MODULE 3:

Internet protocols and the WWW system

The purpose of the module

The module provides basic information about the architecture and protocols of the Internet, important from the point of view of a web application programmer. The assumptions and principles on which Internet protocols are based are explained. Aspects of architecture decentralization and its role in ensuring network flexibility and scalability were indicated.

Learning outcomes

This lesson aims to achieve the following learning outcomes.

Knowledge

After completing the course, the student:

- • Can place key historical events related to the development of the Internet in time, • Characterizes the layered model of OSI protocols and confronts it with the family of protocols the Internet,
- • Characterizes the most important Internet protocols, • Explains the rules of the DNS system, • Characterizes the types of addressing in the Internet, • Characterizes the HTTP protocol and the changes that have been introduced in the course of its development,

Skills

After completing the course, the student:

- • Uses English-language Internet protocol standards specification documents, • I use diagnostic tools to analyze request and response messages HTTP.

Justification

A basic understanding of protocols is important to a web application developer for at least two reasons. Firstly, it defines a common, uniform application programming context regardless of the programming technologies used (including, among others, a set of information about the client that the web application may have at its disposal). Secondly, it imposes certain limitations on the communication model and its costs, which should be taken into account when ensuring proper application performance.

The role of Internet protocols in building web applications

In the previous modules, we dealt with markup documents and their formatting, ie, in the context of web applications, the presentation layer of the software. The related processing (declarative binding of formatting to the markup document) was carried out directly within the consumer program (i.e. a web browser). We haven't addressed the topic of programming in JavaScript on the browser side yet - but here too the situation would be analogous: processing on the consumer's side. So we could completely disregard how the document and related resources reach the client. The situation changes dramatically when we consider the entire web application:

- Instructions generated on the client's side, which must be sent to the server; • The scope of information about the request and about the client that is available on the server software;
- Providing the client with content intended to be presented to him.

In all these aspects, the protocol used in communication plays a key role.

A few important dates

Before we get into technical details - let's look at a few significant dates marking the period of development of the technologies discussed here.

1945 - **Vannevar Bush - "As We May Think"**

An important (at least - from today's perspective) article in the "Atlantic Monthly" magazine. It is considered the origin of the concept of hypertext on which the WWW system was based.

1958 - **ARPA**

The Advanced Research Project Agency (later renamed DARPA) is formed. An agency subordinated to the US Department of Defense, established to conduct large research projects of significance for defense.

1965 - **"On Distributed Communications Network"**

Author - Paul Baran of RAND Corporation. The work outlined the concept of a decentralized and failure/ destruction resistant computer network.

1969 - **ARPA-NET**

In the United States, an experimental ARPA-NET computer network is created (launch of the first nodes).

1971 - **eMail**

Ray Tomilson creates the first e-mail program

1972 - **Telnet**

Another important web service.

1973 - **First international connections**

ARPA-NET reaches international scope: connections to Great Britain and Norway

1974 - **"A protocol for Packet Intercommunication"**

Work by Vinton Cerf and Bob Kahn describing the concept of TCP; also introduces the term "internet"

1975 - **ARPA-NET becomes a commercial network**

The network changes its status from experimental to operational.

1982 - **TCP/IP**

ARPA-NET introduces the family of TCP/IP protocols

1984 - **DNS**

Introduction of the DNS service. From now on, domain names can be used instead of IP numbers to identify computers on the Internet

1991 - **Poland connected to the Internet**

First connection - with Denmark

1990 - **Web concept**

Tim Berners Lee (CERN) specifies the concept of a web system and markup language

1993 - **Mosaic**

The first web browser

1994 - **Spam**

First major case of Internet spam: Canter and Siegel law firm sends letter to 6,000 discussion groups (legal services related to the lottery of work permits in the US)

1995 - **Netscape, Java**

The Netscape browser and the Java language appear

1995 - **WWW the most popular service**

WWW overtakes FTP, becoming the most popular Internet service

1996 - **XML**

The specification of the XML language is created

1996 - **HTTP 1.1**

Significant improvement of the HTTP protocol used in the WWW system

2000 - **Web Services**

Initiation of Web Services standardization

Current control	( )	*reply comment*
The web system was born:		
In the second half of the 1940s, soon after the appearance of the first computers		NO. The WWW was created after the birth of the Internet.
In the first half of the 1970s, it was the first Internet service		NO. Email, Usenet, and file transfer services are much older.
In the mid-1980s - along with the DNS service		NO. DNS is very important to the usability of the web, but the two systems were developed independently.
In the first half of the 1990s - before the creation of the Java language		yes. This extremely popular and momentous service is still a relatively young invention.
At the turn of the 20th and 21st centuries - as a component of the Web Services technology family		NO. Web Services were created later, building on the well-established components of the Web system.

