Изпитна тема № 18: **Мрежови протоколи и технологии**

Мрежови модели и стандарти. Отворен модел OSI. TCP/IP модел. Клиент-сървър модел. Основни мрежови термини – IP адреси, символни адреси и имена на области (домейни), портове, сокети. Комуникационни протоколи – TCP и UDP. Стандартни номера на портове и услуги в интернет. Унифициран локатор на ресурси URL. Протоколи: HTTP/1.1, HTTP/2, WebSocket, FTP, Telnet, SSH, POP, IMAP, SMTP, RPC, AMQP. Еднонишкови, многонишкови и многопроцесни Web сър-върви – предимства и недостатъци.

Дидактически материали: Компютър с подходящ софтуер, задачи и казуси.

Критерии за оценяване на изпитна тема № 18	**Максимален брой точки**
1. Демонстрира знания за мрежовите модели OSI и TCP/IP.	6
2. Дефинира понятието капсулация на данните при модел OSI. Дава пример за капсулация на данни.	6
3. Обяснява предназначението на сокетите и портовете. Демонстрира чрез при-мер как работят сокетите и портовете.	10
4.Дефинира термините: IP адрес, домейн, URL, DNS.	2
5. Обяснява модела клиент-сървър.	4
6. Обяснява и дава пример за използване на UDP. Различава TCP от UDP.	12
7. Описва фазите на комуникация при TCP. Демонстрира чрез програмен код прилагането на TCP.	8
8. Различава HTTP/1.1 от HTTP/2. Различава HTTP от WebSocket.	8
9. Разработва програмен код на клиент и/или сървър чрез FTP и/или SMTP.	6
10. Демонстрира знания за предимствата и недостатъците на еднонишковите и многонишковите Web сървъри.	6
11. Открива грешки в програмен код и го модифицира, така че да реши поставе-ната задача.	8
12. Анализира, определя и допълва програмен код, така че да реши поставената задача.	24
ОБЩ БРОЙ ТОЧКИ:	**100**

1. Демонстрира знания за мрежовите модели OSI и TCP/IP. (6 т.):

Моделът OSI (Open System Interconnect) - абстрактен модел, който описва начина на комуникация в компютърните мрежи. Има 7 слоя, всеки от които е една стъпка в процеса на комуникация.

Приложен слой - служи като посредник между софтуерните приложения и мрежовите услуги. В този слой работят протоколите HTTP, FTP, Telnet, SMTP, POP3, IMAP4, SNMP. Задачата на слоя е да управлява общия мрежов достъп, контрола на потоците от данни и поправката на грешки.

Представителен слой - определя използвания формат за обмен на данните. Тук получените от приложния слой данни се представят във вид на пакети („универсален" формат за пренос). Този слой отговаря за преобразуването на данните:
· компресиране – намаляване на техния размер;
· криптиране – кодиране с цел защита от неоторизиран достъп;
· транслация на протоколи – с цел пренасяне между различни хардуерни платформи и операционни системи.

Сесиен слой - отговаря за изграждане на канал за връзка – сесия – между два компютъра в мрежата.

Транспортен слой - отговаря за транспортирането на пакетите с данни без грешки, в точна последователност и без загуби. В този слой работят транспортните протоколи TCP, UDP от TCP/IP и услугата за преобразуване на имена – Domain Name System (DNS).

Мрежови слой - отговаря за адресирането на съобщенията и за определянето на маршрут, по който да преминат данните от компютъра – източник до компютъра – получател. СВ този слой работи протоколът IP от TCP/IP , а като устройства- маршрутизаторите.

Канален слой - изпраща кадрите с данни от мрежовия слой към физическия слой. Има два подслоя:

Контрол за достъп до преносната среда – Media Access Control (MAC) - разпределя достъпа на компютрите до физическата преносна среда. Той дефинира MAC адресите.

Контрол на логическите връзки – Logical Link Control (LLC). Дефинира логическата топология.

Физически слой - предава потока от битове (единици и нули) от мрежовата карта към преносната среда. Битовете са кодирани като електрически или светлинни импулси (при безжичните системи са електромагнитни вълни). Този слой определя типа на връзката между мрежовата карта и кабела, както и

техниката на предаване на информацията по мрежата.Устройствата, които работят на това ниво са мрежови карти, повторители, хъбове, медиа конвертори.

Моделът DoD , известен с наименованието TCP/IP модел.

