used the *ISL* protocol in this way, other manufacturers also had their own proprietary solutions. These protocols were developed when IEEE 802.1Q did not exist, and there was a reason for using them then.

Currently, only IEEE 802.1Q is used to identify VLANs on trunk ports, so we won't even bother with other protocols.

3.1.3 Inter-VLAN Routing

However, we want to separate these subnetworks from each other, but not completely. We would like to keep some possibility of communication between different VLANs. This is exactly what the router in the picture connected to the root switch can do. In general – communication between different VLANs can only be provided by a device operating at the L3 layer, so one of them:

- router,
- multilayer switch (with the L3 functionality).

In the following, we will mostly mention and show the router for simplicity.

When communicating between devices from different VLANs, the frame is sent through the L3 layer device. In this case, if computer P1 from “blue” VLAN2 sends a frame to computer P5 from “violet” VLAN3, the network path will be as follows:

$$P1 - S1 - S0 - R - S0 - S2 - P4$$

while some testing and routing is done on the router, and the VLAN identifier in the frame is also changed (in fact, on any L3 device, the original L2 header is always removed and a new one is created). The packet nested in the frame is routed from the subnet intended for the source VLAN to the subnet intended for the destination VLAN, which can only be done at the L3 layer.

However, it is important to note that any communication between devices from different VLANs will indeed go through the L3 layer device, with all the consequences including extending the frame path in the network and possibly reducing the link throughput in the network (even if the source and destination are connected to the same switch).

We need an L3 device that has at least as many interfaces (physical or virtual) as there are subnets. Each VLAN will route to its own subnet on L3, it's just a question of whether the path will go through physical or virtual interfaces.

A primitive solution would be to use physical interfaces. But that would mean running as many cables between the router and some switch as there are VLANs, in Figure 3.3 we see this in the situation on the right. There are only three cables, so three subnets/VLANs. Imagine if you needed a few dozen VLANs...

There are two better options (using virtual interfaces) to provide a VLAN connection to L3:

1. *Router-on-a-Stick*: We need a router. A single cable connects one of the switches to the router. From the switch's point of view it will be a trunk, from the router's point of view it will be an interface divided into (virtual) subinterfaces for different (virtual) subnetworks.

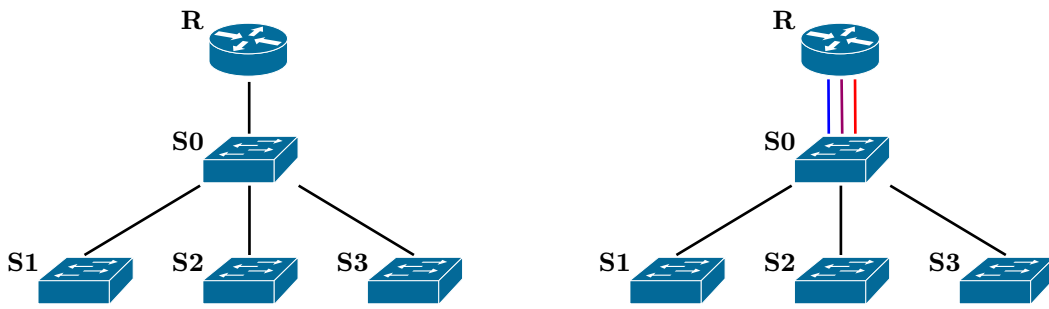


Figure 3.3: L3 device and VLAN

2. *L3 switch with internal SVI virtual interfaces*: SVI is the L3 interface inside the switch. For each subnet/VLAN, we create its own SVI with an address for that subnet, and we enable routing (between these subnets) inside the L3 switch.

In the first case we have virtual paths inside the shared physical path, in the second case it is a virtual interface leading inside the device (L3 switch) to the routing module, i.e. it does not leave the original physical infrastructure.

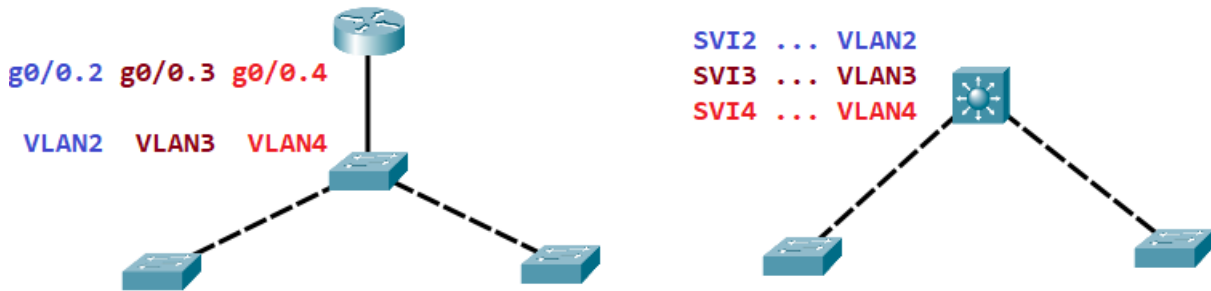


Figure 3.4: Two commonly used types of routing between VLANs

✎ Router-on-a-stick. Between the central switch and the router (someone that “understands” IEEE 802.1Q) there is one link, we connect them *from the switch’s point of view* with a trunk link, *on the router’s side* we create as many virtual subinterfaces on the given network interface as many VLANs will be able to pass through the trunk from the switch. Each VLAN will have its own subinterface (a virtual channel inside the shared link) with its own label and IP address. Subinterfaces are created using the procedure described in the IEEE 802.1Q standard. The situation is indicated in Figure 3.4 on the left.

Specifically – on the switch we configure this port as a trunk port, on the router on the other side of the cable it is a network interface with defined subinterfaces (for example for port `g0/0` the subinterfaces will be `g0/0.1`, `g0/0.2`, . . . , `g0/0.10`, . . . as appropriate, the number of the VLAN after the dot is the number of the VLAN and these numbers are not necessarily sequential). E.g. the subinterface `g0/0.10` has assigned the IP address belonging to the subnet we have designated for VLAN 10.

🔗 Example

Suppose we have port `g0/1` on router R1 connected to some L2 switch (there is also port `g0/1` on its side).

On the switch we create VLANs and configure trunk. In our case we will have VLANs 10, 20 and 30:


```
S1(config)#vlan 10
S1(config-vlan)#name marketing
S1(config-vlan)#vlan 20
S1(config-vlan)#name sales
S1(config-vlan)#vlan 30
S1(config-vlan)#name management
S1(config-vlan)#int g0/1
S1(config-if)#switchport mode trunk
S1(config-if)#switchport trunk native vlan 90
```

We also set up other trunks or access ports. On the router, we do the following (assuming that port g0/1 is currently switched off):

```
R1(config)#int g0/1.10
R1(config-subif)#encapsulation dot1q 10
R1(config-subif)#ip addr 10.10.0.1 255.255.0.0
R1(config-subif)#int g0/1.20
R1(config-subif)#encapsulation dot1q 20
R1(config-subif)#ip addr 10.20.0.1 255.255.0.0
R1(config-subif)#int g0/1.30
R1(config-subif)#encapsulation dot1q 30
R1(config-subif)#ip addr 10.30.0.1 255.255.0.0
R1(config-subif)#int g0/1
R1(config-if)#no shutdown
```

By moving to the subconfiguration mode for the subinterface (for example `int g0/1.10` for VLAN10) we create this subinterface if it does not exist yet. Then we set the encapsulation according to the IEEE 802.1Q protocol (i.e. `dotq1` with the VLAN number, and assign an IP address. Note that in the second octet of the IP address is the VLAN number – not necessary, but it’s more clear. The given IP address will serve as the gateway for the appropriate subnet/VLAN.



 **Multilayer switch using SVI.** This method works similarly to the previous one, but we don’t have the L3 functionality directly at the ports, but only “inside” the L3 switch. We can think of this as a module in the switch that adds L3 layer functionality (i.e. a sort of “integrated router” inside the switch), and while cables from L2 switches might lead to a regular router, here the imaginary lines to the virtual L3 interfaces of this switch lead to the module.

In the configuration of the multilayer switch we create VLANs (we would do this even if it was one of the “common” L2 switches in the network), then we create individual SVI interfaces for different VLANs and assign IP addresses on them. Next, we’ll set up IP routing, or configure filtering (ACLs – we’ll cover this in the master’s course).



Example

We have several switches in our network, SM1 is a multilayer switch. We want to start routing between VLANs on it. A similar configuration to the previous example (but on a single device) would be as follows:

```
SM1(config)#vlan 10
SM1(config-vlan)#name marketing
SM1(config-vlan)#vlan 20
SM1(config-vlan)#name sales
SM1(config-vlan)#vlan 30
SM1(config-vlan)#name management

SM1(config-vlan)#int vlan10
SM1(config-if)#ip addr 10.10.0.1 255.255.0.0
SM1(config-if)#int vlan20
SM1(config-if)#ip addr 10.20.0.1 255.255.0.0
SM1(config-if)#int vlan30
SM1(config-if)#ip addr 10.30.0.1 255.255.0.0

SM1(config-if)#exit
SM1(config)#ip routing
```

First we create VLANs, then we create individual SVI virtual interfaces and set IP addresses on them. If there is no message about the activation of the interface (probably it will appear when it is created), we use the command `no shutdown`.

The command `ip routing` activates the inner L3 module.




Remark

When to use which method? If we have a router, we can choose the first solution, but it will be somewhat computationally more demanding. If we have a multilayer switch, then the second solution is definitely better, it is easier to configure and the operation is faster (part of the logic is implemented in hardware, in addition, all L3 operations are solved on a single device, we can have as many VLANs as we want). At the same time, we save on cables.


In addition, we can strengthen the lines in the network by using EtherChannel technology (we connect two devices with several cables, i.e. a bundle of physical cables forms one logical channel with greater transmission capacity/speed).



 On the previous pages, it was stated that the end devices do not contain an implementation of the IEEE 802.1Q protocol (so “they do not understand VLAN”). However, there are exceptions. It may be advantageous for servers to be included in more than one VLAN (to avoid constantly forwarding traffic through L3 devices), which can be done by implementing IEEE 802.1Q directly on this device. How to do it:

- If it's a Linux server, then it's easy because Linux itself includes an implementation of IEEE 802.1Q.
- For servers with Windows, the situation is more complicated, because Windows itself does not support this protocol. The only way to start VLAN support is to buy a special network card that implements IEEE 802.1Q.

3.2 Switches and Loops

 In the following text, we will focus on the procedure that is standardized for bridge-type devices, however, we will talk about switches. Although there is a certain difference between bridges and switches (switches are not standardized, they have more ports and switching is implemented in hardware), but the principle is the same (they work on the L2 layer, maintain a table of MAC addresses and ports, switch traffic, separate segments).

3.2.1 The STP Protocol

Let's imagine a larger network where we have several switches. Because we want the network to work even if a transmission path is interrupted or a switch goes down, the switches are physically connected more than necessary – some paths are *redundant* (multiplied).

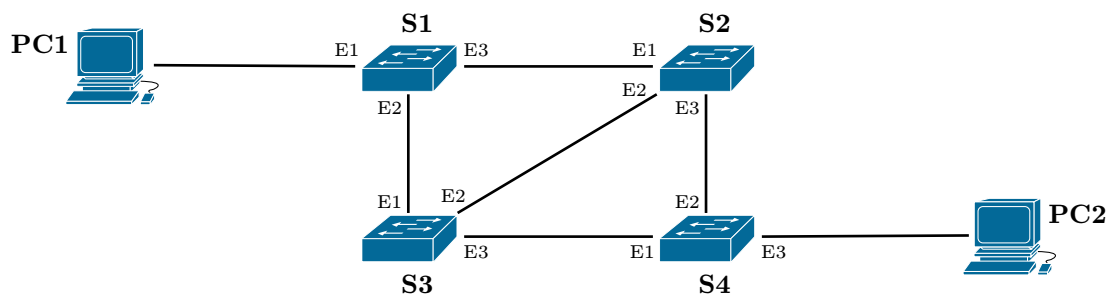



Figure 3.5: Network with loops

Redundant paths are practical for when part of the network goes down, but can cause some problems. Let's imagine a few possible scenarios:


1. PC1 sends the frame to PC2. The first part of the path (switch S1) is unambiguous, but what then? Should the frame be sent from switch S1 via the port E2 or via the port E3?
2. PC2 sends a broadcast frame (intended for all devices on the network). Switch S4 receives the frame on the port E3 and sends it to the ports E1 and E2. What do the other switches do?
 - Switch S2 receives the frame on the port E3 and sends it to the ports E1 and E2.
 - Switch S3 receives the frame on the port E3 and sends it to the ports E1 and E2. It also receives the same frame (but it doesn't know that it is the same frame) on the port E2 (from S2) and sends it to the ports E1 and E3.
 - Switch S2 receives the same frame again on the port E2 (from S3) and sends it to the ports E1 and E3.
 - Switch S1 receives the same frame twice from the port E2 and twice from E3, sends it twice to the port E2, twice to the port E3 and four times to the port E1,...
 - In the meantime, the broadcast frame got back to switch S4, PC2 and then again to other switches...
3. Suppose that PC1 is just connecting to the network, i.e. there is no switch in the MAC table yet. PC2 sends a frame to it (PC2's MAC address is the destination). Switch S4 does not know this address, so it sends the frame to ports E1 and E2, etc. – the frame travels through

the network in the same way as a broadcast frame. It will reach PC1, but copies of it will still be floating around the network for a long time.

 **Remark**

The scenario described in point 2 is called a broadcast storm. It is characterized by identical copies of the same broadcast frame continuously roaming the network, making the network defacto congested and unusable after a certain period of time. 

That implies that something is wrong. Redundancy of paths is good, but we should also prevent uncontrolled repeated wandering of network frames. We can prevent this if we *remove loops* from the network (as far as communication is concerned; loops may remain in the physical topology due to redundancy).

 The *Spanning Tree Protocol* (STP) is standardized as IEEE 802.1D as a mechanism to remove (logical) loops in the graph of a bridge network (i.e. in our case switches). This protocol only sees the bridges (or switches), it is not interested in any other devices (so it will ignore PC1 and PC2 in our network).

The purpose is to reconfigure the switches in the network in such a way that some ports are not used for communication (they are either disabled or used only for service purposes), thus removing loops from the network. The STP protocol basically solves the “graph skeleton search problem”, the purpose of which is to leave only such links in the graph that there is exactly one path (no more, no less) between any two nodes in the network (switches), preferably the fastest one.

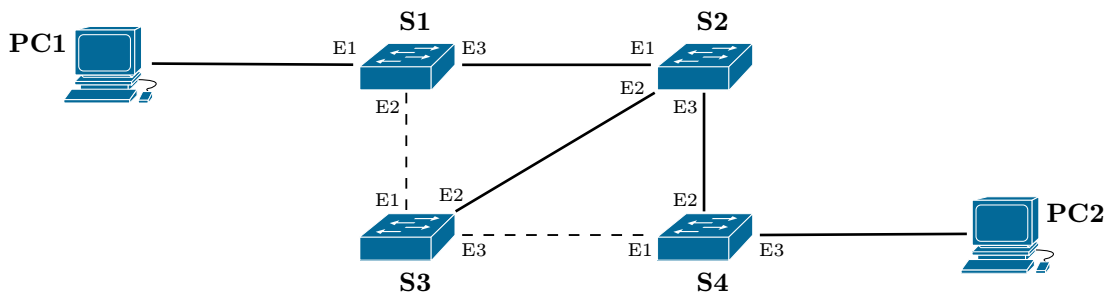


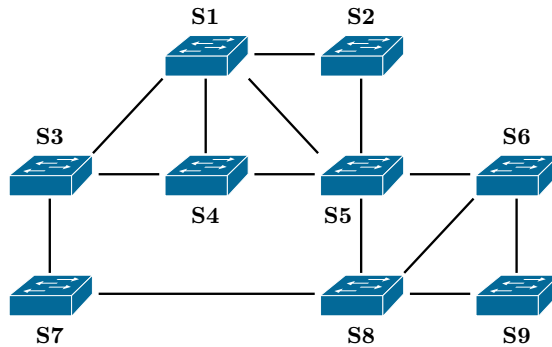


Figure 3.6: Network without logical loops

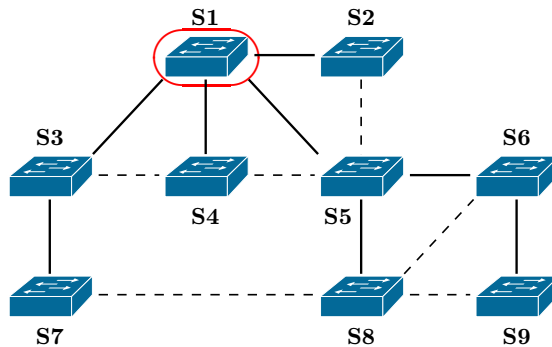
 Actually, we create *tree* from the switch network (“spanning tree” means covering tree). One of the switches is declared to be the root of the tree (*root switch*) and the fastest path to the root switch is searched from each other switch. All other paths are deactivated. In our example, switch S2 could be the root switch and only links S1–S2, S3–S2 and S4–S2 would remain active in the network. Links S1–S3 and S3–S4 would be deactivated.

 **Example**

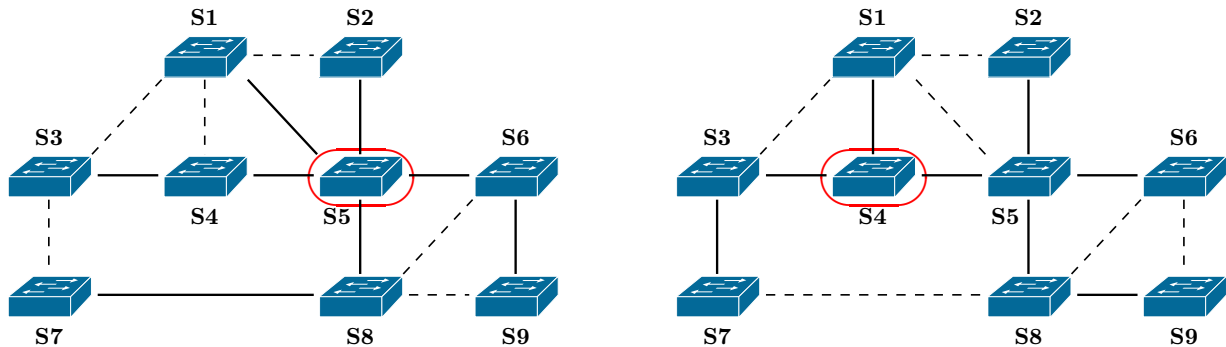
The network in the following figure shows a total of nine switches that are connected in a physical mesh topology. Here we have loops that need to be resolved using STP.



The first task is to select the root switch. We have a choice, but in a real situation we would probably have some criteria for this choice. Here we simply choose switch S1. We block some ports, making a total of six paths unavailable for regular communication.



There are other options – we can choose, for example, switch S5 or S4 as the root switch:



Also in these cases, some ports are deactivated for regular operation, and there is just one (preferably optimal) path from each switch to the root switch.




3.2.2 Convergence of the Switched Network

What determines the root switch? Each switch that “understands” the STP protocol (is implemented on it) has a special identifier assigned to it. This identifier is 8 octets long and consists of two parts:


- Priority (2 octets)
- MAC address (6 octets).

1	2	3	4	5	6	7	8
Priority		MAC address					

By default, there is the number 0×8000 in the priority field, in decimal 32768. It follows that no two switches have the same identifier (they differ at least in MAC address, each switch has a different one).


 STP selects as the root tree the switch whose identifier is the *smallest number*. If we relied only on this procedure, the root switch might be the one we don't like (it would select by MAC address of the switches) – the solution is to change the priority of the switch we selected to *smallest value*. Because the identifier contains the priority first and then the MAC address, this change always has the intended effect.


It remains to determine which connections (or ports) should be deactivated and which should remain. If there is a path to the root bridge through more than one port, then the optimality of that path is calculated for each port (depending on bandwidth/speed, utilization, we talk about the cost of the path, or cost per port). Only the port for the most optimal path will be active.

 So the port is either active or inactive in terms of forwarding frames with regular traffic. We refer to this property as *state*, distinguishing the following port states:

- *forwarding* – working, forwarding all traffic,
- *blocking* – does not forward regular traffic, but receives STP administrative frames (called BPDUs) so that it can be activated when needed,
- *listening* and *learning* – intermediate states in case the port is to transition from the blocking state to the forwarding state; in the listening state the switch processes BPDUs, in the learning state it receives (but does not forward, discards) frames with regular traffic and starts to create a MAC address table,
- *disabled* – forwards no traffic.


Ports move between these states, although they spend most of their time in the forwarding or blocking states.

 From time to time (especially during network changes) a reconfiguration – *convergence process* is performed. The purpose is to ensure that the network is in a converged state (i.e., in our case, that there are no loops in the network and that only the ports for the most optimal path are active). A new root switch may need to be determined, and port states may also change (for example, if a port is to go from the blocking state to the forwarding state, it will spend some time in the listening and discovery states to be able to set all parameters correctly and complete the MAC address table).

 Each port has a specific role, which is determined by the position of the switch in the hierarchy, and also distinguishes whether the port is pointing up to the root switch or down to the child switches. Suppose we have the switches drawn so that the root is at the top. The role determines what state this port will be in at any given time. So what roles are there?

- *root port* – port that points to the root switch; each switch except the root switch has a single root port pointing “up”, the root port is always in the forwarding state,
- *designated port* – port that is active and goes down in the hierarchy, i.e. to subordinate switches (so we have a subtree of switches behind the designated port); the root switch has all its ports except the blocked ones in the designated role, while the switches at the bottom of the hierarchy (the ones with the longest path to the root switch) have no designated ports,

- *blocked port* – this port has not been selected, it is not used for regular traffic (i.e. it is usually in blocking state).

 To make it all work, the switches talk to each other using special frames, which we call BPDUs (Bridge PDUs). They are used to maintain and possibly update network information needed for STP, and are sent to the multicast address that identifies all devices implementing STP: 01:80:c2:00:00:00. BPDUs are sent periodically every 2 seconds, and then whenever the switch network changes.


The BPDU contains, among other things, information about the STP frame and which version, which switch is the root switch (its bridge ID/switch ID), the ID of the sending switch, the cost of the path from the sender to the root switch, etc. We have seen this information in the example in the listings. The BPDU is encapsulated in an IEEE 802.2 LLC frame and then in an IEEE 802.3 MAC frame.



Additional information


<https://wiki.wireshark.org/STP>




 If a new switch is added to a switched network, it must also be added to the STP tree. The new switch considers itself a root switch at first, but leaves its ports in a listening state for now (it receives BPDUs but does not send anything). If it finds the ID of the root switch in the received BPDU that is lower than its ID, it stops considering itself as the root switch and integrates itself into the structure gradually (for each port, it calculates the path cost to other switches using the received BPDUs and selects only one port for each path, blocking the others).

3.2.3 The STP Protocol and its Variants

The original STP has been standardized as IEEE 802.1D.

 **RSTP.** Because the original specification was no longer sufficient for faster standards, the new Rapid STP (RSTP) standard was created as IEEE 802.1w, but is now included in the IEEE 802.1D-2004 revision (so that the original STP and the newer RSTP are merged into one standard).

While the original STP took 50 seconds to converge (during which time the entire network was down, which is an awfully long time), the newer RSTP takes about 1 second, or less. There are other differences between the original STP and RSTP, but this difference is the most significant.

 **MSTP.** The MSTP variant (Multiple STP) has been standardized as IEEE 802.1s, also known as MST (Multiple Spanning Tree). The purpose is to make STP more efficient for when virtual networks (VLANs) are defined in the network. It has also been added to the standard, but different: IEEE 802.1Q, as a revision of IEEE 802.1Q-2005, the extension of RSTP for VLANs.

Basically, the idea is to have a separate STP instance for each VLAN or VLAN group, so there could be a slightly different active link structure for each VLAN (or VLAN group). The purpose is both to make communication within each VLAN more efficient (so that, for example, communication will not be passed through switches that have nothing to do with the VLAN in question), but also to be able to balance the load in the network (a certain port will be blocked

for one VLAN and forwarding for another, for another port it may be the reverse, each will use different paths).

3.3 Overview of the IEEE 802 Standards

From the previous text, it is clear that Ethernet is standardized as IEEE 802.3 (in fact, the term “Ethernet” does not appear in the standard, but it is informally called so), and that there are IEEE 802.2 standards (for the LLC sublayer of the data-link layer) and IEEE 802.1 (some parts of which we have seen in this chapter). But what’s next?

Standard	Meaning
802.1	LAN architecture, bridges, L2 management
802.2	Logical Link Control – LLC (WG is inactive)
802.3	Local networks, CSMA/CD collision method
802.4	Token Bus – bus-based LAN (WG dissolved)
802.5	Token Ring – ring-based LAN (WG is inactive)
802.6	DQDB – MAN network (WG dissolved)
802.7	Broadband LAN (WG dissolved)
802.8	LAN/MAN over optical networks (WG dissolved)
802.9	Integrated services – isochronous networks (WG dissolved)
802.10	Security in LAN/MAN (WG dissolved)
802.11	WLAN Working Group – Wi-fi
802.12	High-speed networks – 100VG-AnyLAN (WG dissolved)
802.14	Broadband network on cables for cable TV (WG dissolved)
802.15	WPAN Working Group (e.g. Bluetooth, ZigBee, UWB), including coexistence with IEEE 802.11
802.16	Wireless broadband MAN (WiMAX) (WG is inactive)
802.17	RPR – Resilient Packet Ring, used in SONET/SDH (WG is inactive)
802.18	Regulation of radio waves
802.19	Coexistence of unlicensed wireless networks
802.20	Mobile broadband wireless LAN/MAN including transport mobility
802.21	Handover Services Working Group – handover between different types of mobile and wireless networks (GSM, GPRS, Wi-fi, Bluetooth, WiMAX, etc.)
802.22	Wireless regional networks using unused TV broadcasting frequencies
802.23	Emergency Services Working Group (WG is inactive)
802.24	Smart Grid Technical Advisory Group: vertical applications – Internet of Things (IoT), Machine-to-Machine communication (M2M), Smart Grid, interconnection of vehicles. . .

Table 3.2: IEEE 802 Standards Family

Table 3.2 provides an overview of the currently existing standards from the IEEE 802 family. Each standard is developed by a specific working group, e.g. IEEE 802.3 is the responsibility of

the *Ethernet Working Group* and IEEE 802.11 is the responsibility of the *Wireless LAN Working Group*.

The most important standards (from our point of view) are highlighted in bold. Some of them we already know, others we will get to know later.



Remark

Many of the IEEE 802 working groups are either inactive or even dissolved. Inactive means that no update or amendment has been issued for a long time (which is not a problem when amendments are not necessary), dissolved means that there are no future activities planned either (typically for technologies that have not become established in practice).

For example, Token Ring, Token Bus and 100VG-AnyLAN networks have been displaced by Ethernet, Isochronous networks offering the possibility to assign priority to some types of data (e.g. during voice transmission) also (Ethernet using IEEE 802.1p/Q can also do this and is faster). IEEE 802.14 has been replaced by the DOCSIS standard.



The IEEE 802.1 standard is the most richest. Thus far, we have encountered the following subordinate standards and amendments to it:

- IEEE 802.1Q – VLAN (Virtual Local Networks),
- IEEE 802.1p – priority setting (3 bits, the range 0–7), used by IEEE 802.1Q, inside IP packet header, etc., it's not really a standard, just a simple prescription,
- IEEE 802.1D – STP (bridges, network without loops),
- IEEE 802.1w – originally RSTP (only a prescription), the specification has been later added to the revision IEEE 802.1D-2004,
- IEEE 802.1s – originally MSTP, the specification has been later added to the revision IEEE 802.1Q-2005.

Of those we have not covered yet, one interesting example is IEEE 802.1X – authentication of devices accessing the LAN, i.e. an authentication server. Authentication according to this standard is often used in corporate networks to authenticate user/device access to the network and determine specific permissions for that user.




Remark


Note that the IEEE 802.13 line is completely missing. Officially, the number 13 is reserved for the further development of fast Ethernet, but in reality the reason is very similar to why many hotels lack a thirteenth floor.




Chapter 4

Network and Transport Layer

 *Quick preview:* In this chapter, we will look at what typically happens at the network and transport layer. We will be interested in addresses, packet/segment format, and the most important processes that are handled at these layers. We will focus on IPv4, IPv6, ICMP at the network layer, and TCP, UDP at the transport layer.


 *Keywords:* IPv4, IPv6, packet, TTL, MTU, fragmentation, ICMP, client-server communication, TCP, UDP, port

 *Objectives:* After studying this chapter, you will be familiar with IPv4 and IPv6 addresses, know what an IPv4 and IPv6 packet looks like, be able to describe the function of the TTL field and its relationship to packet lifetime, and understand the reason for and process of packet fragmentation. You will also understand the TCP connection flow, know what a TCP and UDP segment looks like, and be able to describe the difference between the two.

4.1 Network Layer and Logical Addresses

The network layer (in RM ISO/OSI terminology, the L3 layer), or the Internet layer (according to the TCP/IP network model), works with the logical topology of the network. The role of the L3 layer is to provide a unified network interface for the higher layers (the higher layers are no longer concerned with the actual process of communicating with specific devices), as well as routing, which we will cover in another chapter.

While the lower layer (L2) protocols communicate within a single network, the network layer protocols provide communication across network boundaries, i.e. they connect different networks (even those that use different protocols at lower layers). Typical network layer devices are routers and switches with L3 layer functionality (multilayer switches).

 We refer to network layer addresses as *logical (software) addresses*, as opposed to physical (hardware) data-link layer addresses.

So this gives us a division of responsibilities – the L2 layer with physical (hardware, MAC) addresses is concerned with addressing and delivery (here switching) within the local network,

whereas the L3 layer with logical (software, IP) addresses is concerned with addressing and delivery (here routing) between networks.

4.2 IPv4 Protocol

The main task of the Internet Protocol (IP) is to take data segments from the upper layer, encapsulate them into an IP packet (i.e., add an IP header), and pass them to the lower layer, and vice versa. In fact, some other network layer protocol traffic, such as ICMP, is also encapsulated in IP packets.

4.2.1 IPv4 Addresses

An IPv4 address is 32 bits long, or 4 octets. In the notation, the octets are separated by a period and written either in decimal or binary.



Example


An IPv4 address can look like this:

- 10.6.29.181
- 169.251.220.5
- 169.251.255.255
- 255.255.255.255

The first address listed in binary would be: 1010.110.11101.10110101.


The last one: 11111111.11111111.11111111.11111111.




 IP addresses are *hierarchical* – that is, they take into account a certain division of devices with these addresses into groups and subgroups (networks and subnets), where “similarity” of two devices within a network or subnet (i.e., belonging to the same network or subnet) means that these two devices will have a portion of the address the same. So the hierarchy is reflected in the address as follows:

- *network portion of address (prefix)* – all devices belonging to the same network have this portion identical,
- *host portion of the address* – these devices will differ in the rest of the address.

In addition, this hierarchy can be more complex than just two layers, because the network can be further subdivided into subnets (with devices on the same subnet having the same network and subnet address, differing only in the host portion).

 If we set all bits in the host portion of the address to 0, we get *network address*. If we set all bits in the host portion to 1, we get *broadcast address* for the network.

 As written above, somewhere inside the address is the boundary between the network and host portions of the address. But if that boundary can be in different places, how do we know that particular place? One of the following two methods is used to determine this boundary:

1. *Netmask* – looks like the address itself, but in binary notation there are ones in the (sub)network portion of the address and zeros in the host portion.
2. *Prefix length notation* – after the address, we write a slash and a number specifying the number of bits of the network portion.

**Example**

Let's continue with the previous example. Consider again the address 10.6.29.181, which is binary after adding zeros from the left 00001010.00000110.00011101.10110101. We want to indicate that the network part of the address consists of the first 14 bits, then:

1. we add a netmask – in this case it is:
 - in binary: 11111111.11111100.00000000.00000000
 - in decimal: 255.252.0.0
2. we write the address in the form 10.6.29.181/14, where 14 is the prefix length (i.e. the length of the network portion of the address).

An IP address can be used to identify not only a specific device on the network, but also the network as a whole (because it is a hierarchical addressing system). The network address in our case is

- in binary: 00001010.00000100.00000000.00000000
- in decimal: 10.4.0.0

(we left the original prefix, i.e. the bits that are set to 1 in the mask, but set all other bits to 0).

The broadcast address for the network is

- in binary: 00001010.00000111.11111111.11111111
- in decimal: 10.7.255.255

This means that if a device on this network sends a broadcast IP packet destined for devices on this network, it will use this address as the destination address.



Universal broadcast address has all bits in both the host and network portions set to 1, i.e. it is the 255.255.255.255 address. The target is all devices on the network and is usable even when we don't know the network address (for example, when we are just negotiating an IP address assignment).




At the L3 layer, routing between networks is performed, so it must be possible to connect different networks. For this interconnection we need a device through which all communication between the interconnected networks goes – a router or switch with L3 functionality. In IP terminology, this device is called a *gateway*, and from the perspective of any device on the network, a gateway is an active network equipment to which we forward packets destined for a device on another network – i.e., a gateway is a device for “travel out of the network”.

In order to be able to communicate with someone outside our own network, we need to know the gateway address. So in terms of its own identity and the ability to communicate at the L3 layer, every device should receive this information:

- IP adresa,
- netmark or prefix length,
- default gateway address.

4.2.2 Special IPv4 Addresses

 *Multicast address* is used as the destination address in packets that are to be delivered to multiple destination devices, but not necessarily to everyone on the network. A range of addresses 224.0.0.0 to 239.255.255.255 is reserved for this purpose (it may not be obvious, but these are all addresses where the first three bits are set to 1 and the fourth bit is set to 0). Some multicast addresses are reserved for specific purposes:

- 224.0.0.1 indicates a group of all devices in the local network that “understand” the IPv4 protocol, i.e. it is similar to a broadcast address in the local network,
- 224.0.0.2 is the group of all routers on the local network (so if we want to send a packet to routers, we send it to this address),
- 224.0.1.1 is the group for NTP servers (time servers) used to synchronize time on the network, etc.

Addresses of the form 224.0.0.x (including the first two listed above) are for use within the local network only. They can only be used as destinations in packets with TTL = 1 set, meaning they will not go through the router to another network.




Additional information


The list of registered multicast IPv4 addresses is available at

<http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml>




 *Loopback address* is an address starting with the octet 127, in most cases it is 127.0.0.1. Every device has a loopback address.

Loopback is implemented inside the device as a virtual device, from a network perspective it is actually a “view of itself” in the direction of the network, a kind of mirror. So when we want to test whether or not our network interface is functional (regardless of whether or not it has an IPv4 address assigned to it), we just test the loopback.

 We distinguish between *public* and *private addresses*. Public addresses are globally unique and can be used without any restrictions. Private addresses are unique only within a given network and cannot go outside that network (more precisely: they do not go beyond the company’s border router, where it is assumed that address range collisions for its own network are guarded by the company itself).

The following address ranges are used for private addresses, according to RFC 1918:

- 10.x.x.x
- 172.16.x.x – 172.31.x.x
- 192.168.0.x – 192.168.255.x

 *Undefined address* is an address that a device uses when it doesn’t actually have an address assigned yet. Numerically, this is 0.0.0.0.

4.2.3 IPv4 Packets

The structure of an IPv4 packet is a bit more complex than the structure of the lower layer frame, so we can’t make do with a single “row” in the diagram.

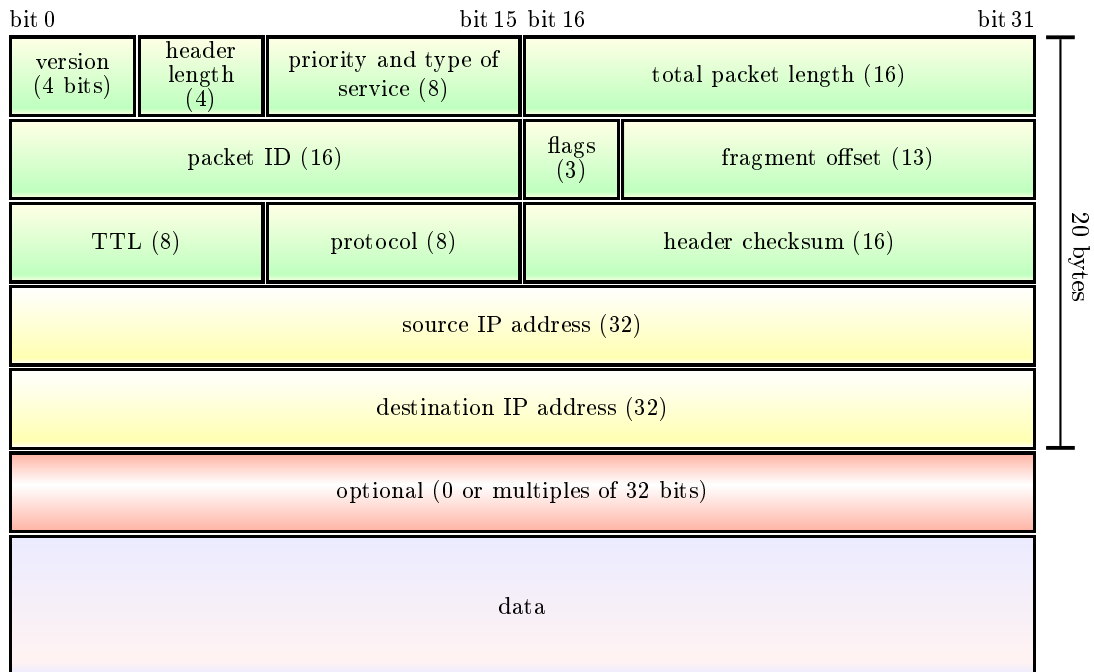



Figure 4.1: IPv4 packet diagram

 The format of an IPv4 packet is indicated in Figure 4.1. The size of the fields is given in bits. The individual fields in the header have the following meaning:

- *Version* (4 bits) – IP version, here it is 4, binary 0100.
- *Header Length* (4 bits) – header size in 32-bit words, i.e. the number of header lines (usually we find here the number 5, binary 0101).
- *Priority and Type of Service* (8 bits) – specifies how this packet is to be handled on the path. The first three bits determine the priority according to IEEE 802.1p, the remaining bits determine whether a path with a better value for less processing delay, throughput, reliability, etc., should be selected for routing.
- *Total Length* (16 bits) – length of the entire packet, including headers and encapsulated data in Bytes.
- *Identification* (Packet ID, 16 bits) – the number assigned to the packet; this number is assigned by the sender during transmission and should be different for each packet sent.
- *Flags* (3 bits) – only two bits are used (the third is reserved), their meaning will be explained later in the section on fragmentation.
- *Fragment Offset* (13 bits) – if the packet had to be fragmented (split into smaller parts), a number for calculating the position of the part of the data transmitted in this fragment relative to the original un-fragmented data is stored in this field.
- *TTL* (Time to Live, 8 bits) – packet lifetime. The number in this field is always decremented by 1 on any active network equipment with L3 functionality (such as a router).
- *Protocol* (Type, 8 bits) – information about what is encapsulated as data inside the packet.
- *Header Checksum* (16 bits) – is counted over the all previous 2-byte sequences.
- *Destination and Source Address* (each 32 bits) – the source address is always unicast.

- *Options* (0 or multiples of 32 bits) – is used for service purposes, for example, to influence packet routing through networks. Depending on the (non-)existence of this field, the Header Length field has a value of 5 or higher.
- *Data* (Payload) – the transmitted data according to the specific protocol indicated in the Protocol field.

Since the total packet length field (the fourth field) is only 16 bits long, the maximum possible number is $2^{16} - 1 = 65\,535$, which implies a limitation on the maximum length of the entire packet. The limit on the length of the encapsulated data is obtained by subtracting the length of the header. However, typically only 1500 B of data can be encapsulated in a frame, unless jumbo frames or a completely different technology than Ethernet or Wi-fi is used, so the length of segments on L4 and packets on L3 is usually adjusted to this limit.

In the *Protocol* field we can find the identifier of the protocol whose data unit is encapsulated in the IP packet. For example, TCP (6) and UDP (17) protocols from the transport layer have their identifier for this field, but also those protocols from the network layer that are encapsulated in the IP packet, such as ICMP (1), and routing protocols, such as OSPF (89). It is even possible to encapsulate a higher version of the same protocol in an IPv4 packet, the ID for IPv6 is 41.



Additional information

A list of all values for the Protocol field can be found at

<http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.



4.2.4 TTL and Packet Lifetime

As written above, in the TTL (Time to Live) field there is a number that is decremented mostly by 1 on each active network equipment operating on the L3 layer. The purpose is to limit the existence of undeliverable packets that would otherwise travel endlessly through the network and congest the links (on the L2 layer we have the STP protocol for this purpose, here it is necessary to use a different mechanism, because loops and redundant links on L3 cannot be avoided).

For some protocols whose PDUs are encapsulated in IP packets, it is directly specified what TTL is to be used (for example, the value 1 is given if the packet is not to leave the network). The maximum value is of course 255 (because we have 8 bits and $2^8 - 1 = 255$), but usually the number 64 or 128 is used (for example, Windows uses 128).