Layered model of Internet protocols When characterizing Internet protocols, it is impossible to omit the classic OSI *(Open Software Interconnection) layered model.* It is not a ready-made recipe to be copied when designing a layered set of protocols, but it systematically and conveniently decomposes the complex problem of electronic communication into hierarchically arranged issues. This model distinguished the following layers of the problem, starting from the lowest and ending with the most specialized:

1. Physical: as the name suggests, it is represented by physical infrastructure elements; provides the transfer of raw data bits over the medium 2. Links: provides the transfer of interpretable sets of bits between two network nodes 3. Network: creates an abstraction layer that allows the use of an internetwork that may consist of different technologies and methods of communication of physical networks; converts logical addresses to hardware, divides logical packets into (used within layer 2) data frames 4. Transport: provides data division, delivery guarantee, speed control 5. Session: establishing and closing connections between cooperating nodes 6. Data presentation: format data exchange, their possible compression or encryption 7. Applications: services provided for specific applications.

Layers 1 through 4 are collectively referred to as transport layers, and layers 5 through 7 as application layers. As you can see, this model covers communication issues, starting from sending signals recognizable as elementary units of information in a given medium, through their grouping into larger portions, then - ensuring their uniform form, independent of the target platform, ensuring the reliability of the message, its proper interpretation, and finally - application-specific functions. Particularly noteworthy in this model is the network layer, as it unifies communication and isolates higher-level solutions from the heterogeneity of hardware solutions in the physical networks it integrates.

Current control ( ) *response comment*		
The layer in the OSI model that provides a common abstraction that hides the details of various physical networks and introduces a unified system of logical addresses is:		
Application layer		NO. We solve this problem in the lower layer
Data presentation layer		NO. We solve this problem in the lower layer
Session layer		NO. We solve this problem in the lower layer No. We solve
Transport layer		this problem in the lower layer
network layer	x	Yes it is. It is in this layer that the concept of internetworking occurs
Link layer		NO. We solve this problem in the upper layer No. We solve
physical layer		this problem in a higher layer

The Internet and TCP/IP

The Internet is defined as a virtual network that implements a model of uniform services. This means that it allows you to connect physical networks and provides a level of abstraction that hides their heterogeneity, making them (from the point of view of the services built on it) a virtual network with a common addressing system and protocol. The role of this "common denominator" is played here by the IP (Internet Protocol) protocol with its system of abstract protocol addresses - IP numbers.

The architecture of the Internet provides far-reaching decentralization and fault tolerance. The failure of a single computer or even the unavailability of a certain physical network may still allow to determine an alternative path *(route)* between communicating machines.
Control over the uniqueness of addresses and names must remain centralized to some extent. This is overseen by the Internet Corporation for Assigned Names and Numbers (ICANN). As the name suggests, this private non-profit entity is responsible for ensuring a unique allocation of IP addresses as well as (see below) domain names. On the basis created by the communication protocols, protocols specialized for individual services for users (e.g. mail, file transfer, www ...) were defined.

The Internet protocol stack has fewer layers than the OSI model mentioned earlier, but it is based on the same philosophy. There are 4 layers here:

Connection - device drivers and network interface card; e.g. Ethernet, token ring. It implements ARP (Address Resolution Protocol) and RARP (Reverse Address Resolution Protocol), ensuring mapping in the local network between logical addresses (like IP number) and physical addresses (like MAC). Also known as the network interface layer. It covers all the details of physical communication.

Network - Also known as the Internet layer. Handles packet traffic over the network. Among other things, packet routing takes place here. It includes the Internet Protocol (IP), as well as e.g. Internet Control Message Protocol (ICMP), Internet Group Management Protocol (IGMP).

Transport - Includes TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). Unlike IP, whose functions are performed by every node along the packet route, transport protocols are end-to-end, which means that only the sender and the final recipient need to perform transport protocol activities. TCP provides a reliable (reliable) flow of data (confirms their delivery, works in connection mode). For this purpose, it performs their portioning, response time control, connection setup. UDP transmits datagrams with no guarantee of delivery.

