Тема 18 - Мрежови протоколи и технологии

1. **Демонстрира знания за мрежовите модели OSI и TCP/IP**

Моделът OSI се състои от седем слоя, като всеки слой изпълнява определени функции, които са задължителни за успешна комуникация.

- Слой 1 (физически слой) се отнася до физическите аспекти на комуникацията, като кабелите, свързващите устройства и физическите характеристики на трансфера на данни.
- Слой 2 (дата-линк слой) осигурява надеждна комуникация между устройствата в мрежата, използвайки протоколи като Ethernet.
- Слой 3 (мрежов слой) осигурява маршрутизация на данни между различни мрежи, като използва протоколи като IP.
- Слой 4 (транспортен слой) осигурява пренос на данни между приложенията на двата края на комуникацията, като използва протоколи като TCP и UDP.
- Слой 5 (сесиен слой) осигурява сесия между две устройства, като управлява начина, по който комуникацията се установява и поддържа.
- Слой 6 (представителен слой) осигурява кодиране и декодиране на данните, като ги прави разбираеми за приложенията.
- Слой 7 (приложен слой) е най-високият слой в модела OSI и съдържа приложенията, които използват мрежовата комуникация.

Моделът TCP/IP се състои от четири слоя.

- Слой 1 (мрежов интерфейсен слой) отговаря за връзката с физическата мрежа.
- Слой 2 (интернет протоколен слой) осигурява маршрутизацията на данните в мрежата, като използва протокола IP.
- Слой 3 (транспортен слой) отговаря за установяване на канал за комуникацията между два устройства, като използва протоколите TCP и UDP, като TCP осигурява надеждна доставка на данни, докато UDP осигурява по-бърза, но по-малко надеждна доставка.
- Слой 4 (приложен слой) съдържа приложенията, които използват мрежовата комуникация, като например HTTP, FTP, SMTP и други.

Една от главните разлики между моделите OSI и TCP/IP е, че моделът OSI е теоретичен модел, докато TCP/IP е практически модел, който се използва в реалния свят. Освен това, моделът OSI има по-голям брой слоеве, като всяка функция е разделена на отделен слой, докато TCP/IP има по-малък брой слоеве и функциите са по-комбинирани.

Независимо от това, и двата модела са полезни за разработване на мрежи и за разбиране на начина, по който мрежовата комуникация функционира.

МБ

2. **Дефинира понятието капсулация на данните при модел OSI. Дава пример за капсулация на данни.**

Капсулацията на данните при модел OSI е процесът на добавяне на заглавия и опции, за да се формира пакет на данни, който да може да се изпрати през мрежата. При този процес, данните на всяко ниво на модела се обработват в рамките на този слой и се добавят хедъри и тейлъри, които съдържат информация за начина на обработка на данните на следващото ниво.

Пример за капсулация на данни може да бъде следният процес на изпращане на данни през мрежата:

- Приложението, което иска да изпрати данни, поставя данните в полезната част на съобщението и ги предава на сесионния слой.
- Сесионният слой добавя своя хедър към данните, който съдържа информация за началото и края на сесията и ги предава на транспортния слой.
- Транспортният слой добавя своя хедър към данните, който съдържа информация за началото и края на транспортната връзка, както и контролни битове, които сигнализират за грешки при трансфера на данните. Транспортният слой предава пакета на мрежовия слой.
- Мрежовият слой добавя своя хедър към данните, който съдържа информация за маршрутизацията на данните, включително IP адресите на източника и получателя, както и други данни за рутирането. Мрежовият слой предава пакета на дата-линк слоя.
- Дата-линк слоя добавя своя хедър и тейлър към данните, който съдържа информация за адресите на мрежовите устройства, както и контролни битове, които сигнализират за грешки при трансфера на данните. Дата-линк слоя предава пакета на физическия слой.
- Физическият слой превръща пакета в битове, които могат да бъдат изпратени през мрежата.

3. **Обяснява предназначението на сокетите и портовете. Демонстрира чрез пример как работят сокетите и портовете.**

a) Сокетите представляват програмен интерфейс, който позволява на приложенията да се свързват с мрежата и да изпращат и получават данни. Сокетите осигуряват абстрактен начин за обмен на данни между различни компютри и могат да се използват за различни видове комуникации - TCP, UDP, HTTP, FTP, SMTP и други.

b) Портовете са номера, които се използват за идентифициране на конкретно приложение в мрежата. Те позволяват на мрежовата инфраструктура да знае къде да изпрати данните, като по този начин осигуряват надеждна комуникация между различни приложения. Всяко приложение може да има свой уникален порт, който му позволява да се свърже с други приложения в мрежата.

За да демонстрираме работата на сокетите и портовете, можем да разгледаме пример със сървър-клиент приложение. Представете си, че имаме едно приложение, което работи като сървър и друго, което работи като клиент.

