MINISTERUL EDUCAŢIEI AL REPUBLICII MOLDOVA   
UNIVERSITATEA DE STAT „ALECU RUSSO” DIN BĂLŢI   
FACULTATEA DE ŞTIINŢE REALE, ECONOMICE ȘI ALE MEDIULUI   
CATEDRA DE MATEMATICĂ ȘI INFORMATICĂ

**REFERAT**   
la disciplina „INFORMATICA GENERALĂ”   
Explicarea concepției de web-socket

**Autor:**

Studentul grupei IS11Z

Igor COSTIUC

**Conducător științific:**

Olesea SKUTNIȚKI

**BĂLȚI 2022**

Cuprins

**[Introducere 2](#_Toc16312)**

**[1. Idea de WebSocket 3](#_Toc24618)**

**[Ce este web socket-ul și cum este diferit de HTTP 4](#_Toc24017)**

[1.1 Protocolul HTTP 4](#_Toc15642)

[1.2 Web Socket 4](#_Toc30455)

[1.3 Unde poate fi utilizat web-socket-ul 5](#_Toc24807)

[1.4 Unde nu trebuie utilizat web-socket-ul 6](#_Toc16494)

**[3 Diferențele dintre HTTP și WebSocket Connection: 7](#_Toc24066)**

**[Semnificația UDP și TCP 7](#_Toc21560)**

**[BIBLIOGRAFIE 10](#_Toc14961)**

# Introducere

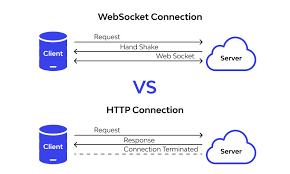
În lumea de astăzi extrem de conectată și constant online, așteptăm să primim orice informație instantaneu. Gândiți-vă la toate aplicațiile pe care le folosim pentru a trimite mesaje sau pentru a primi noutăți care sunt renoite în fiecare zi. WebSocket-urile sunt unul dintre multele instrumente diferite pentru construirea de aplicații web care oferă actualizări și comunicare instantanee în timp real.

# Idea de WebSocket

Ideea WebSockets s-a născut din limitările tehnologiei bazate pe HTTP. Cu HTTP, un client solicită o resursă, iar serverul răspunde cu datele solicitate.

HTTP este un protocol strict unidirecțional - orice date trimise de la server către client trebuie mai întâi solicitate de client. Sondajul lung a acționat în mod tradițional ca o soluție pentru această limitare. În cazul unui sondaj lung, un client face o solicitare HTTP cu o perioadă de timeout lungă, iar serverul folosește acel timeout lung pentru a trimite date către client. Sondajul lung funcționează, dar are un dezavantaj - resursele de pe server sunt legate pe toată durata sondajului lung, chiar și atunci când nu sunt disponibile date pentru a fi trimise. WebSockets, pe de altă parte, permit trimiterea de date bazate pe mesaje, similare cu UDP, dar cu fiabilitatea TCP. WebSocket folosește HTTP ca mecanism de transport inițial, dar menține conexiunea TCP vie după primirea răspunsului HTTP, astfel încât să poată fi utilizat pentru trimiterea de mesaje între client și server. WebSocket-urile ne permit să construim aplicații „în timp real” fără a folosi sondaje lungi.

WebSocket-urile încep viața ca o solicitare și un răspuns HTTP standard. În cadrul acestui lanț de răspuns la cerere, clientul cere să deschidă o conexiune WebSocket, iar serverul răspunde (dacă este capabil). Dacă această strângere de mână inițială are succes, clientul și serverul au fost de acord să utilizeze conexiunea TCP/IP existentă care a fost stabilită pentru cererea HTTP ca o conexiune WebSocket. Datele pot circula acum prin această conexiune utilizând un protocol de mesaj încadrat de bază. Odată ce ambele părți recunosc că conexiunea WebSocket ar trebui să fie închisă, conexiunea TCP este întreruptă.



# Ce este web socket-ul și cum este diferit de HTTP

## 1.1 Protocolul HTTP

HTTP este unidirecțional unde clientul trimite cererea și serverul trimite răspunsul. Să luăm un exemplu când un utilizator trimite o cerere către server această solicitare merge sub formă de HTTP sau HTTPS, după primirea solicitării, serverul trimite răspunsul către client, fiecare cerere este asociată cu un răspuns corespunzător, după trimiterea răspunsului conexiunea se închide, fiecare solicitare HTTP sau HTTPS stabilește noua conexiune la server de fiecare dată și după primirea răspunsului, conexiunea este întreruptă de la sine.