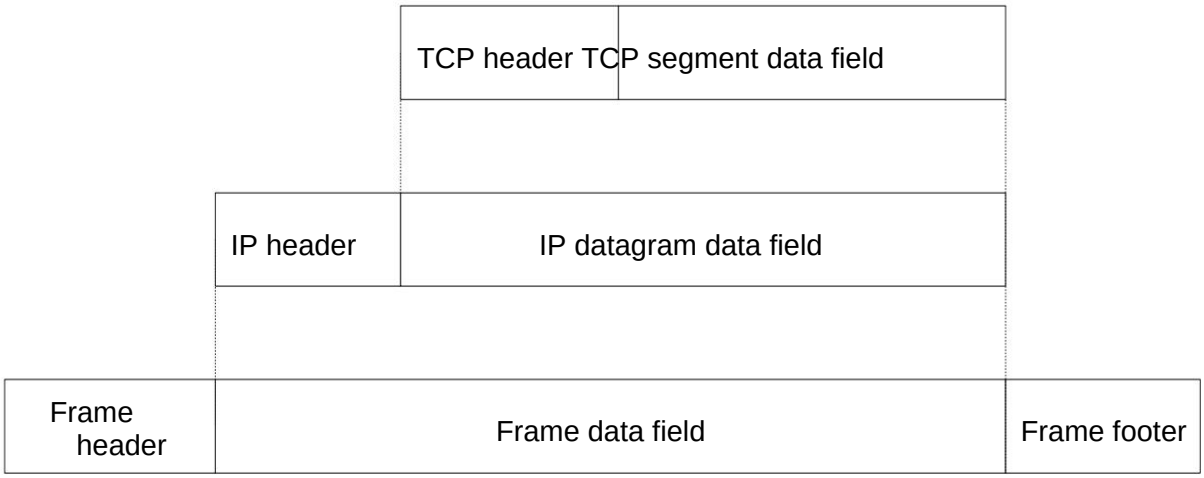
Applications - Includes specialized protocols for utility applications (including Telnet, FTP, SMTP, HTTP...).

Current check ( ) *response comment* Based on the table above, you can		
guess that the popular ping diagnostic instruction uses a protocol from the Internet layer: Connection. Its range is limited to the local network.		
		NO. We solve this problem in a higher layer
network. It uses the ICMP Transport	yes.	
protocol. Only it allows you to reach the target node of the		NO. We solve this problem in the lower layer No. We
Application. It uses the HTTP protocol.		solve this problem in the lower layer

Encapsulation

Referring to the layered organization of Internet protocols, the concept of encapsulation (or encapsulation) of transmitted messages should be mentioned.

The protocol of each layer requires (to be able to perform its tasks) additional data to be added to the message entrusted to it - e.g. address, information on the method of data encoding, requirements regarding the method of delivery, identification of the program or protocol responsible for processing the message on the recipient's side, etc. Placed are in the header and possibly in the footer of the transferred data unit. see the figure below.



The network layer places in the header e.g. sender and recipient IP addresses, allowing for effective delivery of the datagram to the addressee. The header also contains information about what kind of content is in the body of the datagram - this allows you to decide which protocol on the side of the target recipient should be used to interpret the message.

The transport layer, on the other hand, places (in addition to a number of auxiliary information ensuring reliability and appropriate transfer rate) 16-bit numbers of the source and destination port in the header. They identify, within the machines identified by IP numbers, specific processes that communicate with each other.

On the other hand, demultiplexing is a reverse process on the recipient's side: after receiving a frame, successive layers "unpack" the data and, based on identifiers in known headers, transfer it to the appropriate protocol of the upper layer.

Let's simplify this process of encapsulation and demultiplexing using the example of a request to download a web page. On the client machine side, the following is done:

1. Based on the URL address provided in the address bar of the browser - the IP number of the machine on which the server is running is determined, as well as the port number on which the server is listening.
2. The appropriate content of the page download request is prepared as a text string
   HTTP protocol
3. This content is placed in the body of the TCP segment, and the information is sent to the TCP headers about sender and receiver ports.

