Напомена: Да се обрне посебно внимание на ова предавање, односно на теоретските основи на HTTP протоколот, неговите карактеристики и можности!

Што е World Wide Web и што е HTTP?

HTTP, или Hyper Text Transfer Protocol е протокол на апликациско ниво, кој како транспортен протокол го користи TCP протоколот. HTTP е протокол кој се заснова на клиент-сервер архитектура, во која клиентот испраќа барање, а серверот враќа одговор на барањето. HTTP е основата на World Wide Web, кој претставува информативен сервис на Интернет. Развојот на HTTP започна во 1989 на иницијатива на сер Timothy Berners Lee во лабораторијата CERN, a Интернет сервисот World Wide Web влезе со јавна употреба две години подоцна, односно во 1991 година. HTTP најпрво беше замислен и креиран како протокол за размена на текстуални датотеки кои можат лесно да бидат побарани со хиперллинкови.

Тука е интересно да се напомене дека првиот веб браузер беше изработен исто од сер Timothy Berners Lee, но популарноста на веб браузерите драстично порасна со појавата на веб браузерот Mosaic во 1993 година. Mosaic браузерот е основата за развој на браузерот Netscape Navigator од 1994 година (тоа е претходникот на Mozilla Firefox), а воедно Mosaic e и основаta врз која Microsoft во 1995 година го изработи Internet Explorer.

Кои се главните карактеристики на HTTP?

Прво, HTTP е stateless протокол, или, некаде се среќава изразот connectionless кој е со слично значење. Stateless значи дека HTTP врската која се воспоставува помеѓу клиентот и серверот трае само додека не се добие одговорот од серверот. После тоа врската се прекинува и се брише од меморијата на учесниците во HTTP комуникацијата. Секое наредно барање до серверот претставува нова постапка за воспоставување на HTTP комуникација, која ќе трае само до добивањето на одговор.

И Второ, HTTP е независен од типот на податоците кои се разменуваат. Тоа значи дека било кој тип на податоци (како текст, слики, аудио, видео...) можат да се разменуваат доколку и двете страни во комуникацијата (односно и клиентот и серверот) знаат за кој тип на податоци станува збор, и знаат како да ги процесираат. Затоа, секогаш кога се разменува датотека помеѓу клиент и сервер мора да се дефинира типот на датотеката (или content type) со користење на соодветна ознака која се нарекува Media type. Media type, кое порано се викаше MIME-Type (или Multipurpose Internet Mail Extensions Type) е стандард за дефинирање на форматот на датотеките кои се разменуваат преку Интернет.

Основната структура на HTTP пораките е дефинирана во стандардот RFC 822 на IЕTF организацијата, и без разлика дали се работи за request порака од клиентот или response порака од серверот, HTTP пораките се состојат од:

1. Почетна линија, па CRLF (Carriage Return (ASCII-13 ‘\r’) - Line Feed (ASCII-10 ‘\n’)) за крај на линијата.
2. Нула или повеќе Header полиња, односно заглавја, после кои следува CRLF ознака
3. Празна линија, односно линија без содржина која има само CRLF ознака
4. Опционално Message body. Message body во суштина се податоците кои се пренесуваат со http пораките, кои можат да бидат текстуални датотеки, слики видео итн. Доколку постои Message body во HTTP пораката, тогаш во заглавјата Content-Type и Content-Length се дефинираат карактеристиките за содржината во Message body. На пример, содржината во message body може да биде HTML страница.

Почетната линија, ако е испратена од клиент се вика Request-Line, а ако е во одговорот од серверот се вика Status-Line

Најпрво, во request линијата стои методот, потоа URI (Uniform Resource Identifier) па верзијата на HTTP протоколок кој се користи. На крај следува CRLF ознака за крај на редот.

Еден пример за request линија би бил:

**GET /index.html HTTP/1.1**

HTTP протоколот дефинира неколку методи за постапување со податоците кои се испраќаат. Некои од најкористените методи кои се дефинираат во Request пораките се:

GET, HEAD, POST, PUT, PATCH, DELETE и OPTIONS.

GET методот служи за побарување на ресурси од серверот, а можат и да се испраќаат податоци до серверот, како параметри за ресурсите кои се бараат. Тоа е основниот метод кој се користи за добивање на различни документи, како: HTML страници, CSS датотеки, слики, можеби pdf датотеки, видео итн.

HEAD методот е речиси идентичен со GET, со таа разлика што серверот ќе одговори со исти заглавја како и за GET барање, но во пораката нема да има Message body.

POST методот се користи кога се испраќаат податоци на сервер, како на пример податоци од пополнета форма од HTML страница, кои треба да бидат процесирани од страна на серверот.

PUT методот се користи за да се зачува датотеката испратена во Message body на локацијата на серверот дадена во URL адресата, или за замена на постоечка.

PATCH методот е сличен со PUT методот, со разлика што Message body на PUT методот содржи множество инструкции како ресурсот кој е дефиниран во URL адресата треба да се модификува за да се произведе нова верзија од него.

DELETE методот се користи како барање до серверот да ја избрише датотеката дадена со URL адресата во HTTP пораката.