HTTP este un protocol fără stat care rulează peste TCP, care este un protocol orientat spre conexiune, care garantează livrarea transferului de pachete de date folosind metodele de handshaking în trei căi și retransmite pachetele pierdute. HTTP poate rula pe deasupra oricărui protocol de încredere orientat spre conexiune, cum ar fi TCP, SCTP. Când un client trimite o solicitare HTTP către server, o conexiune TCP este deschisă între client și server și, după primirea răspunsului, conexiunea TCP se încheie, fiecare cerere HTTP deschide o conexiune TCP separată la server, de ex. dacă clientul trimite 10 solicitări către server, se vor deschide cele 10 conexiuni TCP separate.

HTTP trimite ca mesaj informația codificată în ASCII , fiecare mesaj de solicitare HTTP compus din versiunea protocolului HTTP (HTTP/1.1, HTTP/2), metode HTTP (GET/POST etc.), anteturi HTTP (tipul conținutului, lungimea conținutului), informații despre host etc. și corpul care conține mesajul real care este transferat pe server. Antetele HTTP au variat de la 200 de octeți la 2 KB, dimensiunea comună a antetului HTTP este de 700-800 de octeți. Când o aplicație web folosește mai multe module cookie și alte instrumente la nivelul clientului care extind caracteristicile de stocare ale agentului, aceasta reduce sarcina utilă a antetului HTTP.

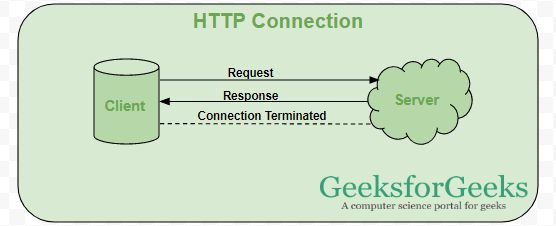


Fig 1.1 Cum lucrează conexiunea HTTP

## 1.2 Web Socket

WebSocket este bidirecţional, un protocol full duplex care este utilizat în acelaşi scenariu de comunicare client-server, spre deosebire de HTTP, începe de la ws:// sau wss://. Este un protocol cu stare, ceea ce înseamnă că conexiunea dintre client şi server va menţine vie până când va fi terminată de oricare dintre părţi (client sau server). După închiderea conexiunii de către client şi server, conexiunea este încheiată de la ambele capete. Să luăm un exemplu de comunicare client-server, există clientul care este un browser web şi un server, ori de câte ori iniţiem conexiunea între client şi server, client-server a făcut strângerea de mână şi decide să creeze o nouă conexiune şi această conexiune va rămâne în viaţă până când va fi reziliat de oricare dintre ei. Când conexiunea este stabilită şi este live, comunicarea are loc folosind acelaşi canal de conexiune până când este terminată.

Așa se face , după handshake dintre client-server, client-server decide asupra unei noi conexiuni pentru a o menține în viață, această nouă conexiune va fi cunoscută sub numele de WebSocket. Odată ce stabilirea legăturii de comunicație și conexiunea sunt deschise, schimbul de mesaje va avea loc în mod bidirecțional până când conexiunea persistă între client-server. Dacă cineva dintre ei (client-server) moare sau decide să închidă conexiunea, ea este închisă de ambele părți. Modul în care funcționează socket-ul este diferit de modul în care funcționează HTTP, codul de stare 101 denotă protocolul de comutare în WebSocket.

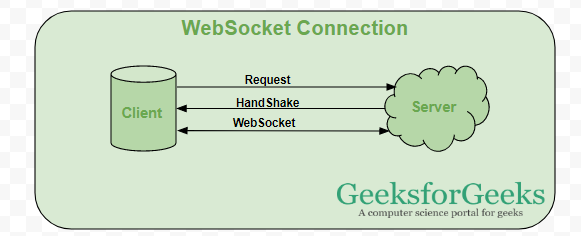


Fig 1.2 Cum lucrează conexiunea webSocket

## 1.3 Unde poate fi utilizat web-socket-ul

**Aplicație web în timp real:** aplicația web în timp real folosește un socket web pentru a afișa datele la capătul clientului, care sunt trimise continuu de serverul backend. În WebSocket, datele sunt împinse/transmise continuu în aceeași conexiune care este deja deschisă, de aceea WebSocket este mai rapid și îmbunătățește performanța aplicației. Pentru ex. într-un site web de tranzacționare sau tranzacționare cu bitcoin, pentru afișarea fluctuației prețului , datele sunt împinse în mod continuu de serverul backend la capătul clientului prin utilizarea unui canal WebSocket.

**Aplicație de jocuri:** într-o aplicație de jocuri, datele sunt primite în mod continuu de către server și, fără a reîmprospăta interfața de utilizare, va avea efect pe ecran, interfața de utilizare este reîmprospătată automat fără a stabili o nouă conexiune, deci este foarte util într-o aplicație de jocuri.