4. The IP segment goes as the body of the IP datagram created for it; headers are filled in, among others information about the sender's and recipient's IP addresses and to indicate that the content of the datagram is a TCP segment.
5. A datagram constructed in this way is entrusted to a network frame in accordance with the technology of a given physical network (e.g. Ethernet); the frame header identifies the destination node within a given physical network (it can be the final recipient - in our case: a web server), or a router that will forward the datagram beyond the boundaries of a given physical network.

The demultipiexing process is analogous.

So we can see that the figure above does not exhaust the full encapsulation stack discussed in this example. Inside the TCP segment, we would also have headers and possibly also (depending on the type of message) the body of the message (and in it - if it was a response from the server - we could encounter header and body elements that make up the HTML document).

Note that the concept of encapsulation, although it has a slightly different meaning here, is motivated by the same as, for example, encapsulation in programming languages: a given protocol treats the content of the message with its information intended for other protocols as an opaque whole and can abstract from its structure.

Current control ( ) *response comment*		
The port number identifying the recipient can be found in:		
frame footer		NO. This data is not needed by the connection layer No. This data
Frame header		is not needed by the connection layer No. This data is not needed
IP header		by the network layer x Yes. The transport layer identifies
TCP header	a	using the port number of the destination
HTTP header		process No. This data must be processed in the lower layer

Addressing

The IP address provides a uniform way to identify machines on the Internet. It is a logical address which, in order to identify the addressee, must be finally mapped to a hardware address in a given physical network. If it identifies a machine on the current network, the mapping can be done. Otherwise - the datagram will go to a machine called a router operating on the border of a given physical network, which, based on the IP address, will determine the next intermediary to whom it should be forwarded.

The IP address in the currently commonly used IPv4 protocol consists of 4 octets (to improve human readability - usually written as 4 decimal numbers from the range 0-255) - a total of 32 bits. This address must distinguish between the address of the network and the address of the machine on that network. Depending on the size of the network - the division in terms of the number of bits identifying the network and the machine in this network may be different. Originally, it was a rigid division, that is - resulting from the address value itself. This is the so-called class approach, which distinguished 5 classes of addresses. In addition, the entire pool of 127.*.*.* are loopback addresses - identifying the machine that is referencing such an address.

With the rapid growth of the Internet, the above class approach has shown a significant inflexibility. It was too lavish in terms of allocating ranges of IP addresses (which are again not so many in IP v4); it also turned out to be sub-optimal from the point of view of routing table growth

maintained by routers. In order to increase the efficiency of using the pool of addresses in IPv4, a specification of the classless approach to addressing was introduced, in which the number of bits defining the so-called network mask is specified explicitly - e.g. 192.9.205.22 /18

The need to address machines in local networks connected to the Internet was also taken into account so that they could access its resources, but were not full members of the Network with a unique address. This gave rise to the need to extract addresses that could be repeated between different private networks of individual entities, but at the same time - so that they would not conflict with public addresses on the Internet. Therefore, 3 pools of addresses were distinguished (historical history - so that such addresses would be available in each of the classes from A to C):

from 10.0.0.0 to 10.255.255.255

from 172.16.0.0 to 172.31.255.255

from 192.168.0.0 to 192.168.255.255

Machines with such addresses can use Internet resources thanks to the use of appropriate NAT (Network Address Translation) techniques.

The ways of addressing networks, machines and their resources on the Internet can be summarized in the following list:

Physical addresses (e.g. Ethernet MAC - 48-bit number) - they allow you to identify machines in the local network. Thanks to the successive layers of the protocol model, the web application developer is insulated from these addressing details.

IP addresses (in IPv4 - e.g. 148.81.141.16) - allow you to refer to a remote machine without knowing its hardware address. They are the basis of routing.

Port numbers – define the way of interpreting the message. Namely, they allow the recipient to decide which of the programs operating on his side should process the message.

Domain names - are a human-friendly equivalent of IP addresses (they take the form of text and are organized hierarchically within the defined main domains). They are translated by the DNS service (see below) to IP addresses.

Identifiers in application protocols - specify references by specifying a resource, e.g. e-mail box name on a given mail server, local address of a specific page on a web server.

<< ANIMATION A1: sending a page download request illustrated in individual communication layers - see attachment >>

The principle of locality in computer networks

In traditional services on the network, local and repeatable references between machines can be observed. This observation has been described as the locality of reference principle: a given computer is more likely to communicate with physically nearby entities. This regularity is complemented by the temporal locality of references: a given computer often communicates with the same machines many times (i.e. again). This principle motivates the development of local networks and the optimization of communication for re-referencing the same machine. As we shall see, this principle is manifested in the design decisions in the protocols as well as in the way the DNS service is organized.