OPTIONS методот се користи од страна на клиентот за да дознае кои HTTP методи или други опции ги поддржува серверот до кој се обраќа.

Од друга страна, статусната линија на response пораките содржи верзија на протоколот, нумерички статусен код и на крај текстуален опис на кодот.

Еден пример за response линија би бил:

**HTTP/1.1 200 OK**

Оваа статусна линија во одговорот на серверот, содржи статусен код кој се претставува со троцифрен број. Овие троцифрени броеви, во зависност од почетната цифра, го имаат следното значење:

1хх: Информативно – барањето е примено и се процесира;

2хх: Успешно примено, разбрано и прифатено барање.

3хх: Редирекција – потребни се следни чекори за реализација на барањето;

4хх: Грешка кај клиентот – барањето е примено но има погрешна синтакса и не може да се процесира од страна на серверот;

5хх: Грешка кај серверот – серверот не успеал да го процесира барањето кое навидум е сосема валидно и не содржи грешки.

Вториот дел од HTTP пораките се заглавјата.

Заглавјата (Headers) ги има 4 типови, и тоа:

General-headers: Кои можат да се применуваат и за Request и за Response пораки.

Request-headers: Само за request пораки

Response-headers: Само за response пораки.

Entity-headers: За дефинирање на дополнителни мета податоци за Message body.

Сите заглавјата имаат иста структура, и тоа: име на заглавјето, две точки, после кои следува листа на параметри кои се одвоени со запирка.

Некои примери од **генерални** заглавја се:

**Cache-control: max-age=3600** – клиентот ќе го прифати одговорот од серверот само доколку одговорот не е постар од 3600 секунди.

**Connection: keep-alive** – се користи за да се извести другата страна дека конекцијата не се затвора после оваа порака, што ќе значи дека примачот на пораката може да очекува и други пораки од истиот испраќач. Ова однесување е default во HTTP1.1, што не беше случај со HTTP1.0.

**Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11, websocket** – се користи за информирање кои други протоколи ги поддржува испраќачот за идна надградба на комуникацијата.

**СЛАЈД 8**

Примери од Request заглавја се:

**Accept: text/html, text/plain** – значи кои типови на датотеки ги прифаќа испраќачот на пораката.

**Cookie: name=daniel;email=daniel@mail.com** – значи дека до серверот се испраќа cookie кое содржи име и емаил на корисникот.

**Host: fikt.edu.mk** – се користи за дефинирање на адресата на серверот до кој се испраќа брањето.

Неко примери од **Response** заглавја се:

**Location: http://www.fikt.uklo.edu.mk/studies/first-cycle/** - се користи за пренасочување на клиентот кон друга локација за комплетирање на барањето.

**Server: Apache/2.2.14 (Win32)** – ова заглавје вклучува информации за серверот.

**Set-Cookie: name=daniel,email=daniel@mail.net; Expires=Fri, 23 Aug 2022 20:24:01 GMT** – дефинирање на cookie кое треба да се чува на страната на клиентот.

Некои Entity заглавја:

**Allow: GET, HEAD, PUT** – Дефинирање на поддржани методи

**Content-Encoding: zip** – се користи за дефинирање на кодирањето на содржината на message body

**Content-Type: text/html; charset=utf-8** – се дефинира типот на содржината во message body, како и кодирањето на текстот.

Пример на Request порака испратена до серверот на ФИКТ

GET /staff HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.fikt.uklo.edu.mk/

Accept-Language: en-us

Accept-Encoding: gzip, zip

Connection: Keep-Alive

Пример за Response порака

HTTP/1.1 200 OK

Date: Wed, 12 Apr 2023 09:45:13 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 05 Feb 2023 19:15:56 GMT

Content-Length: 88

Content-Type: text/html, charset=utf-8

Connection: Closed

<!DOCTYPE html>

<html>

<head>

<title>Факултет за ИКТ</title>

<body>

<h1>Список на вработени</h1>

.

.

.

</body>

</html>

Во врска со развојот на HTTP протоколот, почетната верзија која беше издадена беше верзијата

HTTP/0.9 (1989) — The One-line Protocol

Во Оваа верзија протоколот дефинираше само една линија во која има метод и адреса на бараниот документ.

Единствениот поддржан метод беше GET, а во одговорот можеше да има само статички HTML страници. Исто така, секоја конекција беше затварана веднаш по испораката на одговорот.

Во оваа верзија не постоеја заглавја, ниту пренасочувања, ниту други функционалности кои ги имаат подоцнежните верзии.

Во верзијата HTTP1.0 (1996) беа воведени низа новини:

Воведени се заглавја, а со тоа можност за дефинирање на Content-type со што се обезбедува испорака на различни типови на датотеки, покрај HTML датотеките, а воедно воведено е и дефинирање на HTTP верзија и статусни кодови.

Направено е и проширување на методите со воведување на HEAD и POST покрај GET методот.

Исто така, и во оваа верзија на протоколот конекцијата се прекинува веднаш после испораката на одговорот, но воведена е можност со дефинирање на заглавје Connection: keep-alive да се остави конекцијата отворена и после опслужувањето на првичното барање.