Additional information

- <http://subinsb.com/default-device-ttl-values>
- <http://www.binbert.com/blog/2009/12/default-time-to-live-ttl-values/>



Remark

The TTL field is also called “hop limit”, the maximum number of hops over the nets. This name is primarily used in the next version of the protocol – IPv6.



4.2.5 MTU and IPv4 Packet Fragmentation




Definition (MTU)

MTU (Maximum Transmission Unit) is a value defined for a specific link that determines the maximum packet size for sending over that link. The MTU configured on a particular device is the maximum packet size that the device can receive and process.



MTU is primarily affected by the settings on the active network equipment through which the packet is being directed, but it also depends on the specific L2 layer protocol in which the IP packet is encapsulated.

In the case of an IP packet encapsulated at the L2 layer in an Ethernet frame, we have 1500 as the maximum number (but for example for Token Ring this number is roughly double). Since the vast majority of (primarily local) networks use Ethernet at the L2 layer, a similar MTU value is set on many active network equipment. Given that the data in an IP packet can be up to 65 535 octets long (and we have to add the IP header to that), it is clear that we have a significant disparity.

 *IP Packet Fragmentation* is the ability to divide an IP packet into smaller parts (fragments) according to the size of the MTU and ensure that the target device can reassemble these fragments into the original packet.

Of course, each fragment must also become a packet, so we attach an IP header to each fragment. Most of the fields will be “inherited” from the original packet, but some fields will be different. For fragmentation and especially subsequent assembly at the destination, we need some specific values in the fields of the second line of the packet as shown in Figure 4.1 on page 101:

- *Packet ID* (16 bits) – all fragments have this field the same (inherited from the original packet), it is used in the destination to determine which fragments belong together,
- *Flags* (3 bits) – we’re interested in two one-bit flags:
 - DF (Do not Fragment) – in order to fragment, this flag must be set to 0; if the sender sets this flag to 1, it disables fragmentation on the path during transmission,
 - MF (More Fragments) – we set this flag to 1 in all fragments except the last one,
- *Fragment Offset* (13 bits) – for each fragment, we find out which octet of the original data it starts on, we divide that number by 8 and store it here; this implies that the size of the data in the fragment is always a multiple of 8 octets (so that we can always “fit” a usable address), except possibly the last one fragment.



Procedure (IPv4 packet fragmentation)

If the IP packet is larger than the MTU on the path allows, we must first check if we can fragment at all. If the DF bit in the *Flags* header field is set to 1, we don’t even bother fragmenting and just drop the packet. Otherwise, we do the following:

1. We calculate the data length for the fragment and the number of fragments:
 - we take the MTU value on the following path and subtract the packet header size (usually 20 octets, but beware, it can be larger),

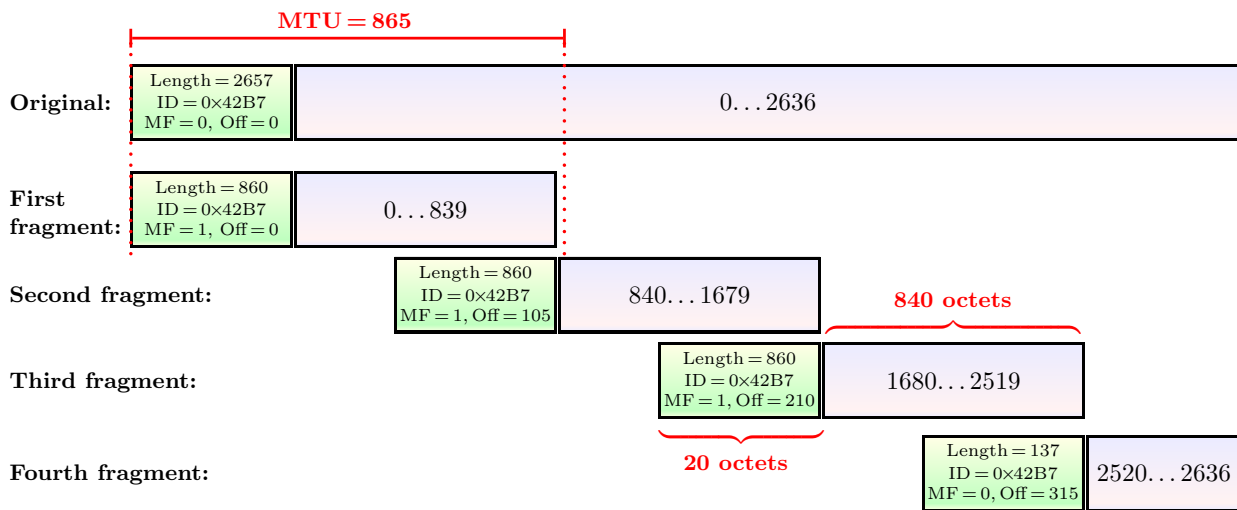



Figure 4.2: fig:fragment

- the resulting number divided by an integer eight (i.e. round down), let's denote this number by X ,
 - the maximum possible size of the data encapsulated in the packet is $X * 8$,
 - if the original data length *after subtracting the header* of the packet is M , then the number of fragments is $M/(X * 8) + 1$ using integer division, adding one is the last fragment with “remainder”.
2. We create the first fragment:
 - we copy most of the headers from the original packet (including *Packet ID*),
 - we set the MF (More Fragments) flag to 1,
 - we put 0 to the *Fragment Offset* field because this fragment is the first in the sequence and it starts with address 0 in the octet sequence of the original packet,
 - we insert the octets with addresses $0 \dots (X * 8 - 1)$ (i.e. the first $X * 8$ octets) into the first fragment as data.
 3. We create other fragments except the last one (n is the order of the packet we are currently processing, starting from $n = 2$):
 - we again take most of the header fields,
 - we set the MF (More Fragments) flag to 1,
 - we put the number $X * (n - 1)$ in the *Fragment Offset* field (the contents of this field multiplied by eight gives the address of the beginning of this fragment in the original data),
 - we insert the octets with addresses $X * 8 * (n - 1) \dots X * 8 * n - 1$ as data (i.e. the n -th $X * 8$ octets).
 4. We create the last fragment ($n = M/(X * 8) + 1$):
 - we include most of the header fields,
 - we set the MF (More Fragments) flag to 0, because there will be no more fragments,
 - we put the number $X * (n - 1)$ into the *Fragment Offset* field,
 - as data we insert octets with addresses $X * 8 * (n - 1) \dots M - 1$.



 Putting the fragments together is always done at the destination, and it may happen that some fragments are fragmented again along the way. Destination device

- collects all packets with the same packet identifier (first field of the second line),
- sorts them by the *Fragment Offset* field, where the last fragment in the sequence should be the fragment with the MF flag set to 0,
- checks if any fragment is missing (for each fragment, it compares its length with the fragment offset field of the next fragment in the sequence multiplied by 8).

If everything matches, it puts the fragments together, but if it finds an error, all fragments are discarded and the transfer must be repeated (at the direction of the parent layer).



Remark

Note that there is no mention anywhere of a request for a retransmission. The IP protocol is in principle unreliable and does not deal with any such thing (it provides the “best effort” service). Retransmissions are negotiated by the higher layers if needed.

There is also no connection established on the network layer (at least this is not done by the IP protocol). This is why *IP datagrams* are often referred to in the literature (instead of IP packets) – recall that a datagram service is a service without connection being established.




4.3 IPv6 Protocol

IP version 6 (IPv6, aka IPng – next generation) is intended to solve the urgent shortage of IPv4 addresses. While IPv4 addresses take up 32 bits, IPv6 addresses are 128 bits long, which should be plenty long enough for any device in the world (theoretically, that’s 10^{38} addresses).

In fact, there are more reasons to move from IPv4 to IPv6 than just address range. For example, more extensive security support is important – IPv4 doesn’t offer this at all, but security is much more important today than in decades past. Other reasons include increased support for mobile devices, the ability to simplify obtaining an IP address for endpoint devices, and more.

IPv4 and IPv6 can be used simultaneously, even on the same node in the network (the dual-stack property).

 *ICANN* (Internet Corporation for Assigned Network Numbers, <http://www.icann.org>) is the main guarantor of IPv6 address allocation, while *IANA* (Internet Assigned Numbers Authority, <http://www.iana.org/>) physically performs the allocation.

The overall structure of address allocation is hierarchical. IANA allocates blocks of addresses to *Regional Internet Registries* (RIRs), which are RIPE (Europe and part of Asia), ARIN (North America), AfriNIC (Africa), LACNIC (Latin America), APNIC (Asia Pacific). The next tier of the hierarchy is made up of *local registrars* (LIRs), which get their blocks from regional registrars. Local service providers obtain their address ranges from local registrars, and from them customers or other entities that can redistribute their ranges.

The upper two tiers of this structure are indicated in Figure 4.3. Below are the LIR registrars for each country.

¹From: <http://whitengreen.com/blog-1124-how-to-trace-and-locate-ip-addresses>

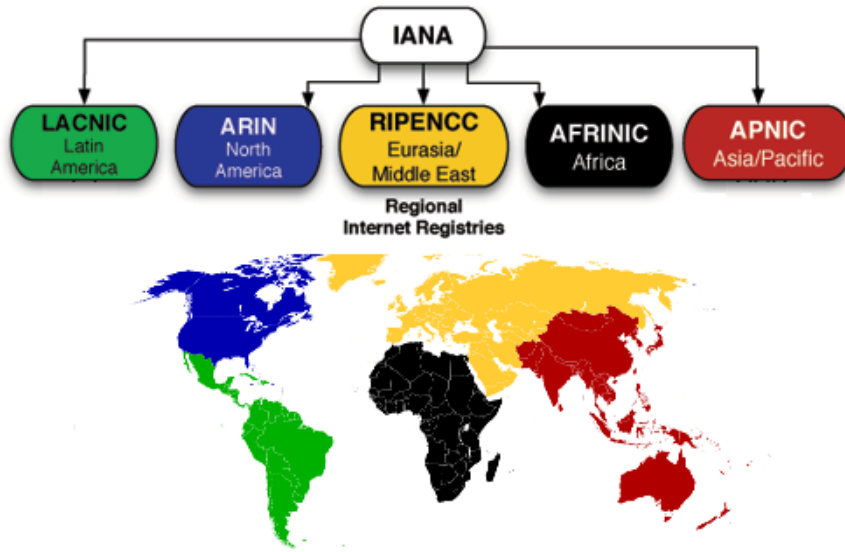




Figure 4.3: The basis of the IPv4 and IPv6 address allocation structure¹

 **Additional information**

LIRs for Europe: <https://www.ripe.net/participate/member-support/info/list-of-members/europe>.




4.3.1 IPv6 Addresses


 The IPv6 address is 128 bits long (i.e., 16 octets, four times the IPv4 address) and consists of two parts – *prefix* and *network interface identifier* (the node address within a single prefix), which is actually similar to what we know in IPv4.

 **Remark**

It is also hierarchical addressing – addresses reflect the physical structure of the network connected by routers, even on a global scale. IANA allocates base ranges (the first 12 bits) to individual regional RIRs, who further distribute their ranges to local LIRs. Further subdivisions are made by ISPs and, of course, by individual organizations for their subnetworks.




 Because IPv6 addresses are very long, they are written in hexadecimal digits in groups of four digits (i.e., two octets) separated by a colon. This implies that the address will have eight parts separated by colons.


 **Example**

An IPv6 address might look like this:

- a4cb:57b1:60aa:000E:113a:b201:042a:02b1
- 2001:0db8:3c4d:0000:0000:a010:0000:0000
- a4cb:57b1:60aa:E:113a:b201:42a:2b1
- 2001:db8:3c4d:0:0:a010:0:0

Note that in the pairs of addresses below each other, the bottom address is actually the same as the top address, we just removed the zeros from the left in each part. Of course, we could have done the same for IPv4.

Colons separate a total of eight parts of the address, each part being written with a maximum of four hexadecimal digits, i.e. occupying two octets. 

 Compared to IPv4, the notation can be simplified by *removing a sequence of null octet groups*. Thus, only one sequence of null groups can be removed per address, and the removal location must be marked with a double colon.

Example


It is only really possible to prune a single sequence of null groups. For example, the address 2001:0db8:3c4d:0000:0000:a011:0000:0000


can (theoretically) be pruned in one of the following ways:

2001:0db8:3c4d:0000:0000:a011:: (respectively 2001:db8:3c4d:0:0:a011::)

2001:0db8:3c4d::a011:0000:0000 (respectively 2001:db8:3c4d::a011:0:0)

mistake: 2001:0db8:3c4d::a011::

because such an address would be ambiguous. It is clear from the number of parts of the address that there are four null parts in total (there must be 8 in total), but when we see two locations of shortening in the address, we cannot tell exactly how the four null parts are positioned between them. For example, it could be a wrong possibility 2001:0db8:3c4d:0000:a011:0000:0000:0000. 

 *Canonical IPv6 address format* is standardized as RFC 5952 and prescribes the following conditions:

- hexadecimal digits are to be written in lower case,
- omitting leading zeros in a group is mandatory (so in the first example on page 106, the correct short form would be the one on the second line of each column),
- the :: mechanism for shortening the number of groups must have the greatest effect, which means that if we have multiple sequences of zero groups, the longer one is selected, and if there are multiple groups of the same length, we select the leftmost one (i.e. In the second example on page 106, only the result of 2001:db8:3c4d::a011:0:0 is in canonical form), and it must absorb all reachable null groups; if there is only one null group in the address, the :: construct is not used (no group is removed).


The canonical form of addresses is deterministic, unambiguous.

4.3.2 Special IPv6 Addresses

 IPv6 uses the following types of addresses:

- unicast (one individual device in the network),
- multicast,
- anycast (addressing to anyone in the specified “group”).


Broadcast addresses are no longer supported.


 For specific device addresses, there are the following possibilities:


- *Unique Local Address* (ULA) – used to send unicast data within the local network (organization etc.), it is similar to a private IP address in IPv4, it must not be visible outside the


organization's network. ULA addresses have the prefix `fd00::/8`, so hexadecimal addresses always start with the octet `fd`.


- *Link Local Address* (local on the segment) – it is not guaranteed to be unique (there may be devices with exactly the same link local address on another network), but it is unique at least within the segment. Packets with this address as the destination do not pass through the router. These addresses always have the prefix `fe80::/10`, so the first ten bits are `1111 1110 10` (note that the first octet is obvious – `fe`, but then it gets more complicated, the third hexadecimal digit can be `8`, `9`, `a` or `b`).
- *Global addresses* – they are always unique within the whole Internet, their prefix is always `2000::/3`, i.e. in binary notation they start with three bits `001` and hexadecimal the first nibble (four bits) is on the value `2` or `3`.

 The uniqueness of the ULA address is ensured by deriving it from the date (time) of address generation and the MAC address of the station.


 Originally there was a *site local* address, which would work exactly like local addresses in IPv4 (including address translation), its prefix is `fec0::/10`. However, this type of address was very soon removed from the standard, and the only one that counts on it is Microsoft in some of its technologies.

 *Loopback* has the same meaning as in IPv4, i.e. it is a test address meaning “outside view of self”. The loopback address in IPv6 is `::1/128`, which could be rewritten as `0:0:0:0:0:0:0:0:1`.

 *Undefined address* (i.e. information that the station has no assigned address) is `::/128`, which means `0:0:0:0:0:0:0:0:0:0`.

 Also in IPv6, *multicast addresses* are used, their prefix is always `ff00::/8` (that is, the first eight bits are set to 1). Some multicast addresses are reserved, for example:


- `ff02::1` – the multicast address for all IPv6 nodes,
- `ff02::2` – all routers,
- `ff02::1:2` – all DHCP servers (to obtain dynamic IPv6 address).

 **Additional information**

A list of all well-known and registered multicast addresses is at

<http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml>



 *Anycast addresses* are in principle also group addresses (with the difference that the recipient is not every member of the group, but only one of them), but in reality they are handled a bit differently. They do not have a dedicated prefix, and they use the range of unicast addresses.

Anycast addresses are practically not used globally (it would be unnecessarily complicated for routers), rather they are encountered in local networks. They are configured in routing tables on network devices (routers, etc.): if anycast destinations are behind multiple router ports, the routing table entry is set to “if the packet has xxx as the destination address, send it to port yyy”, almost the same as for unicast.

Where they can be used, for example: in load balancing over an organization's network or between two ISPs, different paths for packets can be handled by anycast when routing.

4.3.3 IPv6 Packets

The IP packet structure of version 6 is significantly changed compared to version 4. The main difference is the headers structure. While the IPv4 packet header is only one and can be of variable length (depending on the *Optional* field), in an IPv6 packet we have one mandatory (main) header of fixed length (a few most important fields) and we can add extension headers, each of which has a defined purpose.

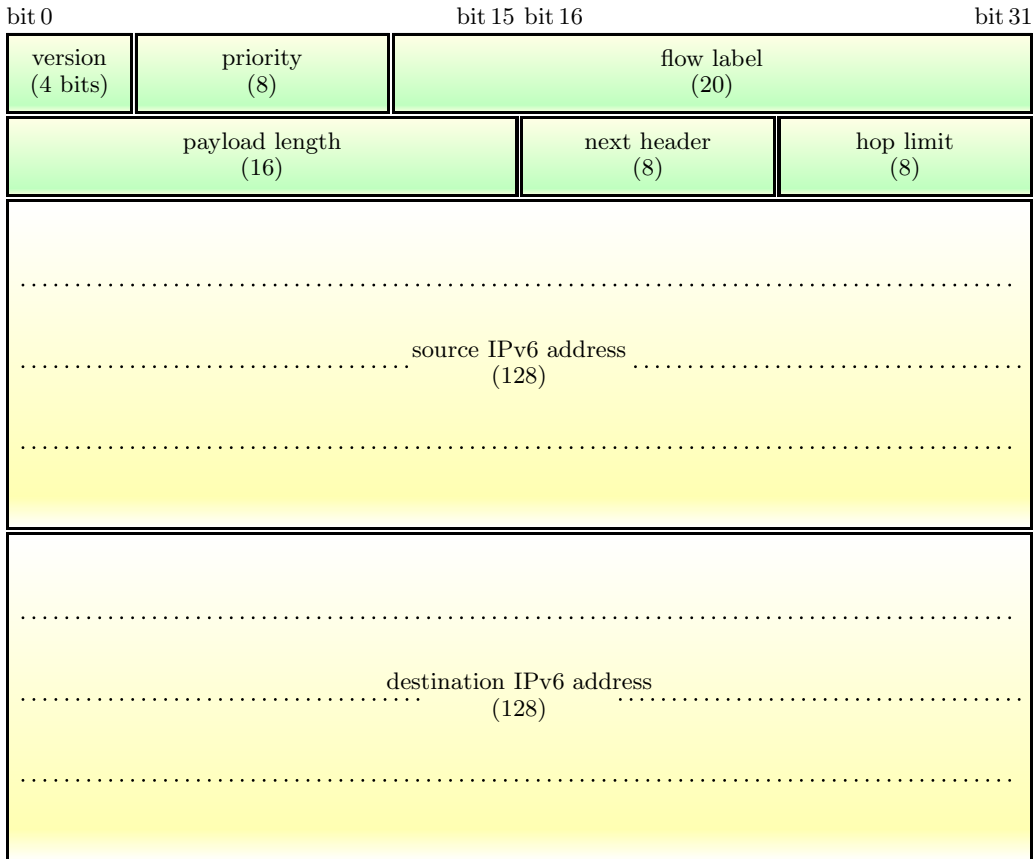


Figure 4.4: IPv6 packet – main header

The *main header* of fixed length is indicated in Figure 4.4. As we can see, the first field is the same as in the previous version (version number), only its content is different – for an IPv4 packet it stores the value 4 (binary 0100), whereas for an IPv6 packet it stores the value 6 (binary 0110). The other fields are completely different, but the receiving device already decides by the first field what version it is and what fields to expect next.

Meaning of the fields:

- *Version* (4 bits) – protocol version (here 6).
- *Priority* (8 bits) – similar to *Type of service* in IPv4, individual bits allow optimization of priorities.
- *Flow Label* (20 bits) – specifies the way of “special handling” on routers for some types of protocols, packets belonging to the same data flow should be handled in the same way (no data flow: = 0). If it is used, it is used together with the previous field (the same priority applies to all packets of the same datastream).

- *Payload Length* (16 bits) – length of the rest of the packet (excluding the main header), i.e. all extension headers and custom data; if 0, it is a *jumbogram*.
- *Next Header* (8 bits) – specifies the type of the next (extension) header of the same packet, or the type of encapsulated data.
- *Hop limit* (8 bits) – similar to TTL for IPv4, indicates the remaining number of routers or other L3 devices from a specified value, decremented by 1 on each L3 device.
- *Source and destination address* (the both 128 bits long, i.e. four lines for each address according to Figure 4.4).

Figure 4.5 is actually the same as the previous one, but the line size is doubled from 32 bits to 64. While years ago computing systems (computers, servers, etc.) were 32-bit, today they are almost all 64-bit and thus 64 bits are always loaded into the processor (and elsewhere) at once. IPv6 is optimized just for 64-bit processing, and as we can see, almost all of the main header fields (all in green) are loaded in a single access, and each address is loaded in two accesses.

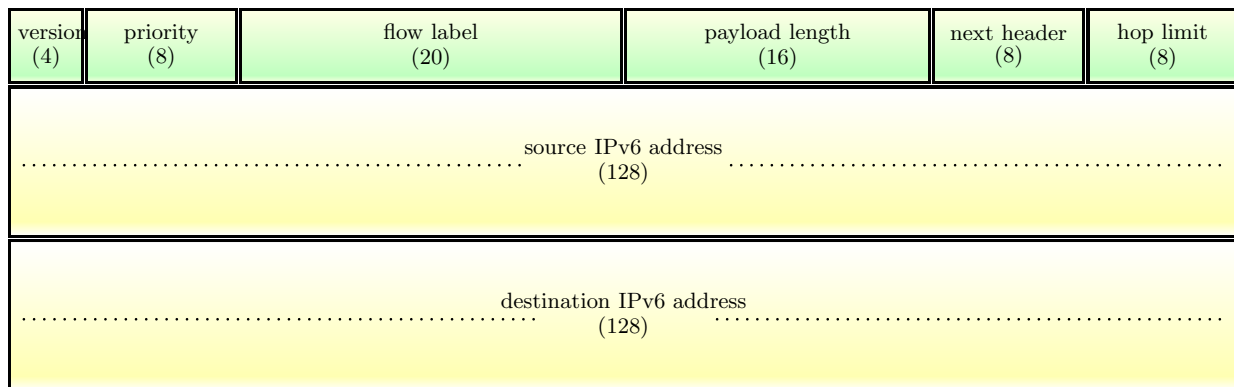


Figure 4.5: IPv6 packet – main header for 64-bit communication

Extension headers follow the main header in a predefined order (i.e., order is important, RFC 2460), some (or all) may be omitted. Each type of extension header has a number, this number is found in the *next header* field of the previous header (i.e., in the main header we find out the type the first extension header, in the first we find out the type of the second extension header, etc.).

 Some of the extension headers are the following:

- *Hop-by-hop Options Header* (0, information for routers on the path, routers read only this header from the extensions),
- *Routing Header* (43, we use if we want to prescribe the path “hard”, here the routers through which the path should lead can be specified, in the reverse order it is also mandatory for the response),
- *Fragment Header* (44, if the packet is fragmented at the source device, this header is used with fragmentation information, similar information to the IPv4 header; the device must know the MTU values on the path),
- *Encapsulating Security Header* (50, encryption information),
- *Authentication Header* (51, authentication information),
- TCP segment (6), UDP segment (17), ICMP packet (58), ...

**Remark**

Note that in the last item of the previous list there are codes for TCP segment, UDP segment, ICMP packet, etc. This implies that the *next header* field in an IPv6 packet also acts as a protocol field from an IPv4 packet – it tells what is encapsulated inside (interestingly, there is a different identifier for ICMPv6 than for ICMPv4).

So the real meaning of the *next header* field is: it specifies what specifically follows the header in which we are reading this field. This can be either some extension header or directly the data passed by a particular protocol from the transport or network layer.



Fragmentation of an IPv6 packet can only be performed by the sending device; no device on the path can perform it. If any intermediate network equipment (router or switch with L3 functionality) detects that the IPv6 packet is larger than the MTU on the path allows, then it drops the packet, it has no other alternative. However, the sender can fragment, in which case it will use the optional header number 44, which is designed for this purpose and contains similar information to the IPv4 packet in the second line of Figure 4.1.

4.4 ICMP Protocol and Control Messages

4.4.1 Purpose of ICMP

Internet Control Message Protocol (ICMP) is also an L3 layer protocol. Its purpose is to provide sending short messages, usually represented by one or two numbers (each such number has a specific meaning), exceptionally with additional data information. Typically, information about errors, problems on the path, or other alerts are sent in this way.



Each message sent using ICMP has a *type number*. The following table lists some common values for ICMPv4:

Type	Meaning
8	<i>Echo Request</i> – request for response (sent when ping is used to find out if a particular device is available)
0	<i>Echo Reply</i> – reply to <i>Echo Request</i> (8) with the meaning “yes, I am here”
3	<i>Destination Unreachable</i> – destination unavailable message (if a router receives an IP packet it cannot deliver, it sends ICMP packet with message 3 back to the source device)
11	<i>Time Exceeded</i> – the wait time has expired; either the TTL value in the IP packet has dropped to 0, or the timeout has expired while waiting for the remaining IP packet fragments, etc. (sent by the router to the source of the IP packet)
12	<i>Parameter problem</i> – there is some problem in the IP packet (usually in its header) that cannot be reported by another ICMP message

Table 4.1: Selected messages of the ICMPv4 protocol




The message type itself is too general in some cases, so the message code can be used to detail it. For example, for message type 3 (*Destination Unreachable*) there are several codes that specify


why the destination device is unreachable:


- code 0 – the destination network is unreachable (the router is not able to find the given network in its routing table),
- code 1 – the destination device (host) is unreachable (the packet has been delivered to the destination network, but the destination device cannot be found in the network),
- code 2 – the destination protocol is unreachable (there is an unknown value in the *Protocol* field),
- code 3 – the destination port is unreachable (this means the port number specified in the encapsulated transport layer segment, for example the specified port is not open or no application is listening on it),
- code 4 – the packet is larger than the MTU on the path, but cannot be fragmented because the DF (Do not Fragment) flag is set,
- etc. (there are a total of 15 different codes for message 3).

For messages of type 11 (*Time Exceeded*), the codes 0 (TTL field value is 0) or 1 (waiting time for remaining fragments exceeded) are usually used.

 So for now, we have the pair message number + code. But the message can be further specified, for example a part of the IP packet that is responded to in this way (the IP header and the first 8 octets of the data part) are appended.

 **Remark**

In fact, even ICMP packets that don't really need a data part have a data part – they add padding because there is a minimum length for encapsulated data (payload) at lower layers. We meet this, for example, in the ICMP message *Echo Request* (8). 

 An ICMP packet has a very simple structure – no addresses, packet length, etc., the header has only three fields:

- *Message type* (ICMP Type, 8 bits) – specify the type of message, e.g. number 8 for Echo Request,
- *Code* (Code, 8 bits) – identifying code,
- *Checksum* (Checksum, 16 bits) – checksum over the previous two header fields.

The format of the entire ICMP packet can be seen in Figure 4.6.

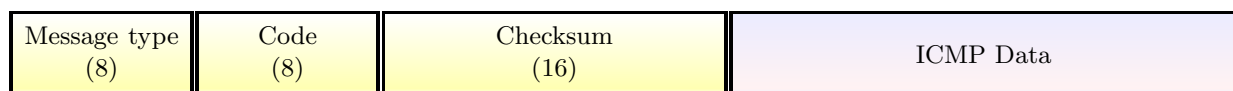



Figure 4.6: ICMP message format

 The ICMP message is encapsulated into IP packet – the ICMP packet itself has no addresses or other important items in its header. In Figure 4.7 we can see the format.

Some message types use TTL = 1, but most use the TTL value set on the sending system (i.e., usually 128 or 64).

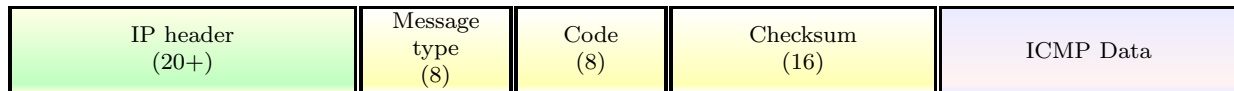



Figure 4.7: ICMP message encapsulated in IPv4 packet


4.4.2 ICMPv6

 When using IPv6, we also need to use ICMPv6. Compared to ICMPv4, some messages have been added, and the scope of this protocol has been expanded (which is related) – ICMPv6 combines the roles of the three original protocols: ICMPv4, ARP and IGMP.


But far from just adding message types for new uses of the protocol, many of the original types were renumbered. Already in ICMPv4 some messages were error messages and others were informational, ICMPv6 carried this distinction over to the numbering of message types – the message type field is also 8-bit, but the first (leftmost, most significant) bit is set to 0 for error messages and to 1 for informational messages.

Type	Meaning
128	<i>Echo Request</i> – request for response, originally number 8
129	<i>Echo Reply</i> – response to <i>Echo Request</i> (128), originally number 0
1	<i>Destination Unreachable</i> – destination unavailability report, originally number 3
3	<i>Time Exceeded</i> – waiting time expired, originally number 11
4	<i>Parameter problem</i> – there is some problem in the IP packet that cannot be reported by another ICMP message, originally number 12

Table 4.2: Some ICMPv6 messages that were also in ICMPv4

 There are several message types in Table 4.2 that can be found in both versions. Note that these messages have different numbers in ICMPv6 compared to ICMPv4, and that the new numbering retains the rule that the most significant bit of the 8 bits of the first field is set to 0 for error messages (i.e. the type number is in the range 0–127, the last three rows of the table) and to 1 for information messages (range 128–255, the first two rows of the table). There are also changes in the codes for some specific message types.

The newly added message types are usually related to either IP to MAC address mapping (which was the job of ARP in the older version) or group management (originally in IGMP), and a few message types for regular or mobile networks. The most important of these are in Table 4.3.

 For instance, there is the interesting message *Packet Too Big* (2), which was not in the previous version, although it would have been useful there too. If a router receives a packet larger than the MTU value on the next path to forward, it behaves differently for various IP versions.

- It is an IPv4 packet: either it tries to fragment it, or (when DF flag = 1) it drops the packet and sends the ICMPv4 message *Destination Unreachable* (3) to its source.
- It is an IPv6 packet: in either case, it will drop it and send an ICMPv6 message *Packet Too Big* (2) to its source.

Type	Meaning
2	<i>Packet Too Big</i> – if the router receives a packet that is too large to forward (larger than the MTU value on the sending port), it drops the packet and sends this ICMP message to the original sender
134	<i>Router Advertisement (RA)</i> – routers send this ICMP message at regular intervals with information about themselves and the network, such as the network address and prefix length
133	<i>Router Solicitation (RS, request for RA)</i> – a station can use this message to force the router to send a Router Advertisement (134) message out of the interval
135	<i>Neighbor Solicitation</i> – we have the IP address of a device in the neighborhood (somewhere in our network), we use this message to ask our neighborhood for the related MAC address (i.e. “Who has this IP address?”, the neighbor who recognizes his IP address will reply with the following message:
136	<i>Neighbor Advertisement</i> – in response to the previous message (if I know my IP address) I will report my MAC address
130	<i>Multicast Listener Query</i> (query to group members) – sent by the router when verifying who in its network belongs to a specific group and therefore receives packets addressed to a given multicast address
143	<i>Multicast Listener Report</i> (group membership notification) – sent by a device when it subscribes to a group (informing the router that it wants to receive packets with a given multicast address as the destination) or in response to the previous message

Table 4.3: Some ICMPv6 messages that are not in ICMPv4



Tasks


A list of all ICMP message types and their corresponding codes is on the web. Browse the following addresses and compare the data for ICMPv4 and ICMPv6.

- IPv4: <http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml>
- IPv6: <http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml>



4.4.3 Reachability Testing

The *ICMP Echo Request* and *ICMP Echo Reply* messages are used to test device reachability.

 If we need to test the reachability of a device on the network, we usually use the `ping` command, which sends the ICMP message *Echo Request* (number 8 or 128 depending on the version). The `ping` command sends several packets with this message, expects a response to each one, and then calculates statistics based on how long it took to respond to each packet. This command can be found in any operating system; it came to Windows from Unix, as can be seen in its syntax.

The `ping` command sends an ICMP message *Echo Request* in the following format:

Each line is 32 bits long. The fields on the first line are part of the ICMP header, from the second line onwards they are data specific to this message type. For *ICMP Echo Request* we send

- *identifier* (16 bits) – for all packets of the same ping call this is the same number,


Type = 8	Code = 0	Checksum
Identifier		Sequence number
optional data, pad		

Table 4.4: ICMPv4 *Echo Request* message

- *sequence number* (16 bits) – the sequence number of the packet sent by the `ping` command, so if we send four packets, there will be numbers incrementing by 1, if we run the command again, it continues the sequence (it doesn't start numbering again), it's used to match the packet and reply to it,
- *optional data* (ICMP Data) – a meaningless data padding, usually 32 or 48 octets long.


Optional data is self-determined by the device. It can be an alphabet or a sequence of special characters and digits.

The device under test responds with an ICMP message *Echo Reply*, which more or less copies the query field (changing the contents of the *Message Type* field and of course the checksum). This message is delivered back to the requesting device, which uses the value of the *sequence number* field in each response packet to compute the times of sending the request and receiving the response and calculate the corresponding statistical information.

 The `tracert` mechanism (in Windows `tracert`) also works with ICMP packets. Unlike the previous command, this is not so much about finding out the availability of the device in question, but rather about mapping the path (tracing, hence the name). If the destination is reachable, a list of all routers on the path is printed, including the duration of the transmission from the previous network node; if the destination is unreachable, a list of routers on the part of the path that is “ok” is printed.


Using `tracert` we can find out:

- which way (through which routers) packets go to a certain destination,
- which part of this path is the slowest (it could mean a bandwidth problem in the network),
- from which router the path becomes problematic.

 In general, this mechanism works as follows (see Figure 4.8):

- packets are sent from our device with the destination address of the “receiving” device, having the TTL set to 1, 2, 3, etc.,
- on each L3 layer device on the path, the TTL value is decremented by 1,
- on that L3 device where TTL = 0 after subtracting 1, the packet is dropped,
- we get back an ICMP Time Exceeded message (number 11 in IPv4, number 3 in IPv6),
- when a packet finally reaches the destination, we get an acknowledgement response and stop sending test packets.

More than one packet is usually sent for the same TTL, usually three packets for each TTL.

 In Windows, test packets are sent in the form of ICMP messages *Echo Request* encapsulated in an IP packet with a given increasing TTL, the acknowledgement from the destination is an ICMP message *Echo Reply*. So by this message we know that it is the last node on the path.

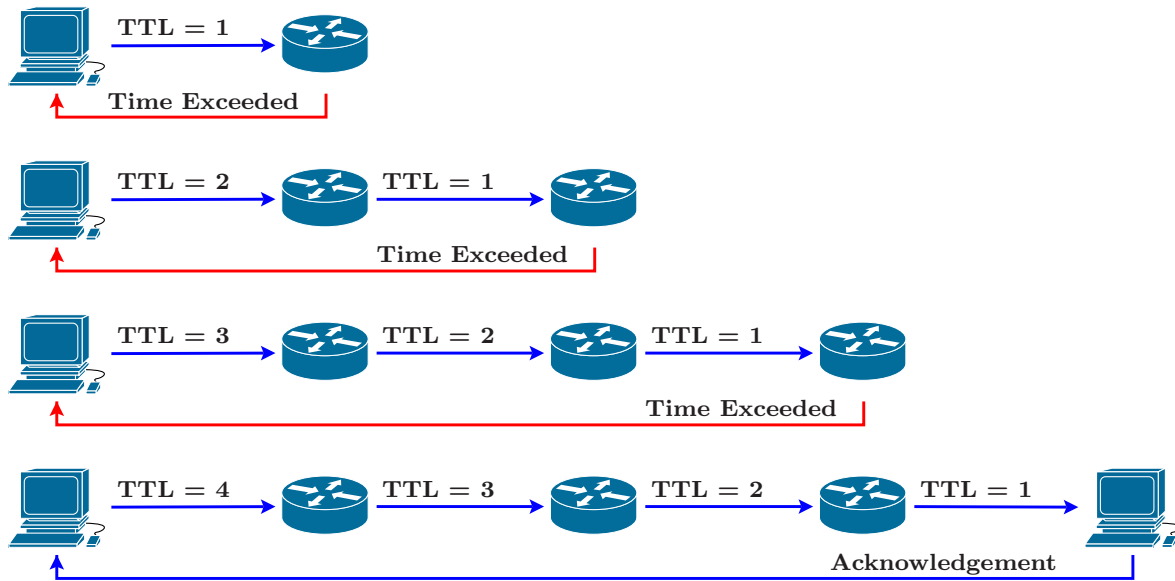



Figure 4.8: Traceroute mechanism

 In Linux, we can choose – by default, these are UDP segments encapsulated in an IP packet with a given increasing TTL, using a “unreal” number as the port number in the UDP segment. Therefore, the acknowledgement from the destination ICMP station is a *Destination Unreachable* message with the code *Port Unreachable*. We can recognize from this packet that our request has reached the destination. In the command parameters, we can choose a different kind of test packets – either ICMP packets like in Windows or TCP segments.


 **Remark**

Many servers and other publicly available network devices do not respond to the `ping` and `traceroute` (`tracert`) commands. This is because either the device itself or the firewall on the path has been configured not to respond to ICMP Echo Request messages or UDP segments with a non-existent port number. The reason for this is security, as these mechanisms are often exploited by hackers to map networks of “interest”.

It is usually a good idea to let these packets respond at least when they come from the internal (trusted) network, but this is up to the administrator.

However, in Linux we can choose TCP segments using the well-known port 80 (HTTP protocol), which usually passes through firewalls. 

These test mechanisms operate at the L3 layer, so if there is a response and availability problem at another layer (either the higher layer or any device on the path in the lower layer), we have no way of knowing.

 **Additional information**

<ftp://ftp.hp.com/pub/hpcp/UDP-ICMP-Traceroutes.pdf>




4.5 Transport Layer

The transport layer is a mediating layer between the transmission-oriented layers (L1–L3) and the application-oriented layers (L5–L7). Although it participates in the negotiation of data transfer (thus close to the lower layers), it does not address devices or networks in any way. On the other hand, it is involved in determining the specific application being communicated (making it close to the higher layers), but it is completely careless about the format of the data and treats all kinds of data basically the same; an application is simply a number to the protocols of this layer, nothing more.

4.5.1 Port Numbers

At the transport layer, no computers or networks are addressed (downward), but applications are addressed (upward). Each application must be uniquely identified by a *port number*, and since there are two communicating parties, we need a source port number and a destination port number. Again, recall that the term port at the L4 layer is not the same as the term port at lower layers.

 The port number takes up two octets, giving us a range of 0–65 535. From this, the

- *well-known* ports in the range 0–1023, these numbers are used for specific commonly known services on servers (WWW/HTTP, FTP, RPC, SQL, Syslog, Kerberos, etc.),
- *registered ports* (official) in the range 1024–49 151, which various companies can register with the IANA organization (for example, Microsoft, IBM, Citrix, Novell, Cisco, Oracle, Eset have their registered ports), this also includes ports for RADIUS, Nessus, or BOINC,
- *dynamic* (private) ports in the remaining range 49 152–65 535 are used on the client side, which communicates with the server.



Remark

Why this differentiation? The first two groups of ports are intended for the server side of communication, while the server side usually runs a single instance of the service (application) in question – for example, a web server runs a single application that provides web services (e.g. Apache or IIS), so one single number is sufficient for the application on the device (server). However, on the client side there can be multiple applications of the same type communicating – for example, multiple web browser windows (actually multiple web browsers) can communicate with the web server, and we can have multiple tabs in each window displaying pages from different web servers.

If we wanted to have only one number on the client side, there would be no way to determine which process/window/tab is specifically addressed for the web page that just arrived from a web server.



Figure 4.9 indicates the meaning of ports for distinguishing individual established connections, of which there may be more than one for a given protocol within a regular computer. When communicating with a web server (which can be, for example, an Apache server), the server side runs the Apache service communicating on TCP port 80 (one of the *well known ports*), whereas on the computer side we have multiple web browser windows or tabs for which we need different port numbers, from the *dynamic (private) port* range.