Application protocols

DNS

The Domain Name System (DNS) is included in the application layer, although it is actually associated with the Internet infrastructure (it is intensively used by a number of other application protocols; it is also logically strongly associated with the network layer). In a word - it is implemented in the application layer, but it is used to obtain information for communication in the network layer.

DNS is a distributed database that maps names and their aliases to IP addresses.
It can be said that it is one of the largest and most intensively used distributed databases. Domain names form a hierarchy:

- Primary DNS servers (they are also replicated - 13 items) know the addresses of the servers responsible for individual root domains, etc.
- The server hosting the domain may delegate maintenance of the "subtree" of the hierarchy to another server.
- The organization to which a given domain has been assigned can therefore independently and flexibly handle your sub-domains.

In accordance with the principle of locality - name mapping starts on the DNS server local to a given user (who independently has knowledge about name mapping from the local domain). On the other hand, according to the principle of temporal locality of references, mapping results are cached for a given time. Caching dramatically improves system performance - especially in evaluating popular domain addresses.

Current control	( ) *reply comment*	
We will characterize the DNS system as:		
A distributed, hierarchically decomposed database	Yes.	
Centralized database replicated on 13 servers		NO. The size and dynamics of changes to the set of domain names would not be supported by a single server. No.
A distributed database divided into servers according to the addresses of the physical networks to which the entries relate		The key to data sharing here is not the physical construction of the network

SMTP

Simple Mail Transfer Protocol (SMTP) - performs reliable message transfer (the sender keeps a copy until he is sure that the recipient has saved the letter in non-volatile memory).
In addition, a check is made to see if the destination mailbox exists.

This protocol is based on the functionality provided by the lower layers, so its specification can focus on specifying the character encoding and information specific to the e-mail message. The content format is defined as a text document in the form:

- Header - a keyword - eg From, To, CC, Date, Subject, Reply-To, X-Charset, X-Mailer, X-Sender, X-Face; followed by a colon and text representing the header value • blank line • actual content

In this form, the standard was formulated in 1982 and met the requirements of the time - limited to the exchange of traditional text messages.

MIME

The dissemination of the e-mail service and the increase in processing capabilities opened up the need for the transfer of binary files, including programs, images or sounds. This meant that the limitations on the volume of mails, character sets or message format adopted in the original version of SMTP had to be revised. In particular, it became necessary to enable the sending of messages with attachments and to specify the method of encoding binary attachments. To this end, a separate specification called Multipurpose Internet Mail Extensions defined additional header lines and established an open system for defining different types of content. This enabled:

• sending multiple separate objects within a single letter • removing restrictions on line length and the size of the entire message • using non-ASCII character sets • sending various media and document formats as attachments

MIME specifies:

• Header row identifying the extension: MIME-Version: 1.0

Predefined set of main content types: text, multipart, application, message, image, audio, video • Content subtypes and Content-Type

header: used to specify the type of content

Encoding mechanism to comply with transport restrictions to non-ASCII characters. This format, called BASE64, represents each three bytes as 4 printable ASCII characters. Hence the well-known phenomenon of the increase in the volume of the shipment over the size occupied on the disk by the attachment file. The Content-Transfer-Encoding: Special header, including mixed content, allows for arbitrary nesting of attachments.

Allows you to define a separator. The resulting definition in the header may take the following form: Content-Type: Multipart/Mixed; Boundary="up to 70-print-characters"

Examples of other application protocols

Here are some more popular application protocols:

POP (Post Office Protocol)

Allows you to retrieve mail from a remote server

FTP (File Transfer Protocol)

File transfer. Usually based on TCP. The control connection remains open throughout the session. Data connections are created separately for each file transfer command.

TFTP (Trivial FTP)

A minimalist replacement for FTP, minimizing network traffic. UDP-based. No interactivity. Ability to send a copy of the file in any direction. It can be used as a hardware boot protocol.

NFS (Network File System)

It offers the usual file operations. The integration of the service with the local file system makes it possible for various programs to work with remote files. Ability to lock a file for concurrent access security. Much more effective when accessing and e.g. modifying a remote file is required.