Състои се от четири слоя:

Слой 4. ***Приложен (application layer)*** – най-горният слой от модела. Обхваща функциите на трите най-горни слоя на OSI модела;

Слой 3. ***Транспортен (хост до хост) (host to host (transport) layer)*** – съответства на транспортния слой на OSI модела. Тук работят TCP, UDP, DNS;

Слой 2. ***Слой интер-мрежа (internetworking layer)*** – съответства на мрежовия слой на OSI. Занимава се с маршрутизацията основана на логическите IP адреси. Протоколът Address Resolution Protocol (ARP) преобразува логическите IP адреси в MAC адреси;

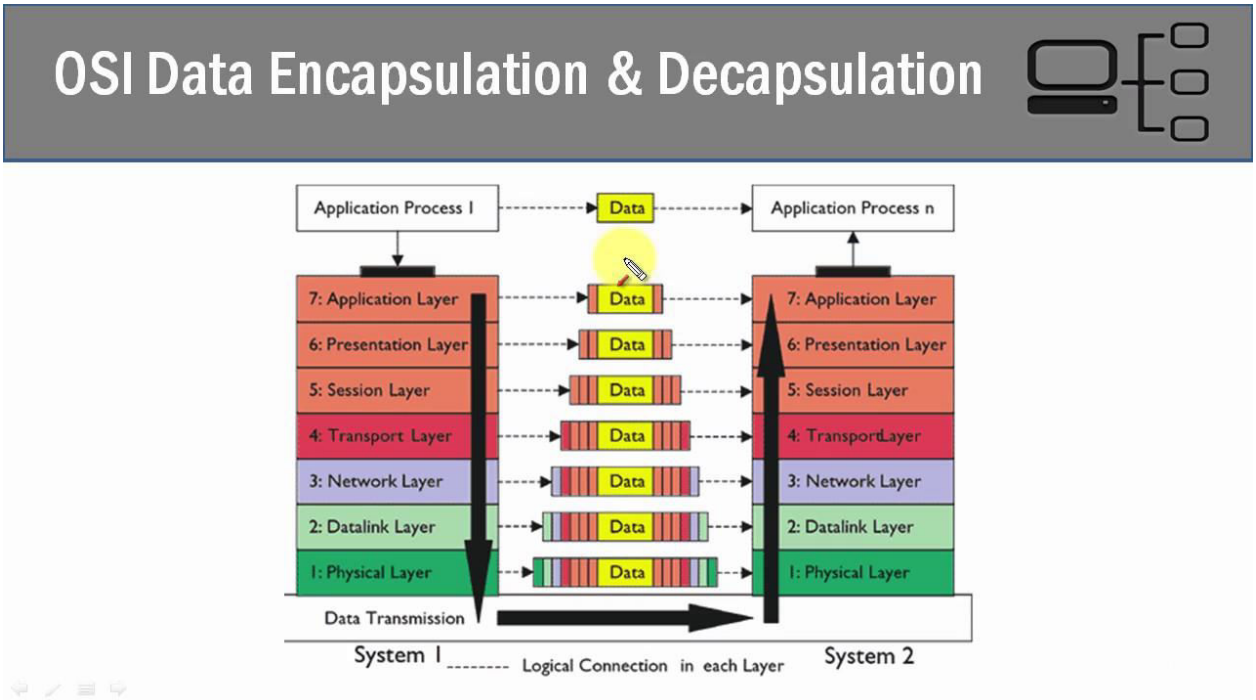
Слой 1. ***Мрежов интерфейс (network interface layer)*** – съответства на двата слоя – канален и физически на модела OSI. Тук работят Ethernet и Token Ring протоколите. В този слой се използват само MAC адреси.

2. Дефинира понятието капсулация на данните при модел OSI. Дава пример за капсулация на данни. (6т.)

Капсулирането е процес на опаковане на данни. Когато хост изпрати данни на друго устройство, те ще бъдат обвити с информация за протокола на всеки слой от OSI (Open System Interconnection) модела.

За да взаимодействат, слоевете използват PDU (Protocol Data Unit), който съхранява контролна информация, която след това се добавя към данните във всеки слой. Данните се добавят в т.н ***header***, могат да се поместят и в края на пакета. Капсулирането се случва, когато протоколът, който е в най-ниският слой и получава данни от протокола на по-високия слой и поставя данните във формат, който се разбира от протокола. Тоест даден слой не трябва да знае броя на другите слоеве, които са над или под него. Те вършат само съответната работа.

При подателя задачата е да се получават данни от най-горния слой, след което да ги обработва според функциите на протокола. На следващо място, просто се добавя *header* и се продължава към слоя под него.



Пример :

Намираме се на web сайт на някаква фирма и с мишката сме кликнали на връзка в страницата. По този начин сме създали заявка към web сървъра да изпрати html документа на връзката. Този документ ще бъде изпратен към компютъра и интерпретиран от браузъра

Данни за двата компютъра (web сървъра и компютъра на клиента)

PC	IP address	MAC address	№ port
Web server	192.168.5.1	5-5-5-5-5-5	TCP 80
PC1	192.168.5.2	6-6-6-6-6-6	TCP 4888

Web сървърът ще създаде пакета, съдържащ web страницата, използвайки следните стъпки:

- Слой 7 получава данни под формата на html документа от приложението
- Слой 6 добавя информация за форматирането
- Слой 5 добавя информация, необходима за създаването на сесия между web сървъра и web браузъра на PC1
- Слой 4 добавя транспортния протокол и номера на изходния и целеви порт – целевия порт ще е TCP port 80 а изходния е TCP port 49152