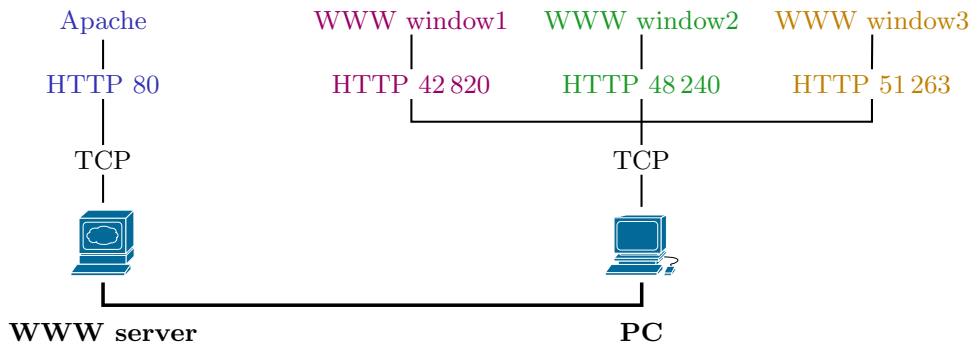


Figure 4.9: The meaning of ports on the transport layer



Additional information


The official list of well-known and registered port numbers is at


<http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>





4.5.2 Transport Layer Protocols


The TCP and UDP protocols work at the transport layer, but for certain specific types of tasks there are others.


 *TCP* (Transmission Control Protocol) provides a reliable connection-oriented service, guarantees ordering, i.e. delivery of multiple data segments in the correct order. It is only used for point-to-point connections, only one end device communicates with another end device at any time.

 *UDP* (User Datagram Protocol) provides an unreliable connectionless service, i.e. a datagram service. It can also be used for point-to-multipoint connections, i.e. the destination can be a group of end devices.

 Other protocols at the transport layer exist: *SCTP* (Stream Control Transmission Protocol), which is similar to TCP, but allows multiple independent parallel connections (streams) to be established, reducing the probability of losing PDUs along the way – each stream can take a different path, and if a device has multiple IP addresses, it can switch between them. This protocol is implemented in the kernel of many Unix systems (FreeBSD, Linux, etc.); third-party implementations exist for Windows.

 The *RTCP* transport protocol is used for resource reservation mainly in communication requiring prioritization in resource allocation (quality of service management), it is encountered in some VoIP technologies.

 The *DCCP* protocol provides a connection-oriented service (like TCP) but does not guarantee delivery in the correct order nor acknowledge (like UDP), its main purpose is to ensure fast data delivery. To this end, it provides a congestion control mechanism (it can adaptively change transmission paths to avoid congested areas). It is used, for example, by Internet radios, some Internet telephony technologies, online video.

 PDUs at the transport layer are usually called *segments*, although in the case of protocols providing datagram service (such as UDP) we also see the name *datagram*.

4.5.3 TCP

 The tasks of the Transmission Control Protocol (TCP) are:

- to establish a connection with the other side (create a virtual channel),
- maintain the connection, acknowledge received segments, control the data flow,
- take an SDU from a higher layer (usually from an application protocol, like HTTP) and process it that way:
 - check the length of this block of data, and if it is greater than the allowed segment length, it breaks it into smaller portions as needed – to *segment*,
 - add a TCP header to each of the portions created in the previous step, specifying port numbers for both communicating sides, thus creating TCP segments,
 - pass the TCP segments sequentially to a subordinate layer (usually the IPv4 or IPv6 protocol),
- terminate the connection at the end of the communication.

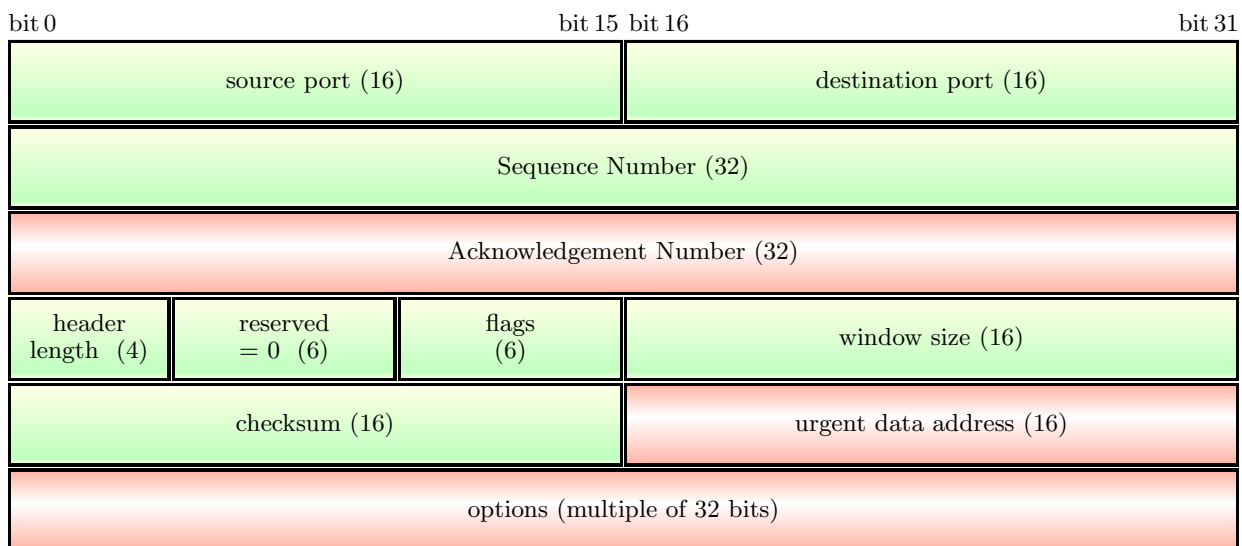




Figure 4.10: TCP segment header

 In Figure 4.10 we can see the TCP segment header. The individual fields have the following meaning:

- *Source Port, Destination Port* (16 bits each) – port numbers on the source and destination device.
- *Sequence Number* (32 bits) – a sequence of data from a higher layer can be split into multiple segments, here is the octet number from the original data stream that the segment encapsulated in this particular segment starts with (octets are numbered from 0).
- *Acknowledgement Number* (32 bits) – the octet number of the beginning of the part of the original data that we are expecting (it can belong either to the next segment in the sequence

or to a previous segment that was not properly delivered). This field is in every segment except the first segment when the connection is established. Header Length/Data Offset (4 bits) – the length of the entire header in multiples of 32 bits (so “number of rows” of the table as shown in 4.10). *Flags* (Control Bits, 6 bits) – array of bits (Wireshark combines them with the previous array of reserved bits), each of which has its own meaning, we will be particularly interested in these flags:

- URG (Urgent) – Urgent data flag (there is data in the segment that has priority in the target during processing), the *Urgent data* field is to be taken into account, ACK (Acknowledgement) – this segment is (among other things) an acknowledgement of the previous segment, the *Acknowledgement Number* field is to be taken into account,
 - SYN (Synchronize bit) – used when establishing a connection,
 - FIN (Finish bit) – used when terminating a connection.
- *Window Size* (16 bits) – specifies the maximum number of octets the sender of this segment wants to receive from its other party without acknowledgement (i.e. after sending so many octets, it sends the acknowledgement segment).
 - *Checksum* (16 bits) – counted over the entire segment including the header plus *pseudo-header*; the pseudo-header is created for this purpose only (on both the source and destination device) and is not transmitted; it contains the most important data from the PDU in which the segment is encapsulated – usually IP (source and destination IP addresses, protocol – 6 for TCP, and TCP segment length).
 - *Urgent Pointer* (16 bits) – if there is also urgent data in the segment and thus the URG bit is set, this is the octet number that *end* the urgent data.
 - *Optional* (multiple of 32 bits) – options that devices need to pass to each other (usually at the beginning of the connection or when parameters need to be changed during the connection), such as maximum segment size, timestamps (for synchronization), or acknowledgement parameters.

 TCP prescribes acknowledgement of delivery, which is what the *Sequence Number* and *Acknowledgement Number* fields are for. These numbers prescribe some continuity between what one sends and what the other receives, and it works the same way in the other direction (so the *Sequence number* going in one direction has nothing to do with the *Sequence number* in the other direction, both sides can send data).

The *Window Size* determines after which quantum of data the acknowledgement should be sent, and usually takes the size of several segments – that is, each segment is not acknowledged separately, but the acknowledging segment is sent after several segments have been delivered. This value usually corresponds to the size of the buffer, but tends to be smaller if many segments are lost along the way.

So how does the confirmation process work? The use of *Sequence Number* is obvious – if I’m sending data, I use this field to specify where the particular part being sent is localized in the original data stream. The *Acknowledge Number* field specifies which part of the data I expect in the other direction.


If we take the *Sequence Number* value from one segment and add the length of the data encapsulated in that segment (i.e. segment length minus header size, both in octets), we get the


Sequence Number value for the next segment. The length of the segment is found in the IP header.

The target device receives as many segments as fit in the window width and checks the *Sequence Number* values to see if they are in the correct sequence. If everything goes as it should (no segments are lost), it sends an acknowledgement segment to the other side, where the *Acknowledgement Number* field contains the number calculated by the previous procedure from the last received segment (it takes the *Sequence Number* and adds the length of the encapsulated data). This corresponds to the next segment to be received.

How do we know if a segment has been lost along the path? In a number of delivered (not yet acknowledged) segments, we perform the calculation outlined above. If the result of one segment does not match the *Sequence Number* of the next segment in the sequence (there is a larger number in the segment than we got), then it means that at least one segment is missing at that location. Therefore, we send an acknowledgement segment with the *Acknowledgement Number* value that we got for the missing segment.

4.5.4 TCP Connection

 The TCP connection must be established first, which is called a *Three-way Handshake*. It is so called because three segments are sent sequentially during the connection initiation with the negotiation of the connection parameters (without data).

 *Connection Establishment* for communication with the web server is shown in Figure 4.11:

1. The client sends the first segment meaning “I want to establish a connection”. This segment has the SYN flag set and contains the initial parameters in the *optional* field, such as the timestamp for synchronization and the maximum segment size that this device can accept.
2. The server sends its first segment in response (which is why it also has the SYN flag set), which is therefore a response and also has the ACK flag set. The *Optional* field has similar type of information as was in the previous packet.

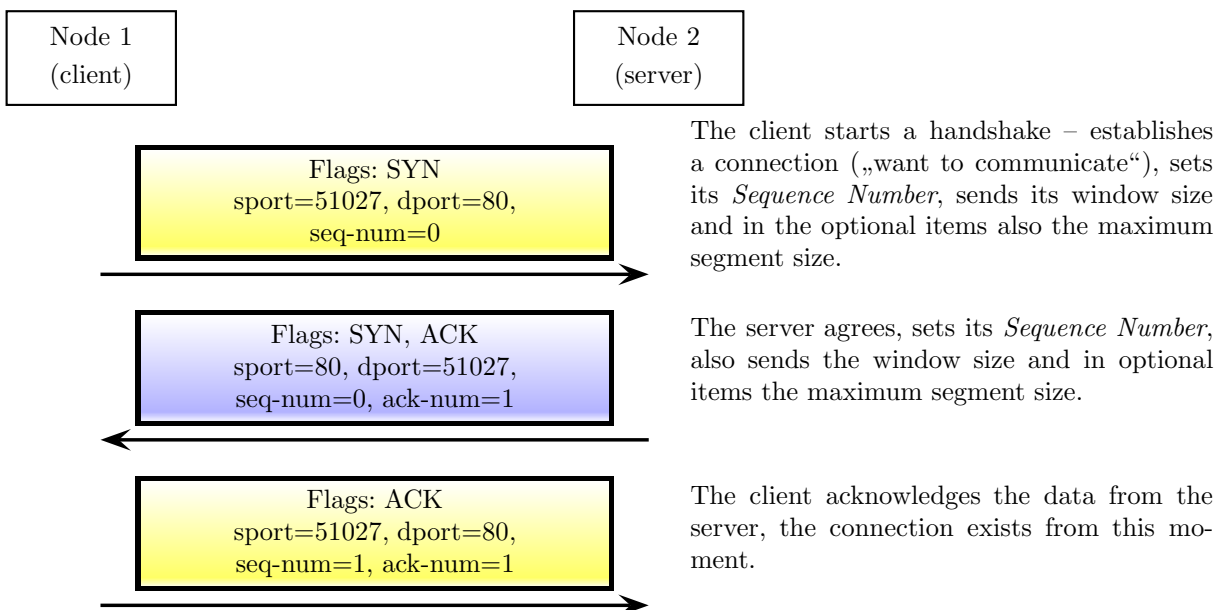


Figure 4.11: TCP handshake – establishing a connection to the web server

- The second segment acknowledged receiving the first one, so the second segment still needs to be acknowledged. The third segment has just this function. It is no longer initializing and does not contain synchronization information, so it has only the ACK flag set. The *Optional* field is usually not used.

The port numbers are obvious – the server uses port number 80 because it’s a WWW server, on the client side we see a dynamic (private) port.

Remark

In the first two segments, the sender sets its default value for *Sequence Number* (Wireshark displays 0 as a relative number instead), in the third segment *Sequence Number* is 1 higher than in the first segment. The *Acknowledgement Number* is used only from the second segment onwards (there is nothing to acknowledge in the first segment) and in the second and third segments it is used 1 higher than the *Sequence Number* in the previous (received) segment. After the connection is established (i.e. from the fourth segment onwards), these numbers are used as described above.

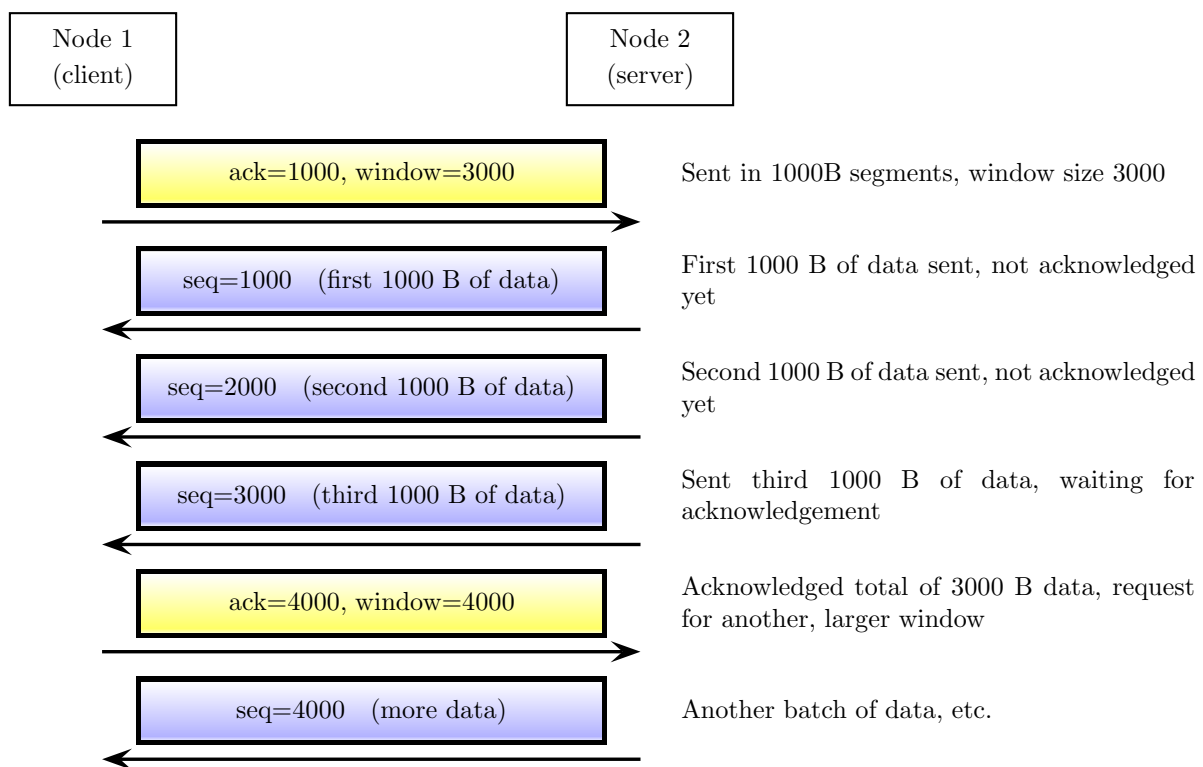


Figure 4.12: Common communication within TCP connection


 If no segments are lost, the acknowledgement is always sent after as many segments as can fit in the window width. In Figure 4.12 we can see the flow of such communication for the case that the segment size is set to 1000 and the window size to 3000 (we assume that the first 1000 B have been sent before). In the acknowledgement segment, *Acknowledgement Number* is set to 4000, because we are asking to send a segment with this *Sequence Number* (thus acknowledging that all previous ones have been delivered).

Figure 4.13 shows a situation when a segment was lost. The data receiver has detected that

the segment for *Sequence Number* 2000 is missing in the segment header, so it enters this number as *Acknowledgement Number* in the acknowledgement segment. If other segments after the lost segment have already been delivered, they will still be delivered again.

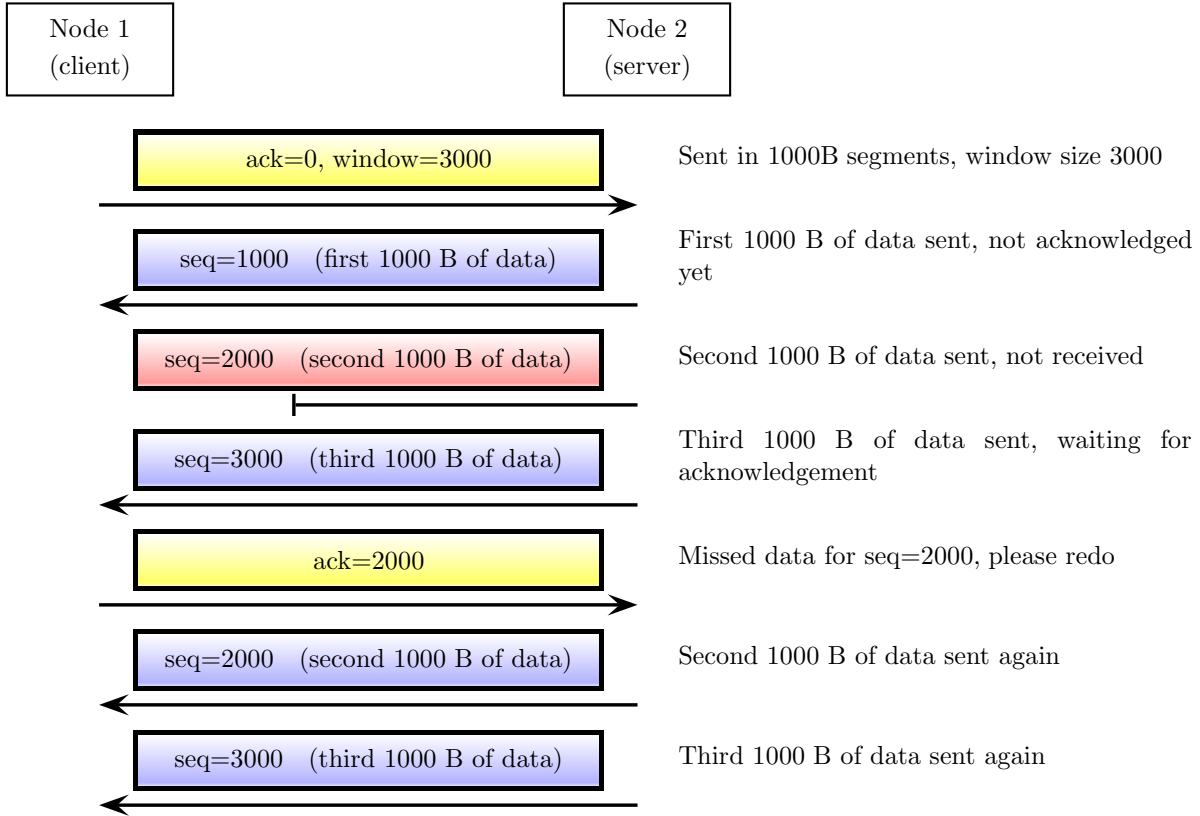


Figure 4.13: Communication within TCP connection, one segment did not arrive


So, in theory, it works as described above. However, nowadays the devices usually handle the SACK (Selective Acknowledgement) function, where only those segments that did not arrive originally are sent (i.e. they fill up “holes”). However, this function must be negotiated by both devices during the three-way handshake.

In Figure 4.14 we can see part of the parameters from the first segment for establishing the connection (in Wireshark). In the Options field, the client sends information about the maximum segment size it accepts, the fact that it can use the SACK function, the actual window size (a multiple for the number specified in the cell for this value), and more.

 **Remark**

It is also important when the data receiver starts to determine whether it has all segments belonging to the current window. There is a maximum waiting time (timeout) for the delivery of segments, and if this time expires, the undelivered segments are considered lost.



 The termination of the connection can be done by any of the communicating devices, and is done by four segments, as indicated in the figure 4.15. First the termination is announced by one party (the FIN flag is set), then the other party acknowledges and sends its own terminating seg-

```

> Internet Protocol Version 4, Src: 192.168.1.3, Dst: 63.116.243.9/
< Transmission Control Protocol, Src Port: 58816, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 58816
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 3851697578
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
  > Flags: 0x002 (SYN)
  Window: 5840
  [Calculated window size: 5840]
  Checksum: 0x9de2 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  < Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  > TCP Option - Maximum segment size: 1460 bytes
  > TCP Option - SACK permitted
  > TCP Option - Timestamps: TSval 1545573, TSecr 0
  > TCP Option - No-Operation (NOP)
  > TCP Option - Window scale: 7 (multiply by 128)
  < [Timestamps]

```

Figure 4.14: SACK negotiation

ment (also with the FIN flag), followed by the acknowledgement of receiving. Both acknowledging segments have only the ACK flag set.

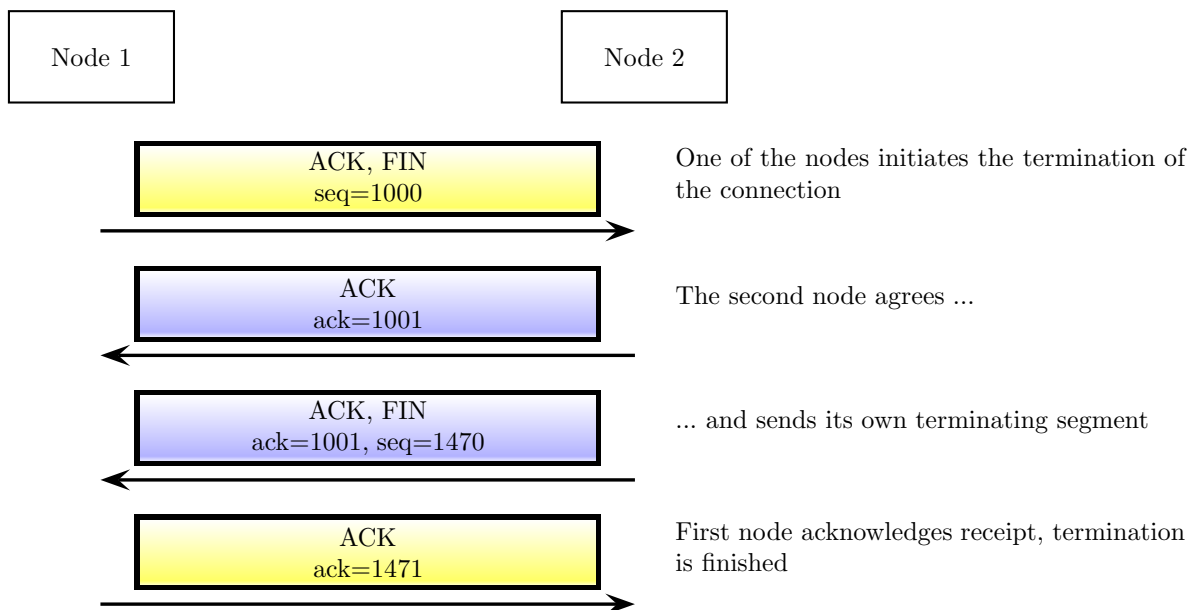



Figure 4.15: Termination of TCP connection

4.5.5 UDP

 As written above, the User Datagram Protocol (UDP) provides an unacknowledged, connectionless service, a simple datagram service. Its tasks are therefore much simpler than TCP – it simply takes an SDU from the higher layer, adds a header, and passes the created segment (datagram) to the lower layer. No connection is established, no acknowledgement is performed, no

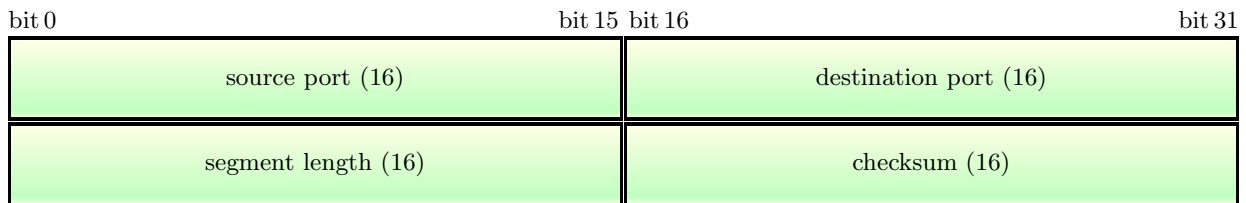


Figure 4.16: UDP segment header

window size is handled, no numbers are calculated for missing segments (in fact, no such numbers are even used).

That’s why the UDP segment header is much simpler, we just don’t need some of the fields there. Figure 4.16 shows the header of a UDP segment – the first line is the same as TCP, the second line is the same checksum, the length of the entire segment (including the header) is “extra” – this is not the case with TCP, and all the other TCP fields are missing.

 The UDP protocol is used when


- is unnecessary to establish a connection (or it is not possible or desirable for some reason),
- sending little data that does not need to be segmented into multiple segments,
- we want the transmission to be fast and not overload the link,
- we want to send data to a multicast or broadcast address (TCP supports only unicast transmissions).


	TCP	UDP
Connection	connection-oriented	connectionless
Acknowledgement	yes	no
Reliability	yes	no
Segmentation	yes	no
Speed	slow	fast
Destination	unicast	unicast, multicast, broadcast


Table 4.5: Comparison of TCP and UDP

Chapter 5

Network Addresses and Routing

 *Quick preview:* In this chapter, we will return to the network layer of the ISO/OSI model (Internet in the TCP/IP model), i.e. L3, but especially at the beginning, we will also move to the L2 (data-link) layer or the border between them. We will focus on working with addresses, address resolution, IP address types, working with subnets, and also the basics of routing and using routing protocols.

 *Keywords:* ARP, NDP, neighbors table, class, loopback, private address, subnetting, VLSM, CIDR (supernetting), DHCP, dynamic allocation, autoconfiguration, EUI-64, SLAAC, privacy extensions, NAT, PAT, routing, routing table, metrics, autonomous system, routing protocol, routing algorithms, RIP, EIGRP, OSPF, BGP

 *Objectives:* After studying this chapter, you will understand the problem of adding MAC and IP addresses to a frame/packet, you will be able to work with IPv4 address ranges, you will learn how to obtain IPv4 and IPv6 addresses, you will understand the problem of IP address translation, you will learn the principle of routing packets between routers, you will understand the operation of routing algorithms.

5.1 Addresses in Frames and Packets

When sending an IP packet, we need the IP address of the source (ours) and the receiver, but we must remember that the IP packet is to be encapsulated in a frame (e.g. Ethernet), and the MAC addresses of the source and destination are also listed there.

In the frame that we send (and is destined outside our network), we put our own MAC address as the source, but what to put the destination one? Usually we can't use the MAC address of the receiver because we don't know it. We only have limited information about the receiver (e.g. a server), an IP or name address, and there might be more than one physical device under the same name or IP address, for example for load balancing or maintenance reasons. The only MAC address we know or can quickly find out is the gateway MAC address.

Example

Figure 5.1 shows the structure of a network through which we have to send the packet. The source device is a computer with IP address 1.1.1.1, the destination device is a computer with IP address 7.7.7.7.

The structure shown is very simplified: as we know, each network interface has its own MAC address and, in the case of a router, its own IP address. So e.g. the router at the top should correctly have different address information for each of the two active interfaces.

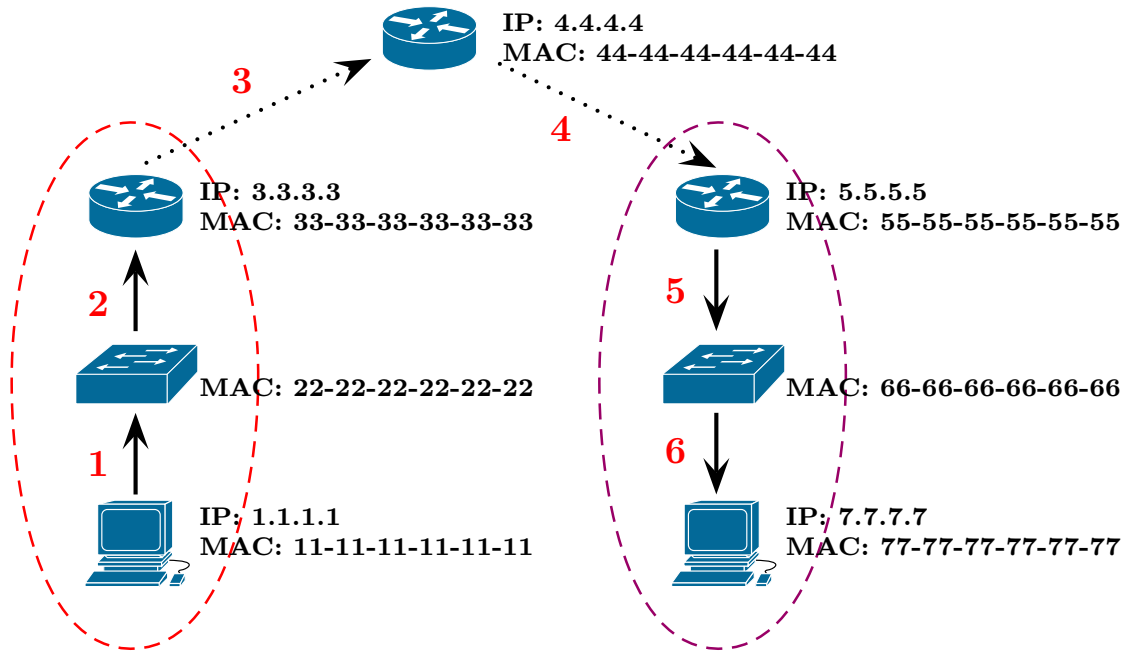


Figure 5.1: Relation between L2 and L3 addressing

How it works:

- On the first part of the path, the PDU goes to the switch, which sees only the frame; the frame and the packet encapsulated in the frame contain addresses:

Source:	IP 1.1.1.1	MAC 11-11-11-11-11-11
Dest:	IP 7.7.7.7	MAC 33-33-33-33-33-33

sender addresses
receiver IP address, gateway MAC address

Note that the IP and MAC address of the destination do not belong together, but that doesn't matter at all.

- On the second part, the PDU is getting to the gateway and the addresses have not changed yet:

Source:	IP 1.1.1.1	MAC 11-11-11-11-11-11
Dest:	IP 7.7.7.7	MAC 33-33-33-33-33-33

sender addresses
receiver IP address, gateway MAC address

- The third part of the path is already outside the sender's network. The gateway router received and de-encapsulated the frame, unpacked the embedded IP packet, and re-encapsulated it again, this time with different MAC addresses.

Source:	IP 1.1.1.1	MAC 33-33-33-33-33-33
Dest:	IP 7.7.7.7	MAC 44-44-44-44-44-44

sender IP address, MAC address for the next part
receiver IP address, MAC address for the next part

- It is similar on the fourth part of the path:

Source:	IP 1.1.1.1	MAC 44-44-44-44-44-44	sender IP address, MAC address for the next part
Dest:	IP 7.7.7.7	MAC 55-55-55-55-55-55	receiver IP address, MAC address for the next part

- Then we are already in the receiver’s network. The gateway in the receiver’s network “knows” the MAC addresses of the nodes in its network.

Source:	IP 1.1.1.1	MAC 55-55-55-55-55-55	sender IP address, foreign gateway MAC address
Dest:	IP 7.7.7.7	MAC 77-77-77-77-77-77	receiver’s addresses

- On the sixth part we just stay on the L2 layer and it makes no sense to change the addresses.

Source:	IP 1.1.1.1	MAC 55-55-55-55-55-55	sender IP address, foreign gateway MAC address
Dest:	IP 7.7.7.7	MAC 77-77-77-77-77-77	receiver’s addresses



It follows that we really don’t need to know the destination’s MAC address, we just need the IP address, and the destination in turn doesn’t need to know our MAC address. The destination does get a frame containing “some” MAC address, but that would only be ours if we are on the same network. There’s also another piece of information – the switch we’re directly connected to is actually “invisible” to us, we can’t access its MAC address in frames (and usually not its IP address either).



Definition (Internetworking)

Internetworking is a mechanism of networking, i.e. providing communication between (inter) networks. This mechanism takes place at the L3 layer, on devices such as routers or switches with L3 layer functionality.




5.2 Neighbors Discovery

5.2.1 Neighbors Table


Consider this situation: we know our IP address, mask (or prefix length) and gateway IP address. We need to send a packet to a foreign network (we know the destination IP address), and thus through the gateway. But where “in which direction” is the gateway? Which network interface is it accessible through and what is its MAC address?

We have to solve a similar problem when we want to send an IP packet to someone in the internal network. For example, I know the IP address of the given device, but what is its MAC address and through which port is this device accessible?

As for “direction”, we have different mechanisms for that, now we will focus on mapping the IP address to the corresponding MAC address.

 The mapping of IP addresses to MAC addresses is performed by the Address Resolution Protocol (ARP) in the case of IPv4 and the Neighbor Discovery Protocol (NDP) in the case of IPv6. The basic functions of these protocols are:

- to keep a neighbors table,
- to update this table using ARP/NDP queries.

 In both cases, the device keeps a *neighbor table* with information about the nearest network neighborhood. In the neighbor table we have the following information for each “neighbor”:

- IP address (IPv4 or IPv6),
- MAC address,
- type (static/permanent or dynamic/learned).

The first two entries map the neighbor’s IP address to its MAC address, the third entry determines how the entry got into the table (whether it was statically inserted or dynamically queried).


One table is maintained for each network interface with its own IP address (including virtual interfaces if virtualization software is installed).



Remark

End devices on the network have one important entry in this table – the address of the gateway to which they direct most of their traffic.




 In the case of Windows, the ARP table and the NDP table are somewhat cluttered – in addition to the gateway, there are also multicast addresses and, in the case of ARP, broadcast addresses including mapping to MAC addresses. The contents of the tables can be displayed as follows:

- `arp -a` for IPv4 ARP table,
- `netsh interface ipv6 show neighbors` IPv6 NDP table,
- `get-netneighbor` for the both in the PowerShell.

(The `netsh` command can also be used to display the ARP table in IPv4 if you type `ipv4` instead of `ipv6`, but who would bother with that when you don’t need . . .).

These commands can also be used to affect the neighbor table (adding new entries or removing them), but in most cases these tasks are handled dynamically by the protocols.

 In Linux and other UNIX systems, the following command is used for displaying the ARP/NDP table:


```
ip neigh show
```

(so we can write `ip neighbor show` or `ip n s` or something in between, but it’s best to compromise – don’t write too much and write enough to be understandable). Also, this command can be used to influence the contents of the table.

Also in UNIX systems there is a command `arp` (to display the table, just enter it without parameters), we can use it when we really only care about IPv4 addresses.

5.2.2 Protocols


The role of ARP and NDP protocols is not only to maintain a table of neighbors, but also to dynamically query what should be in that table when needed. Thus, there are PDUs sent by the given protocol.

 *ARP packets* (i.e. for IPv4) are encapsulated directly into Ethernet II frames or Wi-fi wireless frames. We distinguish between ARP request and ARP response.

If a device needs to find out the MAC address to a certain IP address, it sends an ARP packet with the query “Who has xxx? Tell yyy.”. Since we don’t know the MAC of the receiver (we are still finding it out), the query is sent as a universal broadcast (i.e. to `FF-FF-FF-FF-FF-FF`).

We are asking for someone else’s IP address and want to send the answer to our IP address. Every ARP packet is the same length, no matter which direction it goes. The query and response have all the fields (own IP and MAC address, queried IP and MAC address), but in the query what the sender doesn’t know is filled with zeros.

The ARP request (in the form of a broadcast, i.e. for everyone in the network) is captured, among others, by the owner of the queried address, who fills in the missing field (i.e. his MAC address) and sends it back, this time as a unicast (he knows the requestor’s address).

 *NDP packets* (the mechanism for IPv6) do not have their own specification, they are sent as ICMPv6 packets with special message types and encapsulated in IPv6 packets (like any ICMP message). NDP is a more complex protocol than ARP, it performs more tasks. As far as ICMPv6 messages are concerned, the following are primarily related to neighbors:

- *Neighbor Solicitation* (ICMP message No. 135) – request for neighbor information, meaning corresponds to ARP request,
- *Neighbor Advertisement* (ICMP message No. 136) – response to a previous request.


In contrast to ARP, the format and size of the query and response are different; only the requested address is really found in the query. In the query, the multicast address derived from the multicast address intended for routers is used as the destination address. The response is sent either as unicast (if the query contained the querier’s IP address) or as multicast to all nodes in the network (i.e., to `ff02::1`).

The NDP mechanism is also used for other purposes – communicating with routers to discover network configuration (network address, prefix length, etc.), discovering duplicate addresses, and more. There are separate ICMPv6 message numbers for each purpose.

5.3 IPv4 Address Range

5.3.1 Classes

An IPv4 address takes up 32 bits (4 octets), which would theoretically mean $2^{32} = 4\,294\,967\,296$ of possible addresses, but in reality it’s much less – for “organizational” reasons, among others. The address range is hierarchically divided into networks and subnets to simplify routing.

 Originally, in order to facilitate scaling of this hierarchy, the *five address classes* were distinguished according to the position of the boundary between the network and host portions of the address:

- *Class A* uses the first octet for the network address, the rest is the host portion, the addresses with the first octet 0 are not valid,
- *Class B* uses the first two octets for the network address, the rest is the host portion,
- *Class C* uses the first three octets for the network address, the rest is the host portion,
- *Class D* multicast address range,
- *Class E* for experimental purposes.

Class	Address structure	Fist nibble	First octet		Max. allocated addresses
A	netw.host.host.host	0xxx...	1–127	1–7F	$2^{24} - 2$
B	netw.netw.host.host	10xx...	128–191	80–BF	$2^{16} - 2$
C	netw.netw.netw.host	110x...	192–223	C0–DF	$2^8 - 2 = 254$
D	multicast	1110...	224–239	E0–EF	–
E	experimental	1111...	240–255	F0–FF	–

Table 5.1: IPv4 address classes

The structure of the addresses of these classes is indicated in Table 5.1.

So if we see an address whose first octet is less than 128, then it should be clear that it is a class A address. If the first octet is less than 192 (and greater than or equal to 128), it is a class B address, and if the first octet is less than 224, it is a class C address.



Remark

Note in the 5.1 table that in the last column (Maximum number of devices in the network) there is always a number -2 for the “usable” classes for devices. Why? Because we are subtracting the address where all the host bits are set to 0 (i.e., the network address) and the address where all the host bits are set to 1 (i.e., the broadcast address on the network). These two addresses cannot be assigned to any device, so they are subtracted.¹



Let’s go back to special addresses for IPv4:

- Loopback (i.e. $127.0.0.x$) is obviously class A address.
- We have a part of the range in each class reserved for private addresses:
 - Class A: $10.x.x.x$,
 - Class B: $172.16.x.x$ till $172.31.x.x$,
 - Class C: $192.168.0.x$ till $192.168.255.x$.

Note that these addresses are indeed in the range for the class. If we are set to use addresses of a particular class, we are referred to the private addresses associated with that class.

- Broadcast for a network with addresses of a given class (A, B or C) has the network portion set in the regular way, in the host portion (i.e. the last three, two or one octet) all bits are set to 1.

So if we have a network address for example $10.0.0.0$, the broadcast address on that network is $10.255.255.255$.




Remark

Note that if we keep with classes and class routing, then we don’t actually need a netmask or prefix length. The first few bits of the first octet tell us which part of the address is network and which is host. But we didn’t stay with the classes, so we will need these “additional data”.




¹In fact, some routers can do this (including Cisco), but it is considered a non-standard operation that can sometimes cause problems.

5.3.2 Subnetting

 As we indicated above, the classes gradually became insufficient. Administrators of larger networks simply needed to create a hierarchy, and this was not possible with classes alone. That's why they started using *subnetting*. They divided a single network into several parts – subnets, and membership in a subnet can be determined by specific bits of the address.

Subnetting is nothing more than an administrative intervention in the host portion of the address (nothing is done with the network address and its scope in subnetting). So the bits in the host portion are not all reserved for host station addressing, but a section (on the left, just after the network portion) is reserved for the subnet address.

 So the address consists of three portions when subnetting:

- network portion,
- subnetwork portion,
- host portion.

The sum of the bits of all three portions must, of course, add up to 32. While the number of bits of the network portion is unambiguous (we're still on classes), the number of bits of the subnet portion is no longer unambiguous, and thus a subnet mask (or prefix length) must be added to the address.



Example

If we have a class C network address of the form 192.168.48.0 (i.e. the three octets are the network address) and we want to divide this network into subnets, we need to know first of all:

- how many subnets should there be,
- the maximum number of usable addresses in each subnet we want to have.