The WWW system

The WWW (World Wide Web) system was created relatively recently (early 1990s) and quickly became the most popular Internet service (in 1995 it overtook the FTP service). Due to its popularity and the lack of a convenient Polish equivalent of the term Web, this term is often confused with the term Internet in colloquial speech. The characteristics of the website include that it breaks with the principle of local appeal. With the development of this system, we observe its powerful impact on economic activity on the following levels:

• New marketing communication channel • E-commerce platform • IT integration tool in the business-to-business (B2B) model

The layer of communication with the user of the WWW system assumes the so-called point and click interface, i.e. the use of pointing devices. The WWW is referred to as a hypermedia system, i.e. an extension (presence of information other than text) of the concept of hypertext. Documents are administered independently, so we have to take into account the potentially low stability of links leading to third-party resources. Note that the server's scenario is quite simple: providing documents requested by the client in response to a message from the client. (However, the use of dynamic web document technology significantly changes this state of affairs. We will return to this topic in one of the next lectures.) The browser is a relatively complex tool. We expect it to implement the following components:

• control module reacting to signals from input devices • HTTP client • HTML interpreter • other optional interpreters and clients (FTP, POP, SMTP)

Resources in the web system are identified by a URL (Uniform Resource Locator). In its construction, the following elements can be distinguished:

• specifying the addressing scheme: "http" • host address (domain name or - not recommended - IP number) • port number (by default 80 - can be omitted then) • absolute path to the resource (virtual!) • Possibly. request parameters (so-called *query string* - placed after the "?" symbol)

For example, the URL address may take the following form:

http://www.example.org:8080/publications.php?lang=en&yr=2000

It should also be added that URLs have syntax rules that do not allow the direct use of certain characters. Therefore, such characters require encoding (so-called *URL encoding)* using the symbol "%" of a number written in the hexadecimal system. For example, the space symbol can be replaced with "%20" Content on a web system need not be limited to static content. Web pages may contain code for executing programs

on the server side when the client downloads the page from a given address, or on the client side - after loading the content into the browser environment.

HTTP protocol

The standard defines the use of HTTP (Hypertext Transfer Protocol) in a rather general way, as "transfer of various types of resources over the network". The design of this protocol must take into account the specificity of the web system, namely the fact that viewing hypermedia documents does not show significant locality of references. Thus, we have a different reference pattern than for other types of application programs. The protocol was therefore originally designed as stateless (no concept of a session grouping interactions). The interaction of the browser with the web server takes place according to the connectionless model:

- the request is sent by the client • The server transfers (always at the client's initiative) the requested resources or information about them unavailable • the connection is closed

Communication in the HTTP protocol is transferred in the transport layer via the TCP protocol, which guarantees the reliability of transmission, but also generates significant overheads for establishing the connection (so-called *handshake)* and its subsequent closure.

The protocol defines the request and response format. The structure of messages shows similarities with the previously discussed SMTP protocol. As there, we have the header part and (optionally) the body of the message separated from it by a blank line.

Both the headers and the message body can be present both in the request from the client and in the response sent by the web server (some types of requests and responses exclude the presence of the message body by their nature). Since the task of locating the server rests on lower layer protocols, a request to download a web document (in the older version 1.0 protocol) can take the form of something as simple as:

GET local_path_and_file_name_on_HTTP/1.0 server [empty line]

It should be emphasized that the range of functionalities of the http protocol significantly exceeds the mere downloading of resources located on the server. Among other things, the designers provided for support through HTTP requests for the full set of "CRUD" operations (i.e. Create, Retrieve, Update, Delete), defining the appropriate request methods:

PUT

Placing the resource on the server. The body of the message contains the resource, and the command specifies the URL where the resource should be placed.

GET

It is used to download resources. It does not provide a message body.

POST

Passes data to modify the state of the resource specified in the request. Announcement contains the body.

DELETE

Deletes the resource associated with the URL provided in the request

This set of functionalities makes HTTP somewhat similar to FTP. However, since resources on the server are now often updated by other means, we should first note the following three methods widely used by client software:

GET

It only has a command line and headers. Recall that parameters specifying the request can be attached to the URL in the form of a query string. According to the name, it is to be used only to download resources, without causing side effects. This assumption is reflected in the way the request is handled by the server and by the client software (in particular, it is related to the possibility of caching previously downloaded results of an identical request).

POST

It is used to update the resource on the server. Parameters can be sent in the message body.
Additionally described with Content-Type headers (e.g. application/x-www-form urlencoded) and Content-Length.
The location called (URL) usually points to the handler that is the consumer of the update command.