I. Сървърът трябва да отвори един порт, за да приема входящи връзки от клиенти. Той създава сокет и му задава съответния порт, на който да слуша входящи връзки. Например, можем да отворим порт 8080 за HTTP заявки.

II. Клиентът иска да се свърже със сървъра. Той също създава сокет, който задава IP адреса и порта на сървъра, към който иска да се свърже.

III. Клиентът изпраща заявка към сървъра, като посочва порта, към който се очаква да получи отговора. Сървърът получава заявката и я обработва, като изпраща отговор към клиента чрез този същи порт.

IV. Клиентът получава отговора от сървъра чрез порта, който е посочил в заявката си. Той може да обработва отговора и да изпълни нужните действия.

V. След като свършат нужните действия, клиентът затваря сокета и прекратява връзката със сървъра. Сървърът също затваря сокета, когато приключи комуникацията.

Този пример показва, как сокетите и портовете работят за установяване на комуникация между две приложения. Сокетите позволяват на приложенията да се свързват с мрежата, а портовете позволяват на мрежата да идентифицира конкретните приложения, с които иска да комуникира.

Важно е да отбележим, че портовете са ограничени ресурси и трябва да бъдат използвани внимателно. Обикновено за известни протоколи и услуги са заделени стандартни портове, като например 80 за HTTP, 25 за SMTP и т.н. За динамичните портове, които приложенията използват динамично при връзката си, се заделят номера от 49152 до 65535.

4. **Дефинира термините: IP адрес, домейн, URL, DNS.**
   - IP адрес: Интернет протоколните (IP) адреси са числови идентификатори, които се използват за идентифициране на устройства в компютърната мрежа, като например компютри, маршрутизатори и други мрежови устройства. IP адресите могат да бъдат IPv4 адреси, състоящи се от четири октета, или IPv6 адреси, състоящи се от осем блока от шестнадесетични цифри.
   - Домейн: Домейнът е имена на уеб сайтове, които са част от системата на домейните и са уникални за всеки уеб сайт. Домейните се използват за идентифициране на уеб сайтове и са част от URL адресите.
   - URL: URL означава "Unified Resource Locator" (единен адрес на ресурса) и е адресът, който идентифицира уеб ресурс като уеб страница, изображение, видео или друго съдържание. URL адресите са съставени от протокол (например HTTP), домейн (например www.example.com) и път към ресурса.
   - DNS: DNS означава "Domain Name System" (система за имена на домейни) и е услуга, която превръща домейните в съответните им IP адреси. Когато се въвежда домейн в браузъра, DNS се използва, за да се открие IP адресът на сървъра, който хоства този домейн, и да се осъществи връзка с него.

5. **Обяснява модела клиент-сървър.**

Моделът клиент-сървър е архитектурен модел, който се използва в компютърните мрежи и софтуерните приложения за обмен на информация между клиентски и сървърни системи. В този модел, клиентските системи се свързват към сървърни системи, за да получат услуги или ресурси, които се предоставят от сървърите.

Клиентите са програми или устройства, които използват услугите или ресурсите, предоставени от сървърите. Обикновено клиентските системи имат графичен интерфейс и са лесно за употреба, докато сървърните системи са предназначени да бъдат стабилни и да работят без прекъсване.

117

Сървърите предоставят услуги и ресурси на клиентите, като например уеб страница, база данни, сървър за електронна поща, файлов сървър, игрови сървър и други. Обикновено сървърните системи са по-мощни и по-бързи от клиентските системи, тъй като трябва да обслужват множество клиенти едновременно.
В модела клиент-сървър, клиентите и сървърите се комуникират помежду си чрез мрежова връзка, като обменят информация във формат, който е разбираем за двете страни. По този начин, клиентите могат да поискат услуги и ресурси от сървърите, като същевременно сървърите могат да предоставят услугите и ресурсите на клиентите. Например, когато потребител посети уеб сайт, той използва интернет браузър като клиент, който изпраща заявка към уеб сървъра, за да получи страницата. Сървърът отговаря на заявката, като изпраща страницата към клиента. Този процес на комуникация между клиента и сървъра се осъществява чрез протокола за обмен на информация HTTP (Hypertext Transfer Protocol).

6. Обяснява и дава пример за използване на UDP. Различава TCP от UDP.

UDP (User Datagram Protocol) е мрежов протокол, който се използва за изпращане на данни по мрежата. UDP е протокол без връзка, който означава, че данните, изпратени по този протокол, не гарантират доставка или доставянето им в правилния ред. Той е бърз и лек протокол, който не изисква създаването и поддържането на връзка между източника и местоназначението, както при TCP.

Един от най-честите примери за използване на UDP е в онлайн игрите. При онлайн игрите е изключително важно да се намали латенсът (закъснение) при обмяна на информация между клиента и сървъра. TCP се грижи за доставката на данните, но има малко закъснение при установяване на връзка и изпращане на данни. Вместо това, UDP може да изпраща данни много бързо и без да ги проверява, като по този начин може да се намали латенсът и да се подобри преживяването на играчите.