The two are related – if we want more subnets, we'll have to reserve more bits for the subnet portion of the address, leaving fewer bits in the host portion, and thus fewer hosts in each subnet.

So let's assume that we don't want too many subnets – we'll make do with two bits (i.e. at most $2^2 = 4$ subnets).

Thus far we have used up $3 \times 8 + 2 = 26$ bits for the network and subnet portion of the address, so we have $32 - 26 = 6$ bits left for the host. Since $2^6 - 2 = 62$, there can be 62 devices on each subnet (note, any devices including the router). The bit mapping in the address – network, subnet and host portion:

nnnnnnnn.nnnnnnnn.nnnnnnnn.sshhhhhh

So what will the addresses of each subnet look like: the subnet address portion of the different subnets will take the values 00, 01, 10 and 11. For each subnet, we set the network address (bits in the host portion = 0) and the broadcast address (bits in the host portion = 1), and the addresses between them are intended for hosts. In real life:

- Subnet 1: the first two bits of the last octet are 00, so:
 - subnet address is 192.168.48.0/26 (the last octet in binary 00000000),
 - broadcast address is 192.168.48.63/26 (the last octet 00111111),
 - address range for hosts is 192.168.48.1/26 till 192.168.48.62/26.

- Subnet 2: the first two bits of the last octet are 01, so:
 - subnet address is 192.168.48.64/26 (the last octet in binary 01000000),
 - broadcast address is 192.168.48.127/26 (the last octet 01111111),
 - address range for hosts is 192.168.48.65/26 till 192.168.48.126/26.
- Subnet 3: the first two bits of the last octet are 10, so:
 - subnet address is 192.168.48.128/26 (the last octet in binary 10000000),
 - broadcast address is 192.168.48.191/26 (the last octet 10111111),
 - address range for hosts is 192.168.48.129/26 till 192.168.48.190/26.
- Subnet 4: the first two bits of the last octet are 11, so:
 - subnet address is 192.168.48.192/26 (the last octet in binary 11000000),
 - broadcast address is 192.168.48.255/26 (the last octet 11111111),
 - address range for hosts is 192.168.48.193/26 till 192.168.48.254/26.

Note that the subnet address is 1 higher than the broadcast address of the previous subnet.

Table view containing the same information as the entire previous list:

Subnet	Address	Host range	Broadcast
LAN1	192.168.48.0/26	192.168.48.1/26 till 192.168.48.62/26	192.168.48.63/26
LAN2	192.168.48.64/26	192.168.48.65/26 till 192.168.48.126/26	192.168.48.127/26
LAN3	192.168.48.128/26	192.168.48.129/26 till 192.168.48.190/26	192.168.48.191/26
LAN4	192.168.48.192/26	192.168.48.193/26 till 192.168.48.254/26	192.168.48.255/26


We just need to write down the parts of the address where there are differences, in this case the last octet:

Subnet	Address	Host range	Broadcast
LAN1	...0/26	...1/26 till ...62/26	...63/26
LAN2	...64/26	...65/26 till ...126/26	...127/26
LAN3	...128/26	...129/26 till ...190/26	...191/26
LAN4	...192/26	...193/26 till ...254/26	...255/26



In the tables in the previous example, we can clearly see the relationship between the addresses of the individual subnets: we change the last octet by adding the same number, 64. By incrementing by 64 we get the sequence 0, 64, 128, 192, 256, without using the last number.

When creating subnets of equal length, it's easier than working with bits to just figure out which number to add and in which octet, and then it's easy.


 The number we will add is called the *magic number*. We can find this number and the location where it will be added by using the subnet mask.

Example

Once we know how many bits we need to use for the subnet, we can write the subnet mask. In our case, this is the mask

255.255.255.192, in binary

11111111 . 11111111 . 11111111 . 11000000

Where is the last digit 1 in the binary mask? In our case, it is the second bit of the fourth octet. The value of this bit is $2^6 = 64$. It means, that we will increment the fourth octet by 64. This is applicable for subnet addresses as well as for broadcast addresses. 

When creating subnets, it is useful to keep track of how many hosts can be placed in a subnet with a specific prefix length. This table may help:

Prefix length	/22	/23	/24	/25	/26	/27	/28	/29	/30
Host portion length	10	9	8	7	6	5	4	3	2
Addresses	1024	512	256	128	64	32	16	8	4
Hosts	1022	510	254	126	62	30	14	6	2

We just need to remember the column for /24. In the left direction we multiply the row “Addresses” by two, in the right direction we divide by two.

Example

Let’s try creating larger subnets. The base address is 10.0.0.0/8, we need at least 10 subnets. However, the real number of subnets is always a power of two (because so many possibilities give us a certain number of bits), so we find the next higher (or equal) value, which is $16 = 2^4$. This means that we will use four bits for subnetting, whose values are 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111 16 possibilities in total. So the network part takes up 8 bits, the subnet takes up 4 bits, for a total of 12 bits. That leaves 20 bits for the host part.

The last digit 1 in the binary mask is in the second octet:

255. 11110000 .0.0

So, we will increment the second octet by $2^4 = 16$.

Subnet	Subnet address	Host range	Broadcast
LAN1	10.0.0.0/12 second octet: .00000000	from 10.0.0.1 till 10.15.255.254	10.15.255.255 second octet: .00001111
LAN2	10.16.0.0/12 second octet: .00010000	from 10.16.0.1 till 10.15.255.254	10.31.255.255 second octet: .00011111
LAN3	10.32.0.0/12 second octet: .00100000	from 10.32.0.1 till 10.47.255.254	10.47.255.255 second octet: .00101111
LAN4	10.48.0.0/12 second octet: .00110000	from 10.48.0.1 till 10.63.255.254	10.63.255.255 second octet: .00111111
• • •			
LAN15	10.224.0.0/12 second octet: .11100000	from 10.224.0.1 till 10.239.255.254	10.239.255.255 second octet: .11101111
LAN16	10.240.0.0/12 second octet: .11110000	from 10.240.0.1 till 10.255.255.254	10.255.255.255 second octet: .11111111

Table 5.2: Creating sixteen subnets for the network 10.0.9.0/8

The address ranges of the subnets cover the entire address space of the original network, they follow each other, and some addresses are unusable for host devices. In each subnet we need two

reserved addresses (subnet address and broadcast), in the original network there were 2 addresses in total, whereas after subnetting there are $2 * 16 = 32$ addresses, so by subnetting we lost $32 - 2 = 30$ addresses. However, creating a hierarchy is worth it, we can use different subnets for example for VLANs.



So far, we have been within one octet of the address (subnet addresses differed by a single octet). All we had to do was figure out how much the subnet addresses would differ, and then just increment. But what if the subnetting bits spread across the boundary between two octets?

Example

A larger company with about 800 offices around the world uses a 10.0.0.0/8 network, with each office needing its own subnet. Each branch office is expected to have up to 30,000 devices with an IP address (from mobile phones and servers to machines in production). Our task is to create subnets to meet these requirements:

- 800 subnets – $2^9 = 512 < 800 < 1024 = 2^{10} \Rightarrow$ we must select at least 10 bits for the subnet portion of the address,
- 30 000 usable addresses – $2^{13} = 28\,561 < 30\,000 < 57\,122 = 2^{14} \Rightarrow$ we need at least 14 bits in the host address portion.

The network portion of the address takes up 8 bits, and if we add everything up, we get $8 + 10 + 14 = 32$. So we get it exactly right, there is no other option (if it was less than 32, we could choose either more subnets or more clients in the subnet).

The length of the subnet prefix is 18 (sum the bits for the network and subnet portion), the third octet of the mask is binary 1100 0000, the mask is 255.255.192.0. Let's focus on the last 1 of the mask. It's in the third octet, it's the second binary digit from the left, and so when converted to decimal we get $2^6 = 64$. So we're in the third octet and we have the number 64 \Rightarrow in the third octet of the subnet addresses we're going to increment the number 64.

10.0.0.0/18
 10.0.64.0/18
 10.0.128.0/18
 10.0.196.0/18

But what now? After all, $196 + 64 = 256$, but that's already outside the allowed octet range. It's the same situation as, for example, adding 20 incrementally in the decimal system. Up to 80 is no problem, but if we want to add 20 again, we have to zero out the digit we made the changes to and add 1 to the higher order. We do the same thing here, but instead of digits we have octets.

...
 10.1.0.0/18
 10.1.64.0/18
 ...
 10.1.196.0/18
 10.2.0.0/18
 ...
 10.255.196.0/18

The result is in Table 5.3 on page 136.

Subnet	Address	Host range	Broadcast
LAN1	10.0.0.0/18 second and third octet: .00000000.00000000	from 10.0.0.1 till 10.0.63.254	10.0.63.255 second and third octet: .00000000.00111111
LAN2	10.0.64.0/18 second and third octet: .00000000.01000000	from 10.0.64.1 till 10.0.127.254	10.0.127.255 second and third octet: .00000000.01111111
LAN3	10.0.128.0/18 second and third octet: .00000000.10000000	from 10.0.128.1 till 10.0.195.254	10.0.195.255 second and third octet: .00000000.10111111
LAN4	10.0.196.0/18 second and third octet: .00000000.11000000	from 10.0.196.1 till 10.0.255.254	10.0.255.255 second and third octet: .00000000.11111111
LAN5	10.1.0.0/18 second and third octet: .00000001.00000000	from 10.1.0.1 till 10.1.63.254	10.1.63.255 second and third octet: .00000001.00111111
LAN6	10.1.64.0/18 second and third octet: .00000001.01000000	from 10.1.64.1 till 10.1.127.254	10.1.127.255 second and third octet: .00000001.01111111
• • •			
LAN1023	10.255.128.0/18 second and third octet: .11111111.10000000	from 10.255.128.1 till 10.255.195.254	10.255.195.255 second and third octet: .11111111.10111111
LAN1024	10.255.196.0/18 second and third octet: .11111111.11000000	from 10.255.196.1 till 10.255.255.254	10.255.255.255 second and third octet: .11111111.11111111

Table 5.3: Creating 1024 subnets for the network 10.0.0.0/8




Why all this?

- Subnets simplify routing within a large network (in routing tables, it is enough to have a single row for one subnet).
- We optimize network traffic as a result of simplifying routing.
- We can also restrict broadcast domains to a subnet, so broadcasts don't traverse the entire network, again reducing network traffic.
- We can implement various restrictions (such as ACLs), mapping to L2 layer VLANs.


In the exercises we worked with VLANs (virtual local area networks), which is usually understood as an L2 layer mechanism. In fact, the division of the network into VLANs is distributed to the L3 layer as well, just by subnetting – devices belonging to the same VLAN will be at the same subnet, devices from different VLANs will be at different subnets.

5.3.3 VLSM

 *Variable Length Subnet Masks* – VLSM – are a mechanism to extend the subnetting capability. While with subnetting we have fixed how many bits will be used for the subnet address, with

VLSM this number of bits can vary for different subnets. We still stick with class-based routing, but increase the flexibility of the design.

What's it good for? Basically, we are increasing the benefits of subnetting. With subnetting we have the same maximum number of hosts for each subnet (all subnets are the same size), and with VLSM we can have different numbers of hosts for different subnets (different sized subnets) as needed. But beware, uniqueness is important here as well, so that the different subnets (their address ranges) cannot not overlap in any case.

 **Remark:**

We can imagine this by creating a hierarchy of subnets, stopping subdivision earlier in some “branches” (fewer bits for subnet, more bits for host) and stopping subdivision later in other “branches” (more bits for subnet and thus more subnets, fewer bits for host).

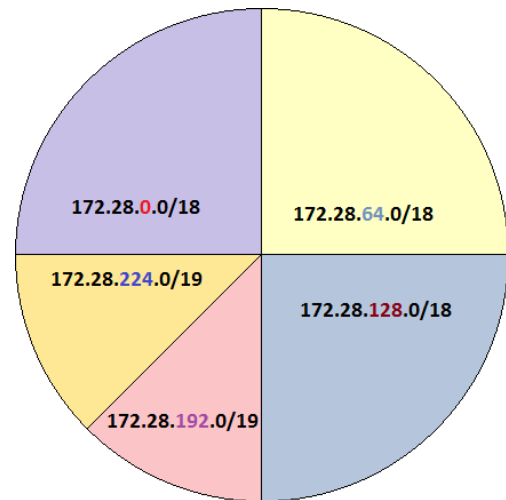



Figure 5.2: Dividing the 172.28.0.0/16 network into three larger and two smaller subnets

 **Example**


Again we have a class C network address of the form 192.168.48.0 and we want to divide this network into subnets. We know we will need one large subnet and two smaller ones. We choose a prefix length of 25 for the first subnet (so we reserve one bit for the subnet), and 26 for the second and third (two bits for the subnet).

Subnet Address	Pref. length	Hosts	Broadcast	Mask
192.168.48.0 last octet: .00000000	/25	from 192.168.48.1 till 192.168.48.126	192.168.48.127 last octet: .01111111	255.255.255.128 last octet: .10000000
192.168.48.128 last octet: .10000000	/26	from 192.168.48.129 till 192.168.48.190	192.168.48.191 last octet: .10111111	255.255.255.192 last octet: .11000000
192.168.48.192 last octet: .11000000	/26	from 192.168.48.193 till 192.168.48.254	192.168.48.255 last octet: .11111111	255.255.255.192 last octet: .11000000

Table 5.4: Creating three variably sized subnets for the network 192.168.48.0/24

The first subnet has a total of 128 addresses, of which 126 can be assigned to host devices. The second and third subnets have 64 addresses, of which 62 can be assigned.

5.3.4 CIDR

 *Classless Routing* – CIDR (Classless Inter-Domain Routing) already fully removes classes in the sense that it removes the dependency between the form of the first octet and the leftmost boundary for the network portion of the address (which is what classes do). The main motive was

to make routing at the higher levels of the hierarchy, i.e., at the RIR and LIR network providers and at the Internet backbone, clearer and simpler. In the following, we will use the terminology introduced in Section 4.3 from page 105.

How it works with CIDR: IANA allocates base ranges to each RIR, i.e. a certain number of bits in the address from the left. This gives a certain base prefix length. Each RIR allocates an address range to its customers (i.e., individual LIRs) such that the LIR *inherits* what the RIR has allocated (i.e., all customers share this part), and to this are added additional bits specific to the LIR in question – subnetting is performed, each LIR will have (relative to the RIR) one subnet. So the LIR has an address with a longer prefix. The LIR also has its customers, to which it does the same thing – each customer “inherits” the prefix from the LIR plus the customer-specific part.

This implies that CIDR actually builds a hierarchy of networks and subnetworks nested within each other to the extent needed. The main purpose of CIDR is precisely to align the hierarchy of IP addresses with the hierarchy of (physical) networks and especially routers.



Remark

How CIDR affects routers: let’s go back to the previous example. On a router that has a common path to subnets 2 and 3, we can have a common routing table entry for both subnets – simply for the address 192.168.48.128/25 (the last octet binary 10000000), because the first 25 bits of the address are the same for both subnets, and only in the next bit do they differ. However, this is only possible if the path to both networks is through the same interface.

In fact, the router that addresses subnets 2 and 3 in this simplified way does not even distinguish between these two subnets (it does not know that the 192.168.48.128/25 range is somehow internally structured). But that’s alright, there is no need to overwhelm the external elements with internal subdivision.



Summarization (also aggregation) of paths, or supernetting is what is described in the remark. If multiple subnets share a path through the same router port, there is no reason to have a separate row for each in the routing table. They will all be summarized in one line, with the prefix truncated (moving the network portion of the address boundary to the left) to include all of these subnets. Even before the original A/B/C class boundary. This aggregate row representing multiple (sub)networks is called the *CIDR block*. Of course, the condition is that the physical networking actually matches the IP address hierarchy.




Remark

The difference between subnetting (including VLSM) and supernetting (CIDR) is, among other things, that in subnetting we move the boundary between the (sub)network and the host to the right, whereas in supernetting we move it to the left.

Subnetting applies to every element of the hierarchy (the company gets an address range from the ISP and can divide it up as it wishes), while supersetting applies more to ISPs running very busy routers (they need to reduce routing tables just by using CIDR).



 It was the CIDR system that came up with the simplified marking of the boundary between the network and host portion using a slash and prefix length, so in class routing we should theoretically use mask notation. The slash and prefix length notation is called *CIDR notation*.

 **Remark**

The IPv6 address range system is naturally already of the CIDR type. That is why we started with the description of the hierarchy for IANA, RIR, LIR, etc., by describing the principle of IPv6 addressing.




5.4 How to Get an IP Address


Every device communicating over the network needs at least one IP address, but it needs to get it from somewhere. In fact, we've already covered the mechanisms for obtaining an IP address (the previous chapter had a section on DHCP), so now we'll go through various options.

5.4.1 How to Get IPv4 Address

We can obtain an IPv4 address

- by dynamic DHCP allocation,
- statically, either entering the address manually into the configuration or using automatic static allocation (also using DHCP).

 *Static IP address* is therefore a fixed address for the device, the device does not even receive another one. Typically, static IP addresses are used for servers that are always to be reachable under the same address. In contrast, a *dynamic IP address* is only ever leased for a certain period of time (for example, one day), after which the device can be reassigned the same address or a completely different one.

 The DHCP server has a range of addresses it can allocate (Address Pool, Address Stack) and in this range it notes to whom (which MAC address) it has allocated which IP address so far and for how long (leased time). This table is actually the DHCP server's state information (it determines the status of address allocation), and thus the use of the DHCP server for address allocation is referred to as *DHCP stateful mode*.

5.4.2 How to Get IPv6 Address

In an IPv6 enabled network, each router sends out an ICMPv6 message *Router Advertisement* – *RA* at regular intervals. The router (or switch with L3 functionality) puts all the important information about itself and the network into these messages, such as the network prefix (i.e. network address), prefix length, gateway address, MTU value, etc.

If a device wants information for an IPv6 address, it either listens on the network or actively queries ICMPv6 with the *Router Solicitation* – *RS* message, which forces the router to send an RA.

**Remark**

The RA (ICMP message 134) and RS (ICMP message 133) messages are part of the NDP (Neighbor Discovery Protocol) mechanism.



Under IPv6, we have the following possibilities for obtaining an IP address:

- by dynamic allocation using DHCP (DHCP state mode),
- autoconfiguration using EUI-64 (SLAAC – Stateless Address Autoconfiguration),
- autoconfiguration with Privacy Extensions,
- static configuration.



The first possibility – *dynamic allocation by DHCP* – is practically the same as for IPv4 (the difference is mainly in the type and format of the messages sent). It is the only one that is *stateful*, i.e. the DHCP server stores stateful (variable, dynamic) address allocation information, the other options are stateless (no such records are stored anywhere).



Autoconfiguration using EUI-64 (SLAAC, Stateless Autoconfiguration) is intended to reduce unnecessary traffic for DHCP servers and especially for the network. The host part of the address is built by the device itself, it just needs to get the other information (network part of the address, etc.). How it works:

- The host part of the address (here exactly half, 64 bits) is EUI-64, which is made from the MAC address. The MAC address is 48 bits long, so we still need to add 16 bits (2 octets). We modify:
 - exactly in the middle of the MAC address, inserting two octets FF-FE,
 - set the U/L bit to 1 (second bit from the right in the first octet).
- The network part of the address (the remaining 64 bits) must be obtained in another way, for example from the router’s RA message.

The U/L bit is one of the bits in the first octet of the MAC address – the second from the right. If it is set to 1, it means that the address is only locally valid. If it is set to 0, it is a universal globally valid MAC address. However, this meaning is not important in the transformation to EUI-64.

**Example**

We will demonstrate the creation of EUI-64 and its integration into an IPv6 address. Consider the following data:

- the MAC address is 50-E5-49-C2-AC-27,
- via RA we found that the prefix (network part of the address) is 2001:718:2601:265.

First we create the EUI-64. The MAC address consists of six octets, so we insert two octets FF-FE between the third and fourth (i.e. in the middle):

50-E5-49-FF-FE-C2-AC-27


Then we set the U/L bit in the first octet – binary the first octet is originally 01010000, after the change 01010010, which is hexadecimal 52. So the resulting EUI-64 is:

52-E5-49-FF-FE-C2-AC-27

We complete the whole address, i.e. we modify EUI-64 to “optically” match the IPv6 part of the address (groups, colons) and add the prefix.

2001:718:2601:265:52E5:49FF:FEC2:AC27/64



 *Autoconfiguration with Privacy Extensions* is the third possibility to obtain an IPv6 address. It works basically similarly to autoconfiguration with EUI-64, only the host part of the address is created differently.


- The host part of the address (also exactly half, 64 bits) is dynamically generated from various hardware and software characteristics of the system and changes every few days.
- The network part of the address is obtained, for example, from the RA message of the router.

While EUI-64 is primarily used on network devices and servers (generally devices where “user does not sit” – due to GDPR), Privacy Extensions are commonly used on end-user devices or Windows servers.

 **Remark**

The intent is to make it as difficult as possible to identify endpoint devices on the network, but in reality, this practice also makes it difficult for administrators. Network administrators need to keep track of what is going on in the network, what devices are currently connected and where they usually connect, the IP address is an important identifier in monitoring and accounting. If this identifier changes frequently, it causes chaos in the network.



 *Static Configuration* is the last possibility to get an address, it means a situation when the device gets a pre-assigned address. As with IPv4, this can be done manually or by static allocation (the device queries the DHCP server but always gets the same IP address).

Statically, either the entire address or just the host portion can be configured, with the network portion being added by the device as described for the previous possibilities.

 **Remark**

Static allocation is *stateless use of DHCP*. Although information must be stored about which IP address to allocate to the device with a particular MAC address, it is not dynamic (variable) information.



5.5 NAT and Private Addresses

We already know something about *private addresses* – there are ranges reserved for them in each class and packets with this address are not allowed to “to the router”. So what are they for?

Their main purpose is to save public IP address ranges (most end devices do not need a public IP address, just a private one that is not globally unique). The second purpose is a slight increase in security, since private addresses need to be resolved at the network boundary to public ones (or to private ones operating on the higher level network), which can be seen as an additional protection mechanism.

Private addresses need to be resolved at the network boundary, but in fact we can use this resolution for other types of addresses as well. Resolving basically consists of swapping one IP address for another in a packet.

**Remark**

Let's not confuse the terms:

- public address is globally valid, it can enter the Internet (or behind the router),
- private address is only locally valid, can't behind the company's main router.

We also distinguish between statically and dynamically assigned addresses:

- static address is fixed to the device, another device cannot get it,
- dynamic address is currently assigned to the device, but tomorrow a completely different device may have it.

A device address can be private and dynamic or private and static. Public addresses tend to be static, but not every static address is public.



NAT (Network Address Translation) is a mechanism for translating IP addresses at the network boundary. In a packet, we can translate only the source address or only the destination address or both, usually combining the translation of the source address in one direction and the translation of the destination address in the other direction (if we think about it, we find that it is actually the address belonging to the same device).

We distinguish the following types of NAT:

- Static NAT (stateless) – it is fixed which address from the internal network is resolved to which address valid in the external network and vice versa.
- Dynamic NAT (stateful, Masquerade) – dynamically determines the pair of addresses to be resolved, requires storing a record of the resolution to be used for back resolution.
- PAT (Port Address Translation, NAT overload) – adds an additional mechanism to Dynamic NAT to differentiate individual conversations.



Static NAT means that we replace the source IP address of our device in the outgoing traffic packet with another one under which this device should be visible “outside”, and on the contrary, in the incoming traffic we perform a back translation to the (this time destination) IP address of this device. Static NAT requires that we have two addresses statically reserved for the device – one for the internal network and one for the external network.

It is typically used to hide the IP addresses of our servers. The server is “outside” visible under a different IP address than it actually has. Another reason to use static NAT is to temporarily change the network situation, for example for maintenance.



Dynamic NAT (Masquerade, “masquerade”) is already a stateful configuration. It is used when the internal addresses we are translating are dynamic, and thus cannot be “statically stored”. In order to save address range, we have dynamically allocated addresses on the internal network and resolve them outward to a single address (common to everything dynamic on the internal network).

The NAT server (this is usually the router performing NAT translation) keeps a connection table in which it records for each connection between the internal and external network the address of the device on the internal network and the address of the device on the external network (typically a server on the Internet with which the connection is established). The principle is outlined in Figure 5.3.

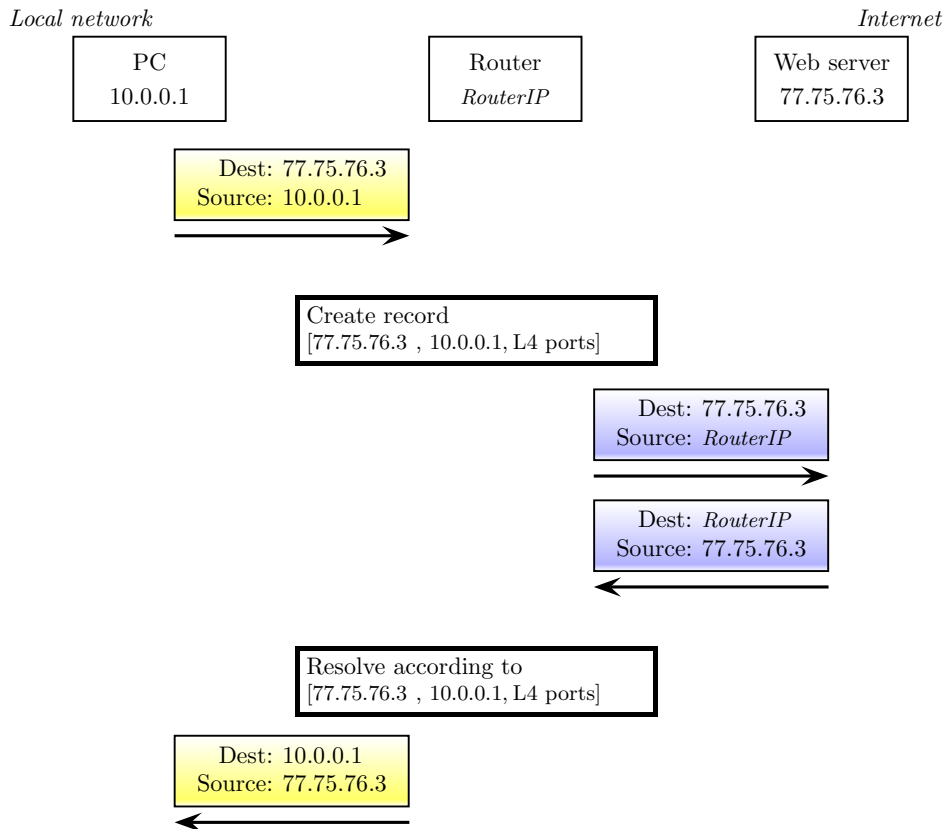



Figure 5.3: Simplified dynamic NAT scheme

Uniqueness is ensured if each device from the internal network communicates with a different server. The problem arises when multiple devices from the internal network want to communicate with the same “external” server, then the entries in the connection table would no longer be unambiguous. In the client-to-server direction, a packet could be sent, but we could not deliver the response from the server to the appropriate client because we would not know which one.

 *PAT (port translation, NAT overload)* solves this problem. IP packets usually encapsulate TCP or UDP segments using source and destination port numbers. Client port numbers tend to be dynamic, i.e. there is a high probability that two different clients on our network communicating with the same server will have different port numbers.

So instead of just storing IP addresses in the table, we add port numbers (remember that a socket is identified by the pair IP address + port number). If there was still a situation where two clients from the internal network used the same source port number when communicating with the same server, not only the IP address but also the port number would be translated on the NAT server to ensure uniqueness.

**Remark**

Let's summarize the address translation mechanisms discussed so far:

- ARP and NDP perform the translation of a logical IP address to a physical MAC address,
- DNS performs the translation of a name (domain) address to an IP address,
- the NAT mechanism translates between internal and external IP addresses.



5.6 Routing

5.7 How Routing Works

Routing takes place at the L3 layer, and actually means the process of determining the path to specific IP networks. This information is further used to determine the path for a particular IP packet.



Definition (Routing table)

At least one routing table is kept on each router and all other devices of the L3 layer, in which we have the following information for each record:

- the (sub)network address with the mask or prefix length,
- the outgoing network interface or the neighbor (its IP address) leading to the given network,
- the type of the record (static, dynamic gained by some protocol, identification of such protocol),
- metrics (quality of the chosen path) and other information.

The routing table also includes information about the gateway (also called “gateway of the last resort”), i.e. determining the direction for everything to send for which we do not have a specified path elsewhere in the routing table.



In a device supporting the both IPv4 and IPv6 protocols there are at least two routing tables, one for each of these protocols.



This table needs to be filled in and further updated according to the current situation (some paths will cease to be valid, or, conversely, another path is made available, metrics are changed or a new subnet is being added). This is either done manually (*static routing*) or left to *routing protocols* (*dynamic routing*).

In practice, both are combined, i.e. we insert some paths manually and let the rest be added and updated dynamically. Static routing data is considered more trusted, so it is usually preferred over data obtained dynamically using protocols.



Metric is a number indicating the quality of a path, it has been calculated according to certain criteria. Each routing protocol uses slightly different criteria and calculates metrics from them in a slightly different way. For example, the following can be used as criteria:

- number of routers at the path (it means the path length),
- path throughput (for example, whether it is Fast Ethernet or Gigabit Ethernet),
- latency (delay in ms),
- path reliability (low error rate, few dropped packets, etc.),
- the path load,
- the maximum MTU value over the path (i.e. the maximum packet size that can be sent),
- the credibility of the source from which the path information was obtained, etc.

Each routing protocol has its own metric. It is not only that it uses different criteria, but also that the metrics of the different protocols are not comparable. The same path is evaluated completely differently by two different protocols.

**Remark**

Usually: the smaller the metric number, the better the path. Directly connected addresses have the lowest metric.



When and how to work with the routing table: if the router receives a packet, it needs to find out to which port this packet should be forwarded. It goes through the routing table and finds out which record in the table corresponds to the packet's destination address, i.e. it determines whether the address belongs to the subnet specified in the given record.

If no path is found in the routing table, either the packet is passed to the gateway because the packet appears to belong to another network, or dropped (when no gateway is available). This procedure corresponds to the procedure from the previous paragraph, because the gateway of the last resort is specified by the address 0.0.0.0/0, which is the least specific possible address, and therefore will be used only if no other record from the routing table is selected.

5.8 Cooperation of Routing Protocols

Various organizations use various routing protocols. Some of them are intended for interior usage inside a network of routers of a single organizations, others are able to provide routing among these relatively homogenous areas.

5.8.1 Autonomous System



An autonomous system is a group of routers that are managed by the same organization. Each autonomous system is identified by a 16-bit or, more recently, a 32-bit *ASN* number.

- the *interior routing protocols* can only route within their own autonomous system (they send everything “foreign” simply to a single gateway), they do not distinguish between other autonomous systems (they distinguish only “own” and “foreign” network).
- the *exterior routing protocols* provide routing even between different autonomous systems (they can work with the ASN number, the routes are characterized by ASNs of the autonomous systems through which the destination can be reached. They can distinguish between different “foreign” paths.

**Remark**

Does each organization have to manage its own autonomous system and its own ASN? No. Smaller companies only need to be part of their ISP's autonomous system (or of the autonomous system of the ISP of their ISP), and they usually don't even know its ASN. If the company uses just interior routing protocols, there is no reason to have its own ASN (mainly to pay for it).



Each routing protocol works according to some *routing algorithm*. This is usually one of the following algorithms (or some modification): Distance-Vector Algorithm and Link-State Algorithm.


5.8.2 Distance Vector Routing Algorithm

The Distance-Vector Algorithm determines path metrics primarily by distance (number of routers on the path). A router operating under this algorithm (protocol) first has its neighbors (the routers to which it connects) in the table and receives information from them about other routers and their subnets. If it receives a packet from its neighbor with information about a network with a specific metric (according to that neighbor), it adds that network to its routing table:

- the address of the (sub)network he found out from a neighbor,
- the interface is the address or port of the respective neighbor (because the network is accessible through it),
- as a metric will use a number greater by 1 (or some other small constant) than the neighbor (i.e. the path from the neighbor to the network must include the path from the neighbor to me).

Each router sends its routing table to all neighbors at regular intervals, informing them of the currently reachable networks and their metrics, which can overwhelm the network quite a bit.

As we can see, a similar problem can arise here as in the switch network, where we had to use STP to remove loops. Loops can occur here as well (we have a network of routers using a given algorithm/protocol), and the consequences could be similar. Therefore, a maximum number of hops is defined, which determines the maximum size of the network (in number of routers on the path). If information is received from a neighbor about a network that is reachable by more hops than what the maximum number is, then this entry is ignored (the given network is considered unreachable).

 A typical routing protocol using the distance vector algorithm is RIP (Routing Information Protocol), which uses the number of hops to the destination (i.e., the number of routers on the path) as a metric.

The first version of RIPv1 is currently unusable (classful routing only, does not support masks or prefix lengths, only counts on very small networks, routing information sent as broadcast, IPv4 only).

RIPv2 sends network records including the mask in update packets and thus supports classless routing. Nowadays, it is only used in small networks because it converges very slowly (network updates are time consuming, it does not route during) and by default allows a maximum of only 15 hops.

Another version is RIPng, which is actually RIPv2 with added support for IPv6 addresses.



Remark

A typical characteristic of protocols strictly implementing the distance vector algorithm is that they directly “know” only their neighbors and have no information about the network topology. They only know that a specific (sub)network can be accessed through a specific neighbor and the path takes a specific number of hops.



Another protocol that uses the distance vector algorithm is Cisco’s IGRP (proprietary) and its successor EIGRP (its specification has been published in an RFC and can be implemented by

other vendors). IGRP is no longer used (it worked with IP address classes), but EIGRP is still widely used. Unlike RIP, the EIGRP metric is quite complex and is considered to be of high quality. Compared to the basic implementation of the Distance Vector algorithm in RIP, the implementation in (E)IGRP is much more sophisticated and includes elements that are not part of the basic algorithm.

5.8.3 Link-state Algorithm


The Link-state Algorithm (the shortest path algorithm) requires building a *network topology database* (a kind of network map in the form of a matrix). A router with a protocol following this algorithm maintains three tables:

- neighbors table, which is the first one it creates right after switching on,
- a table representing the network topology database, it creates this table one by one after powering up according to the update information from other routers,
- the routing table, which it creates and maintains according to the topological database.

Each router constantly monitors the quality (state) of the links leading to the neighbors (this refers to the first mentioned table). If it detects a change (a link/neighbor becomes unavailable or changes some parameter), it informs its neighborhood. The routers that receive this information will record the change in the topology database (second table) and calculate the new path to the destinations according to it (update the routing table).

It is typical for the protocols based on the link state algorithm that they have an overview of the network topology, they flood the network less (only changes to specific links are sent, and usually only when the topology changes) and the time of convergence is significantly shorter (updates are only forwarded, the change is calculated by each router itself, they do not have to wait for each other).

The actual calculation of routes and finding the shortest (most optimal) path to a given (sub)network is done according to the topological database using *Dijkstra's algorithm* (shortest path first algorithm) or its modification. This is one of the most famous graph algorithms, whose author is the Dutch computer scientist Edsger Dijkstra.


 The most well-known link state protocol is *OSPF* (Open Shortest Path First). It works exactly as described above, and its big advantage is that it is an open protocol (a manufacturer who wants to implement it in his device does not have to pay licensing fees). It is currently the most widely used routing protocol for internal routing, and it is also somewhat applicable to external routing (it can handle autonomous system numbers).

OSPF supports classless routing, has no IPv6 problem, and allows (unlike RIP) more than one route to the same destination. It uses a more complex metric where it primarily considers the bandwidth on the path, so that the metric more closely matches the actual throughput of the path. Convergence is very fast (which is related to the algorithm used).



Remark


A typical feature of link-state algorithm protocols is that they have knowledge of their network or a substantial part of it (an entire group of routers can be divided into relatively separate *areas* that

are interconnected by border routers, and then it is sufficient for the protocol to have knowledge of the area it is in). Another typical feature is fast convergence and less vulnerability to loops 

Another protocol in use that uses a link-state algorithm is IS-IS, which until recently was particularly popular with network connectivity providers (ISPs). Unlike other routing protocols, IS-IS operates at the L2 layer, so it is used e.g. in IEEE 802.1aq (bridges in industrial automation) to support the Shortest Path Bridging (SPB) mechanism.

5.8.4 BGP

While the previous mentioned protocols are rather internal routing protocols (they mostly route within their autonomous system and have only a rough or no idea about “the rest of the world”), we will now focus on one external routing protocol.

 *Border Gateway Protocol* (BGP) is an external routing protocol, so it is typically used for routing *between* autonomous systems, not within them. It is primarily used in WANs to interconnect LANs, MANs, and smaller WANs, and thus BGP routes between autonomous systems in these networks. In fact, it is the most widely used external routing protocol on the Internet.

BGP of course supports classless routing (CIDR) and IPv6, otherwise it would not be usable in today’s WANs.


It is an implementation of the *path vector algorithm*, which is a hybrid of the distance vector algorithm and the link state algorithm. If multiple paths lead to an autonomous system, it will select the one that leads through fewer passing autonomous systems.

A path to a network represented by BGP is a sequence of numbers of autonomous systems (a shorter sequence with fewer numbers means a better path).




Remark


While the packets of most of the previous named routing protocols (RIP, IGRP, EIGRP, OSPF) are encapsulated directly into IP packets, and IS-IS even works on L2 (they do not involve the transport layer at all) and thus they all use the datagram service, BGP packets make a “detour” through the transport layer and are encapsulated into TCP segments (i.e. routers in the BGP network communicate with each other using a connection-oriented way) and then into IP packets.


The reason for the use of connection-oriented communication is nature of the WANs in which it is used. 

Chapter 6


Wireless Local Networks

 *Quick preview:* In this chapter, we will stay at the L1 and L2 layers, but this time we will focus on wireless local area networks. First, we'll take a general look at wireless transmission characteristics and related concepts, then we'll focus on Wi-fi at the L1 and L2 layers, including the transmission and antennas themselves. The last section is about Wi-fi network security.

 *Keywords:* Wi-fi, IEEE 803.11, radio waves, frequency range, ad-hoc, infrastructure, OFDM, QAM, antenna, diversity, MIMO, MU-MIMO, access point, repeater, extender, AP client, WDS, WISP, BSA, ESA, SSID, BSSID, ESSID, Beacon frame, CSMA/CA collision method, RTS/CTS, AAA, WEP, WPA, WPA2, IEEE 802.1X, Radius, Diameter, WPS

 *Objectives:* After studying this chapter, you will understand the differences between “cable” and wireless local area networks, the flow of communication in a wireless network and the use of radio signals for data transmission, the operation of Wi-fi at the L1 and L2 layers, and the principles of wireless network security.

6.1 Wireless Technologies

 Transmission technologies can be divided into the following categories according to the transmission medium:


- *wired* – transmission is via cable (optical or metallic),
- *wireless* – transmission is without cable.


In this chapter, we will discuss wireless technologies.

So what transmission medium do we use for wireless technologies? We could say air, that's more or less true, but the actual transmission medium is what we modulate the signal to (yes, in wireless technologies we usually modulate the signal). The options are:

- *radio waves* of a certain frequency – Wi-fi, WiMAX, Bluetooth, NFC, ZigBee, Z-Wave, etc.,
- *sound waves* (sonic wireless technology) – e.g. ultrasound,
- *light* (optical wireless technology) – laser, infrared light (IR), flag waving. . .

We will be primarily interested in the first option. Networks based on radio wave transmission can vary in size – from personal (PAN, e.g. Bluetooth or NFC) to local (LAN, e.g. Wi-fi) and metropolitan (MAN, e.g. WiMAX) to wide area (WAN, mobile networks).


 *WLAN* (Wireless LAN) – This abbreviation refers to wireless local area networks. Be careful not to confuse this abbreviation with VLAN, the given terms are completely different.

 Wireless networks can also be classified according to the degree of mobility:

- *mobile* – the connected client devices will not lose the signal (handover) even when moving relatively fast, here we include mainly mobile WANs such as GPRS, EDGE, LTE, 5G, etc.,
- *fixed* – connected client devices may lose signal when moving faster, which happens e.g. with Wi-fi.

Nothing is just black or just white. Even for Wi-fi there are standards that bring the technology closer to mobile technology, but at the base it is still a fixed wireless technology.

As for the frequency bands in which Wi-fi wireless radio networks transmit, most use the unlicensed band, but some transmit in the licensed frequency band.

 An *Unlicensed frequency band*, also called *ISM* (Industrial, Scientific, Medical, defined by ITU, and by ETSI in Europe), is a frequency band such that if a device transmits in that band, its operator does not need to apply for a license, as long as it follows certain rules set out in the general license. There is no legal protection against interference in the license, only a rule stating that “it is not allowed”. The ISM band is used for Wi-fi, NFC, Bluetooth, baby monitors and other similar devices.

Licensed frequency band – the user of such a device must ensure *individual authorisation* to operate in a given frequency band in the area concerned, this service may be paid for. The user also has the right to protection in case someone else transmits in his assigned frequency band and therefore interferes with the signal.