HEAD

It works similarly to GET, but it only checks the availability of the resource: it returns an unbodyed message in response. The headers allow you to specify when the resource was last modified and the suggested amount of time that downloading clients should cache the current version.
This facility plays an important role in reducing network traffic. We will learn more about the HTTP protocol, its extensions and how developers use it in the course of the next lectures.

Current control ( ) *response comment*		
Of the HTTP request methods listed above, the following methods predict the presence of a message body:		
PUT, POST, HEAD, GET, DELETE (all listed)		Wrong answer.
GET, POST, HEAD		Incorrect answer x
PUT, POST	Yes.	
POST		Incomplete answer.
PUT		Incomplete answer.

HTTP 1.1 and further development of protocols for web applications The original shape of
the HTTP specification, formalized in version 1.0, was created in completely different realities of web use. It was created for relatively simple (perhaps even pure hypertext) pages that:

• could be downloaded with a single request; • contain
hyperlinks to other documents on the same site or • to external sites.

For this type of application, the connectionless and stateless nature of HTTP 1.0 was sufficient, even optimal. This situation was radically changed by the following factors:

- increasingly rich websites (containing related resources: CSS sheets, scripts, graphic elements, photos, etc.) - often requiring dozens of requests to download a complete document (implemented separately, it would require re-negotiating the TCP connection for each transfer);

- rapid increase in the popularity of WWW - also among smaller companies that would not need a separate dedicated machine for their WWW server; •
implementation of the application functionality via the web interface - the need to remember the context of interaction with the customer (eg the fact of logging in, setting preferences, etc.).

HTTP 1.1 met these needs by offering:

- For hosting multiple sites on the same server - additional request identification by adding the Host: header.
- For retrieving multiple document-related resources - the ability to have the TCP connection kept open for future requests. • To save on redundant transfers - additional header information
and request options to support caching of previously downloaded resources. • To streamline the transfer of dynamic documents - the ability to send in parts (the server
can send a fragment of the page while the rest of its content is still being generated).

- To maintain selected data on the course of previous interaction of a given client with the website - the mechanism of transferring the so-called protocol headers in the protocol headers. cookies.

Some of the above issues will be developed in subsequent modules. In particular, we will use mechanisms based on cookies.

<<SCREENCAST S3 – analysis of HTTP communication with a web server using a web debugger tool>>

Finally, let us note that the improvements to the HTTP protocol are not exhaustive about the desired improvements. The protocol remains relatively wasteful in terms of network traffic generated, and new applications (e.g. gaming, live streaming, remote collaboration tools, etc.) create new demands for the protocol. We will return to these issues in one of the next modules.

Ongoing audit [ ] *reply comment* Which new challenges does the HTTP		
version 1.1 specification address?		
Data compression		Incorrect answer x
Improvement of downloading pages consisting of many related resources Support	Correct answer. / Incomplete answer.	
for XML uploads Hosting		Incorrect answer x
multiple sites on a single IP address Start sending a	Correct answer. / Incomplete answer. x Correct answer. /	
response while the request is not yet fully processed	Incomplete answer.	

Task

Acquire and familiarize yourself with the basic functionality of the selected diagnostic tool for analyzing HTTP protocol messages (these tools are usually called Web-debuggers). It can be one of the components of the development tools that is an integral part of the browser, or a separate tool (e.g. https://www.telerik.com/fiddler/fiddler-classic).

Use this tool to check how many and what http calls are made when fetching the start page of a selected site.

Check how the last modification date of the downloaded resource declared in the headers behaves for HTML pages (remember that nowadays these are very often dynamic documents, i.e. generated on the web server side in response to a request), and as for other, probably less frequently changing resources (CSS sheets, graphic elements, etc.).

Use one of the publicly available sites as an example, find a document to use as an example of a conditional download of a resource (request "download if modified after <date, time>").

Show how the request sent by the client to the server is structured and attach the headers (plus information whether the body of the message has occurred or not) for the two variants:

a) When the client specifies a date earlier than the last modification date of this resource on the server;
b) When the customer specifies a later date.

To the above When solving the task, include explanations of the meaning of all protocol headers that were included in the request and both versions of the response.

If necessary - try to check the meaning of the appropriate constructions in the HTTP 1.1 protocol specification available, among others, on through the website http://www.rfc-editor.org