**Aplicație de chat:** aplicațiile de chat folosesc WebSockets pentru a stabili conexiunea o singură dată pentru schimbul, publicarea și difuzarea mesajului între abonați. Reutilizează aceeași conexiune WebSocket, pentru trimiterea și primirea mesajului și pentru transferul de mesaje unu-la-unu

.

## 1.4 Unde nu trebuie utilizat web-socket-ul

WebSocket poate fi utilizat dacă dorim orice fluxuri de date actualizate sau continue în timp real care sunt transmise prin rețea. Dacă vrem să obținem date vechi, sau vrem să obținem datele o singură dată pentru a le procesa cu o aplicație, ar trebui să mergem cu protocolul HTTP, datele vechi care nu sunt solicitate foarte frecvent sau preluate o singură dată pot fi interogate prin simpla solicitare HTTP, deci, în acest scenariu, este mai bine să nu utilizați WebSocket.

Notă: Serviciile web RESTful sunt suficiente pentru a obține datele de pe server dacă încărcăm datele o singură dată.

# 3 Diferențele dintre HTTP și WebSocket Connection:

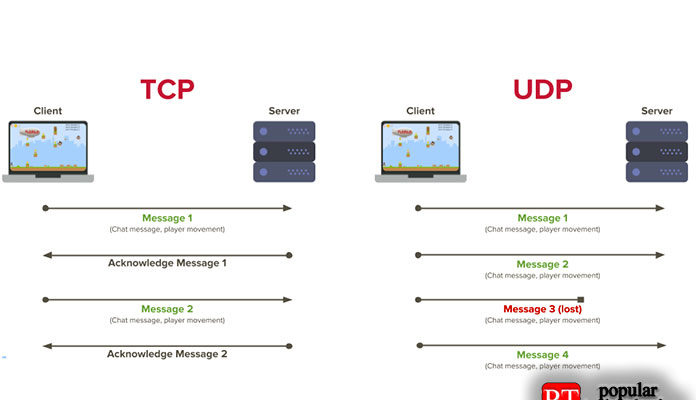
WebSocket este un protocol de comunicare bidirecțional care poate trimite datele de la client la server sau de la server la client prin reutilizarea canalului de conexiune stabilit. Conexiunea este menținută vie până când este terminată fie de client, fie de server.

Protocolul HTTP este un protocol unidirecțional care funcționează pe deasupra protocolului TCP, care este un protocol de nivel de transport orientat spre conexiune, putem crea conexiunea utilizând metode de solicitare HTTP după ce conexiunea HTTP de răspuns se închide.

Semnificația UDP și TCP

**TCP** este un protocol de transport pentru transferul de date în rețelele TCP/IP, care stabilește mai întâi o conexiune la rețea.

**UDP** este un protocol de transport care transmite mesaje datagrame fără a fi nevoie să stabiliți o conexiune într-o rețea IP.



Diferența dintre protocoalele TCP și UDP este așa-numita „garanție de livrare”. TCP necesită un răspuns de la clientul căruia i se livrează pachetul de date, o confirmare de livrare, iar pentru aceasta are nevoie de o conexiune prestabilită. De asemenea, protocolul TCP este considerat de încredere, în timp ce UDP a primit chiar numele de „protocol de datagramă nesigur. TCP elimină pierderea de date, duplicarea și amestecarea pachetelor, întârzierile. UDP permite toate acestea și nu necesită o conexiune pentru a funcționa. Procesele cărora le sunt trimise date prin UDP trebuie să se descurce cu ceea ce primesc, chiar și cu pierderi. TCP controlează congestionarea conexiunii, UDP nu controlează nimic altceva decât integritatea datagramelor primite.

**CONCLUZIE**

Web socket-urile ne permit nouă să comunicăm cu serverul direct, cu o conexiune constantă, aceasta ne permite să contruim aplicații de tip messenger sau altele, în care conexiune dintre client-server este strict necesară.

# BIBLIOGRAFIE

**<https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/>**

**https://sookocheff.com/post/networking/how-do-websockets-work/#:~:text=A%20WebSocket%20is%20a%20persistent,between%20a%20client%20and%20server.**

**http://pyatilistnik.org/chem-otlichaetsya-protokol-tcp-ot-udp/#:~:text=TCP%20%E2%80%93%20%D1%82%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D1%8B%D0%B9%20%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85,%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B8%20%D1%81%D0%BE%D0%B5%D0%B4%D0%B8%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B2%20IP%2D%D1%81%D0%B5%D1%82%D0%B8.**