Licensed frequency band	ISM band
individual authorisation for one applicant	general authorisation (no application)
usually for a fee	usually no frequency fee
protection from misuse, interference	no legal protection from misuse, interference
usually quality of service guarantee can be provided	usually no quality of service guarantee is provided

Table 6.1: Comparison of licensed and unlicensed frequency bands

6.2 Wi-fi

Wi-fi (Wireless Fidelity) technology is a wireless technology operating in the unlicensed band around 2.4 GHz, 5 GHz and 6 GHz (depending on the specific sub-standard).

The basic standard for Wi-fi is IEEE 802.11, and substandards (annexes) add one- or two-letter abbreviations to this designation (the alphabet is short, one-letter is no longer sufficient). The Wi-fi Alliance takes care of the standard and device compatibility (including certification).

 A wi-fi network can take one of two forms, respectively operate in one of two modes:

- *ad-hoc* – connect two end devices directly, using no active network equipment (DCE),
- *infrastructure* – use (at least one) active network equipment (DCE), end devices do not communicate directly, all communication goes through this infrastructure.

6.2.1 Physical Layer

There are several different sub-standards on the physical layer with very different properties, and from time to time a new sub-standard is added.

Specification	Frequency	Max. throughput	Designation
IEEE 802.11	2.4 GHz	2 Mbps	
IEEE 802.11a	5 GHz	54 Mbps	Wi-Fi 1
IEEE 802.11b	2.4 GHz	11 Mbps	Wi-Fi 2
IEEE 802.11g	2.4 GHz	54 Mbps	Wi-Fi 3
IEEE 802.11n	2.4 GHz, 5 GHz	250 Mbps per antenna, up to 600 Mbps total	Wi-Fi 4
IEEE 802.11ac	5 GHz	433 Mbps per antenna and stream	Wi-Fi 5
IEEE 802.11ad	60 MHz	cca 7 Gbps, depending on the channels used	WiGig
IEEE 802.11ax	2.4 GHz, 5 GHz	units Gbps	Wi-Fi 6
IEEE 802.11ax-2021	2.4 GHz, 5 GHz, 6 GHz	tens Gbps	Wi-Fi 6E
IEEE 802.11be (2024)	2.4 GHz, 5 GHz, 6 GHz	hundreds Gbps	Wi-Fi 7

Table 6.2: Substandards for the Wi-fi physical layer

Table 6.2 gives an overview of the existing substandards for the physical layer (in fact there are many more, but the rest are rather additional sub-standards).

The information in the throughput column should be taken with a grain of salt; sometimes this information can even be considered marketing. In reality, the speed is reduced, for example, by the need to communicate in both directions, which is a problem when using the same frequency, a larger number of stations may be associated, the distance between the communicating nodes and possible obstacles also have an impact. It also depends on the number of antennas, the channel width used, the number of streams, etc.

Devices that support the Wi-fi standard have a label indicating which frequencies and other features they support. For example, IEEE 802.11b/g/n means that the device has implemented sub-standards b, g, n of the above, whereas IEEE 802.11a/n/ac/ax has implemented sub-standards a, n, ac, ax.

Figure 6.1 shows the frequency range used by IEEE 802.11b. It is definitely not just the 2.4 GHz frequency, but the surrounding frequency range.

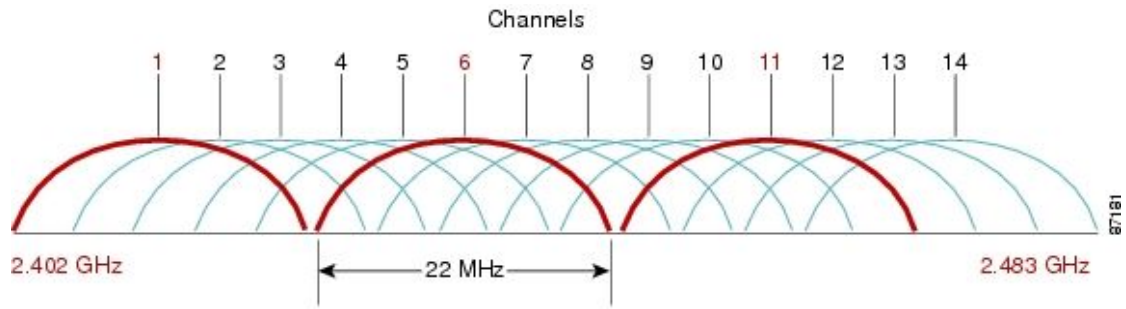



Figure 6.1: IEEE 802.11b frequency spectrum¹

 The whole spectrum is divided into *channels*, the channel width is 22 MHz. The device (both DTE and AP) has a specific channel on which it communicates. The AP usually chooses its own channel based on spectrum interference analysis, but in some cases this value has to be determined manually (because the AP cannot sense which channels are interfered with on the client side).

However, beware that different countries may have different “allowed” channels. In most EU countries, these are the ones with numbers 1–13.

If a device communicates on a specific channel, the signal is actually on the surrounding channels. Theoretically, a total of five channels are occupied (for example, if the device communicates on channel 6, then it occupies channels 4–8, according to the “arc” in Figure), but practice is usually a bit more rough – on more distant frequencies the signal strength does not drop as sharply as the arc prescribes, and weak interference is still several channels away. The channels overlap in practice somewhat more than theory suggests.

 **Remark**

So how many devices can communicate in the 2.4 GHz band without interfering with each other? Theoretically three (channels 1, 6 and 11). In practice, even just two devices can interfere with each other in this relatively narrow spectrum, depending on their signal strength and other antenna characteristics.

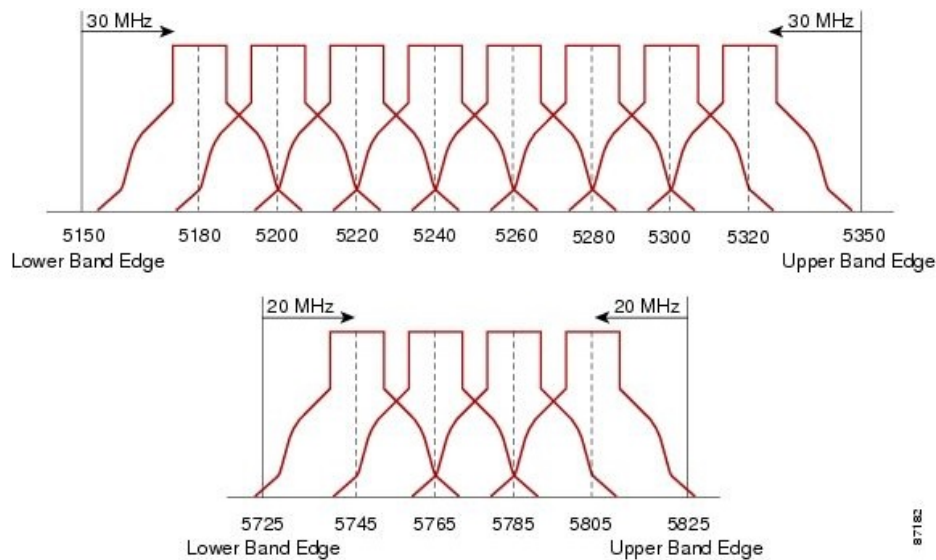


Microwave ovens also operate in the same frequency spectrum (if you think that the microwave oven is so thoroughly isolated during operation that no signal can get out, then know that this is not always the case) and most baby monitors, so mutual interference is far from being limited to Wi-Fi devices.

Figure 6.2 indicates the division of the spectrum for IEEE 802.11a into channels. The 20 MHz wide channels overlap in a slightly different way (less) than for IEEE 802.11b and the spectrum is divided into two separate parts.

The multiplexing method used in IEEE 802.11a is OFDM (mostly with QAM modulation), which is more efficient than DSSS from IEEE 802.11b. Higher speeds compared to IEEE 802.11b are achieved mainly due to more efficient modulation and the transition to higher frequencies (at twice the frequency we can transmit twice as much data in the same time unit, all other parameters being equal).

¹From: <http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob30dg/RFDesign.html>

Figure 6.2: IEEE 802.11a frequency spectrum²

IEEE 802.11g. This specification uses the same frequency spectrum as IEEE 802.11b (see figure 6.1). OFDM is used for multiplexing, and subcarriers are also modulated in the same way as in IEEE 802.11a, using QAM.

IEEE 802.11n (Wi-Fi 4). This specification was the first to allow operation in two frequency bands. The maximum throughput was increased up to 600 Mbps (sum over all antennas), which was achieved as follows:

- typically more than one antenna is used,
- we use both frequency bands, even in parallel,
- but the usual channel width is 40 MHz,
- better modulation (OFDM, but with different parameters),
- changes are also at the L2 layer.

The more antennas, the more throughput we theoretically get (but practically we can't just take the sum over all antennas).

While the IEEE 802.11a/b/g specification only differs at the physical layer (the MAC sublayer of L2 is the same), the IEEE 802.11n specification also changes the MAC sublayer.

When devices supporting different sub-standards (in the same frequency band) coexist, the AP can run in one of the following three modes:

- *Legacy Mode* – only the legacy standard is used (from the intersection of supported standards across all devices in the cell). Usually the speed falls down to the lowest common one.
- *Mixed Mode* – each device in the cell operates according to its best substandard, as long as the number and settings of the AP antennas allow.
- *Greenfield* – opposite of Legacy mode, only IEEE 802.11n is used and all devices that do not support it are out of luck.

Of course, it depends on what the AP in question allows to be set up. In terms of network throughput, the third option is the best.

²From: <http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob30dg/RFDesign.html>

IEEE 802.11ac (Wi-Fi 5). This standard has fully migrated to the 5 GHz band. It uses OFDM multiplexing with QAM modulation on subcarriers. Of course, the exclusive use of higher frequencies has a positive effect on throughput, but a negative effect on range.

The frequency band is actually slightly wider than that of IEEE 802.11a. The channel width can be 20 MHz, 40 MHz, 80 MHz or 160 MHz, the specific value is chosen according to the expected number of clients.

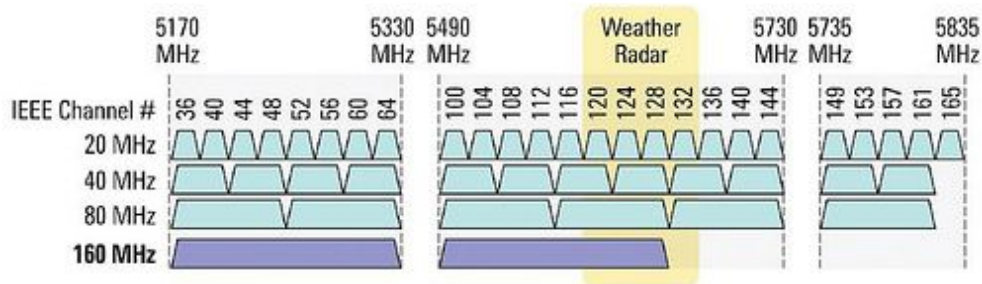


Figure 6.3: IEEE 802.11ac frequency spectrum³

IEEE 802.11ac increases speed over older versions as follows:

- uses better modulation,
- allows more antennas (up to 8) and handles them much more optimally,
- operates in the 5 GHz frequency band, and the channel width can be determined adaptively according to the network load.

OFDM multiplexing is used, and the subcarriers are then processed by QAM modulation, up to 256-QAM. In each generation, this parameter was varied as follows:

- 802.11g: 16-QAM (one symbol = 4 bits, $2^4 = 16$ options for symbol value),
- 802.11n: 64-QAM (one symbol = 8 bits),
- 802.11ac: 256-QAM (16 bits).

The consequence was a gradual increase in transmission efficiency (data transmitted per time).

The above applies to more or less ideal conditions. For interference or longer distances, when the quality of the delivered signal deteriorates and frame loss is more frequent, a more robust modulation is switched (symbol is encoded into more bits).


IEEE 802.11ad (WiGig) is a bit of a step aside, it's not so much next generation as it is a specialty for a specific application. The main purpose has been to create a wireless replacement for HDMI cable, i.e. high-speed transmission of multimedia data over short distances in the range of meters. Typical range is up to 10 meters, and this is without barriers. According to a more recent specification, up to 6 channels are available (each country has it differently, e.g. China only 2 channels, USA all 6, in Europe we use 4 channels) with a width of 2.16 GHz.

Using a frequency range around 60 GHz has both positive and negative consequences:

- + theoretical throughput up to units of GHz, which is sufficient for the transmission of uncompressed UHD video,
- higher frequencies are more sensitive to interference and also pass poorly through barriers, even air has a dampening effect on the signal.

³From: <http://www.dailywireless.org/2012/04/page/3/>

A typical application is to wirelessly connect a computing device (PC, laptop, HTPC, etc.) to a display device (TV, projector, etc.), but we often find that devices do not support this standard.

 **IEEE 802.11ax (Wi-Fi 6)** uses spectrum around 2.4 GHz and 5 GHz, as well as IEEE 802.1n. Channels are up to 160 MHz wide, modulation is up to 1024-QAM with OFDMA multiplexing (as with LTE), and the number of parallel streams has increased to 8 (all of which will be discussed below). WPA3 is expected for security.

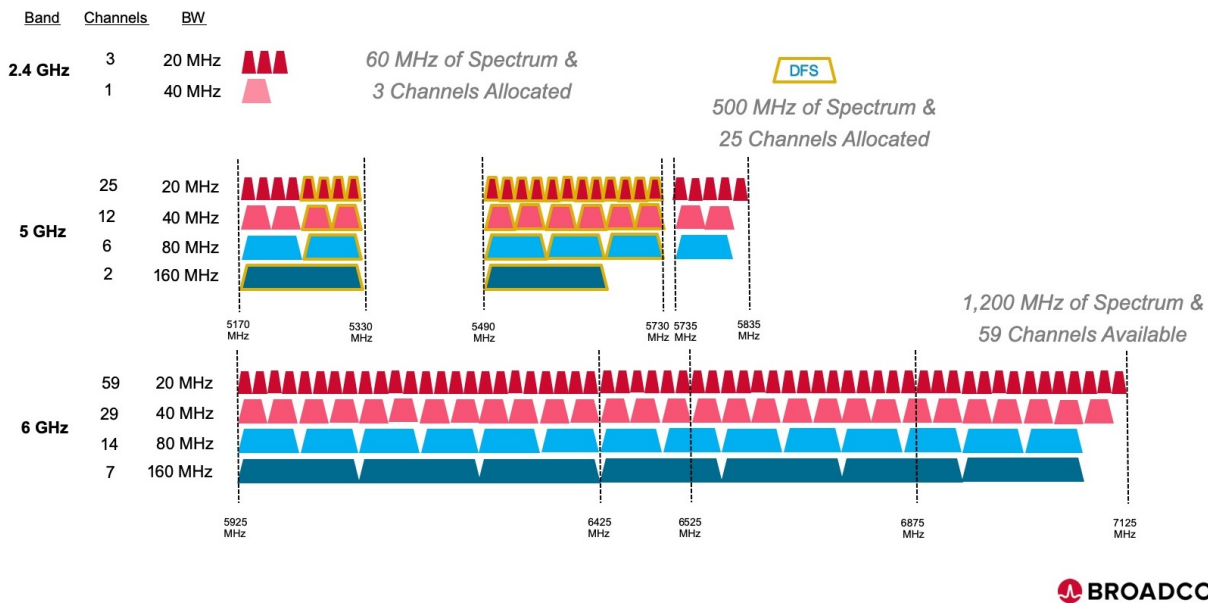



Figure 6.4: Frequency spectra used by various generations of IEEE 802.11, non-overlapping channels⁴

 W-Fi 6 has one extra feature – it is designed to incorporate IoT (Internet of Things) devices, and many things are tailored to this, including the new *TWT* (Target Wake Time) feature – intelligent scheduling to put IoT devices to sleep and wake them up.

Additional information


- <https://www.wi-fi.org/discover-wi-fi> (there are various standards and other topics in the menu on the right)
- <https://www.cisco.com/c/en/us/products/collateral/wireless/white-paper-c11-740788.html> (comparison of IEEE 802.11ax and IEEE 802.11ac)



6.2.2 Signal and Antennas

There are various types of antennas – omnidirectional with different signal shaping methods or directional for point-to-point links. Antennas can be either *internal* (e.g., in a laptop, antennas are usually laid along display) or *external* (interchangeable antennas are usually connected with an N-type or BNC coaxial connector).

⁴From: <https://jp.broadcom.com/info/wifi6e>

 There are various types of antennas. They can be divided according to directionality into:

- *omni-directional* – the most typical for Wi-fi, they transmit in all directions in the range of 360°,
- *semi-directional* – covering a certain angle usually in the range 30–180°,
- *highly-directional* – they are directed to a specific point to communicate with (another antenna), either parabolic or yagi are used.

Most of the following text will deal with omnidirectional antennas.



Figure 6.5: Highly directional antennas – parabolic, parabolic mesh, yagi⁵

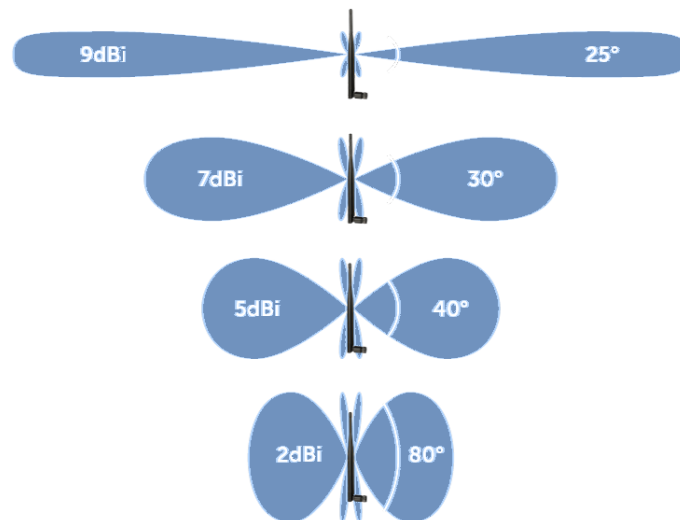



Figure 6.6: Signal coverage for omnidirectional antenna⁶

It is also necessary to consider the vertical propagation of the signal, for example, if we want to cover neighbouring floors or garden, which is lower than the occupied floor of the house, or vice versa if we do not want to unnecessarily interfere with the signal of neighbours from other floors. When we refer to an antenna as omnidirectional, this refers to the horizontal plane (perpendicular to the antenna), not the vertical plane.

The actual signal coverage in 3D may look like in Figure 6.6 (shown vertically). The main coverage area is a kind of “pneumatics”, then in the up and down directions are the side lobes (of course, imagine these in 3D too). In the top-down direction the drawn antennas have less and less gain.

⁵From: <https://www.abctech.cz>, <https://www.tvsatshop.cz>

⁶From: <https://insmart.cz/zlepsit-wifi-signal-domaci-siti/>

 *Antenna gain* is an expression of the relationship between directivity and efficiency: the ratio of the energy radiated in a particular direction to the radiation intensity that would be obtained if radiated in all directions simultaneously, compared to a reference antenna.

We usually compare with a so-called *isotropic antenna*, which is a “ideal” omnidirectional antenna radiating without losses. The gain is given in units of dBi (decibel isotropic), with an isotropic antenna having a gain of 0 dBi. We can see in Figure 6.6 that higher gain means more range horizontally, but less range vertically.

Obviously, if we want to boost signal on a single floor, it is better to use an antenna with higher gain. On the other hand, if we want to send signal to the surrounding floors, we need either a lower gain antenna or to tilt the antennas appropriately (in this case they should have an angle of about 45° between them).

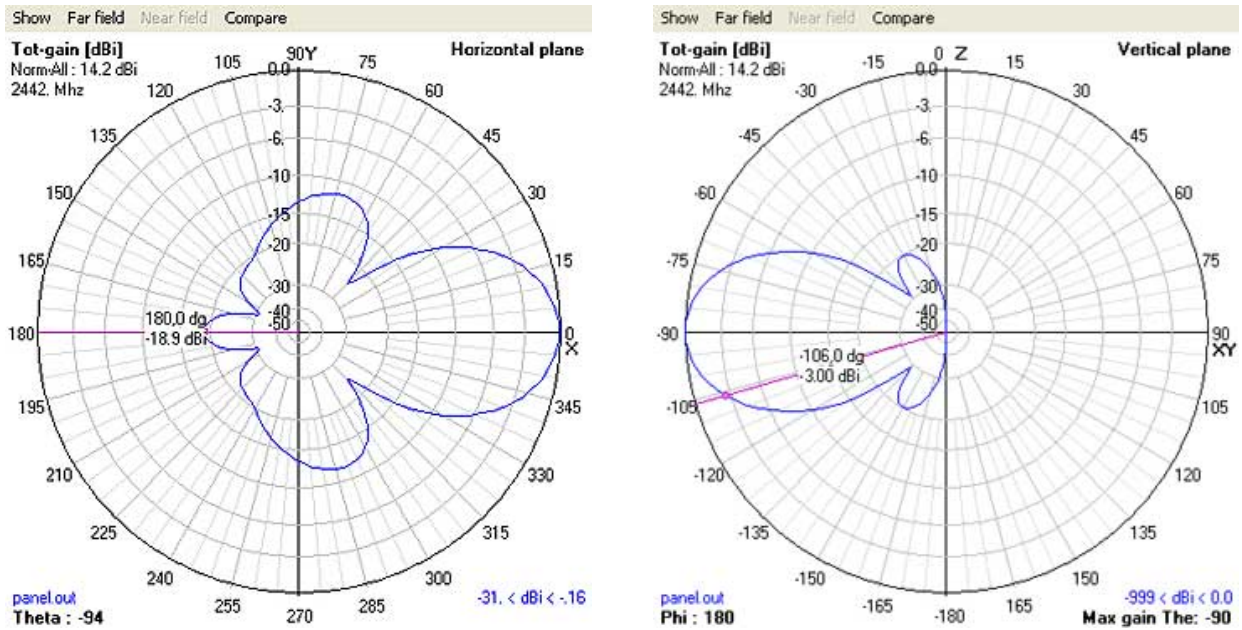



Figure 6.7: Sector antenna radiation pattern – horizontal and vertical⁷

Figure 6.7 shows the radiation patterns of a sector antenna (in horizontal and vertical direction) and the following (here on the right) shows the 3D model created according to these radiation patterns. As we can see, even the sector antennas do not radiate “only” at the specified angle. The gain of sector antennas is typically higher than that of omnidirectional antennas, here specifically 14.2 dBi (this value can be found in the radiation pattern images on the top left).

The situation expressed by the radiation diagrams and the 3D model is, of course, only valid if the signal is not obstructed or interfered with by other signals.

 Antenna power is given in units of dBm (decibel milliwatt) and it is the power per mW (milliwatt). The absolute power

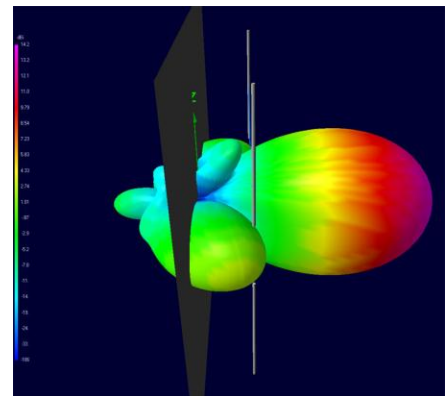


Figure 6.8: 3D model according to the radiation diagrams shown in Figure 6.7⁸

⁷From: <https://www.svethardware.cz/jak-zapojime-sit-wifi-bez-tajemstvi/12953-3>

⁸From: <https://www.svethardware.cz/jak-zapojime-sit-wifi-bez-tajemstvi/12953-3>

per 1 mW is 0 dBm. The value of the result is often a negative number. The power of a Wi-fi antenna can be several tens of dBm “below zero”, for negative numbers the closer to zero the higher the transmit power (note, no absolute value). If we want a somewhat usable signal, it should not be too close to -100 dBm (that would be practically unusable).

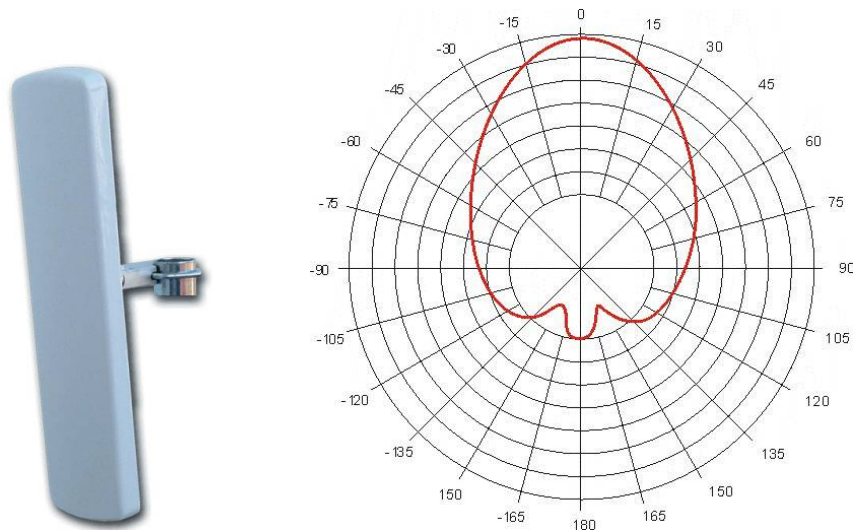


Figure 6.9: Sector antenna radiating at an angle of 65° and its horizontal radiation pattern⁹



Remark

For a person who has no background in electronics or physics, the preceding text may seem “strange”. So what is it with these units?

The unit dB (decibel) has a slightly different basic meaning than the usual units of length, volume, etc. It defines the ratio between two quantities, and the added letter determines which quantities are actually involved. The formula for the calculation is as follows:


$$dB = 10 * \log_{10} \frac{\text{quantity1}}{\text{quantity2}}$$

For example, to calculate the antenna power we can use:

$$dBm = 10 * \log_{10} \frac{\text{quantity1}}{1\text{mW}}$$

The meaning of the logarithm implies that if the antenna has a nominal power less than 1 mW (which can happen with smaller devices) and we substitute this number for “quantity1”, we get a number less than 1 (although positive) in the logarithm parameter, the resulting logarithm will then be a negative number.



 Wi-fi transmits in unlicensed bands, where the transmission power is regulated so that devices do not interfere with each other too much. These limits are always set by the relevant regulatory authority for the country (different countries/continents may have different limits). For the 2.4 GHz and 5 GHz bands, this is often a power limit of 20 dBm.

⁹From: <https://www.zive.cz/clanky/pruvodce-vyberem-spravne-anteny-na-wi-fi-pripojeni/sc-3-a-126652/>

**Remark**

What can affect signal range? Besides power (how much energy I put into the antenna) and gain, barriers and interference. Frequencies around 2.4 GHz are difficult to pass, especially materials containing water (wet walls, many organic materials including leaves, etc.), higher frequencies (5 GHz, 60 GHz) are not bothered by water. Beware of aquariums, they are impassable for 2.4 GHz. Another problematic material is metal (among other things, mirrors have a metallic reflective layer applied).

As far as “solid” materials are concerned, in the case of the 5 GHz frequency band, the wavelength is 6 cm, and anything thicker than 6 cm (like a wall) is harder for such a signal to pass through (the signal tends to be reflected). For frequencies around 2.4 GHz, this “critical” length is 12.5 cm. The higher the frequency, the shorter the wave, and hence the width (thickness) of the potential obstacle decreases. The more the obstacle exceeds a given length, the harder it is for the signal to pass through.


It is not said that the signal will not go through a too “thick” barrier at all. Usually, omnidirectional antennas are used, which generate and receive signals in various directions, so that the signal can reach its destination via a very twisty path, using reflections from other objects.


Interference for the 2.4 GHz band can be generated by Bluetooth transmissions, baby monitors, but also, for example, microwave ovens.

**6.2.3 Multiple Antennas**

Current Wi-fi network interfaces usually have more than a single antenna. Sometimes we are not able to judge this at a glance, because nowadays antennas are (unfortunately) often integrated, or some are external and some are integrated.

Even if we have external antennas, we can’t fully judge how long they are: we usually only see the plastic casing, which can be several centimetres larger than the antenna itself. While it is more or less true that the larger the antenna, the higher the performance may be, in reality multiple parameters affect performance. Also, the higher the frequency, the smaller the antenna can be for the same performance.


 In the IEEE 802.11a/b/g specification we encounter *diversity*, i.e. the possibility of using multiple antennas for a given frequency band. The way diversity works is that when communicating with a particular device, all antennas receive first, and then after evaluating the quality of the received signal from all antennas, one antenna is determined to be used for that device.


 In the IEEE 802.11n specification, the successor to diversity – *MIMO* (Multiple Input, Multiple Output) [maimo] – specifying the use of multiple antennas and an algorithm for combining signals from those antennas. Unlike diversity, (almost) all antennas can operate simultaneously.


When using MIMO, the antennas are divided into Tx (transmitting) and Rx (receiving) to ensure full duplex communication, and another parameter is the number of parallel streams (i.e. how many independent channels can be fit into the spectrum). It is written as follows:

$$\begin{array}{rcccc} \text{Tx} & & \text{Rx} & & \text{streams} \\ 3 & \times & 3 & : & 2 \end{array}$$

which means 3 transmitting antennas, 3 receiving antennas and 2 streams.

 MIMO technology has one disadvantage – an AP can only communicate with a single client at a time. This is addressed by the *MU-MIMO* (Multi-User MIMO) technology used in the IEEE 802.11ac specification. At any one time, the AP can communicate with multiple different clients in parallel (maximum 8 streams, 2 for each client, so a maximum of 4 clients).

 In IEEE 802.11ax (Wi-Fi 6), there is a transition from OFDM to OFDMA multiplexing. What does this mean? Actually, the meaning of the “end” of these abbreviations is different: OFDM is Orthogonal Frequency Division Multiplexing, OFDMA is Orthogonal Frequency-Division Multiple Access. In OFDM, it is possible to use MU-MIMO, but only in a form where different channels are assigned to different end devices. OFDMA goes further: it divides channels into sub-channels (called RU = Resource Unit). Also within the subchannels, orthogonal division is used so that the individual subchannels can be easily separated.

 *Beamforming* is a technology to improve the transmitting/receiving signal of devices with multiple antennas. The purpose is to amplify the “correct” signal towards the device and in contrast to suppress interference. By having more than one antenna, the router can better locate the end device (the direction towards it) and adjust the signal to reach that particular end device with the best quality. Reflections from walls and other barriers can also be used, and for these directions, the time delay is calculated relative to the “straight path” and the phase shift adjusted accordingly so that the signal at the destination is composed correctly without interference from various reflections.

The beamforming technology is already part of IEEE 802.11n, but unfortunately the specification has not been completely finalized and the implementation in various devices (from various manufacturers) may not be compatible with each other. In the case of IEEE 802.11ac and newer, the specification is already complete and any device working according to this standard should be able to handle beamforming.



Remark

In fact, beamforming is not just a Wi-fi issue. It is also used in other wireless and mobile technologies (including LTE). In general, it is a technique for improving the signal coming from a sensor array, and in Wi-fi this sensor array is represented by antennas.





Additional information

- <http://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/802-11ac-solution/q-and-a-c67-734152.html>
- <https://www.howtogeek.com/220774/htg-explains-what-is-beamforming-on-a-wireless-router/>





6.2.4 Access point

 The basic active network equipment in a Wi-fi network is the *access point* (AP). Basically, this device works on the L2 layer (and also L1, of course), so the IEEE 802.11 protocol with its sub-standards implements the L1 and L2 layers (like Ethernet, corresponding to a switch). It is intended to be placed under TCP/IP or other network stack for L3–L7 layers.


 However, the DCE can also support higher layer protocols than just L2, then of course it provides additional functions related to the higher layers, or conversely it can implement only the L1 layer. So a DCE in a Wi-fi network can work in these *modes*:


- *Access Point (AP)* – basic variant, simply implements L1 and L2 layers. It’s similar to a switch from Ethernet networks.
- *Wi-fi router* – it is an AP with added L3 layer functionality. By allowing the IP packet headers, it can route and offers features such as address translation (NAT), DHCP server, hardware firewall (filtering by IP headers), etc.
- *Wi-fi repeater, extender* – works only on L1 layer, so no frames. It receives the signal and retransmits it amplified, or re-generates it. This type of device has compatibility problems (especially when communicating between devices from different manufacturers), and it also reduces network throughput (it usually receives and transmits in the same band, so throughput can be halved). In homes, this is quite a useful feature for increasing signal range.
- *AP client* – communicates wirelessly with a full-featured AP, Wi-fi clients are usually connected to it by cable. So unlike the previous solution, here we have cables, in relation to the AP it pretends to be a client device and thus only mediates the communication of “real” clients with the AP. There is not such a large reduction in network throughput. WDS (Wireless Distribution System) – we link multiple APs in a network to cover a larger area, a mobile client is being “passed” between the linked APs. It’s not a standardized solution, there are compatibility issues with devices from different manufacturers. In addition, network throughput is significantly reduced, similar to a repeater.
- *Gateway* – mediates communication with another network working with different protocols. In the form of a module, this functionality is often present in ADSL/VDSL Wi-fi routers, whereby the module connects the local network (WLAN) to the access network (ADSL or VDSL). Thus, the device has a WAN port (i.e. a port leading to the access network) of a given type, for example ADSL (this would be a port with an RJ-11 interface, the cable would lead to a telephone socket). WISP AP (Wireless Internet Service Provider AP) – wireless is the WAN port (i.e. you communicate wirelessly with your ISP), whereas ports for a local area network are, for example, Ethernet ports.


 The AP (of whatever type) transmits a signal, the area covered by this signal is called *cell* or *Basic Service Area (BSA)*. If we connect multiple APs together in a distribution system (WDS), then the area covered by the signal of the connected APs is called *extended service area (ESA)*.

 Client devices need to identify the network they are connecting to or working on. That’s why we have the following *identifiers*:

- *SSID (Service Set ID)* is the identifier of the entire network, the network name. It is a string of up to 32 characters that every device that logs into the network must know.
- *BSSID (Basic Service Set ID)* is the identifier of the access point. This is usually the MAC address of this AP, so the length is 6 octets. Each device in the cell needs to know the BSSID of its AP because it places this address in the header of the frames it sends.
- *ESSID (Extended Service Set ID)* is the WDS identifier when we have multiple APs connected to a distribution system.

 **Definition (Beacon frame)**

Each AP sends a *Beacon frame* at regular intervals to inform about its parameters. Among other things, the Beacon frame contains the SSID string. 

 AP provides the following services in the wireless network:

- *authentication* – decides whether to allow a specific end device into its cell and thus into the network,
- *association* – if an end device is allowed to access, it needs to establish a link between the AP and this device, i.e. technically provide the device with the ability to communicate in the cell,
- *de-association* – to break this binding.

6.2.5 Wi-fi at L2 Layer

IEEE 802.11 implements only the lower L2 sublayer – the MAC sublayer. It leaves the upper sublayer to the LLC protocol (IEEE 802.2), which means that we use LLC frames (as outlined in Chapter 2, starting on page ??), which we encapsulate in an IEEE 802.11 MAC frame.

 There are three types of MAC frames according to IEEE 802.11:

- *Data Frames* – contain the payload in the LLC frame,
- *Control Frames* – used, for example, to acknowledge received frames,
- *Management Frames* – this includes Beacon frames, frames for negotiating client device connection to the network (association) or device disconnection, etc.

Let's focus on the MAC frame header according to IEEE 802.11. It is 30 octets long, which is a bit longer than Ethernet (in addition, a checksum trailer is also added, just like Ethernet). We won't go through them in detail, we'll just show how to work with the two flags and the addresses.

All of this information in the header changes depending on which part of the Wi-fi network the frame is currently passing through. We distinguish the following cases:

1. We are in an ad-hoc network, i.e. the connections are always between two client devices.
2. We are in an infrastructure type network, the frame is on the link between the source of the frame (client device) and an AP (segment I. in Figure 6.10).
3. We are inside the infrastructure, the frame is on the link between two APs (segment II. in Figure 6.10).
4. We are in an infrastructure type network, the frame is on the link between an AP and the destination of the frame (client device) (segment III. in Figure 6.10).

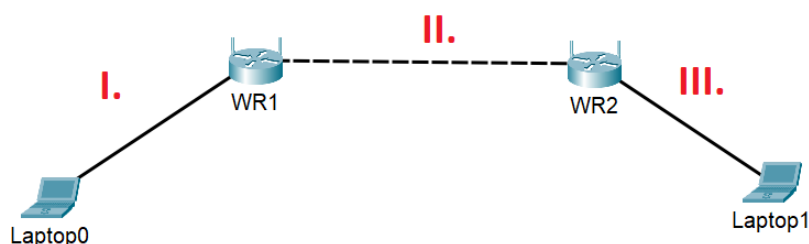


Figure 6.10: Infrastructure type network, segments of a frame path

So it depends on what type of link the frame is currently on, or between what two types of nodes in the network it is currently passing through (who is the current sender and receiver).



We have the following two flags (i.e., single-bit values) in the header:

- **FromDS** – is set to 1 if an AP (DCE) is the current sender on the link, and is set to 0 if an end device (DTE) is the sender.
- **ToDS** – is set to 1 if an AP (DCE) is the current receiver on the link, and is set to 0 if a DTE is the receiver.

The “DS” abbreviation in the flag names means *distribution system*.



It follows that in the four cases outlined above, the flags are set as follows:

1. In an ad-hoc network both flags are always set to 0, because no APs can be on the path.
2. **I.:** On the link between the frame source (DTE) and the nearest AP is **FromDS=0** (the sender is a DTE), **ToDS=1** (the current receiver is an AP).
3. **II.:** On the link between two APs, both flags are set to 1: **FromDS=1**, **ToDS=1**. There are APs at both ends of the link, so we are inside the distribution system.
4. **III.:** On the link between an AP and the frame destination (DTE), i.e. at the end of the path, there is **FromDS=1**, **ToDS=0**.



Now briefly on *addresses*. In an Ethernet frame, we always have two addresses – the MAC address of the destination and the MAC address of the source. In Wi-fi frame, we also use MAC addresses (the same type and structure of address as Ethernet), but the number of addresses changes dynamically according to the current position in the network, similar to the two flags mentioned above.

In an ad-hoc network we only have the MAC address of the destination and the MAC address of the source, we don’t need anything else. But in infrastructure mode, we add the addresses of the closest APs on a given path segment – we have the largest number of addresses in the header if the frame is currently going between two APs (the header contains the addresses of the source, destination, and the two neighboring APs). The number and order of addresses in the header changes over time, but it is always the case that the first two addresses in the sequence are for the current path segment, and on that segment the end-of-segment address is always first in the sequence, then the beginning-of-segment address.

6.2.6 Collision Method

The IEEE 802.11 standard also defines an access (collision) method that is used at the MAC sublayer of the L2 layer. Unlike Ethernet, it is used permanently in wireless networks because collisions can never be completely avoided with shared media.



Definition (CSMA/CA Collision method, RTS/CTS mechanism)

According to IEEE 802.11, the *CSMA/CA* access method is used:

- CS (Carrier Sense) – we listen on the carrier,
- MA (Multiple Access) – shared transmission medium,
- CA (Collision Avoidance) – collisions are avoided, i.e. prevented (it is not enough to detect them).


Any device that wants to transmit must first ask the AP for permission (this is a short administration frame) and once it gets it, it can send data. CSMA/CA can be implemented in several different ways, one of them is the *RTS/CTS* (Request to Send/Clear to Send) mechanism:

- DTE sends an RTS signal (in a short control frame) to request permission to transmit data,
- if the transmission can be permitted, the AP sends a CTS signal (in control frame) to the requesting DTE,
- the DTE can send a data frame (MAC according to IEEE 802.11 with LLC frame, containing data),
- an acknowledgement (ACK) is received back, also in the control frame.



As we can see, Wi-fi uses acknowledged transmission in this case without establishing a connection.

However, even with such a designated access method, collisions may occur because the transmission medium is shared. They occur, for example, if two devices send an RTS signal at the same time or if frames with RTS and the start of a data frame transmission meet in this way.

 *Hidden stations problem* occurs when the cell is larger and two stations belonging to the same cell are “not visible” to each other (each of them can reach the signal from the central AP, but not their signals to each other). This is quite common; stations may be too far apart or shielded from each other by furniture or walls, and the signal of end devices is typically weaker than the AP signal.




Remark

The RTS/CTS mechanism more or less solves the problem of hidden stations – it reduces the number of collisions to a minimum by making short control frames much less likely to collide than large data frames, and the AP will also not send a CTS signal with transmission permission if another (hidden to the requester) node in the network is currently communicating on that frequency.




6.3 Security in Wireless Communication

6.3.1 AAA

 AAA (Authentication, Authorization, Accounting) is a term that refers to access control, in this case access to a wireless network. As the acronym suggests, it involves three phases:

- *Authentication* – the client (and in some cases also the other side) has to prove its identity, so it is an identity check. The authenticated party proves its identity with credentials (name and password, fingerprint, access token, etc.), and the authenticator verifies them.
- *Authorization* – when the identity of the client is established, it is necessary to assign access permissions to the client, to define the allowed activity.
- *Accounting* – we record the specified client activities in the network (according to the configuration), or react when attempts to exceed the permissions assigned in the authorization phase.