- Слой 3 добавя изходния и целевия адрес – изходния е 192.168.5.1 , целевия е 192.168.5.2
- Слой 2 научава целевия MAC адрес и добавя изходния и целевия MAC адрес – изходния е 5-5-5-5-5-5 , целевия е 6-6-6-6-6-6
- Слой 1 преобразува целия пакет в битове и го изпраща по мрежата към PC1

PC 1 ще приеме пакета и ще го обработи по следния начин:

- Слой 1 приема битовете в електрическа форма , обработва ги и ги подава на слой 2
- Слой 2 разглежда целевия MAC адрес(установява, че това е неговия собствен MAC адрес) и предава останалите данни към горния слой
- Слой 3 разглежда целевия IP адрес, проверява дали е неговия собствен и предава останалата част на пакета към следващия слой
- Слой 4 разглежда изходния порт и предупреждава браузъра, че идват http данни, пропуска тази част от пакета и предава останалата част на горния слой
- Слой 5 използва информацията, която е създадена от web сървъра, за да създаде сесия между web сървъра и браузъра и предава останалата част от пакета на горния слой
- Слой 6 изпълнява всички преобразувания и предава останалата част от пакета на слой 7
- Слой 7 приема HTML документа и го подава към приложението (в случая web браузъра) и отваря документа в прозореца

3. **Обяснява предназначението на сокетите и портовете. Демонстрира чрез пример как работят сокетите и портовете. (10т.)**

Портът в компютърните мрежи е крайната точка на комуникация в операционната система. Портът винаги е свързан с IP адреса на хоста или вида на комуникационния протокол. Той завършва възлагането на адреса на сесията. Портът се идентифицира за всеки протокол и адрес, като се използва 16-битов номер, известен също като номер на порт. Често се използват конкретни номера на портове за идентифициране на конкретни услуги.

Множество приложения на един компютър могат едновременно да пращат и получават данни през TCP/UDP - портът служи за да може да се различават пакетите за дадено приложение от тези за друго в рамките на една компютърна система. Едно приложение може да ползва повече от един порт за изпращане и получаване на данни

Тъй като за номера на порт в хедърите на протоколите е заделено 16-битово поле, възможните му стойности са 0 – 65535.

● от 0 до 1023 – ***добре известни портове (well known ports)***. Към тези номера са назначени често използвани услуги като уеб (HTTP), FTP, IRC и др.

● от 1024 до 49151 – регистрирани портове (registered ports) IANA назначава тези портове за ползване от частни програми

● от 49152 до 65535 – динамични (или частни)портове (dynamic ports). Известни и като Ephemeral (краткотрайни) портове. Обикновено се присвояват динамично на клиентски приложения, иницииращи сесия. Услугите не използват такива портове (с изключение на peer-to-peer file sharing програми).

Протокол	№ на порт	Тип в
FTP	20, 21	File transfer
FTPS	989,990	File transfer (secure)
SFTP	22	File transfer (secure)
SSH	22	Remote login (secure)
HTTP	80	Web
HTTPS	443	Web (secure)
DNS	53	Find IP address
PoP3	110	IMAP mailbox
SMTP	25	Internet mail
Telnet	23	23 Remote login
NNTP	119	Usenet newsgroups
NNTPS	563	Usenet (secure)

В контекста на компютърната мрежа, сокетът е крайна точка на двупосочна комуникация, която се осъществява в мрежа, която се основава на интернет протокола. ***Сокетът се състои от IP адреса на системата и номера на порта на програма в системата.*** IP адресът съответства на системата, а номерът на порта съответства на програмата, където данните трябва да бъдат изпратени:

Сокетите могат да бъдат класифицирани в три категории: stream(поток), datagram(дейтаграма), and raw socket(суров soket). Поточните сокети използват ориентирана към връзката мрежова точка за изпращане и получаване на данни. Този тип сокети обикновено използва TCP, за да позволи на процесите да комуникират един с друг. Сокетите за дейтаграми използват мрежови протоколи без връзка като UDP, за да позволят комуникация на процеса. Суровите сокети са ориентирани към дейтаграми и позволяват на процесите да използват ICMP за комуникационни цели.