Верзијата на HTTP која моментално е во најширока употреба е 1.1 (1997, 1999, 2014). Во оваа верзија воведени се промени за оптимизација на перформансите и нови функционалности. Во оваа верзија нема потреба од дефинирање на Connection: keep-alive бидејќи протоколот HTTP1.1 по правило функционира на ваков начин. Доколку треба да се прекине конекцијата после одредено опслужување, треба да се испрати заглавје Connection: close.

Во оваа верзија воведени се и нови методи, како PUT, DELETE, OPTIONS и други.

Можеби најважната придобивка во HTTP1.1 верзијата е воведувањето на заглавјето Upgrade. Со ова заглавје сe обезбедува можност да се започне конекција со општо познат протокол, како што е самиот HTTP1.1, а подоцна да се премине кон комуникација со понапредни протоколи, како што се на пример HTTP/2.0 или пак, WebSocket.

Верзијата HTTP/2.0 (2015) е верзија која обезбедува низа на напредни можности, како мултиплексирање на пораките (паралелни барања) во една TCP конекција, компресија на заглавјата. Со HTTP/2.0 серверот може да испраќа „PUSH“ пораки до клиентот без клиентот експлицитно да ги побарува содржините за секоја порака и друго.

HTTP/3 е најновата верзија на HTTP протоколот, кој сѐ уште е во развојна фаза. Драфт верзијата на протоколот е од 2018 година и сѐ уште трае фазата на тестирање.

HTTP/3 е замислен да го користи новиот QUIC протокол, наместо TCP.

Да видиме каков е начинот на комуницирање со HTTP во споредба со протоколот WebSocket.

HTTP е еднонасочен протокол. Тоа значи дека иницијатор на комуникацијата секогаш е клиентот, а серверот е тука само да даде одговор. Никогаш, во никој случај, серверот не може да иницира комуникациска сесија.

Е, сега, во однос на различните начини на комуницирање, најпрво да го спомнеме начинот HTTP Short polling.

Short polling значи дека клиентот испраќа барање до серверот, а серверот веднаш враќа одговор без разлика дали барањето можело да се процесира или не. На пример, сакаме преку HTTP протокол да добиваме информација од некој сервер за резултатот на еден фудбалски натпревар. Да замислиме дека податоците на серверот ги ажурира администратор кој веднаш става информација доколку биде постигнат гол.

Short poling значи клиентот испраќа барање до серверот во кое вели „дај ми моментален резултат“. Серверот испраќа одговор веднаш и комуникацијата е затворена. Доколку клиентот сака постојано да биде во тек со резултатот, користејќи Short polling ќе треба да испраќа ново барање на одредено кратко време, на пример на секоја пола минута или минута и да го чита новиот резултат. На овој начин, бројот на разменети пораки ќе биде голем, што значи се трошат мрежни ресурси, а промена во резултатот може и да нема во стотици разменети пораки.

Втора можност за реализација на оваа идеја со користење на HTTP е да се користи начинот long polling. Long polling значи клиентот испраќа прво барање до серверот, со прашање каков е резултатот, серверот испраќа одговор, а потоа клиентот веднаш испраќа ново барање со прашање дали има промена во резултатот. Серверот во овој случај го задржува барањето се додека не се промени резултатот, а потоа го испраќа одговорот. Веднаш по добивањето на одговорот, клиентот испраќа ново барање до серверот, а серверот ја задржува отворена комуникацијата и чека на промена на резултатот за да испрати одговор. Со ова се трошат ресурсите на серверот. Уште една негативна страна на овој long poling e тоа што нема матична имплементација во HTTP, туку програмски се дефинира вакво однесување на серверот.

Една релативно нова можност за реализација на PUSH пораки од страна на сервер е Server Sent Events на HTML5 стандардот. SSE е исто така еднонасочна комуникација со која клиентот само испраќа барање за поврзување на SSE извор, а понатаму серверот испраќа PUSH пораки според неговата дефиниција. Ако е дефинирано да испраќа пораки штом се промени резултатот, само тогаш ќе ги испраќа пораките до сите кои се приклучиле на тој извор.

Поради овие можности и начин на функционирање, HTTP протоколот е непрактичен за примена во Chat апликации на пример, токму поради фактот што комуникацијата е практично еднонасочна, односно само клиентот може да иницира комуникација.

Чекор погоре во можностите за мрежни комуникации е протоколот WebSocket дефиниран во IETF стандардот RFC 6455 од 2011 година.

WebSocket протоколот му овозможува и на клиентот и на серверот да испраќаат push пораки во секое време без никаква поврзаност со некое претходно испратено барање. Овој протокол обезбедува и Full-duplex двонасочна комуникација што значи дека и клиентот и серверот можат да испраќаат пораки меѓу себе во исто време, што не е случај со HTTP. Вообичаено, најпрво клиентот и серверот се поврзуваат со HTTP протокол преку TCP конекција, а потоа со надградба (Upgrade) на протоколот се преминува на WebSocket протокол со користење на истата воспоставена TCP конекција.