Each of the following options provides encryption both in the first phase of communication (authentication and association) and during regular operation. However, various procedures and various encryption algorithms are used (also with respect to security).

 A short note on encryption. For any encryption, we need an *encryption key* that both parties to the communication should have (the sender for encryption and the receiver for decryption).

There are two basic types of ciphers – *symmetric and asymmetric*. Given the encryption keys, there are the following differences:

- Symmetric ciphers use the same key for encryption and decryption. The main problem (and disadvantage) is how to *securely* get the key to the other communicating party.
- Asymmetric ciphers use a pair of keys for each direction of communication – private and public, and what I encrypt with the public key must be decrypted with the private key (and vice versa – they are interchangeable in functionality). The user generates his key pair, stores the private one, and passes the public one (somehow) to the other party. The other party does the same with their own key pair. If I want to send something, I encrypt the data with the receiver’s public key, send it, and the receiver decrypts it with his private key.


The disadvantage of symmetric ciphers is the problem of passing the key to the other party, but the advantage is the high speed of encryption and decryption. For asymmetric ciphers it is exactly the opposite – the advantage is the possibility to transport the public key even over an insecure channel, the disadvantage is the high computational complexity (it slows down the transmission a lot).

Therefore, in practice, both approaches are combined in some ways – we use *hybrid encryption*. After the connection is established, the transmitted data is encrypted symmetrically so as not to slow down the communication too much, but beforehand the symmetric key is passed on using asymmetric encryption (instead of the data, we simply encrypt the symmetric key, which allows us to pass it on to the other side relatively securely).

To increase security, in addition to the (symmetric) static encryption key, a dynamically changing addition – the *IV vector* (initialization vector) – is used for each frame sent. The purpose is to make it as difficult as possible to eavesdrop on the communication. It’s just that when something is constantly changing, we need a mechanism to let the receiver know what form the IV vector takes for that particular frame (it’s nice that the hacker doesn’t decrypt our frame, but the receiver should be able to). There are two approaches for this: either both parties have an algorithm to determine the IV vector for a given frame, or we simply send the current IV vector within the given frame to the receiver. The first option is of course much safer.

6.3.2 Security Protocols

There are several possibilities for Wi-fi to secure communication.

 **WEP (Wired Equivalent Privacy).** This is a security mechanism that cannot be called secure anymore. The only advantage is full compatibility with anything that has implemented IEEE 802.11.


The same encryption (if any) is used for the first phase of communication and for regular traffic. Only symmetric algorithm is used for encryption, i.e. the key must be somehow passed to

the receiver (client). The IV vector is transmitted as part of the (unencrypted) header in the vast majority of cases.

WEP offers three modes:

- open network (no encryption),
- symmetric RC4 cipher with a 64-bit key, with the IV vector being its 24 bits (so 40 bits is the static part of the key),
- symmetric RC4 cipher with a 128-bit key, with the IV vector also being its 24 bits (104 bits being the static part of the key).

In any case, it only takes a few minutes to crack the key (using AirCrack), so WEP is considered a completely unsatisfactory security method.

 **WEP and IEEE 802.1X.** The WEP mechanism was soon found to be inadequate, but there was no other option for a long time. Therefore, an additional login security option in the form of an IEEE 802.1X implementation (note, not 11) began to be used in corporate networks. This consists of providing authentication and authorization using an authentication server that maintains a database of “authorized” users with credentials and authorization information. Authentication servers implement the AAA mechanism.

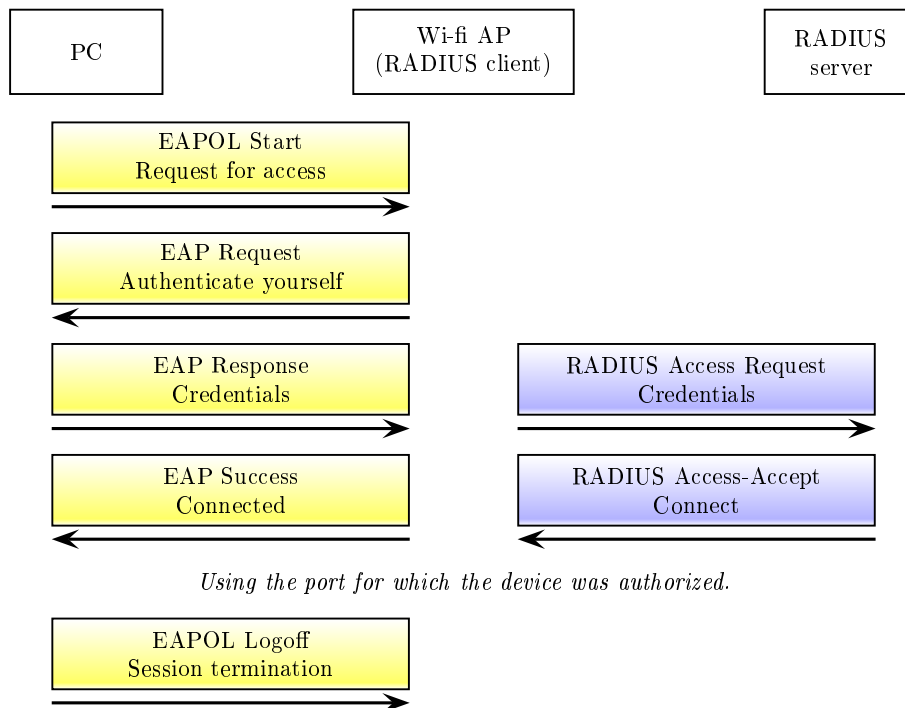




Figure 6.11: Simplified network communication when using a RADIUS server

 There exist several authentication servers in use – RADIUS, DIAMETER, and TACACS+. There are freely available implementations for RADIUS and DIAMETER (for example, the open-source projects FreeRADIUS and freeDiameter), whereas TACACS+ is a proprietary solution (Cisco).

The principle is that it is not enough for a users to have a key, but they must also have their login credentials (for example, a name and password or perhaps a hardware token). Figure 6.11

shows the communication with a RADIUS server – the client logs in to a cell to the AP, but instead of the AP deciding on the connection directly, it just forwards the login credentials to the RADIUS server and actually acts as an intermediary during authentication and authorization. In addition to the credentials, the RADIUS server may request additional information to decide on authentication and specific authorization parameters.


A RADIUS network contains only two nodes – a RADIUS server and a RADIUS client (but there can be multiple RADIUS clients), which is the AP in question, the connection is point-to-point. This connection should be well secured (no wi-fi, there should be a cable, and encryption too). The client itself (the supplicant) is not part of the RADIUS network.

 **WPA (Wi-fi Protected Access).** The WPA mechanism was originally a temporary solution, created while waiting for the WPA2 standard. Today, it is considered to be *something between* the insecure WEP and the secure WPA2, and unlike WPA2, it is also suitable for older devices on which WPA2 cannot be used. On such a device, a firmware upgrade was sufficient (i.e. a software issue), WPA did not require hardware modification. Simply put, WPA can be thought of as WPA2 pruned of its hardware-dependent components.

The basis is similar to WEP – an RC4 cipher with a 128-bit key whose IV vector is 48-bit. However, the management of dynamic IV vectors is handled by the Temporal Key Integrity Protocol (TKIP), so the IV vector can no longer be eavesdropped on from an unencrypted frame header. In addition, the TKIP protocol adds a 64-bit checksum (rather, a digital signature of the frame) to the frame, called MIC (Message Integrity Check, commonly referred to as “Michael”), which is used to tell if the frame has been corrupted or intentionally modified along the way.

WPA works in two variants:


- WPA-Enterprise – for the corporate sphere, basically the successor of WEP+RADIUS, there must be an authentication server in the network,
- WPA-Personal (WPA-PSK) for home and small business, a shared key (PSK = Pre-Shared Key) is used for authentication, we don’t have an authentication server, but the user still needs authentication information (password 8–64 characters or 64 hexadecimal digits) from which the key is dynamically determined.

 **WPA2 (IEEE 802.11i).** There is a standard for this mechanism. Compared to WPA, the Counter-Mode CBC MAC Protocol (CCMP) is used instead of TKIP, and encrypts with the symmetric AES cipher. And just using AES encryption requires hardware support on devices (nowadays most processors already include AES instructions, so there is usually no problem getting it to work).

As with WPA, each user needs login credentials that are used either for the authentication server or for the PSK protocol, so again we have two options:

- WPA2-Enterprise – for the corporate sector, with an authentication server,
- WPA2-Personal (WPA2-PSK) for homes and small businesses.

In 2017, a report emerged that the WPA2 mechanism is vulnerable to Key Reinstallation Attacks (KRACK), which can be used to bypass security with encrypted connections.

 **WPA3** was introduced in 2018. Again, there are two modes – WPA3-Enterprise and WPA3-Personal. For Enterprise use, AES-256 encryption is prescribed in GCM mode with HMAC SHA-384 hashing. In Personal mode, AES-128 can be used with CCMP as in WPA2, but key exchange

is done using the SAE (Simultaneous Authentication of Equals) algorithm, which is a variant of Dragonfly Key Exchange (works similar to Diffie-Hellman).

The arrival of WPA3 was delayed by detection of serious security vulnerabilities in WPA2, but the WPA3 mechanism itself has not escaped security vulnerabilities. For example, the aforementioned Dragonfly algorithm (used in WPA3 to establish connections for secure key exchange) had a security weakness, DragonBlood, which was exploitable in several different ways.



Additional information

- <https://www.wi-fi.org/discover-wi-fi/security>
- <https://medium.com/@reliancegcs/wpa3-explained-wi-fi-is-getting-major-security-update-2b6dca8f3aff>



6.3.3 WPS

The WPS (Wi-fi Protected Setup) mechanism emerged as more and more various types of devices were connected to the Wi-fi network (IoT devices, smart home, W-fi extenders, etc.). Many devices do not have a keyboard or a proper display, so entering passwords and keys would be problematic. WPS is just used to simplify authentication and association of such devices to a Wi-fi network.

There are two options for using the WPS mechanism – WPS button or WPS PIN.



WPS button (PBC – Push Button). On the AP we press the button labeled WPS (or QSS or PBC) or in the AP administration click on “software” button. Within a few tens of seconds, the AP scans its cell for new devices. If (any) device in the cell starts to report within this time interval (which is achieved by also pressing the button on the device in question or doing something similar as instructed), it is considered trusted and the AP will negotiate an association with it (so we don’t need to enter any passwords).

The problems are mainly as follows:

- In that time interval, our network is virtually unprotected, anything can log in and get in.
- We have to hit the button on the associated device in time, so in some cases it’s more of a two-person job.



WPS PIN. For a device that we want to get into the network this way, we need to find out its PIN – an 8-digit number that we can find either on a sticker on the device itself or somewhere in the documentation. We then enter the PIN in the administration of our AP, then the AP and the device negotiate all the parameters and the association is done.

However, there are problems with this method as well:

- AP with WPS PIN support *listens constantly*.
- PIN is an 8 digit number, the last digit is calculated from the previous ones, it is similar to a checksum. If the PIN is entered incorrectly into the administration, the AP returns information from which one can deduce which half is incorrect – if the second half is incorrect, one only needs to guess three digits, which is a time-saving game for the average hacker.


If our Wi-fi AP has this feature, it’s better to turn it off, and only turn it on when we really want to associate a new device this way.


6.3.4 So how to Secure a Wireless Network

So what can we do to make our Wi-Fi network as secure as possible? First of all, set up WPA2 or WPA3 encryption if all our clients can handle it.


The web administration interface is usually accessed via the gateway address (i.e. we list the IP address and related data on the connected device, for example in Windows using the `ipconfig` command). This address is entered into the web browser. And then we need the login credentials, which can be found either in the documentation or on the web. Today, unfortunately, we can also encounter devices that are managed by our ISP, where this procedure does not help.

Most of the time we have at least basic access via the mobile app. If nothing else is available, at least this.

 **Credentials to administration.** AP administration credentials are set to some default values by default (typically `admin-admin` or something like that). Under no circumstances should we leave it that way! So right after purchase during the initial configuration, the first thing we should do is to set our own AP administration credentials.


 **Hiding SSID.** The AP broadcasts a Beacon frame at regular intervals, which contains, among other things, the SSID (i.e. the network name). The SSID is needed to be connected into the network, so if the AP does not broadcast Beacon frames with the SSID, theoretically we would prevent login attempts by people who have no right to be on the network. Practically, however, the SSID can be eavesdropped in the authentication communication of “allowed” devices, so we just need to force some such device to reconnect (drop its communication).


In addition, hiding the SSID can cause problems when finding the source of interference on another network – there may be a situation where we can see that something is interfering with our network, but we can’t see what it is because the owner has a hidden SSID.


 **MAC address filtering.** In AP administration we can create a black list (list of forbidden) or white list (list of allowed, no one else will be associated) MAC addresses. But just note that the MAC address can be changed (in Linux it only takes one command, in Windows a registry edit or a special program), so again a toothless operation.

Chapter 7

Application Protocols

 *Quick preview:* In this chapter we will focus on the L7 layer, the application layer. We will look at the services provided at this layer, and the protocols that define the technology that implements these services (i.e., the application protocols).


 *Keywords:* DNS, domain, name address, zone, DNS server, host table, DNS packet, WHOIS, URL, URI, URN, HTTP, HTTPS, SMTP, IMAP, MTA, MDA, MUA, MIME, FTP, SMB, DHCP, Telnet, SSH, SNMP

 *Objectives:* After studying this chapter, you will gain an overview of the application layer protocols. You will know how DNS translation works, how a client device communicates with a web server, how e-mail works, how files are transferred, how computers communicate on a simple peer-to-peer network, how addresses are assigned, and how to communicate with a remote device.

7.1 DNS Service

7.1.1 Domains and Name Service

We know that computers and networks are addressed by IP addresses, and without these addresses (or their equivalent in competing technologies) they cannot exist on the network. It's just that IP addresses (especially for IPv6) are hard for humans to remember, and people would probably revolt if we forced them to write these addresses in, say, the address bar of a web browser.

 People use *named (domain) addresses* of devices, which are easier to remember and easy to write down. For example, we can type `www.google.com` in the address bar of a web browser and not have to bother with `216.58.214.196` or `2a00:1450:400d:802::2004`.

But network devices will definitely not use such addresses. So we need a mechanism that performs a mapping between a specific device name address and its IP address, and that is *DNS (Domain Name System)*.

**Definition (Domain, domain name)**

Domain is a network or group of networks under common management and sharing a common (domain) name. Each *Domain Name* must meet the following conditions:

- may contain letters of the English alphabet, numbers and a hyphen, the hyphen not being at the beginning or end,
- the maximum length of a single name is 63 characters,
- the maximum length of concatenated domain names is 255 characters,
- within a domain, subdomain names are unique.

A domain can have multiple names assigned to it. One of them is *canonical name* (main), the other names are called *aliases*.



The IP address space is said to be hierarchical, with addresses in the left-hand direction (devices on the same network have the left-hand part of the address the same), with the network divided into subnets and their relationship can be plotted using a tree. The domain address space is also *hierarchical*, but in the right-hand direction – devices in the same domain have the part of the name address the identical, domains are built hierarchically. We can draw their relationship using a tree as well.



The top levels of the domain tree also have their names:

- At the root of the tree there is the *root domain*.
- *Top-Level Domains* (TLD) are at the first level of the tree, e.g. `.org`.
- There are *Second-Level Domains* (SLD), third level, etc.


A domain or group of domains is managed by a specific entity – *domain administrator*. According to our picture, for example, the domain `.slu`, including subdomains, is managed by Silesian University, the domain `.cz` is managed by the main domain registrar for the Czech Republic, CZ.NIC.

The list of TLD domains is maintained by IANA (Internet Assigned Numbers Authority), similarly as IPv4 and IPv6 address spaces, at the Root zone database (database present at the root servers).


There are several types of the TLD domains:

- generic TLD (gTLD): created in early development of the Internet, named after the focus:
 - `.com` – commercial organizations,
 - `.edu` – educational institutions,
 - `.gov` – USA government,
 - `.mil` – USA military,
 - `.net` – organizations maintaining network standards, nodes for distributed computing, anything related to computer networks,
 - `.org` – various organizations,
- country code TLD (ccTLD): international country codes, national abbreviations (`.cz`, `.uk`, `.sk`, etc.) or for groups of states (`.eu`),


- infrastructure TLD (ARPA): the domain `.arpa` is intended for the network infrastructure management, for example the subdomain `.in-addr.arpa` is used for reverse DNS lookup (translation of an IPv4 address to the corresponding domain address) or `.ipv6.arpa` (the same for IPv6),
- sponsored TLD (sTLD): hundreds of new TLDs, for example `.audio`, `.cafe`, `.download`, `.help`, `.chat`, `.training`, etc.

 *FQDN* (Fully Qualified Domain Name) is the full name of a device, the full domain address. This address is compiled by going up from the leaf (the given device/domain from the request) upwards to the root and separating the domains on the path with a dot, for example `www.slu.cz`. The maximum length of the FQDN is 255 characters.


7.1.2 Zones and DNS Servers

 We associate one or more domains into a common *zone* (typically part of a subtree of domains). A specific *authority*, i.e. a responsible organization, is determined for each zone.


Zoning (as grouping of domains) is related to responsibility and technical administration, zones usually do not overlap, except for their boundaries (there must be some communication between the zones). For example, the entire subtree `slu` forms a zone managed by the Silesian University, while the upper domain `cz` belongs to the zone managed by CZ.NIC.

 Zone management is ensured on the *domain (DNS) server* performing the following tasks:

- keep an address table (host table), it is stored in the *zone file*,
- answer DNS requests according to this table, e.g. translate a name address to the corresponding IP address.

 We have a single primary DNS server in the zone and we can also have auxiliary servers there – secondary DNS servers and caching-only servers. In relation to the zone file:

- The *primary DNS server* maintains and guarantees a zone file with the host table, we always make changes to the table on this server and the records on this server are always trusted, *authoritative*.
- The *secondary DNS servers* copy the zone file from the primary DNS server at regular intervals (synchronize, the procedure is called *zone transfer*), their purpose is to distribute the load so that the primary server is not overloaded. Their answer is *authoritative* as well.
- The *caching-only servers* do not keep zone file. When such a server receives a request for an address, “asks” the primary or secondary server, but stores the response in cache for a short time and, if it is requested for the same address, uses the information from cache (so it doesn’t have to query further). The caching-only server response is not authoritative, but usually sufficient.

 *DNS resolver* is a component that provides DNS requests and keeps the results of previous requests in its cache for some time. This component can be found in any system that is connected to a network using DNS, including computers and mobile devices. The most important information that a DNS resolver needs is the address of the responsible DNS server.

DNS servers also have their own DNS resolvers, because even these devices need such DNS records that they do not have in their zone file. Their resolver queries another DNS server and stores the result in its cache (so both primary and secondary DNS servers have a cache, not just caching-only servers).




Procedure (Possibilities of DNS request evaluation)

The DNS resolver gradually tries the following options when evaluating a DNS request:

- the file `/etc/hosts` (UNIX-like systems) or `... \system32\drivers\etc\hosts` (Windows) contains a list of pairs
IP address + name address,
- the result is in the DNS cache if we requested the same name address in the near past, but some UNIX-like systems usually do not use DNS caching,
- the last and most time consuming option is to query a DNS server.



 Medium and larger companies have their own DNS server, while smaller companies and households use the DNS server services of their ISP. In addition, there are many *public DNS servers* that we can use when the one from our ISP does not work as it should or is not stable enough, for example:


- Cloudflare: 1.1.1.1 (figures) and 1.0.0.1,
- Google: 8.8.8.8 (snowmen) and 8.8.4.4,
- Cisco OpenDNS: 208.67.222.222 and 208.67.220.220,
- Quad9: 9.9.9.9 and 149.112.112.112,
- Verisign: 64.6.64.6 and 64.6.65.6, etc.

We can set it up on the end device, but it is more practical to do it on a router with DHCP server function, so that this information reaches all end devices during obtaining address, and on laptops we avoid problems if we bring the device to work and try join Active Directory.

7.1.3 Evaluation of DNS Requests


DNS servers, as mentioned above, keep information about its domain, the subdomains in the zone file, as well as information about the DNS server of the upper-level domain. Each DNS server knows at least one root DNS server.

DNS querying can be distributed – the DNS server often cannot respond to a request immediately according to its records, so it turns or references some related TLD DNS server or one of the root servers.

 A questioner (actually a resolver) sends a query (what is the IP address of the device named `www.abcd.org`?) To the so-called *local DNS server* (the closest one). If this server already knows the response, it will provide it. If not, the next steps differ depending on the type of query used. We distinguish two types of DNS queries:

- a *recursive query* – the questioner receives a ready-made answer or an error message (domain does not exist),


- an *iterative query* – the questioner gradually gets references to servers where a better answer should be got.

 **Recursive query.** If the requested DNS server knows the response, it will reply back immediately. Conversely, if it does not know the response (the address is not from its domain), it contacts a relevant TLD server or a root server. The next queried DNS server behaves similarly – if it knows the destination, it replies with the address, if it does not know the destination, it continues to query (usually in its subdomains).

In this way, the query is sent recursively down the domain tree. When it finally reaches a DNS server that knows the response, that server sends the response in the recursive path back.

Thus, the procedure is as follows:


- If the queried name is from a subordinate zone (subtree, the right part of the address is the same), the name server finds the first domain in the FQDN that is not its from right-to-left. It has links to all domains on the boundary with subordinate zones (for example, in the zone file for `cz` there is a link to the DNS server of the domain `slu`), so it passes the query to the DNS server of this subordinate domain. In the lower-level domain, the query is processed or resubmitted below. The response is found recursively downwards, which is then distributed back to the first resolver in the same way.
- If the name being queried is neither from the own zone nor subordinate zones, the DNS server forwards the query to the DNS server from the relevant TLD zone (if it does not know the relevant TLD server, the query is forwarded to some of the root servers). It either handles the query or passes it to a low-level node. After processing, the reply is again distributed in the same way to the first resolver.

 **Example**

If a device from the domain `fpf.slu.cz` asks for the IP address of the server `www.fpf.slu.cz`, the DNS server of the domain `fpf.slu.cz` responds immediately, because this web server belongs to its zone.

If a device from the domain `fpf.slu.cz` queries the IP address of the server `www.cuni.cz`, the DNS server of the domain `fpf.slu.cz` deals with the query first. The requested domain is not in the zone file, but the TLD domain is `.cz`, so the name server for the given TLD domain is the next part of the request chain.

This TLD server finds out that the queried address belongs to the lower-level domain `cuni.cz`, in the zone file it finds a contact to the DNS server from that domain, which already finds the response – the IP address of the web server `www.cuni.cz`, because it belongs to its zone. The response goes back to the first querying device in the same way as the query.

 **Iterative query.** Here the activity is mainly on the side of the resolver. If the queried (local) DNS server does not know the answer, it does not formulate further queries, but instead sends back the list of DNS servers that might know the answer (“I don’t know, ask xxxx”). The resolver chooses one of the recommended DNS servers and sends query further. Each queried server again sends either a final response or a recommendation with the names of other DNS servers (so called best possible answer).



Example

If we need to find out the IP address of the computer `www.something.org`, we contact the local DNS server. It will recommend that we send the query to one of the TLD servers of the `org` domain. The next queried root server either looks up the address in its records or returns only the addresses of the name servers that may know the response.

The query is evaluated distributively in this way (gradually according to the nesting of domains of different levels), the last queried DNS server returns the required IP address.



In the real world, recursive and iterative querying are combined. The recursive method is used in the first phases of the path, when a simple resolver is queried, in subsequent phases iterative querying takes place. Some DNS servers (as software) can only work recursively, others can also work iteratively.

7.1.4 Host Table



A key element for DNS resolution is the *host table* stored in the zone file. We can think of it as a table in which each *record* (Resource Record, RR) has its own row, and on that row we have the following data (in columns) – for the most common types of records:

- name (e.g. name address to map),
- record type,
- class – usually “IN” as Internet,
- TTL,
- IP address to map or other data, depending on the record type.

The TTL value is the lifetime of the resource record in seconds in case this record is downloaded to cache of another device.




The fact that “record type” (the third item of the previous list) is present means that there are different types of DNS records. The most used record types are:

- A – to map a name address to an IPv4 address,
- AAAA – to map a name address to an IPv6 address (four “A”s are here because IPv6 addresses are four times longer than IPv4 addresses),
- CNAME (Canonical Name) – to map an alias to a canonical name,
- SOA (Start of Authority) – administrative information for the given domain,
- NS (Name Server) – record for IP addresses of authoritative DNS server(s) for the given domain,
- MX (Mail eXchanger) – e-mail server for the given domain,
- SRV (Service locator) – similar meaning as NS and MX, but more general, for various types of services,
- TXT – variable type record used, for example, to store public keys that anyone can use to verify that an e-mail sent from our domain has actually passed through our mail server or has been spoofed,
- typ PTR – for reverse mapping (to map an IP address to the relevant name address),

- IPSECKEY, OPENPGPKEY, SMIMEA, SSHFP, TLSA – records to store security related information such as public keys,
- etc.

We usually need records of type A or AAAA, when we know the name address and we need either an IPv4 address or an IPv6 address.

 We distinguish between *forward lookup* (finding an IP address by domain name) and *reverse lookup* (finding a domain name by IP address). The reverse lookup is performed through the domain `in-addr.arpa`. The main problem here is that both name and IP addresses are hierarchical, but each in a “different direction” – the TLD domain in the name address is on the right, while the network address (main part) in the IP address is on the left. The PTR records are used for reverse lookups. They map IP addresses to canonical names.

Example

Suppose that somebody wants to know the domain name of the IPv4 address 77.201.75.176. The appropriate PTR record holds the IP address with bytes in the reverse order, followed by the infrastructure domain `in-addr.arpa`:

`176.75.201.77.in-addr.arpa`

This PTR record leads to the searched domain (its canonical name).



7.1.5 DNS Protocol and DNS Packet

DNS is not just a service, it is also the application protocol that provides this service. This protocol determines what a DNS packet should look like, how it should be handled, how the entire DNS system works, including how to work with zone files, their transfer between DNS servers (zone transfer), and mutual communication between DNS servers.

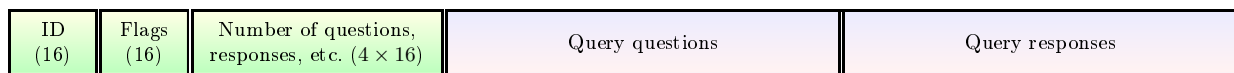




Figure 7.1: DNS packet


 The DNS protocol defines client-server communication, i.e. we distinguish between *query question* and *query response*. The same *DNS packet format* is used for both question and response. The header holds

- *identifier* (similar to IP) used to match the question and the response,
- *flags*, the most important of which is the one that determines whether it is a question or response packet, or, for example, in the case of a response, whether the server being queried is authoritative,
- several two-octet numerals (number of questions, responses, authoritative responses, additional information),
- question(s),
- response(s).

 The DNS response packet is created as follows:

- we use the same identifier as in the query question,
- in the flag field we set the flag for the answer and then a few more as needed,
- we copy the field for the number of questions, enter the number of found records from the zone file in the field for the number of responses,
- we copy the query questions field,
- we fill in the field for the response with the content of the found records from the zone file.

The authoritative response comes from a primary or secondary DNS server. From the DNS packet we also know whether the given response is authoritative or not, which servers are authoritative in the given zone, or it is possible to force an authoritative response by setting a flag in the query question packet.

 DNS packets are encapsulated into UDP or TCP segments, on the DNS server side it communicates *on the port 53* (for both UDP and TCP). UDP segments are preferred because speed is important for this type of communication. TCP is typically used when a DNS packet needs to be split into multiple segments, and when executing a zone transfer.

Because there can be several different clients communicating with the DNS server, each such client (for example, a web browser window, a mail client, Skype, etc.) must have its own dynamic port number in order to distinguish these communications.

Tasks

If you haven't done so already, browse a few different DNS packets (there are usually a lot of them wandering around the network, or use a web browser). Compare the query question and the corresponding query response. Notice how the query and the response are mapped – how the client recognizes that a particular response belongs to a particular query.



7.1.6 WHOIS Database

As mentioned above, each domain has its own responsible administrator. How to find information about a domain including responsibility? The *WHOIS* service is used for this.

The WHOIS database is distributed among the responsible organizations – the individual RIR providers (each maintains its own database for its own domain and subdomains), similarly, the information is distributed below (LIR, etc.).

Example

If we want to search for an SLD domain within the Czech TLD domain, we look in the WHOIS database maintained by the main Czech registrar CZ.NIC. If we want to search for a TLD domain from Europe or most of Asia, we look in the WHOIS database of the RIPE registrar.

However, it is often easier to use “general services” servers that are not regionally limited, such as <http://www.whois.com/whois/> or <http://www.whois-search.com/>.





Tasks

Find information about the following domains:

- `slu.cz`
- `google.com`
- `cloudflare.com`
- `ietf.org`
- `netacad.com`



7.2 WWW Service and HTTP Protocol



We know the HyperText Transfer Protocol (HTTP) primarily from the address bar of a web browser, but generally, this protocol is used to transfer structured information; transferring web pages in HTML or XML format is just one of its many uses.

7.2.1 Addresses



URI (Uniform Resource Identifier) is a sequence of characters identifying a particular resource. It can be a locator (specifies the location of the resource) or a name (no location). Thus, there are two kinds of URIs:

- *URL* (Uniform Resource Locator) is a locator type URI,
- *URN* (Uniform Resource Name) is a name type URI.

So the difference is whether or not we have a resource location encoded in the identifier. For example, a phone number is a name-type URI (URN) because it does not specify a location.

A URN usually requires some form of translation or routing that automatically determines the location of a given resource so that we can access that resource.



The *URL of the HTTP protocol* is a string starting with the protocol name (in this case `http://`) followed by the server's name address and, if necessary, the specific page in the server's directory structure.


Full specification of the HTTP URL is `protocol://server:port/path-to-file` where

- `protocol` determines which protocol is used for communication (`http`, `ftp`, etc.),
- `server` is the server address, usually a FQDN or IP address,
- `port` is the number of the TCP/UDP port to be used for communication (we specify if a port other than the usual one is to be used),
- `path-to-file` specifies the path to the requested resource within the given server (in the directory structure).

Some parts are optional. If they are not entered, the default option is selected. For example, if the web server address is entered and we do not add a port number, port 80 is selected.

7.2.2 HTTP Communication

HTTP PDUs are called messages and are encapsulated in TCP segments (i.e., connection communication with acknowledgement). In the vast majority of cases, port 80 is used on the server side (we can also encounter port 8080), on the client side we again have a separate port number from the dynamic port range for each process above the application layer. The entire communication is referred to as a *HTTP session* and includes all messages within that connection.

 HTTP prescribes request-response communication. Thus, an HTTP message can be either a request or a response. A query can be made using one of the following *methods*:

- HTTP GET – the most commonly used method. The URL includes a specification of the data.
- HTTP POST – used for sending completed web forms, the data sent is not put in the URL, but in the body of the message.
- Other methods – for example, PUT transfers data to the server, DELETE allows you to delete objects on the web server, HEAD works like GET but doesn't send data (just provides metadata in the header), CONNECT adds port number information to the address.

The first header information is the method used, the rest of the header is a sequence of pairs of the form “item: value”. The items passed in this way are, for example, the content type, the server designation, the specific address of the object on the server, the encoding used, and in the User-Agent item we find the most complete specification of the client-side program requesting the web page.


This is followed by the body of the HTTP message. If there is something to transmit, when sent from the server to the client we usually find the HTML code of the requested page there.

 HTTP communication is as follows:

- A TCP Three-Way Handshake takes place. The port number field on the server side is 80, even though we have no HTTP message encapsulated inside these TCP segments.
- The client then sends a request to the server in a TCP segment that already contains an encapsulated HTTP message with a request header.
- The server responds with a TCP segment with an encapsulated HTTP message with a response header.
- Requests and responses are repeated as needed.
- Followed by the end of the HTTP session and termination of the TCP connection (without encapsulated HTTP messages).

If the client detects from the previous server response that the page whose text it has already received contains the address of an embedded object to be loaded (image, frame, etc.), it sends a new HTTP request to the server, this time for the embedded object. This can also be repeated recursively; there may also be another nested object in the nested object.

Of course, in addition to the above mentioned steps, it is necessary to take “complementary” steps, such as using the DNS protocol to find out the IP address of the specified server (and then the Three-Way Handshake is performed). Before that, it may be necessary to use ARP or NDP to find out the gateway's MAC address.

 **Example**

We will show on a practical example (only HTTP messages, no “context” in the form of TCP or DNS) how the page request and its delivery looks like.

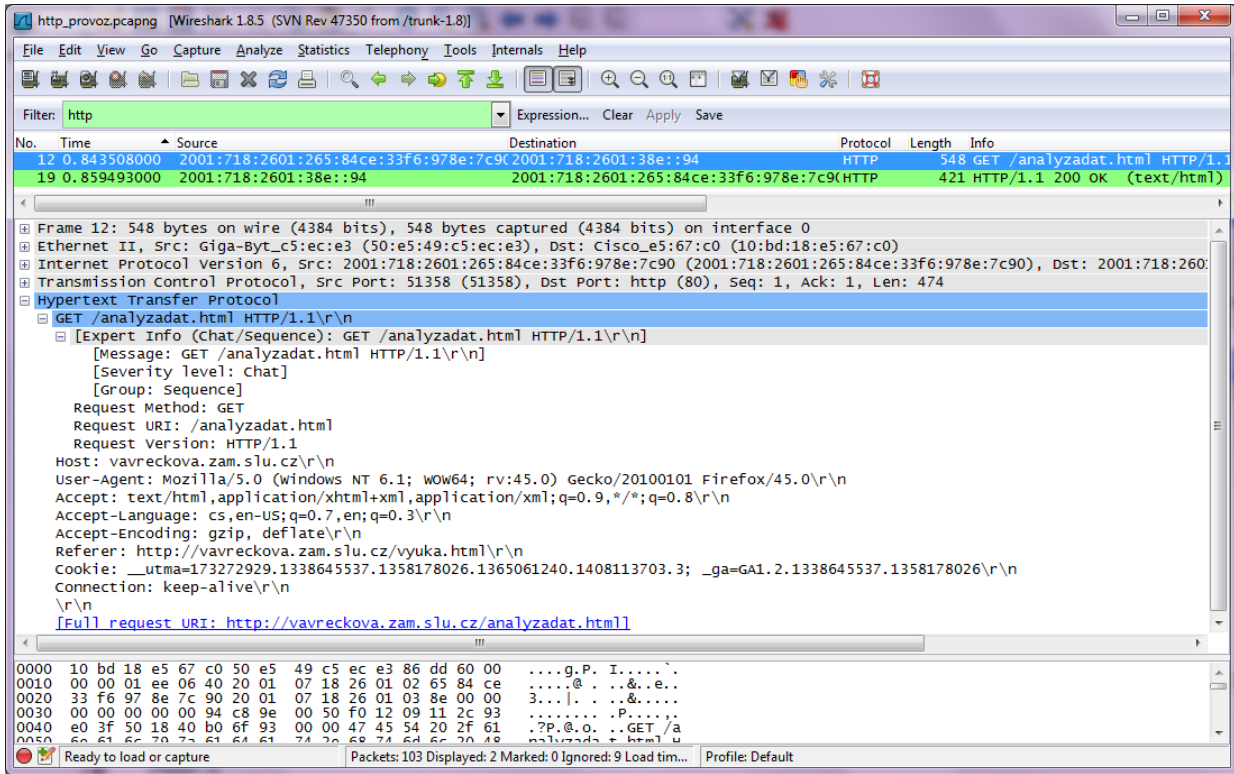



Figure 7.2: HTTP GET Message

Figure 7.2 shows an HTTP message with a GET request. Note that the last part of the URL follows the GET, which is actually a specific file (if this file were embedded in a directory, the whole path would be there). The line `Host: ...` follows the GET query information, which is the part of the URL corresponding to the domain name. The individual header entries are terminated by a pair of `\r\n` characters, which is a line break. At the end of the entire header (in the case of a query, this also corresponds to the end of the entire HTTP message), this termination is doubled, so that the target knows the end of the header.

Note also how verbose our web browser is – not only the target, but also anyone on the way will know what web browser we have and what operating system we’re running, including versions (here Firefox on 64-bit Windows version 6.1, which is Windows 7).

And what did we learn from the response? For example, that the server is running the Apache web server, and the operating system is CentOS (that’s one of the Linux distributions). The header again ends with a double line break, but this time we also have a data part – **Line-based text data**, where the MIME type is given first (more about MIME in the next section). As we can see, this is followed by the HTML code for the web page.

 The best known web (HTTP) servers are Apache, nginx and MS IIS, most websites use Apache or nginx. Typically, it is a service or daemon (a daemon is the equivalent of a service in UNIX



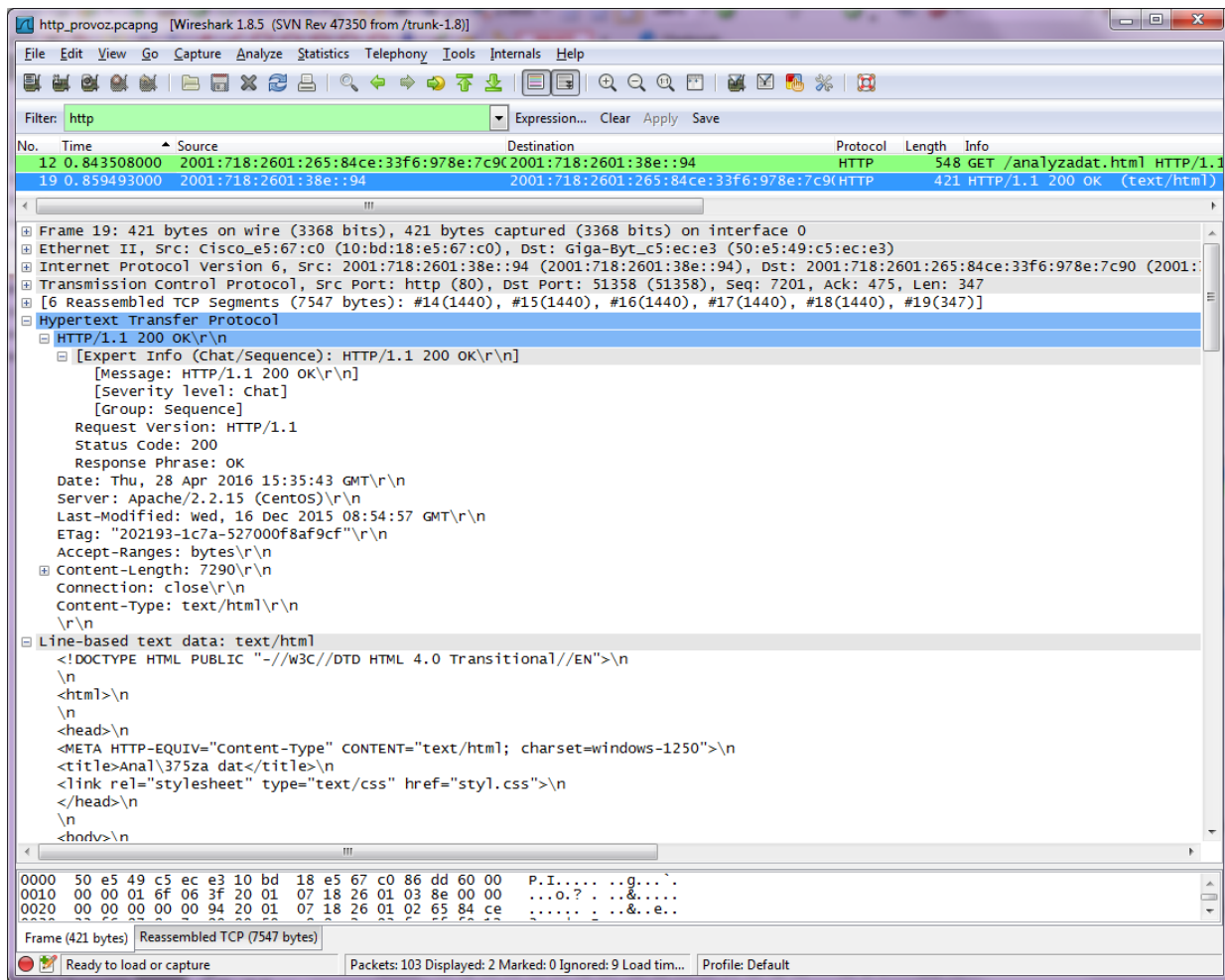


Figure 7.3: Response to HTTP GET

systems) – a process running in the background and constantly evaluating requests coming in from the network.



Remark

How does a web server (like Apache) actually know that it has just received an HTTP message? The daemon or service itself has to take care of that. We call it *listens* on a port (usually port number 80), and whenever a request arrives on that port, the listening process is notified.