4. **Дефинира термините: IP адрес, домейн, URL, DNS. (2т.)**
   - **IP (Internet Protocol**) е протокол за комуникация. Предназначението му е да позволи адресация на информацията, която се изпраща по мрежата. На всеки хост в мрежата се дава уникален адрес. Когато се изпраща информация през мрежата, тя се разделя на малки пакети, наречени IP пакети. Към всеки пакет се прикрепя т. нар. хедър, който съдържа IP адреса на подателя и получателя и други служебни данни. С помощта на тези адреси компютрите, през които минава пакетът, решават какво да правят с него. IP адресите са два вида. IPv4 адрес - използва 32 битово адресиране. 32 битовият адрес се записва като поредица от четири 8 битови числа (октети). Всеки един от тези октети може лесно да се преобразува в десетично число. IPv6 адрес - използва 128 битов адрес. Представя се като осем 16 битови двоични числа. 16 битовите числа се записват в шестнадесетична бройна система и се разделят помежду си с двоеточия.
   - **Домейн –** част от йерархичното пространство на глобалната мрежа, което има собствено уникално име (име на домейн). За да бъде заредена web страница, домейна е името,което се изписва в полето на браузъра. Целта на домейн имената е да се означат ресурси или услуги в онлайн пространството, които имат собствено уникално име, но трябва също да отговарят на определени изисквания. Зад името на домейна abv.bg стои IP адреса 194.153.145.104, който е труден за запомняне. Затова всеки уеб сървър изисква сървър на DNS (Domain Name System), за да превежда имената на домейни в IP адреси. Разширенията като .com се наричат домейн от първо ниво. Други примери са: .org, .net, .bg и тн.
   - **URL - Uniform Resource Locator** . Структурите на URL адресите се състоят от три части:

     1. наименование на протокола
     2. име на хост или адрес
     3. файл или място на ресурса

   пример :

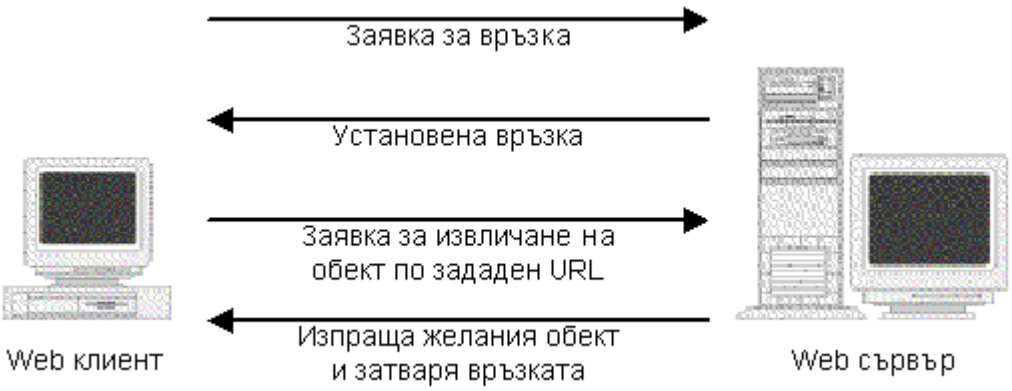
   https://security.googleblog.com/2018/01/todays-cpu-vulnerability-what-you-need.html

Където: https е протоколът (като FTP е протокол), който определя вида на сървъра, с който комуникирате.**security** -
името на хоста, използвано за достъп до този уеб сайт.**googleblog** е името на домейна.**com** е това, което се нарича домейн от първо ниво (TDL) **/ 2018/01 /** представлява директории, използвани за организиране на уеб страница или файл. **Tadays-cpu-vulnerability-what-you-need.html** е действителният файл, към който URL-ът сочи.

- **DNS** – система за преобразуване на логическите имена в IP адреси . DNS превежда имената на домейни в IP адреси, така че браузърите да могат да те приземят на правилната страница – превежда буквите в цифри.Всяко устройство, свързано към Интернет, има уникален IP адрес, който други машини използват, за да се свържат с него. DNS сървърите премахват необходимостта хората да запомнят IP адреси като 192.168.1.1 (IPv4) или по-сложни буквено-цифрови IP адреси като 2400: cb00: 2048: 1 :: c629: d7a2 (в IPv6).

5. Обяснява модела клиент-сървър. (4т)

Клиент-сървър - основен модел за обмен на информация в Интернет. Това е тип мрежова архитектура, която отделя клиента от сървъра и най-често се използва в компютърни мрежи. WWW е Интернет услуга, в основата на която стои също модела клиент/сървър. Комуникацията между Web клиента и Web сървъра се осъществява чрез използване на протокола HTTP. HTTP (Hyper Text Transfer Protocol) служи за обмен на документи между сървър и клиент, и е част от протоколния стек TCP/IP за управление на поток от данни в Интернет. Всъщност протокола HTTP функционира на базата на проста схема от тип “въпрос–отговор". Клиентът изпраща заявка към сървъра, на която сървърът отговаря.



Освен при HTTP подобна схема на комуникация се прилага и при други протоколи – например FTP (File Transfer Protocol). Web сървърът изпраща поисканите от клиентите заявки и файлове. Разликата между това дали ще използваме FTP или HTTP сървър се състои в по–богатата функционалност на HTTP сървъра. Ако даден сайт с Web страници е изграден под форма на хипермедийни документи, то освен текста има асоциирани с него графични, звукови или видео компоненти. Тогава, при условие, че клиент е отправил заявка към сайта, като резултат ще му бъдат изпратени всички елементи на документа, т.е в документа-резултат ще влязат всички съставни компоненти. Освен това според естеството на заявката и средствата за нейната обработка, от сървъра към клиента може да се изпращат и генерирани динамично данни.