Тъй като UDP е без връзка, той не осигурява гаранция за доставка на данните или контрол върху тяхното последователност. Затова, приложенията, които използват UDP, трябва да се грижат за проверката на доставката и подредбата на данните, ако това е необходимо.

В сравнение с UDP, TCP е протокол с връзка, който осигурява надеждна и гарантирана доставка на данните в правилния ред. TCP използва механизми като потвърждения на получаване на данни и повторна изпращане на загубени или повредени пакети, които гарантират, че данните безпроблемно достигат до местоназначението си. Това го прави по-надежден, но и по-бавен и по-тежък от UDP.

7. Описва фазите на комуникация при TCP. Демонстрира чрез програмен код прилагането на TCP.

TCP (Transmission Control Protocol) е един от основните протоколи за комуникация в интернет. TCP използва три основни фази на комуникация:

- Установяване на връзка (Three-way handshake) - тази фаза се използва за установяване на връзка между два устройства (клиент и сървър). Тя се състои от три стъпки:
    - Клиентът инициира връзката, като изпраща заявка (SYN) към сървъра.
    - Сървърът потвърждава заявката на клиента и инициира собствената си заявка към клиента (SYN-ACK).

- Клиентът потвърждава заявката на сървъра (ACK), като това установява успешна връзка между двете устройства.

- Предаване на данни (Data transfer) - след като връзката е установена, данните се предават между двата устройства. Тази фаза на комуникацията включва:
  - Изпращане на данни от клиента до сървъра.
  - Потвърждение от сървъра на получаването на данните.
  - Изпращане на данни от сървъра до клиента.
  - Потвърждение от клиента на получаването на данните.
- Разпадане на връзката (Connection teardown) - когато комуникацията между двата устройства е приключила, връзката се разпада. Тази фаза се състои от два стъпки:
  - Клиентът инициира заявка за разпадане на връзката (FIN).
  - Сървърът потвърждава заявката на клиента и инициира своята собствена заявка за разпадане на връзката (FIN-ACK).
  - Клиентът потвърждава заявката на сървъра (ACK), като това установява успешно разпадане на връзката.

8. Различава HTTP/1.1 от HTTP/2. Различава HTTP от WebSocket.

HTTP/1.1 и HTTP/2 са две различни версии на протокола за обмен на данни HTTP (Hypertext Transfer Protocol).

Основната разлика между HTTP/1.1 и HTTP/2 е, че HTTP/2 използва мултиплексиране на потоците, което позволява да се изпращат множество заявки едновременно по една и съща TCP връзка. HTTP/1.1 използва серийно изпращане на заявки, което означава, че заявките се изпращат последователно, една след друга, по една и съща TCP връзка.

WebSocket е протокол за двупосочна комуникация между уеб браузър и уеб сървър. Той позволява на уеб браузъра да изпраща заявки към сървъра и да получава отговори, но също така и на сървъра да изпраща съобщения към браузъра, без да бъде подавана заявка от браузъра. Това прави WebSocket подходящ за приложения като чат системи, онлайн игри, реално време аналитика и други.

HTTP и WebSocket са различни протоколи, макар че могат да бъдат използвани за подобни цели. HTTP е протокол за заявки и отговори, който се използва за изтегляне на ресурси от уеб сървъри. WebSocket, от друга страна, е протокол за двупосочна комуникация, който се използва за реално време обмен на данни между браузъра и сървъра.

9. Разработва програмен код на клиент и/или сървър чрез FTP и/или SMTP.

118

10. **Демонстрира знания за предимствата и недостатъците на еднонишковите и многонишковите Web сървъри.**

Еднонишковите и многонишковите уеб сървъри имат различни предимства и недостатъци в зависимост от конкретното приложение, за което се използват.

- Еднонишковите Web сървъри използват единична нишка за обслужване на всички заявки от клиентите. Това ги прави по-прости за изграждане и по-лесни за управление, но има няколко недостатъка. [illegible], еднонишковите сървъри не могат да обслужват множество заявки едновременно, което може да доведе до забавяне или дори блокиране на приложението. [illegible], тъй като една нишка обслужва всички заявки, всеки проблем в нишката може да доведе до прекъсване на обслужването на всички заявки.
- Многонишковите Web сървъри използват множество нишки, като всяка нишка обслужва една заявка. Това им позволява да обслужват множество заявки едновременно и да постигнат по-високи нива на производителност и отзивчивост.
    - Един от най-популярните многонишкови Web сървъри е [illegible]. [illegible] е друг често използван Web сървър, който използва модел на една нишка за обслужване на множество заявки, но има вграден механизъм за управление на многонишковост при нужда.

11. **Открива грешки в програмен код и го модифицира, така че да реши поставената задача**
12. **Анализира, определя и допълва програмен код, така че да реши поставената задача**