Today, we often communicate with a web server securely. This means that between HTTP and TCP, one of the SSL or TLS protocols is used to encrypt the communication. The combination of HTTP and one of these security protocols is called *HTTPS*. Immediately after the TCP connection is established, the parameters of the secure connection are negotiated and from that point on the communication is encrypted.

HTTPS communication takes place on a port other than 80, on the server side port 443 is used. So the web server actually listens not only on port 80, but also on port 443 (and possibly other ports depending on the configuration).

7.2.3 HTTP Information Codes

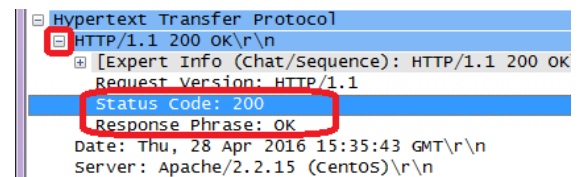
It is not always possible to get the website to the client as it should be. Many various errors can occur, on the client side, server side and along the way. The client-side user may (or may not) then be informed by the web browser that something is wrong. In addition to errors, of course, the client is informed in general about the status of the request.



The HTTP protocol informs the client with a three-digit code. Codes beginning with a specific digit have the following meaning:

- 1xx – informational, the server is processing the request, for example code 100 means that the server has successfully received the request and started working on its solving, these codes are usually not encountered at all,
- 2xx – notification of successful processing of a request or part of a request, for example, code 200 indicates successful completion of the operation,
- 3xx – for successful processing of the request it is necessary to perform some non-standard operation (for example, redirection to another address), this code is unfortunately sometimes abused by hackers,
- 4xx – client-side error,
- 5xx – server-side error.

The first and second code types are progress information, from the third type onwards they are error information.



The most common client-side error is probably a 404 (Not Found – the requested resource was not found on the specified server), which usually means that we misspelled the last part of the URL (the file name or some directory on the path to it), or the file was moved, renamed or deleted on the server.

The 401 error is sent to the client if it requests access to a resource that requires authentication (i.e. we must enter login information and then the resource will be made available).

From the server-side errors we can encounter for example error 500 (Internal Server Error), which occurs when a process running on the server side does not react as it should (for example, it freezes), or 503 (Service Unavailable), if the server is shut down (for example, for maintenance purposes, probably Windows is running on it).

This code is part of the HTTP message header. Notice in the image 7.3 on page 181 (also in the image on the previous page) the line `Status Code: 200`, this is exactly it, the server tells the client that everything is done and sends the requested page (generally an object) in this message.



Additional information

All status codes are explained in (older) RFC 7231, chapter 6, or (newer) RFC 9110, chapter 15.



7.3 E-mail Service

7.3.1 Infrastructure

We have our mailbox on the mail server. When we send an e-mail, we send it to this server (which ensures forwarding to the receiver's mailbox), and we look at incoming e-mails (of which we are the receiver) in our mailbox. So this server is the mail service provider for us. But the whole infrastructure is more complex than just mail servers with mailboxes, as we can see in Figure 7.4.

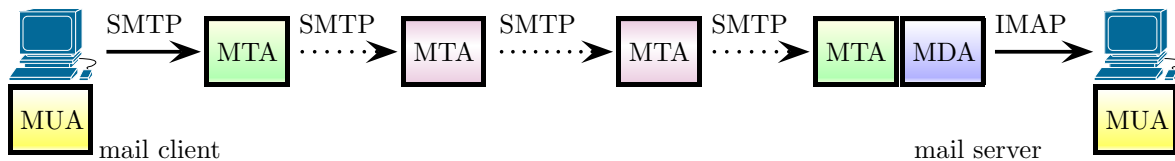



Figure 7.4: Mail servers infrastructure


 The following elements are part of the infrastructure:

- *MTA* (Mail Transfer Agent) – its task is to receive an e-mail message, check it and determine by address where to forward it. The role of MTA can be performed by mail servers such as MS Exchange, postfix, qmail, sendmail, etc.
- *MDA* (Mail Delivery Agent) – manages the mailboxes of users, receives from MTA messages destined to these mailboxes. MDA functionality can be included in MTA tools.
- *MUA* (Mail User Agent) – an application that communicates directly with the user (usually locally on the computer) on one side and with his/her mailbox (maintained by the MDA) on the other side. This is usually either a client program (Outlook, Thunderbird, Evolution, etc.).

So if we send an e-mail, our MUA application first establishes a connection with our mail server acting as an MTA. It sends an e-mail to it, which is then forwarded to the next MTA along the way, etc. (the message travels up the parent domains and networks and then down the receiver's network hierarchy in the opposite direction), looking for MX-type records in the DNS records of the domains it traverses. The last MTA already forwards the message to the destination MDA, which ensures placement in the receiver's mailbox.



Remark


If we can do that, we can also avoid MUA. The connection to the MTA can be established directly via Telnet or SSH, sending the information that would otherwise be sent by the MUA agent. 

7.3.2 Protocols

To work with electronic mail we need two types of protocols:

- protocol ensuring the sending of the message and its transfer to the receiver's mailbox, nowadays SMTP protocol is almost exclusively used,
- protocol for accessing messages in the mailbox, there is a choice between POP and IMAP protocols.


All of this (including the diagram on the previous page) is true if we are really using an e-mail client. If we communicate with the mailbox via the web interface, then the first and/or last part of the path is usually HTTP or better HTTPS.


 *Simple Mail Transfer Protocol* (SMTP) is used to transfer (forward) messages towards mail servers. It is also according to this protocol that we speak of SMTP servers (these are servers providing the MTA service).

SMTP communicates on port 25, SMTP messages are encapsulated in TCP or UDP segments (mostly TCP).

With HTTP, you will have noticed that the header is quite variable (depending on the direction and phase of the communication). SMTP is the same way. Between the MUA and the MTA, a TCP connection is first established (with port number 25), security is negotiated (so we don't transmit login credentials as plain text), and then "topic" SMTP messages are sent with sender and receiver addresses, subject and other parts. The message is then completed by our local MTA.

On the next path (when we find the MTA on both sides of the connection), communication is easier. The message is being forwarded "in whole", and on each MTA a new record is added at the beginning of the message – the information about the MTA. This implies that the size of the SMTP message grows incrementally along the way, and the recipient can then verify in the header which MTA (SMTP) servers the message went through.

 *Post Office Protocol* (POP) version 3, i.e. POP3, is already somewhat outdated. Its purpose is to access the mailbox in order to download messages to local storage. In version 3, it communicates on port 110, using the TCP protocol.

 *Internet Message Access Protocol* (IMAP) is a modern replacement for POP. It allows to access the mailbox, but not only to download messages – it offers an online management service with everything, for example:

- to read, copy and delete selected messages directly in the MDA mailbox,
- to sort messages, tag, more freely use the dedicated storage space in the mailbox,
- to view or download not only full messages, but also just message headers, saving network bandwidth,
- mailbox and individual messages can be accessed from different client devices and each can have its own copy of the messages in the mailbox,
- advanced automation options, such as automatic message backup.


IMAP communicates on TCP port 143. IMAP is almost exclusively used today, with POP3 rarely encountered.

7.3.3 Communication and Settings

In the case of e-mail, it is also a connection-oriented communication, i.e. the TCP protocol into which SMTP and IMAP messages are encapsulated. This always means client-server communication.

In SMTP, a client is the device that sends the SMTP message. The SMTP server (that is, the MTA) listens on port 25 and waits for incoming messages.

The IMAP client is the receiver of the message, and must query the IMAP server for changes (usually at appropriate intervals). The IMAP server listens on port 143 and waits for these queries.

 When we configure the MUA (mail client), we have to tell it who its counterpart is – we enter the name address of the SMTP server and the port number for communication with it, and the address of the IMAP server with the port number.

If we use only the web interface to the mailbox instead of a mail client, then we don't need to do anything like this, because the configuration is part of the web application and is related to the address we type in the address bar. It usually communicates with the SMTP/IMAP server via HTTPS.




Remark

Nowadays it is common to communicate with mail servers in a secure way. Note that this does not mean that messages are encrypted without breaks all the path, or digitally signed. Only the connection between two communicating devices is encrypted so that, for example, the login details or the message itself cannot be eavesdropped on, the SMTP server can “see” what we write.

Secure communication is a little different than simple unsecured communication. The security protocol TLS is inserted between SMTP or IMAP and TCP, which means, among other things, using a different port number. So if we are securing SMTP communication, it is port 465, for IMAP it is port 993 and for POP3 it is port 995.



 Mail protocols use addressing based on a similar principle to HTTP, i.e. URL. The protocol is `mailto:` and the rest of the locator contains the mailbox name and the MDA name address.



Example

The URL for e-mail looks like this:

`mailto:john.smith@company.org`

Compared to the HTTP URL, the meaning of the second and third part is switched – in the second part we have the name of the mailbox, and only the third part is the name address. The @ wildcard is usually read “at” (john smith at company.org).



Additional information

- <https://www.siteground.com/tutorials/email/pop3-imap-smtp-ports.htm>
- <https://www.port25.com/how-to-check-an-smtp-connection-with-a-manual-telnet-session-2/>




Remark

Relay servers (Open Mail Relay) are SMTP servers that do not verify that the sender's name and address match when communicating with the user or MDA sending the e-mail. They are often misused to send spam, so a network with such a server is considered untrusted.



7.3.4 MIME

 MIME (Multipurpose Internet Mail Extensions) is a standard for defining the formats of data sent. It does not use its own PDUs, but specifies how various types of data are to be represented in the PDUs of other application protocols. This standard is very often used in mail messages, where it is used to represent the same information in various formats (plain text, HTML, etc.) and also to represent embedded attachments.

Some of the most well-known MIME types are:

- *text* – for text formats (e.g. text/plain, text/html, text/rtf),
- *image* – for images (e.g. image/bmp, image/gif, image/png, image/jpeg),
- *audio* – for audio files (e.g. audio/mpeg, audio/mp4, audio/ogg),
- *video* – for video files (e.g. video/h263, video/h264, video/mp4),
- *application* – for application files, modules and application data files, simply binary files (e.g. application/java-vm, application/json, application/msword, application/octet-stream, application/pdf),
- *multipart* – “multiformat”, which may, for example, indicate that the same data is followed sequentially in several different formats (multipart/alternative), eventually consecutively several different objects, for example, that a file or digital signature is attached, etc.

```

...
Received: from ...
...
Content-Type: multipart/mixed; boundary="-----WLFzCqNdParrZ2rAJJm6AUD6"
MIME-Version: 1.0
...
This is a multi-part message in MIME format.
-----WLFzCqNdParrZ2rAJJm6AUD6
Content-Type: multipart/alternative;
boundary="-----Z2gIXt5PZfmUp1DRr62FcfR0"
-----Z2gIXt5PZfmUp1DRr62FcfR0
Content-Type: text/plain; charset=UTF-8; format=flowed
Content-Transfer-Encoding: quoted-printable
Ahoj Sarko,
posilam.
H.
...
-----Z2gIXt5PZfmUp1DRr62FcfR0
Content-Type: text/html; charset=UTF-8
Content-Transfer-Encoding: quoted-printable
<html>
...
  <body>
    Ahoj Sarko,<br>
    posilam.<br>
    H.<br>
  ..
-----Z2gIXt5PZfmUp1DRr62FcfR0--
-----WLFzCqNdParrZ2rAJJm6AUD6
Content-Type: application/pdf; name="prihl_CCNA3.pdf"
Content-Disposition: attachment; filename="prihl_CCNA3.pdf"
Content-Transfer-Encoding: base64
JVBERi0xLjcNJeLjz9MNCjE5NyAwIG9iag08PC9MaW51YXJpemVkeVtCAzMjIzOTMvTyAx
...
Ng0KJSVFT0YNCg==
-----WLFzCqNdParrZ2rAJJm6AUD6--

```



Figure 7.5: Combination of several MIME formats

When MIME is used, instead of plain text, the message first contains (one or more) MIME headers followed by MIME content. In the MIME header we find the MIME version, the content-type, the encoding method for the following data, and more as needed.

Figure 7.5 shows a fragment of a mail message (the content has been shortened considerably) where we can see a combination of several MIME types. The outer area (multipart/mixed, borders are marked by blue arrows) defines the email itself and its attachment, in the first nested area there is the area of the e-mail itself (multipart/alternative, red arrows), in which two nested areas are alternative to each other (text/plain and text/html).

The boundaries between the parts that make up the multipart are separated by randomly generated strings (the same string is used at the beginning, end and boundaries between the parts). The purpose is to make these boundaries easy to trace. The coloured arrows just point to these boundary strings.



Additional information

<http://www.iana.org/assignments/media-types/media-types.xhtml>



7.4 File Services

7.4.1 FTP

A file server provides a data storage service and unified access to that data. The client therefore stores data on the file server (upload), downloads data (download), or updates data.

 *FTP* (File Transfer Protocol) is used to communicate with file servers and has dedicated ports 20 and 21, whereby:

- port 21 is used to send control information (commands),
- port 20 is used for sending data.

So the FTP server listens primarily on ports 20 and 21, more precisely it listens on only one of them – by default on 21 (for commands), and if a data transfer is negotiated over port 21, it works with port 20 only for a change. As far as transport protocols are concerned, both TCP and UDP can be used, but only TCP communication is practically usable directly for FTP, since many file servers require authentication, for which we need to establish a connection.

There is a lightweight version of the FTP protocol, namely TFTP (Trivial FTP). TFTP communicates over UDP, so it does not establish a connection. It is used for a simple one-time transfer of a file to/from a TFTP server, such as a network device configuration file for a backup.




Example

The URL for a file server looks similar to HTTP: `ftp://fileserver.company.org`




Secure communication with the FTP server is similar to the previous protocols, i.e. between FTP and TCP we use the TLS protocol (referred to as FTPS), but a better alternative is to directly use the secure SSH protocol (port 22). SFTP refers to FTP transfer over SSH, which is a slightly more complicated communication.

The FTP protocol is typically used to download data from the web, because the HTTP protocol (which could also handle it) tends to make it unnecessarily difficult to work with large files. Another typical use is to manage one's own web pages – working with the files that make up those web pages on the server (there is usually an FTP server on the server running the HTTP server).

 While an FTP server must run on the server side (as a process, service, daemon), we need *FTP client* on the client device. FTP clients are usually part of web browsers and file managers (for example Total Commander or Free Commander), or we can choose a specialized program (FileZilla, WinSCP, etc.).

In order to use a given FTP client to access a specific server (or a part of it that is dedicated to us), we usually have to create a profile for this communication (this is not necessary if users do not have to authenticate). In the profile, we need to specify the server name (i.e. domain name), location (server URL), protocol number (e.g. 21) and authentication information (username and password), or we can specify this information dynamically each time we access the server.

 We communicate with the FTP server using FTP commands (or by clicking the mouse, depending on the client). Some of the most basic commands:

- `open server_name` – opens a session to the given server, the parameter is the name of the file server,
- `get file` – we want to download the specified file (from the server to itself),
- `send file` – we send the specified file from ourselves to the server,
- `!` – this command is used to switch between the data space on our computer and the server's data space, it applies to moving around the directory structure on our computer or the server,
- `lcd directory` – move to another directory either on our machine or on the server, switches the location of the action with `!`, the same directory entry methods as in the command line are used (including `..` to move up a level).



Additional information

There are of course many more FTP commands. A brief tutorial and overview can be found, for example, at

<http://www.computerhope.com/issues/ch001246.htm>.



Example


In fact, a simple FTP client is available on every operating system, even Windows. When we type `ftp` at the command line, we get to the command environment of this client. We can then open a session with a particular server, whereby we are prompted for a username and password, and after a successful login we can enter FTP commands. So the beginning of the session looks like this:

```
ftp
open fileserver.company.com
...
```

We are then asked for a username and password, after which we can enter FTP commands. We end the session with `quit` or `bye`.



7.4.2 Sharing Resources on a Local Network


 In addition to FTP, *protocol SMB* (Server Message Block), also called CIFS (Common Internet File System), can be used for sharing resources on local networks (often peer-to-peer). It is an application protocol that provides authorized access to resources such as files, printers, etc. It provides access to file and print servers.

SMB was originally developed by IBM in collaboration with Microsoft and is still popular today, especially in networks with Windows servers and clients, but there is also an open-source implementation for other operating systems called *Samba*.


The protocol defines client-server communication – the resource requestor is the client and the resource provider is the server. SMB messages are called *blocks* (it's also in the protocol name), their format is the same in both directions of communication, only the requested data can be added to the response.

7.5 IP Address Assignment and DHCP

Every device on the network needs to get an IP address somehow. We can either enter it manually on each device or let the DHCP server assign it dynamically to connected devices.

 An IPv4 address can be obtained in the following ways:


- *dynamic allocation*: the user doesn't have to worry about anything, when a device is connected to the network it is automatically assigned an IP address, it can be different each time,
- *static*: the user has a predefined IP address, he gets into the configuration like this:
 - the user *manually* enters it at the appropriate place in the network interface configuration,
 - *static allocation*: this address is automatically assigned to the device when it connects to the network (as with dynamic allocation, but the address is reserved for the device).

 We will be interested in the first case and from the second case the possibility of static allocation, which all takes place according to *DHCP* (Dynamic Host Configuration Protocol). Thus, DHCP provides a mechanism for distributing the configuration of a network interface. The client device can receive various information from the DHCP server:

- IP address, network mask,
- DNS server address,
- default gateway address,
- other information.

So it's definitely not just about the IP address and the netmask. The client also learns which way the network path goes and who on the network can translate name addresses to IP addresses. The server can also distribute other information as needed.

With the transition from IPv4 to IPv6, new versions have been created for other protocols as well (we have seen this with ICMPv4/v6 so far), and the same has happened with DHCP. The changes in IP were simply so extensive that this protocol had to be changed considerably as well.

 The older DHCPv4 is simpler and uses Bootstrap (Bootp) as its carrying protocol down to the transport layer. The overall structure is as follows: The DHCP message is encapsulated in

a Bootstrap message, and the resulting message is encapsulated in a UDP segment. The port number is 67 on the server side and 68 on the client side.

DHCPv6 is already more complex and does not use the Bootstrap protocol, it creates the messages itself at the application layer. DHCPv6 messages are also encapsulated in a UDP segment, using port numbers 547 on the server side and 546 on the client side.

Remark

Note that unlike previous application protocols, DHCP client uses a port from the “well known” port range and not dynamic ports. Note that while DHCP operates at the application layer, it certainly does not communicate with applications and processes in different windows or tabs – there cannot be a situation where two different applications want to allocate an address. Therefore, the client does not need to use dynamic ports to distinguish between applications – there is nothing to distinguish.

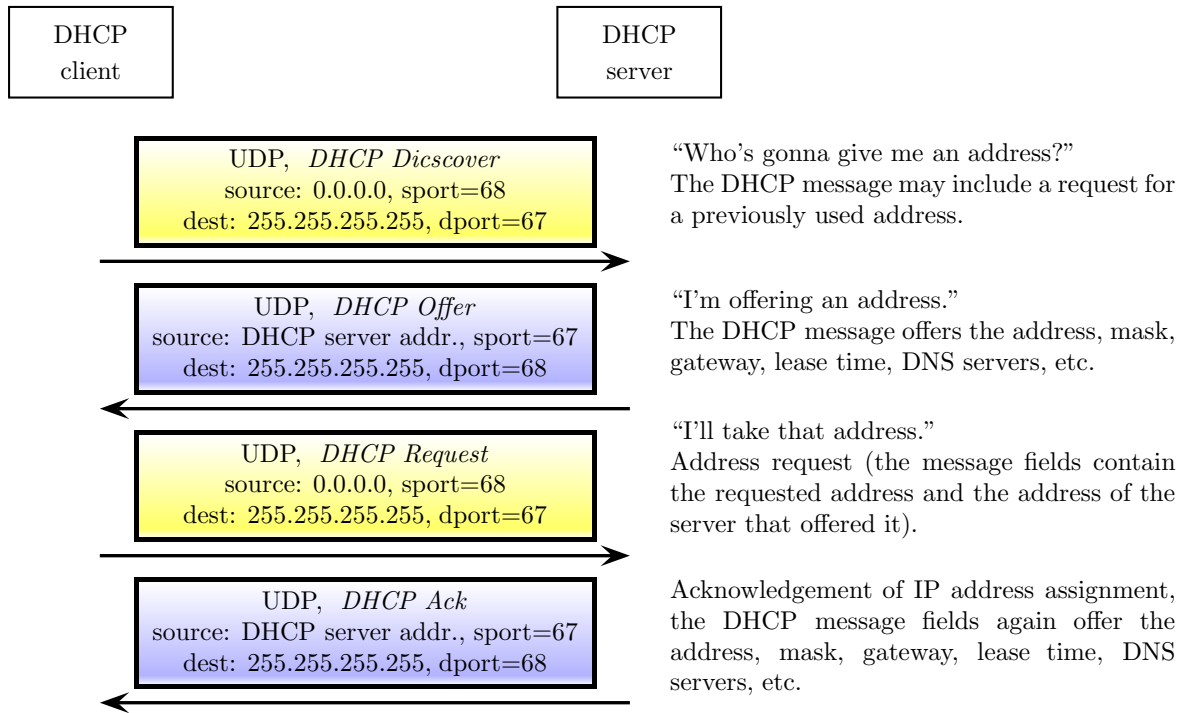



Figure 7.6: Obtaining an address via DHCPv4

Figure 7.6 shows the procedure for obtaining an IPv4 address. The communication is only broadcast, because the client has not yet been assigned an IP address (in both directions – the client does gradually find out the server’s address, but still sends broadcasts). Steps:

1. *DHCP Discover* – the client discovers which device on the network can assign it an address (it does not know the address of the DHCP server). In the request, it sends its MAC address and a list of parameters it requests (Parameter Request List – netmask, DNS server addresses, router/gateway address, etc.).
2. *DHCP Offer* – The DHCP server has received the request and responds by sending the offered address (or range of addresses), the address validity period (Lease Time), its own

address, DNS server addresses, etc. If there are multiple DHCP servers on the network, the client may receive more than one offer. The offered address is temporarily reserved for a short period of time (e.g. 1 minute).


3. *DHCP Request* – the client accepts the offered address and sends again the list of other requested parameters. If more than one offer is received, it will respond to only one.
4. *DHCP Ack* – the server confirms the IP address allocation to the client and (again) sends the values of the other requested parameters. The address is finally assigned to the client and the DHCP server has it recorded in the database.

 Regarding *reassignment of the same address* (the client has an address assigned, but it is about to expire – *Lease Time*), only the last two steps take place, because the client knows its address provider (it doesn't have to ask) and the offer has already been done. So just a request and a confirmation. The client resolves this before expiration, while it still has an IP address assigned, so the communication can be unicast.


The client cannot use the real IP address before DHCP Ack when it is first being assigned, until then it has 0.0.0.0 (undefined), in case of Windows client during DHCP communication the APIPA address from the range 169.254.0.0/16 is used.

When using DHCPv6, there are multiple options for how communication can take place, we will discuss these options in the next chapter. One of the differences from DHCPv4 is that a custom message format is already defined for this protocol; Bootstrap messages are no longer used.


7.6 Other Server Types

 *Database server* is a server running some database system (MySQL, PostgreSQL, Oracle, MS SQL, etc.).

It is typical for database servers that the user usually does not communicate directly with them, but there is an intermediary – a web server providing a user interface to the database server. This is both for user-friendliness and for security reasons; user input needs to be carefully checked. So the user is presented with a web page where they enter what they actually want from the database, send it to the web server (usually as a form using the HTTP POST method), it checks everything and sends a query to the database server. The response again goes through the web server, which “formats” it into a web page to make the output more user-friendly and secure.

 *Print Server* is a server that provides print services (maintains print queues of connected printers, receives requests from clients on the network and queues them).

If we have a printer connected to a regular computer that we share on the network, then our computer also acts as a print server. Nowadays, the print server is included in many network printers, so that there is no need to run other “mediating” devices alongside them.

 *Application server* is a server providing services of a certain (network) application to the network. It is actually a process (service, daemon) running on a given hardware server, and this process receives requests from the network (more precisely: listens on the appropriate ports and responds to them).


In most cases, there is a web server between the client and the application server that will mediate the communication. This is not only for the convenience of the users, but also for better protection of the application server.

It may be an information system server, an authentication server or other. We can also consider a database server as a special type of application server because it runs the database system as a special process.

7.7 Remote Access and Remote Communication


If we need to configure a device that for some reason we can't sit at directly (it's far away, it's less accessible, it doesn't have a keyboard and screen, etc.), we need a protocol that can provide remote access to the device to the same extent as if we were sitting at the device. When managing a server, we usually make do with a text-based interface (yes, even a Windows server can be installed without a GUI for security and performance reasons – a “Server Core” installation), whereas when managing user devices remotely, the transmission of the user's GUI can be useful.

7.7.1 Telnet

 *Telnet* protocol is used for interactive remote (text) access of the client-server type to devices over the network, with the possibility of authentication (we enter the name and password). In real life, it performs *terminal emulation* – a terminal is a specialized device consisting of a screen and a keyboard and connected e.g. to a server over a network. If we emulate a terminal, it means that our computer can behave like a terminal, even though it is not a terminal.

Basically, Telnet can be run on any device (client or server), but it is usually blocked or otherwise prevented from being used. What is the problem? Telnet was created when the network was a secure place (and much smaller) and there was no need to encrypt anything. When authenticating, it transmits the username and password in plain text, and anyone who accesses the network can eavesdrop on this information.

Nowadays, Telnet is only used in environments (internal networks) that the administrator considers secure (note that I have not written “in secure networks” here – there is no such thing as a secure network). A more logical use is in debugging server software, whereby the client and server are separated from any network and no one can technically get into the communication.

 Telnet uses connected communication (TCP protocol) via port 23 (the client uses dynamic ports). TCP communication is first established on server port 23. If authentication is required, authentication data is sent and confirmed, and then it depends on what we want to do.



Remark


If we can access Telnet on our computer, we can start a Telnet session as follows:


```
telnet
open server.address portNum
...
```

We enter the port number only if it is different from 23. For example, if we want to access a web server, we use port 80. We are then asked for a username and password (if the server in question is protected by authentication) and then we can enter commands.




In UNIX servers it is quite common to access them remotely by terminal emulation (either via Telnet or some more secure protocol), this way we can enter almost any command as if we were sitting right at the device in question. We simply get a prompt and enter the command. In Windows, this remote access is somewhat more limited because, unlike UNIX systems, there are many tasks that simply cannot be done in text mode, and Windows is somewhat less good at remote access.

 Often (especially during testing and debugging) we use the fact that many application protocols are text-oriented (not binary) and we can send their messages via telnet. This can be done for example with HTTP, SMTP and others.

 **Example**


Pokud chceme přes Telnet otestovat webový server, postupujeme takto:

```
telnet www.server.org 80
GET /index.html HTTP/1.1
```


First we established a connection to the specified server, on port 80 (because we want to communicate with the process listening on this port). Web servers usually do not require authentication, so no credentials will be required. The next command is to send an HTTP request of type GET (it must be in upper case, otherwise an error will be reported), we ask to send the file `index.html` from the root directory of the server, also adding the HTTP version information. In the response we get the HTTP header and also the body of the message (i.e. the HTML code). 

For example, we can communicate with the SMTP server on port 25 in a similar way, we can even send an e-mail this way, it's just a bit more time consuming than requesting a web page.

7.7.2 SSH


 *SSH* (Secure Shell) is a more secure variant of the Telnet protocol, but in addition to the security of the communication itself, it has additional functionality compared to Telnet. It is used to securely access a remote device, in a variety of ways. In addition to enabling secure configuration, it also performs a similar (but secure) role as FTP (file transfer), allowing monitoring of processes and resources on the remote system, creating encrypted VPN tunnels (long-term secure connections), etc. Currently SSH version 2 or 3 is used.

This is a client-server communication over TCP (connection is established), on the server side there is port 22. Since this is mainly about security, it is better to configure a different port than 22 on the server. The procedure of this configuration depends on the specific product, usually it is a change in a configuration file. However, if we change the port number for SSH on the server, the client must use that number (and thus be informed of it) when initiating establishing the connection.

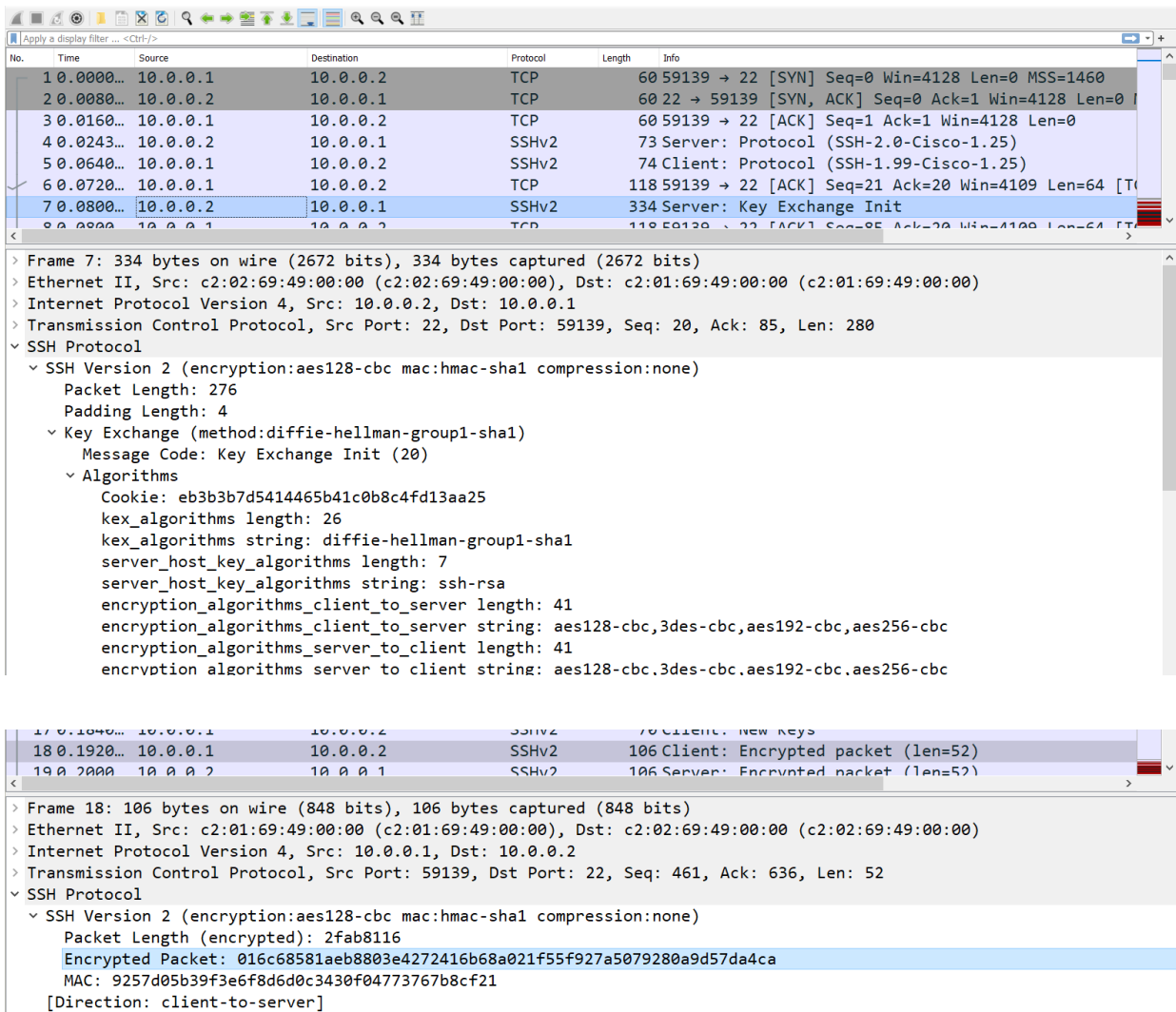
 The most well-known implementation of the SSH protocol is the open-source project *OpenSSH*. There is both a client and server version of this product (for both UNIX and Windows systems), and it is usually already installed on UNIX systems including Linux and MacOS.

There is also a program *PuTTY*, unfortunately only in the client version. It is available for

Windows and then for some Linux distributions (based on Debian, including Ubuntu, Mint and others).

 The SSH conversation looks like this:

- First the TCP connection is established, the client starts.
- SSH client and server “introduce” themselves (tell each other their version), the server starts.
- The server tells the client which encryption algorithms it can handle (see Figure 7.7, upper part), the client chooses between them in response.
- The following is the security procedure – the cipher string is exchanged, etc., according to the chosen algorithm. The purpose is to ensure that in subsequent communications, someone who could eavesdrop on the traffic does not interfere with the connection.
- After the security procedure is completed, the next traffic is already encrypted (see Figure 7.7, lower part), including authentication if required.



The image shows a Wireshark network traffic capture of an SSH session. The top pane displays a list of captured packets. The middle pane shows the details of the selected packet (Frame 7: 334 bytes on wire, 334 bytes captured). The bottom pane shows the details of another selected packet (Frame 18: 106 bytes on wire, 106 bytes captured).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000...	10.0.0.1	10.0.0.2	TCP	60	59139 → 22 [SYN] Seq=0 Win=4128 Len=0 MSS=1460
2	0.0080...	10.0.0.2	10.0.0.1	TCP	60	22 → 59139 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0
3	0.0160...	10.0.0.1	10.0.0.2	TCP	60	59139 → 22 [ACK] Seq=1 Ack=1 Win=4128 Len=0
4	0.0243...	10.0.0.2	10.0.0.1	SSHv2	73	Server: Protocol (SSH-2.0-Cisco-1.25)
5	0.0640...	10.0.0.1	10.0.0.2	SSHv2	74	Client: Protocol (SSH-1.99-Cisco-1.25)
6	0.0720...	10.0.0.1	10.0.0.2	TCP	118	59139 → 22 [ACK] Seq=21 Ack=20 Win=4109 Len=64 [T...
7	0.0800...	10.0.0.2	10.0.0.1	SSHv2	334	Server: Key Exchange Init
8	0.0800...	10.0.0.1	10.0.0.2	TCP	118	59139 → 22 [ACK] Seq=85 Ack=20 Win=4109 Len=64 [T...

Frame 7: 334 bytes on wire (2672 bits), 334 bytes captured (2672 bits)
 Ethernet II, Src: c2:02:69:49:00:00 (c2:02:69:49:00:00), Dst: c2:01:69:49:00:00 (c2:01:69:49:00:00)
 Internet Protocol Version 4, Src: 10.0.0.2, Dst: 10.0.0.1
 Transmission Control Protocol, Src Port: 22, Dst Port: 59139, Seq: 20, Ack: 85, Len: 280
 SSH Protocol
 SSH Version 2 (encryption:aes128-cbc mac:hmac-sha1 compression:none)
 Packet Length: 276
 Padding Length: 4
 Key Exchange (method:diffie-hellman-group1-sha1)
 Message Code: Key Exchange Init (20)
 Algorithms
 Cookie: eb3b3b7d5414465b41c0b8c4fd13aa25
 kex_algorithms length: 26
 kex_algorithms string: diffie-hellman-group1-sha1
 server_host_key_algorithms length: 7
 server_host_key_algorithms string: ssh-rsa
 encryption_algorithms_client_to_server length: 41
 encryption_algorithms_client_to_server string: aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc
 encryption_algorithms_server_to_client length: 41
 encryption_algorithms_server_to_client string: aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc

No.	Time	Source	Destination	Protocol	Length	Info
17	0.1840...	10.0.0.1	10.0.0.2	SSHv2	70	Client: new keys
18	0.1920...	10.0.0.1	10.0.0.2	SSHv2	106	Client: Encrypted packet (len=52)
19	0.2000...	10.0.0.2	10.0.0.1	SSHv2	106	Server: Encrypted packet (len=52)

Frame 18: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)
 Ethernet II, Src: c2:01:69:49:00:00 (c2:01:69:49:00:00), Dst: c2:02:69:49:00:00 (c2:02:69:49:00:00)
 Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.2
 Transmission Control Protocol, Src Port: 59139, Dst Port: 22, Seq: 461, Ack: 636, Len: 52
 SSH Protocol
 SSH Version 2 (encryption:aes128-cbc mac:hmac-sha1 compression:none)
 Packet Length (encrypted): 2fab8116
 Encrypted Packet: 016c68581aeb8803e4272416b68a021f55f927a5079280a9d57da4ca
 MAC: 9257d05b39f3e6f8d6d0c3430f04773767b8cf21
 [Direction: client-to-server]

Figure 7.7: SSH communication captured in Wireshark, the first Key Exchange message and one of the encrypted packets



Additional information

- <https://www.openssh.com/>
- <https://github.com/PowerShell/Win32-OpenSSH>
- <http://www.debianhelp.co.uk/ssh.htm>



7.8 Network Management

We won't go into network management in detail here, just talk about the protocol that is used for this purpose.



Simple Network Management Protocol (SNMP) is used to collect data from a local network for the purpose of managing that network. It is also a client-server communication, and the client part is now implemented on almost every device in the network (computers, active network devices, sensors, printers, etc.).

The client, called *agent* in SNMP terminology, collects data related to the device it is running on, while the server part (*manager*) can request this data from various devices on the network and analyze it at any time. The communication can be of one of the following types:

- *query-response*, where the manager sends a query and the agent responds,
- *trap* – the activity is on the agent side (informing the administrator of an event on the device that needs to be responded to).

There are two versions currently in use – SNMPv2 and SNMPv3. Version 2 is no longer considered very secure.



SNMP usually communicates over UDP, but can also use TCP. Different versions use different ports.

- In request-response communication (the manager asks and the agent answers) the agent listens on port 161 according to SNMPv2, according to SNMPv3 it is port 10161. The manager uses a dynamic port.
- For trap communication, the manager listens on port 162 according to SNMPv2, on port 10162 according to SNMPv3. The agent uses a dynamic port.

7.9 Protocols and Ports Overview


In this chapter we have learned that application protocols use TCP (for connection-oriented acknowledged communication) or UDP (for connectionless unacknowledged, i.e. datagram service), which means that their messages are encapsulated in TCP or UDP segments and in the case of TCP the connection is also established.


In TCP and UDP segments, the header contains the port number, which on the server side specifies the service that is being communicated with (it can also mean the type of encapsulated message, but there is not always something inside the segment), on the client side it is usually a particular application or its part.


HTTP(S)	SMTP	IMAP	FTP	Telnet	SSH	DNS	DHCP	SNMP
80 443	25 465	143 993	20, 21	23	22	53	67, 68 547, 546	161, 162 10 161, 10 162
TCP						UDP		

Table 7.1: TCP and UDP ports with protocols

Introduction to Network Security

 *Quick preview:* This chapter contains a mixture of security oriented topics. First, we look at a few basic types of attacks on computer networks. This is followed by an explanation of VPNs, an overview of common network analyzers, and finally we find a section on firewalls.


 *Keywords:* Mapping, network sniffing, SQL injection, spoofing, DoS, DDoS, Man-in-the-Middle, social engineering, VPN, remote access, site-to-site, tunnel, encryption, IPSec, GRE, L2TP, OpenVPN, SSH, IP-in-IP, network analyzer, firewall, demilitarized zone (DMZ), packet filter, network ACL, SPI, IDS, IPS, DPI, proxy


 *Objectives:* After studying this chapter, you will have a basic understanding of computer network security. You will be familiar with the most common types of network attacks, know what a VPN is, characterize some types of VPNs, learn about the most common network analyzers, and be able to describe the principles of a firewall and its types.

8.1 Network Security

8.1.1 Types of Attacks

In a computer network, there are various types of attacks to be considered, some of which are interrelated. In particular, we are interested in the following attacks:

 **Mapping, Reconnaissance, recon** – this is actually a kind of preparation for the following attacks. The hacker gets as much information as possible (open ports, IP address ranges used, operating systems and their versions, etc.) so that the actual attack on a particular device can be as successful as possible.


 **Network sniffing** – a packet sniffer diverts traffic on the network, captures packets and extracts information from them (then sends them on to the destination, undelivered packets would mean its discovery). Network sniffing is also classified as a mapping attack.

The purpose of sniffing is primarily to obtain passwords transmitted in text form, as well as other sensitive information. It is also possible to intercept and modify the contents of packets


or even read and modify information on nodes in the network (including configuration files or passwords).

Effective defence is primarily based on reliable encryption (including network authentication) and physical network security, as this type of attack requires physical access to the network. This can be a problem with wireless types of connections.

However, a packet sniffer (such as Wireshark) can be used quite legally by an administrator to monitor network traffic.

 **SQL Injection** – the hacker manages to inject the SQL statement somewhere where it has no right to be, or by appropriate choice of strings “produce” and send the SQL statement to the server. Most of the time this is an abuse of web forms or a string passed over HTTP GET, where the attacker fills in a special “harmful” string containing symbols with special meaning instead of a regular string. A typical exploited symbol is for example the apostrophe.


This type of attack can be defended against if the programmer of the web application (i.e. the web interface to the database) ensures a careful analysis of user inputs, and above all, it is necessary to have correctly set permissions on the database (e.g. requests for destructive operations such as DROP, DELETE, but also changes to data or requests for protected information should not be performed by a normal user from an unprotected part of the network, i.e. the Internet).