6. Обяснява и дава пример за използване на UDP. Различава TCP от UDP. (12т.)

UDP (User Datagram Protocol) - осигурява негарантирана доставка на данни. UDP осъществява изпращането на данните и не се занимава с проверка на получаването им. Съобщението, което се предава се нарича дейтаграма (datagram). UDP осигурява единствено контролна сума, гарантираща целостта на данните.
Програмите, използващи UDP протокола трябва да реализират самостоятелно:

- препредаване на загубени дейтаграми;
- игнориране на повторения;
- фрагментиране и обединяване на големи потоци данни.

В локалните мрежи грешките при предаване на информация са пренебрежимо малки. Използването на UDP протокол ще генерира по-малък трафик. Обикновено, при избора на UDP протокол се търси не намаляване на трафика, а по-скоро намаляване на времето за доставка и отговор. Например, при използване на UDP при предаване на глас, загубата на

сегмент ще доведе до дефект – пукане. Използването на TCP при същото приложение ще доведе до забавяне – накъсване на звука, сериозно забавяне времето за отговор.

UDP се използва при пренасяне на глас по Интернет (VoIP), интернет радио, реално видео и аудио. Услугите multicast и broadcast могат да се реализират единствено чрез UDP.

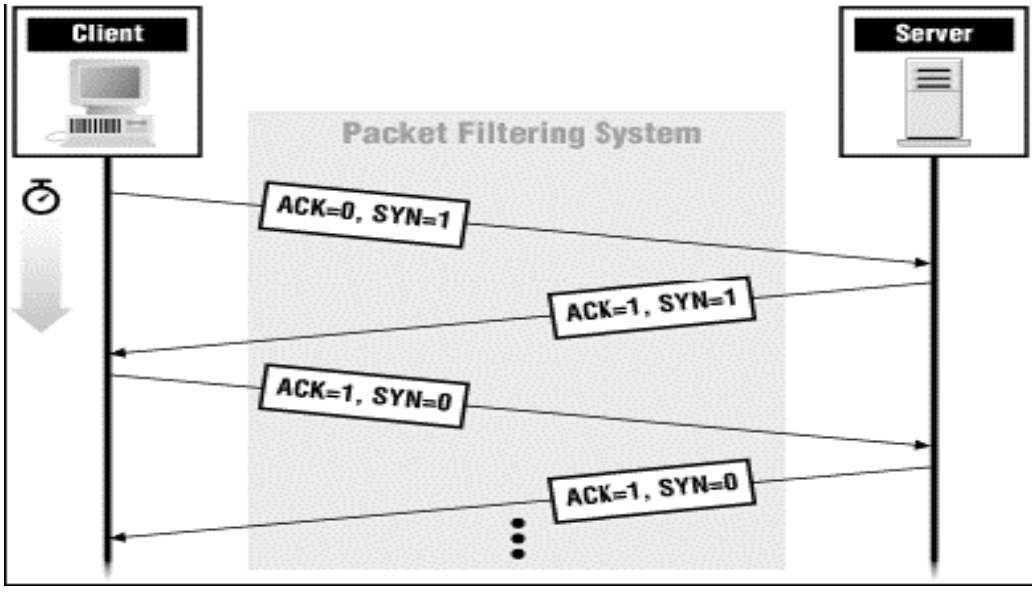
TCP (Transmission Control Protocol) - осигурява зависим от връзката трафик между две устройства в мрежата. Когато се изпраща съобщение, между компютъра, който изпраща и компютъра получател трябва да се изгради сесия (връзка). Съобщението, което се изпраща се разделя на сегменти, като всеки сегмент има пореден номер. При достигането на сегментите при получателя, те се подреждат по тези номера, за да се получи оригиналното съобщение. Ако някой от сегментите не е пристигнал се изисква неговото повторно изпращане. Максималният брой препредавания на сегмент се определят от параметъра TcpMaxDataRetransmissions (стойност по подразбиране 5 пъти). Протоколът TCP осигурява гарантирано достигане на информацията до получателя. За да направи това, освен необходимите за изпращане данни се включва допълнителна служебна ‘статус’ информация. Този статус включва съобщения за потвърждаване на получаването (ACK), рестартиране – изчистване на буферите (RST), синхронизация на последователността (SYN), край на сесия (FIN) и др. Потвърждаването става чрез изпращане на сегмент статус ACK съдържащ следващия поредния номер.
Потокът информация получена от приложението се разделя на сегменти в зависимост от параметъра MSS (Maximum Segment Size). Сегментите се поставят в буфер – наречен прозорец (с размер – параметър TCPWindowSize). Изпращачът изпраща всички сегменти съхраняващи се в прозореца. За всеки сегмент се съхранява брояч RTO (retransmission timeout). Изпращачът препредава сегмент, ако не получи потвърждение за RTO секунди. Това време се определя динамично и зависи от качеството на връзката. При изграждането на сесията стойността RTO е 3 секунди. При успешно предаване на сегменти последователно, RTO се намалява. При всяко препредаване на сегмент RTO се удвоява.

7. **Описва фазите на комуникация при TCP. Демонстрира чрез програмен код прилагането на TCP. (8т.)**

TCP гарантира доставянето на пакети. TCP е по-бавен от UDP. Когато TCP приеме поток от данни (съобщения), той разбива потока на сегменти и задава поредни номера на всеки байт в дейтаграмата, преди да започне доставяне с помощта на IP. Тези поредни номера изискват да бъдат връщани със сответните потвърждения от местоназначението, за да се гарантира, че то е приело от изпращащата страна всеки сегмент от дейтаграмата.. Ако не получи потвърждение, протоколът приема, че дейтаграмата е изгубена и я предава отново.

За да осигури надеждно доставяне на данни между процеси, TCP трябва първо да изгради връзка. TCP първо изгражда връзка между портовете на отдалечените хостове и я поддържа по време на целия разговор и я унищожава (разпада) когато вече не е нужна.При установяване на връзката се резервират ресурси за времето, в което трае тази връзка. По време на фазата по предаване на данните, те се предават по вече установения път като в отсрещната страна данните пристигат в реда, в който са изпратени. Резервираните ресурси се освобождават, когато връзката вече не е необходима и тя се прекъсва.

Хостове, които използват TCP протокол на транспортен слой, изграждат връзка помежду си преди да започнат предаването на необходимата им информация. Установяването на тази връзка става по метод, наречен **Tree-way Handshake**. При него става синхронизиране на връзката в двата края преди предаването на информацията. Хостовете си обменят пакети, съдържащи началният номер на последователността от пакети, които ще предават помежду си с цел следене на това дали цялата информация, която изпраща единият хост към другия и обратно се получава в отсрещната страна.



1. Клиентът изпраща пакет с вдигнат SYN флаг, което от сървъра се възприема като начало на инициализиране на TCP сесия.
2. Втората стъпка от установяването на сесията настъпва когато от страна на сървъра към клиента също се изпраща пакет с вдигнат SYN флаг. Освен него в пакета се съдържа и вдигнат ACK флаг, който е в отговор на получения от сървъра SYN.
3. Третата стъпка по установяването на сесията е когато клиентът потвърди получения от сървъра пакет с вдигнат SYN флаг, чрез изпращането на пакет с вдигнат ACK флаг.
4. Сървърът отговаря със същия пакет и връзката е установена.

Създаване и свързване на TcpClient

За създаването на обект от класа **TcpClient** можем да подходим по два начина – да създадем несвързан клиент, който после да свържем чрез метода Connect(…), или да създадем клиент, който още при инициализирането си да опита да се свърже с дадения сървър.

Несвързан клиент създаваме чрез конструктора **TcpClient()**. Той инициализира TCP сокет, който се обвързва с локалния мрежов интерфейс и със случайно избран от операционната система свободен порт. За да образуваме истинска TCP връзка, трябва да зададем и втория сокет, който може да бъде както на отдалечена машина, така и на локалната.

Задаването на другия край на връзката правим чрез метода **Connect(address, port)**. Методът Connect(…) се опитва да осъществи връзка към указаната комбинация от адрес и порт. Ако това стане успешно, TcpClient обектът минава в свързано състояние и по него вече могат да се предават данни. В противен случай се хвърля SocketException с описание на грешката. Затова ограждаме извикването към метода Connect(…) в try-catch-finally блок, като във finally частта да затваряме сокета чрез метода **Close()**.

Методът Close() затваря създадения сокет и извършва действията по освобождаването на ресурсите, свързани с него.

```
TcpClient client = new TcpClient();
try
{
  client.Connect("www.abv.bg", 80);
}
catch (SocketException se)
{
```

```
  Console.WriteLine("Could not connect: {0}", se.Message);
}
finally
{
  client.Close();
}
```

Променливата client декларираме извън try блока, за да можем да я използваме и след свързването.

8. Различава HTTP/1.1 от HTTP/2. Различава HTTP от WebSocket. (8т.)

HTTP 1.1 е следващата версия на **Hypertext Transfer Protocol (HTTP).** Той сигурява по-бърза доставка на уеб страници от оригиналния HTTP и намалява уеб трафика.

HTTP 1.1 прави информационния поток по-бърз: Вместо отваряне и затваряне на връзка за всяка заявка за приложение, HTTP 1.1 осигурява постоянна връзка, която позволява множество заявки да бъдат групирани или пренасочени към изходен буфер. Подлежащият слой на протокола за управление на предаването може да постави множество заявки (и отговори на заявки) ***в един TCP сегмент***, който се препраща към слоя на интернет протокола за предаване на пакети. Тъй като броят на заявките за свързване и прекъсване за поредица от заявки за получаване на файл е намален, по-малко пакети трябва да преминават през Интернет. Тъй като заявките са конвейерни, TCP сегментите са по-ефективни. Общият резултат е по-малко интернет трафик и по-бърза производителност за потребителя. Когато браузър, поддържащ HTTP 1.1, укаже, че може да декомпресира HTML файлове, сървърът ще ги компресира за транспортиране през Интернет, осигурявайки значителни сумарни спестявания на количеството данни, които трябва да бъдат предадени.