 **IP Spoofing** – spoofing of an IP address. A communicating node pretends to be the owner of an IP address that actually belongs to another node (or to no node). This is usually the case when an attacker tries to pretend to be a proper member of a private network using a specific range of IP addresses, or attempts to impersonate a specific node on the network.

A common use case is the misuse of certain protocols, including SMTP for sending e-mails. It is also used as a means for more sophisticated attacks, such as DoS (the source address is spoofed in packets sent to the compromised device).

 **Other types of spoofing** – Identity spoofing can be performed in other ways, often by hacking some address tables, for example:

- ARP Spoofing means spoofing entries in ARP tables on devices (the IP to MAC address translation mechanism is compromised), for example an attacker can declare himself as the default gateway,
- DNS Spoofing is spoofing the records in the zone file on the DNS server (the mechanism for resolving the name address to an IP address is compromised).


 **DoS (Denial of Service)** – this is enforcing denial of service on legitimate users. This attack is carried out either by exploiting a bug in the code (some protocol or operating system), by induced network congestion, by overloading some other component of the target device over the network (such as memory), or by spoofed messages – network status packets. The server is overwhelmed with connection requests (TCP handshake) or data requests that it is unable to handle, and therefore subsequent traffic is held up.


 **DDoS (Distributed DoS)** – a very dangerous variant of the previous type of attack. The unwilling “participants” of DDoS attacks are bot networks. A *bot* is an infected computer controlled remotely by a hacker (without permission and usually without the knowledge of the rightful owner). Bot networks, even very large ones (or their services), are rented on the underground market for


DDoS attacks, among other things.

A DDoS attack can also be carried out using otherwise quite useful mechanisms. For example, the *Ping Flood* type of attack takes place by flooding the attacked machine with ICMP Echo Request messages (these are the ones sent by the `ping` command). This overwhelms both its input capacity (receiving) and its output capacity (sending) – when it tries to respond. To hide its identity, the attacker spoofs the source IP address in the IP packets encapsulating these messages. The DDoS variant of the attack means that ICMP messages are sent by devices involved in the botnet.


Smurf Attack has an even higher distribution rate. It consists of a large number of devices (typically servers and routers) receiving ICMP Echo Request messages whose source address is spoofed – set to the victim’s address. These devices reply with ICMP Echo Reply messages to the victim’s address, thus congesting its input capacity.

 **Hijacking** – These attacks are mainly related to dial-up connections, but also generally any paid connection requiring authentication (e.g. private Wi-Fi networks or websites requiring login). An effective defense is to use an appropriate encryption algorithm during authentication.

 **Man-in-the-Middle** – A hacker gets between two communicating devices and eavesdrops or even alters the communication between them. This type of attack is related to some of the previous ones – connection hijacking, password mining, etc.

 **Password Discovery Attacks** – passwords can be discovered by brute-force attacks, Trojan horses, and some of the methods described above. Another possibility is social engineering, which is currently on the rise (the user defacto discloses this information alone and of his/her own free will, being tricked out of it).

A brute-force attack can mean trying all possible combinations of characters in a string of a suitable length, but can be conducted as *dictionary*, i.e. all strings from a created dictionary (including names, date of birth, etc.) are automatically tried. This can be defended against by setting a maximum number of failed logins, access is completely blocked once this limit is exceeded. It is also possible to require users to use a “strong” password (a sufficiently long one, containing letters, numbers and other characters – colon, percentage, call sign, etc.), but this is often problematic (users prefer to choose a password they can remember).

 **Human Factor** – “enemy” can be inside the network, even one that doesn’t know it. Especially recently, social engineering methods are often chosen for attacks. Social engineering is a method of extracting information from a user, even without using technology, by misleading (tricking) the user. The user is believed to be giving information to a trusted person for absolutely necessary reasons.

The attacker impersonates, for example, an employee of the security department, a repairman, an employee of the phone company, a new colleague, etc., and discreetly extracts everything he needs from his victims (especially passwords), or “tests” a new cool computer or otherwise gets access to the technology in the workplace. This happens especially in larger companies, where it is not assumed that all employees know each other, but contact can also be “impersonal” via phone or email. Employees are able to say incredible things about their company over the phone, including things that are trade secrets.

Alternatively, many users misunderstandably trust e-mail: for example, just sending an e-mail informing them that their account has apparently been compromised, with the login credentials having to be sent back for the administrator to check that everything is OK (plus the e-mail has to be graphically reproduced), and many users blindly accept.

Social engineering can also be of a “material” nature – for example, free “dropped” portable devices that many people find completely attractive. According to DHS¹, around 60 % of regular users are able to connect a randomly found USB flash drive to their computer, even though they have no idea if it contains malware. Other portable devices can be similarly misused, and the best protection here is to be honest, i.e., turn in the found object to the “lost and found”.

How to defend yourself?

- Not to believe everything anyone says. Especially when I am, for example, an administrator of a corporate network, I have to verify everything (the identity of the applicant for a new password after forgetting it, the necessity to perform the required intervention in the system, etc.).
- We do not keep passwords or other similar data on a piece of paper stuck to the monitor (or in other “usual” places). We choose strong passwords. It’s a good idea not to use the same password on multiple systems: if an attacker cracks the password on one system, they will usually try to use the same password on another system where they find the victim’s account.
- Any password-protected system (including operating systems) can be set to lock out an account after a specified number of failed login attempts.
- Banks and other institutions do not ask their employees to send passwords, certificates, TANs, credit card numbers and similar data by email or any other non-secure means (only through secure sites directly when logging in). And they certainly don’t ask for it by e-mail or phone.
- The network administrator should always have a “paper” support for any request to access employee accounts or systems.
- Attackers using social engineering do “homework” – gathering as much information (including personal) as they can use. A company should consider what it publishes (and the same goes for home users, also on discussion forums and social networks, for example). A shredder is not a useless device.


8.1.2 Security at Various ISO/OSI Layers


Data can be encrypted at any layer. In the ISO/OSI model, all higher layer data is encrypted as data flows downwards, i.e. including the header:

1. Encryption is performed on the data-link layer (L2) according to the transmission technique (e.g. according to IEEE 802.11).
2. At the network layer, the security of the connection between two nodes is implemented, including the creation of a tunnel in a VPN connection using IPSec or the same functionality built into IPv6.

¹DHS (Department of Homeland Security, <http://www.dhs.gov>)

3. SSL/TLS works on the application layer of the TCP/IP model (the inclusion in the ISO/OSI reference model is debated, because its functionality apparently affects both the presentation and the session layer).
4. Security solutions based on PGP (GPG, OpenPGP, etc.) are used at the application layer.

 The *IPSec* protocol is an additional mechanism for securing IPv4 communication at the L3 layer, and is already directly included in IPv6. It is used to secure the connection between two networks (the *site-to-site* type), typically when providing a connection between two branches of a company over a WAN.

 *SSL* (Secure Sockets Layer) and its successor *TLS* (Transport Layer Security) are typically used for mobile or home worker connections, type *remote access*. Beware, SSL up to and including version 2.0 is not considered secure, so definitely use at least version 3.0!

TLS, unlike SSL, is standardized (TLS version 1.2 is described in RFC 5246), and while the two protocols are functionally very similar, they are mutually incompatible (both parties must use either SSL or TLS, they cannot be combined).


While IPSec is a network solution (transparent to different applications), SSL/TLS is an application solution (i.e., it must be supported by the specific application whose communication is to be encrypted). SSL/TLS support has long been built into web browsers, so there are no problems with the functionality of this solution, at least when communicating over HTTP or similar protocols. If an application does not support SSL/TLS, this can be addressed by e.g. *STunnel*.²

While SSL/TLS supports mutual authentication, we often meet solutions using one-way authentication (the external employee is authenticated). Data integrity and encryption are also ensured by this protocol, only the transmitted data is encrypted. A combination of public and private keys is used and certificates can be used as well.

SSL/TLS is considered a less secure but more flexible solution (IPSec has problems with the NAT mechanism, whereas SSL/TLS can handle it quite well).


8.2 VPN

8.2.1 Principle of VPN

 A VPN (Virtual Private Network) is a secure connection through an untrusted environment (usually the Internet). It is the creation of a tunnel, a communication channel between two points (these can be specific end devices or routers, for example, then we connect networks) by wrapping the original packet or frame in a PDU with a header intended for network devices outside our networks. If we encrypt the internal PDU, we talk about a VPN.

The existence of a tunnel does not automatically imply encryption, it depends on the specific protocol. Thus, for some “tunneling” protocols (such as GRE), we need to use an encryption protocol (such as TLS) in addition to the tunneling protocol to create a VPN.


²<http://www.stunnel.org/>

 VPN is used in the following cases:

- *Remote Access*: a mobile employee needs secure access to the company's (local) network to access the company's information system and other secure resources while travelling, or an employee working at home (Home Office).
- *Site-to-Site* (or also *Network-based*): the need to interconnect remote local networks (e.g. branches of the same company).
- It is necessary to communicate with a business partner through a secure communication channel, but at the same time we don't want to let him directly into our local network. This can be *site-to-site* or *remote access*, depending on the actual VPN configuration, this option is actually a special case of both the previous ones.


In all these cases, we need to build a secure encrypted *tunnel* for communication through an untrusted environment – first of all, we provide suitable encapsulation with possible address translation (because packets will go through a foreign network with different address ranges and possibly different protocols), and we provide encryption. In the third case, we also need to use a firewall that will only let through communications allowed for that purpose.

A VPN tunnel to a corporate network terminates at a VPN concentrator (VPN gateway), which is usually either directly on a router with a firewall at the boundary of the local network that is visible on the Internet, or in a demilitarized zone, depending on where in the local network one needs to access through the tunnel.

 The VPN solution should provide the following:


- authentication – it is necessary to verify the identity of both communicating points (e.g. a user with a laptop on a business travel and a VPN-enabled firewall in the company LAN), or authentication of transmitted PDUs is ensured,
- authorization – determination of specific access permissions,
- data confidentiality – the transmission is always encrypted,
- data integrity – detection of packet modification or corruption along the way.

There are a number of protocols that can create a tunnel. They differ in various parameters, such as the type of tunnel (site-to-site or remote access), encryption options, and more. We'll look at a few of the most common solutions:

 **IPSec** (Internet Protocol Security) provides both tunnel creation and encryption. It is combined with IPv4, and is already included in IPv6 (in optional headers).


It works at the network layer, making it transparent to application protocols. It mostly encapsulates IP packets. It is used for both site-to-site and remote access solutions.

The disadvantages of IPsec are problems with NAT, but also that it cannot carry multicast traffic.

 **Additional information**

For details on Linux in particular, see the links at the end of the VPN section, especially <http://www.ipsec-howto.org/ipsec-howto.pdf>



 **GRE** (Generic Routing Encapsulation) works at the network layer and is typically used for site-to-site tunnels.

The usual procedure is that the original packet is given a GRE header (very simple, a couple of fields) and then a carrier protocol header is added, usually IP (there is a number 47 in the Protocol/NextHeader field that specifies the GRE protocol). Note that there is no mention of encryption in the procedure, because it can't do that.

Its advantage is its universality, it can encapsulate other types of ISO/OSI network layer PDUs than just IP. It can even encapsulate other layer protocols, including frames from the L2 layer. It can also transport multicast and broadcast over the tunnel. Another advantage is that this protocol is supported virtually everywhere (on Linux, on Windows, on network devices from different vendors).


On the other hand, the big drawback of GRE is that it does not perform encryption. In practice, GRE is combined with IPSec, TLS or other solutions that can encrypt.



Additional information

- <https://www.ietf.org/rfc/rfc2784.txt>
- <https://www.cloudflare.com/learning/network-layer/what-is-gre-tunneling/>
- https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5500/interfaces/61x/b-ncs5500-interfaces-configuration-guide-61x/b-ncs5500-interfaces-configuration-guide-61x_chapter_01101.pdf
- <https://networklessons.com/cisco/ccie-routing-switching/how-to-configure-gre-tunnel-on-cisco-ios-router>



 **L2TP** (Layer 2 Tunneling Protocol) is a descendant of the older PPTP (Microsoft) and L2F (Cisco) protocols. It is standardized by the IETF, the version L2TPv3 from 2005 is in RFC 3931.

As the name suggests, this protocol works on the L2 layer. The data-link layer VPN protocols are interesting in that they usually make a “loop” to a higher layer: they encapsulate traffic from the upper layer, but they encapsulate themselves in a TCP or UDP segment.

The main purpose of the L2TP protocol is to tunnel (encapsulate) PPP packets (which is used for data transmission over the telephone network, including ADSL/VDSL), and is very popular with various ISPs. It is encapsulated in UDP segments, it does not use TCP (the tunnel has no support at the L4 layer).

Like GRE, L2TP does not perform encryption, so it is usually combined with IPSec in transport mode, which is standardized in RFC 3193.


L2TP is supported on both Linux and Windows. In Linux it is recommended to use OpenL2TP (using IPSec), in Windows there is an L2TP/IPSec client for the transport mode.



Remark


In Windows, we can also meet with the Secure Socket Tunneling Protocol (SSTP), which transmits PPP or L2TP encrypted using the SSL protocol.



 **OpenVPN** runs on most known UNIX-like systems, including Linux and MacOS, there is also a variant for Windows. It is an open remote access solution that handles both tunnel creation and encryption, only transmitted data is encrypted.

This solution works at higher layers (above L3), which means that, for example, there is no need to solve problems with NAT or other IP address mappings. SSL or TLS protocol is used for encryption, authentication and key management, mostly UDP is used at the transport layer, but it is also possible to use TCP. The TLS protocol is standardized, but the entire OpenVPN solution is not.

While TLS is standardized, the entire OpenVPN solution is not.

 **SSH** (Secure Shell) is a more secure variant of the Telnet protocol, but in addition to securing communication itself, it has additional functionality compared to Telnet. It is used for secure access to a remote device in various directions. In addition to enabling secure configuration, it also performs a similar (but secure) role as FTP (file transfer), allows monitoring processes and resources on a remote system, creating encrypted VPN tunnels, etc. SSH version 2, issued in 2006 by the IETF (RFC 4254), is currently in use.

This is a client-server communication via TCP, there is a port 22 on the server side. Due to the fact that this is mainly about security, it is better to configure a different port than 22 on the server.

The best known implementation of the SSH protocol is the open-source project *OpenSSH*. There is a client and server variant for this product, and it is usually already installed on UNIX-like systems including Linux and MacOS. There is a client variant for Windows and it is used to access UNIX servers and network devices.


For Windows and Debian Linux systems, we also have the program *PuTTY*, which, however, exists only in the client variant. Again, it is used for remote access to UNIX servers and network devices.



Additional information


- <http://blog.trackets.com/2014/05/17/ssh-tunnel-local-and-remote-port-forwarding-explained-with-examples.html>
- <https://www.ietf.org/rfc/rfc4254.txt>
- <http://www.debianhelp.co.uk/ssh.htm>



 **IP-in-IP** is simply encapsulating an IP packet into an IP packet. It can be the same version or different versions, for example, this often solves the transport of IPv6 packets over a part of the network that “does not understand” IPv6 (so we encapsulate the IPv6 packet into an IPv4 packet).

An important feature of IP-in-IP is the change of IP addresses in the header, which is actually one of the features of tunneling: in the outer header, the IP address of the device (router) at the beginning of the tunnel is used as the source, the end address of the tunnel is used as the destination.

In the case of IPv6, the possibility of encryption is built directly into this mechanism (in the optional IPv6 header there would be an optional security header), in IPv4 we would have to provide encryption differently (for example using the IPSec protocol).

 **MPLS VPN.** MPLS networks are mostly used for the implementation of interconnection of corporate branches through a tunnel (or a company and a business partner), i.e. a VPN of the site-to-site type, the connections are of the point-to-point type. A header stack is used in MPLS

networks, one of the given headers is intended for VPNs. The solution works at the border of layers L2 and L3.

There are two basic solutions for MPLS VPN: MPLS Layer-3 VPN, MPLS Layer-2 VPN. Both IP packets and Ethernet frames (Ethernet over MPLS, EoMPLS solution) or PDUs of other protocols on these layers can be encapsulated.

The disadvantage of MPLS VPN is that we need an external provider for this service (usually an ISP, through whose WAN network our tunnel will run), we cannot create these tunnels ourselves. On the contrary, the advantage is that packets transmitted through the tunnel do not pass through the untrusted Internet, the configuration is on the side of the service provider and the service also includes QoS, so that despite relatively high financial costs, this service finds its customers.

8.2.2 Tunnels in Linux

The `iproute2` mechanism is a popular set of commands for working with networks (`ip`, `iw`, `ss` and others). It can also be used to create IP/IP tunnels. It can be used to create a VPN tunnel, encapsulate IPv6 into IPv4 on a path without IPv6 support, create a bridge between multiple network interfaces on a single device belonging to different networks, create a remote bridge, etc.

It is possible to work with several different types of tunnels, but first we must have the appropriate module for the tunnel type loaded in the kernel. The most common are

- *IP-IP tunnels* (the `ipip` module) – they encapsulate only unicast traffic (point-to-point tunnels),
- *GRE tunnels* (the `ip_gre` module) – they encapsulate various types of traffic, the both unicast and multicast, they are widely used,
- *SIT tunnels* (Simple Internet Transition, inside the `ipv6` module) – to interconnect IPv6 networks over IPv4 networks.

In commands for working with tunnels, the tunnel type is reflected in the mandatory parameter `mode` (i.e. the tunnel mode), it determines the way the transported packet should be handled (especially how and in what it should be encapsulated). We use the following command:

 `ip tunnel ...` examples of usage:

```
ip tunnel show mytunnelname    displays information about the created tunnel with the given
                               name, we can see the tunnel type (mode), remote and local IP address (tunnel endings),
                               and other features such as the network interface label, TTL value for outgoing packets,
                               the TOS value, etc.
```

```
ip -s tunnel show mytunnelname  in addition to the above information, tunnel statistics
                               similar to the output of the command ip -s link show will be displayed
```

```
ip tunnel add mytunnelname mode ipip local 194.50.20.42 remote
                               195.84.152.140 ttl 32  we create a new tunnel of IP-IP type with specified local and
                               remote address and TTL value
```

```
ip tunnel del mytunnelname     this tunnel is destroyed
```

```
ip tunnel change mytunnelname remote 195.35.84.15  changing settings of the given tun-
                               nel (the remote address has been changed)
```

In order to use these commands, we must first have the appropriate module loaded in the kernel.



Example

We will demonstrate establishing a GRE tunnel. Suppose we want to connect two remote networks (belonging to the same company) with private IP addresses through a tunnel. The address ranges in these networks should not overlap to avoid routing problems.

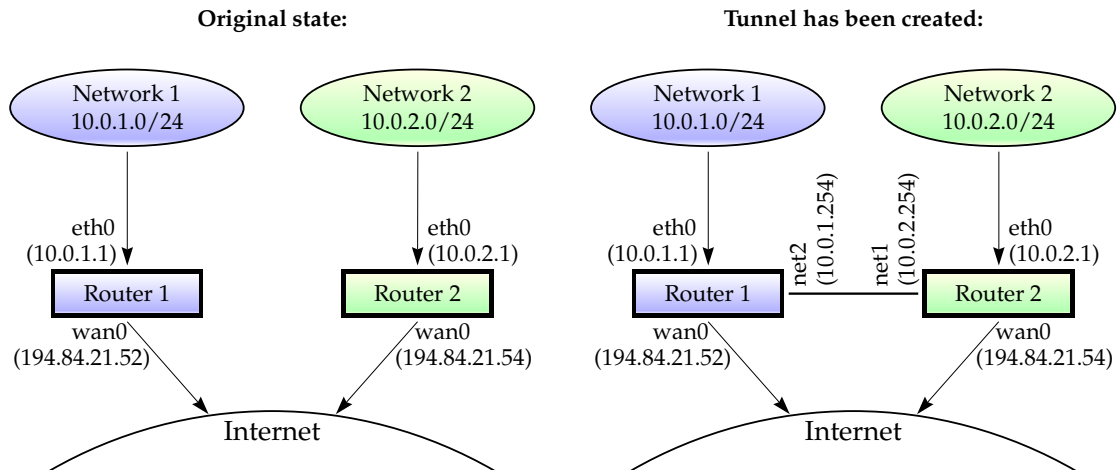


Figure 8.1: GRE tunnel establishing

The situation including addresses is shown in Figure 8.1 on the left. We can see from the prefixes that the address ranges in the networks do not overlap. For routers, through which networks connect to the Internet, we also see the address visible inside (it is from the address range of the internal network) and the address visible outside (it is assigned by the ISP). In order to build a functional tunnel, we need to know all these addresses (they are entered in commands).

We enter the following commands on the *Router1*:

```

modeprobe ip_gre    load the GRE tunnel support module into the kernel
ip tunnel add net2 mode gre local 194.84.21.52 remote 194.84.21.54 ttl 255
    create a tunnel, note the higher TTL value (we don't know how many routers the tunnel
    goes through, so it's better to use a higher value)
ip link set net2 up    activate the new virtual interface (i.e. our tunnel)
ip addr add 10.0.1.254 dev net2    assign an IP address to our end of the tunnel, it should be
    visible in our network and should not be used for other devices
ip route add 10.0.2.0/24 dev net2    add a rule to the routing table for routing to the network
    behind the tunnel, the closer end of the tunnel will act as a gateway
  
```

Similar commands for the *Router2*:

```

modeprobe ip_gre    load the GRE tunnel support module into the kernel
ip tunnel add net1 mode gre local 194.84.21.54 remote 194.84.21.52 ttl 255
    create a tunnel
ip link set net1 up    activate the new virtual interface (i.e. our tunnel)
ip addr add 10.0.2.254 dev net1    assign an IP address to our end of the tunnel
  
```

```
ip route add 10.0.1.0/24 dev net1    add a rule to the routing table for routing to the network
    behind the tunnel
```

Figure 8.1 on the right shows the situation after tunnel creation. Then we should test if the packets are delivered correctly, for example on the first router we try `ping` on the other end of the tunnel:

```
ping 10.0.2.154
```

If it does not work, it is possible that the packets were dropped by a firewall. If ICMP packets are enabled and we have no restrictions on the virtual interface we created, we should still check if GRE packets (protocol number 47) can pass through, for example in the FORWARD chain we would enable protocol 47 as follows:

```
iptables -A FORWARD -p 47 -j ACCEPT
```

Also, the problem may be in disabling the port number used by the tunnel, so if enabling protocol 47 doesn't help, we can try

```
iptables -A FORWARD -p tcp -dport 1723 -j ACCEPT
```

Possibly in other tables or with other parameters, depending on the current firewall settings.



We should note that GRE tunnels are not encrypted.




Additional information


- <https://kovyrin.net/2006/03/17/how-to-create-ip-ip-tunnel-between-freebsd-and-linux/>
- <https://www.informit.com/articles/article.aspx?p=29039&seqNum=3>
- <https://lartc.org/howto/index.html>
- <http://www.faqs.org/docs/Linux-mini/Remote-Bridging.html>
- <http://linux-ip.net/gl/ip-cref/>
- <http://linux-ip.net/html/index.html>



8.3 Network Analyzer


 A network analyzer is a device connected between a network element (PC, server, bridge, router, etc.) and a LAN, it can also be part of another device. A standalone network analyzer (portable) can be very small, it can resemble a mobile phone. We have already encountered an analyzer for Ethernet.


It works on the physical or higher layers, depending on which characteristics it can track. On the physical layer, mainly media status, traffic (load), it can generate its own traffic, on higher layers it can also support virtual networks, monitoring the status of available network nodes, frame or packet monitoring, etc.

 **Hardware by Fluke** is widely known in this field. The most used devices:

- *LinkRunner* – works on the physical layer, identification of Ethernet link properties – negotiation (10/100/1000 Mb, duplex, etc.), cable fault detection including distance to fault, interfaces, segment operation, etc.,


- *NetTool* – network layer, physical layer tests (cabling, negotiation, segment utilization), link layer (collision and error frame detection, VLANs, MAC address lookup in the network, etc.), network layer (IP address lookup in the network, subnets, routers and other sources, ping, etc.), PoE measurement, VoIP parameter monitoring, etc,
- *AirCheck* – Wi-fi network analysis.

 **Hardware by Embedded Technologies** usually contains its own variant of embedded Linux (i.e. Linux adapted for special purposes, here for network devices) together with other necessary software. Something similar can be built in-house from an older computer with the necessary hardware interfaces, and embedded Linux distributions are available on the Internet.

 **Software network analyzers:** Network analyzers can also be software – they monitor and possibly affect network traffic related to the computer on which they are installed (ideally a server, but it can be any computer on the network). Probably the best known software network analyzer is *Wireshark*.


8.4 Firewall

8.4.1 Principle of Firewall

 A firewall is a network equipment used to control (filter) traffic among networks with different levels of trust or security. The firewall defines *rules* for communication between networks, according to which it responds either by enabling communication, disabling it, or requesting the user's opinion. The rules usually cover the following information:

- the source and destination of communication, can be represented by an IP address,
- port number (source and destination),
- protocol, connection status, etc.

In respect to security, the combination of a simple hardware firewall in a router (for example an VDSL router) and a software router on a computer, each of a different type, is considered ideal in households and small companies.

 **Network Areas.** Firewall divides network into several areas:

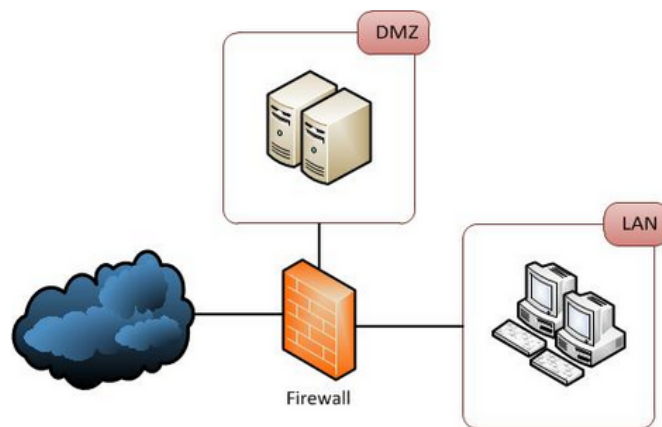
- *private network* (trusted, internal network, intranet) – our network with our devices, we usually trust these devices,
- *public network* (external network, internet) – untrusted network (of our ISP, WAN, Internet),
- *demilitarized zone (DMZ)* – somewhere between private and public network; we own everything inside DMZ, this area is usually intended for servers.

Inside DMZ, we have devices that should be accessible from the external network (for example, web server, mail server, DNS server). The servers in the DMZ should be secured as if they were really directly on the Internet, even though they are separated from the Internet by a firewall.

Figure 8.2 shows the above listed areas. The firewall in the middle of the schema filters the communication in the following way:

- an attempt to establish a connection from LAN wherever is allowed (passes),


³Resource: <https://resources.infosecinstitute.com/topic/virtual-dmzs-cloud/>


Figure 8.2: Areas divided by firewall³

- an attempt to establish a connection from the Internet to the DMZ is allowed,
- an attempt to establish a connection from the DMZ to the Internet is allowed,
- everything else is denied (will not pass).

If a device from the Internet tries to establish a TCP connection to LAN, it is denied, as well as if a device from DMZ tries to establish a connection to LAN.


On a network device that supports DMZ (e.g. firewall or a computer with Linux), we usually have one or more (hardware) ports labeled this way, or we can configure some ports ourselves so that they are treated as DMZ.

 A *zone-based policy firewall (ZPF)* allows to flexibly configure a various number of zones with different filtering methods (with different sets of rules). Each firewall interface is assigned to a specific zone and the rules set for that zone are valid for such interface. Usually we have one or more “inner” more or less trusted zones including DMZ and one “outer” untrusted zone. More than one interface can belong to a single zone.


 **TCP/UDP ports.** A regular user usually does not know how to set up (software) ports. On the Internet we can find pages with lists of known and registered ports used by different protocols.⁴ However, these lists are useful for monitoring outgoing traffic or when setting ports on a server. In fact, applications may use ports other than those common to the protocols they work with.

8.4.2 Filtering

We distinguish between different types of filtering according to which ISO/OSI layer a given method can be assigned to (and therefore which information in the headers we reach).

 **Packet filter.** This is the simplest filtering on L3 and partially L4, only IP addresses and port numbers are specified in the rules. It is a simple and fast solution (operation is delayed only minimally), which used to be commonly applied mainly to intermediate network elements, today we can find them in some of the cheapest routers. The disadvantage is the inability to look at the ongoing communication in more complex protocols.

⁴List of ports and services can be found e.g. at <http://www.iana.org/assignments/port-numbers>, http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers, <http://www.ports-services.com/>.


 **ACL at the network layer.** ACL (Access Control List) is a method widely used in both desktop and server operating systems, as well as in network devices. The purpose is to determine access rules for different clusters of users/communications. In essence, it is a functional extension of the packet filter method.

The access control list is actually a list of items marked by categories:

- *deny* – not allowed traffic,
- *permit* – allowed traffic,
- *deny all else* – what was not previously allowed must not pass.

The ACL is usually implemented in the form “what is not allowed is denied”, so we will find only *permit* items (and what is not in these items will not pass). The order of the items is also important.


It is usually filtered according to the destination or source IP address, their combination, and possibly according to other criteria (for example items in the PDU header of the network layer – IP protocol, ICMP, TCP/UDP ports, etc.).

 **Stateful packet inspection (SPI).** At the transport layer (strictly: L3+L4), SPI is performed. While on L3 packets are taken as “individuals” without mutual binding, on L4 it is possible to take into account their mutual relations. For example, we can distinguish packets that establish a connection from packets that belong to an existing connection.

The segment establishing connection is thoroughly scanned (data from the L3 and L4 layers, such as source and destination addresses, protocols, source and destination ports, flags set by each protocol, anything in the PDU headers), and if it passes, a record in the *status table* is created for this connection. If the packet fails the check, the record is not created and packet is denied.

Packets that belong to (or impersonate) an already established connection are filtered according to whether their connection is recorded in the status table.

Currently, SPI is basically a standard for quality firewalls. Compared to a conventional packet filter, it offers greater security with a relatively small slowdown in filtering operation.

 **Proxy firewalls** (also called application gateways) operate on the TCP/IP application layer. They work with application protocols, for example they understand HTTP, FTP, IMAP, they can work with packets containing elements for ActiveX, etc.

This type of firewall actually works as a man-in-the-middle (but legal), i.e. it divides the path between the client and the server into two parts. Server “looks at” proxy instead of client, client “looks at” proxy instead of server. This can cause problems, especially for encrypted connections.


All communication is processed using so-called *proxy* – software gateways either programmed for a specific type of communication (protocol) with a relatively high level of security (for example for FTP or HTTP), or using a generic proxy usable generally for various protocols.

 We distinguish between two types of proxy:

1. *standard proxy* – everything that was previously written about proxy applies to it. It filters all packets according to data from PDU protocols of the application layer and lower layers, the analysis is time consuming and can have a great influence on the throughput of the line.
2. *Dynamic proxy* – behaves differently in different packets; to the initiating connection it behaves the same as the first type of proxy, but to packets belonging to the already established


connection it behaves as an SPI (i.e. at the lower TCP/IP layer). The result is faster processing of incoming and outgoing traffic.

8.4.3 IDS/IPS and Deep Packet Inspection

 *IDS* (Intrusion *Detection* System) is a passive appliance (the traffic flow does not go through this device, but is mirrored to it). *IDS* does not interfere in the data flow, it only announces in some way that something is wrong.

IPS (Intrusion *Prevention* System) is able to actively respond to detected problems. The response is intended to prevent an attack, so the specific location of the *IPS* probe has to be planned so that it can also interfere with the configuration of other network devices (for example, change the rules in the firewall) or drop packets (*IDS* cannot drop packets). The main problem of *IPS* is that its overload or e.g. technical problems might affect network traffic, and the performance of this device may affect the network (slower, worse latency, etc.) even during normal operation.

A firewall can have an integrated *IDS/IPS* function (module), but it is generally safer (especially for larger networks) to deploy *IDS/IPS* separately from the firewall. Thus *IDS/IPS* is either a separate device, or a hardware module in a router or firewall, or a software module or application.

 *IDS* and *IPS* appliances search for malicious traffic patterns and compare them with *signatures* obtained from trusted security sources (clouds of security providers). Additional functionality is using *heuristics* and detection of *unusual network traffic*.

Unusual network traffic is called *anomaly-based detection*, it means real-time comparing of host/network behaviour with a learned *baseline*. A baseline is a model for common host/network behaviour, for example a network baseline is characterized by the network performance, current protocols in the traffic, the ratio of source and destination addresses from the internal and external network, the dynamics of changes in the operation and use of the network, etc.

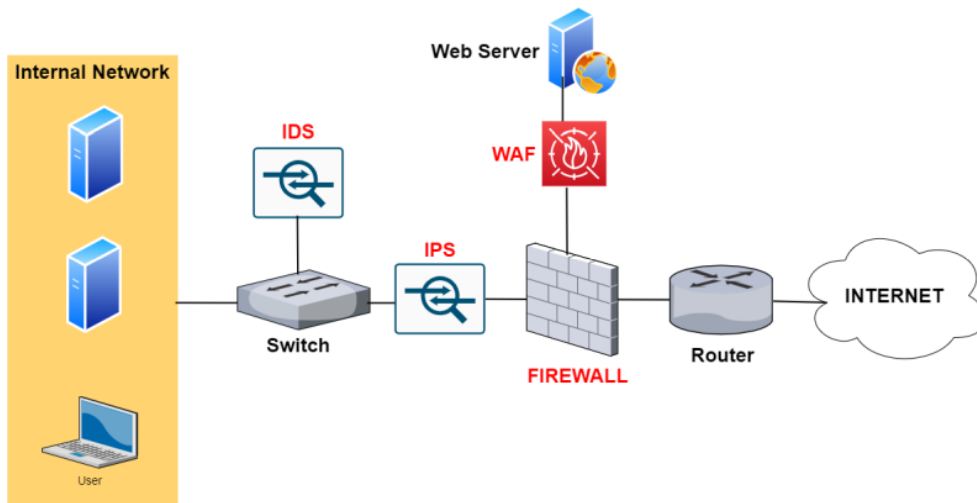




Figure 8.3: Scheme of possible *IDS/IPS* connection⁵


 Stateful packet filter with *deep packet inspection* (*DPI*) is just a firewall with an integrated *IDS/IPS* module. This type of firewall adds additional options for defining rules related to the

⁵Resource: <https://www.networkstraining.com/firewall-vs-ips-vs-ids-vs-waf/>

usual properties of communication with a given (known) application or protocol. For example, if it detects that HTTP is being used for a different type of communication than with the Web server, it blocks this request as suspicious.

 *HIDS, HIPS* (Host-based IDS/IPS) is a module in an end device (server, computer, etc.) intended for protection of the host device, it is an agent-based system.

Examples of HIDS/HIPS: OSSEC (Open Source HIDS Security), Safetica, AlienVault USM, Tripwire and its open-source variant AIDE, Cisco AMP. OSSEC uses a central server for IDS/IPS management and agents installed on individual hosts (computers, servers).

 *NIDS, NIPS* (Network-based IDS/IPS) is implemented as a sensor through which the data flow in the network passes (NIPS) or the data flow is mirrored to it for monitoring purposes (NIDS).

Examples of NIDS/NIPS: Snort, Squil, Suricata, Fortigate, FortiOS, Zeek (Bro), OpenWIPS-ng. Most of them are freely available, except Fortigate and FortiOS.

In Figure 8.3 we see a possible scheme of connecting firewalls and IDS/IPS in a larger network. The whole network is protected by a firewall, behind which is the IPS. All communication goes through the IPS, so it can respond to anything suspicious in any packet. The firewall defines three zones, one of them is DMZ (there is a web server inside, protected by a web application firewall, WAF, possibly with a built-in IDS/IPS module). A switch mirrors all traffic leading to and from the internal network to an IDS.



Additional information

- <https://www.networkstraining.com/firewall-vs-ips-vs-ids-vs-waf/>
- <http://www.symantec.com/connect/articles/intrusion-detection-terminology-part-one>
- <http://www.symantec.com/connect/articles/intrusion-detection-terminology-part-two>



Recommended Resources

- [1] *Aruba (HP) OS 6.1 Documentation* [online].
URL: https://www.arubanetworks.com/techdocs/ArubaOS_61/ArubaOS_61_UG/ [cit. 2020-10-30]
- [2] CHAUDHARY, Harry. *Hands on Computer Networks 1500+ MCQ Test Series: Handy Book Series for All IT Exams & Interviews*. STCD Company, 2018. ISBN 978-0-359-03847-3.
Most pages available at: <https://books.google.cz/books?id=fpJqDwAAQBAJ&printsec=frontcover>
- [3] *CCNA Cybersecurity Operations Companion Guide*. Indianapolis, IN: Pearson Education, Cisco Press, 2018. ISBN 978-1-58713-439-5. Most pages available at:
<https://books.google.cz/books?id=FxRbDwAAQBAJ&printsec=frontcover>
- [4] *Check Point Support Center* [online]. URL:
<https://supportcenter.checkpoint.com/supportcenter/portal> [cit. 2020-10-30]
- [5] *Cisco Documentation* [online]. URL: <https://www.cisco.com/c/en/us/tech/index.html> [cit. 2020-10-30]
- [6] ELENKOV, Nikolay. *Android Security Internals*. San Francisco, USA: No Starch Press, 2014. ISBN 1-593-227581-1.
- [7] *Fortinet Docs Library* [online]. URL: <https://docs.fortinet.com/> [cit. 2020-10-30]
- [8] GORALSKI, Walter. *The Illustrated Network: How TCP/IP Works in a Modern Network*. 2nd edn. Cambridge, MA: Elsevier, 2017. ISBN 978-0-12-811027-0. Most pages available at:
<https://books.google.cz/books?id=6nDtNA6VJ5YC&printsec=frontcover>
- [9] GUBBI, Jayavardhana, et al. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, Elsevier, 2013, Volume 29, Issue 7. Pages 1645–1660, ISSN 0167-739X.
- [10] HUBERT, B. et al. *Linux Advanced routing & Traffic Control* [online].
URL: <http://lartc.org/howto/>, others <http://lartc.org/> [cit. 2020-10-30]
- [11] HURA, Gurdeep S. a Mukesh SINGHAL. *Data and computer communications: networking and internetworking*. Boca Raton, FL: CRC Press, 2001. ISBN 08-493-0928-X. Most pages available at: https://books.google.cz/books?id=BViV0PoH_voC&printsec=frontcover

-
- [12] *Introduction to Networks v6: Companion Guide*. Indianapolis, IN, USA: Cisco Press, [2017]. ISBN 978-1-58713-360-2. Also available at:
<http://ptgmedia.pearsoncmg.com/images/9781587133602/samplepages/9781587133602.pdf>
- [13] *Juniper Documentation* [online]. URL: <https://www.juniper.net/documentation/> [cit. 2020-10-30]
- [14] LAMMLE, Todd. *CCNA: Routing and Switching: Study Guide*. Indianapolis, Indiana: SYBEX, [2013]. ISBN 978-1118749616. Available at: <http://index-of.co.uk/Various/CCNA>
- [15] *Linux Home Networking* [online]. Books on computer network management in Linux. URL: <http://www.linuxhomenetworking.com/wiki/index.php> [cit. 2020-10-30]
- [16] *Routing and Switching Essentials: Companion Guide*. Indianapolis, IN: Cisco Press, c2014. Cisco Networking Academy Program series. ISBN 978-1-58713-318-3. Also available at:
<http://ptgmedia.pearsoncmg.com/images/9781587133183/samplepages/1587133180.pdf>
- [17] SANDERS, Chris. *Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems*. 3rd edition. San Francisco: No Starch Press, [2017]. ISBN 978-1-59327-802-1. Sample capture files available at: https://nostarch.com/download/ppa3ecaptures_updated.zip
- [18] *Scaling Networks v6: Companion Guide*. Indianapolis, Indiana: Cisco Press, [2018]. ISBN 978-1-58713-434-0. Also available at:
http://ptgmedia.pearsoncmg.com/images/9781587134340/samplepages/9781587134340_sample.pdf
- [19] TAUB, Mark L. *Connecting Networks v6: Companion Guide*. Indianapolis, Indiana: Cisco Press, [2018]. Cisco Networking Academy Program series. ISBN 978-1-58713-432-6. Also available at:
http://ptgmedia.pearsoncmg.com/images/9781587134326/samplepages/9781587134326_sample.pdf
- [20] *The TCP/IP Guide* [online]. URL: <http://www.tcpipguide.com/free/index.htm> [cit. 2020-10-30]
- [21] VERMA, Dinesh C. *Content distribution networks: an engineering approach*. New York, N.Y.: Wiley, c2002, xv, 182 p. ISBN 04-714-4341-7.
- [22] WALRAND, Jean and Shyam PAREKH. *Communication Networks: A Concise Introduction*. Second Edition. Morgan & Claypool Publishers, 2018. ISBN 978-1681-73615-0. Most pages available at: <https://books.google.cz/books?id=dcBIDwAAQBAJ&printsec=frontcover>