Освен това HTTP 1.1 предоставя и възможността множество имена на домейни да споделят един и същ интернет адрес (IP адрес). Това опростява обработката за уеб сървъри, които хостват редица уеб сайтове в това, което понякога се нарича виртуален хостинг.

HTTP/2 е последната версия на Протокол HTTP, която се използва от браузърите за комуникация с уеб сървъри. При него има по-бързо доставяне на съдържание и подобрено потребителско изживяване, включително:

Двоични протоколи – Бинарните протоколи консумират по-малко честотна лента, анализират се по-ефективно и са по-малко податливи на грешки от

текстовите протоколи, използвани от HTTP/1.1. Освен това те могат по-добре да се справят с елементи като празно пространство, главни букви и завършвания на редове.
Мултиплексиране – HTTP/2 може да инициира множество заявки паралелно през една TCP връзка. В резултат на това уеб страниците, съдържащи няколко елемента, се доставят през една TCP връзка. Тези възможности решават проблема с блокирането на хедъра в HTTP/1.1, при който пакет в предната част на линията блокира предаването на други.
Компресиране на заглавки – HTTP/2 използва компресия на заглавки, за да намали излишните разходи, причинени от механизма за бавно стартиране на TCP.
Натискане на сървър – HTTP/2 сървърите избутват вероятно използвани ресурси в кеша на браузъра, дори преди да бъдат поискани. Това позволява на браузърите да показват съдържание без допълнителни цикли на заявка.
Повишена сигурност – Уеб браузърите поддържат само HTTP/2 чрез криптирани връзки, повишавайки сигурността на потребителите и приложенията.

WebSocket е двупосочен комуникационен протокол. Той осигурява пълна дуплексна (двупосочна) връзка. Той е различен от HTTP. И двата протокола са разположени на слой 7 в OSI модела и зависят от TCP на слой 4. Въпреки че са различни, WebSocket "е проектиран да работи през HTTP портове 443 и 80, както и да поддържа HTTP прокси сървъри и посредници, ", което го прави съвместим с HTTP.
HTTP обменя информация, като отваря нова връзка във всеки цикъл на отговор на заявка. Затова всеки път, когато се изпраща заявка или се получава отговор, HTTP установява нова връзка. „Връзка“ е просто сигурна линия за прехвърляне на информация. Веднага след като информацията бъде доставена, HTTP прекратява връзката.

WebSocket поддържа връзката отворена, така че може да се обменя информация по същия ред, докато някой не прекрати връзката.

9. Разработва програмен код на клиент и/или сървър чрез FTP и/или SMTP.(6т)

Програма за изтегляне на файл

```
<a href="/images/exampleimage.jpg" download="example">
  <img src="/images/exampleimage.jpg" alt="example">
</a>
```

Атрибутът за **download** се използва само ако атрибутът href е зададен.

Стойността на атрибута е името на изтегления файл. Няма ограничения за разрешените стойности и браузърът автоматично ще открие правилното файлово разширение и ще го добави към файла (.img, .pdf, .txt, .html и др.).

Можете също така да се посочи стойност за атрибута за изтегляне, който ще бъде новото име на изтегления файл. Ако стойността е пропусната, се използва оригиналното име на файл.

10. Демонстрира знания за предимствата и недостатъците на еднонишковите и многонишковите Web сървъри.(6т)

С постъпване на HTTP заявка, Web сървърът започва да търси заявеният от клиента ресурс. Докато сървърът е зает, същите или други клиенти могат да изпратят нови заявки. Web сървърът може да пренебрегне или да обработи успоредно тези заявки. Web сървърите, които пренебрегват или нареждат на опашка получените нови заявки, се наричат ***еднонишкови***. Това означава, че те не могат да се справят с натоварения при Web сървъра трафик. Тези сървъри обаче са много добри за сайтове, които имат слабо натоварен или умерен трафик, защото осигуряват много добра скорост на обработка на заявките. Примери за еднонишкови Web сървъри са **thttpd, Medusa и Zeus**.

Web сървърите, които обработват едновременно новопостъпилите заявки, изпълняват тази задача по два начина. Те могат да стартират нов процес или нова нишка на първоначалния процес. Сървърите, които започват нов процес за всяка нова заявка, се наричат ***многопроцесни***, докато тези които стартират нова нишка – ***многонишкови*** Web сървъри.

IIS (Internet Information Services) на Microsoft е пример за многонишков Web сървър. Сървърът **Apache** за платформата UNIX е типичен пример за многопроцесен Web сървър. Поради ограниченията на операционната система Windows (липса на поддържка за разклоняване), Apache работи в многонишков режим, когато е инсталиран за Windows.

Apache Web Server е най-използваният в момента Web сървър. Той, подобно на операционната система Linux, скриптовия език PHP и сървърът за бази данни MySQL (които са често използвана комбинация от програмни средства във Web програмирането), е с отворен код. Apache може да работи с външни модули и всеки който има познания може да създаде код който да увеличи

функционалността му. Някои от предимствата му са - стабилност, бързина, лесно добавяне на допълнителни възможности, възможност за лесно преконфигуриране и не на последно място - той е безплатен.

Apache може да се ползва от почти всички компютърни платформи. В началото Apache е бил свързан най-често с Unix. Apache не само се ползва с повечето, ако не всички варианти на Unix, но също така и с Linux, Windows и много други сървърни операционни системи. Apache работи най-добре с Unix и Linux, но и версиите за работа с Windows са стабилни и сигурни. Apache предлага много възможности, включително индексиране, псевдоними, управление на дъщерни процеси, докладване за HTTP грешки, сървър ориентирани карти, онлайн наръчници и др.

Web сървър IIS във Windows

IIS осигурява интегрирани, надеждни и сигурни възможности на Web сървър за Intranet или Internet. IIS е стабилна и сигурна платформа за изпълнение и на динамични мрежови приложения. Той осигурява сигурен хост за Web сайтове в Internet, хост и управление на FTP сайтове и обслужване на Web news или E-mail чрез използване на Network News Transfer Protocol (NNTP) и Simple Mail Transfer Protocol (SMTP). IIS работи с ASP.NET Core (ASP.NET Core framework е най-новото поколение Active Server Page (ASP), сървърна скриптова машина, която създава интерактивни уеб страници)

IIS се развива заедно с Microsoft Windows.

IIS 1.0	Windows NT
IIS 4.0	Windows NT 4.0
IIS 5.0	Windows 2000
IIS 6.0	Windows Server 2003
IIS 7.0	основен редизайн с Windows Server 2008
IIS 7.5	Windows Server 2008 R2
IIS 8.0	Windows Server 2012 (Windows Server 2012 R2 използва IIS 8.5)
IIS 10	Windows Server 2016 и Windows 10

С всяка итерация на IIS Microsoft добавя нови функции и актуализира съществуващата функционалност. Например, IIS 3.0 добавя ASP за динамични скриптове; IIS 6.0 добавя поддръжка за IPv6 и подобрена сигурност и надеждност; и IIS 8.0 внедрява многоядрено мащабиране на хардуер за нееднороден достъп до паметта, централизирана поддръжка на SSL сертификати и индикация за име на сървър.

IIS 10 също добавя редица нови характеристики и функционалност:

- IIS 10.0 поддържа протокол HTTP/2, което позволява множество подобрения спрямо HTTP 1.1 и води до ефективно повторно използване на връзките и намаляване на забавянето. Поддръжката на HTTP/2 е добавена към Windows Server 2016 и Windows 10 .
- IIS 10 работи на модела за минимално разгръщане на сървър Nano Server под Windows Server 2016 и може да изпълнява работни натоварвания на ASP.NET Core, Apache Tomcat и PHP на IIS на Nano Server.
- IIS 10 работи в контейнер и виртуална машина, така че разработчиците и администраторите имат по-голяма гъвкавост при избора на внедряване, както и плътността за настаняване на широк набор от уеб приложения.

11. Открива грешки в програмен код и го модифицира, така че да реши поставената задача.(8т)

12. Анализира, определя и допълва програмен код, така че да реши поставената задача.(24т)

11. Открийте грешката в кода и го модифицирайте, така че да запазва файла в папка **myFolder**

```
<?php
  $folder = “myFolder”
  if($_FILES['fileInput_1']['tmp_name']!="") {
    $temp = explode(".", $_FILES["fileInput_1"]["name"]);
    // will be changed by the time and user's username
    $filename1 = round(microtime(true)) . '_1' . $_COOKIE["user_id"] . '.' . end($temp);
    //declaring variables
    $filetmpname = $_FILES['fileInput_1']['tmp_name'];
    //folder where images will be uploaded
    //function for saving the uploaded images in a specific folder
    move_uploaded_file($filetmpname, $folder.$filename1);
    //inserting image details (in eg. image name) in the database
  }
```

?>

12. Обяснете дадения код и го допълнете така, че потребителят да получи **e-mail** от сървъра и да се изпише съобщение за невалиден **e-mail** и неуспешно изпращане

```
<?php
   $email = "pgtk@gmail.com";
   if($email != ""){
      $receiver = $email;
      $subject = "Forgotten password";
      $body = "To change your password, click the link: http://" . $_SERVER['HTTP_HOST'] .
"/forgotPassword";


      if(filter_var($email, FILTER_VALIDATE_EMAIL)){
         if(mail($receiver, $subject, $body, $sender)){
            echo '
               <script>
                  alert("Email is valid and it was send successfully!");
               </script>
            ';
         }
         else{
            echo '
               <script>
                  alert("Email is invalid and it wasn’t send successfully!");
               </script>
```

```
            ';
        }
    }
?>
```