**1. Ce este un Socket?**

**Socket** permite comunicarea intre doua procese diferite pe aceeasi masina sau diferite.

A Socket is used in a client-server application framework.

**Server** este un proces care indeplineste unele functii la cerere de la un client.

Majoritatea protocolelor de nivel de aplicatie precum FTP,SMTP si POP3 folosesc sockete pentru a stabili conexiunea intre client si server, apoi pentru schimbul de date.

**2. Scopul protocolului HTTP.**

HTTP sta la baza schimbului de date pe internet.

HTTP este prototcol de interactiune client-server, ceea ce inseamna ca solicitarile sunt intiate de destinatar, de obicei web browser-ul.

Client si Server comunica prin prin schimbul de mesaje individuale( nu si flux de date).

Mesajele trimise de Client de obicei un web browser, se numeste “request”, iar mesajele trimise de Server ca raspuns se numesc “responses”.

HTTP este protocol client-server: cererile sunt trimise de o singura entitate, user-agent ( proxy).

De cele mai multe ori user-agentul, este un web browser, dar poate fi orice. De exemplu, un robot care acceseaza cu crawlere web-ul pentru a popula si a mentine un index al motorului de cautare.

**3. Ce este un fir de executie?**

Un fir de executie este cea mai unitatea de executie a unui proces ce poate fi programată spre execuție de către sistemul de operare. Fiecarui fir de executie i se asociaza o secventa de instructiuni, un set de registri CPU si o stiva. Procesul nu executa instructiuni, este un spatiu de adresare comun pentru unul sau mai multe fire de executie. Firele de executie sunt cele care executa instructiunile.

**4. Diferența dintre TCP si UDP.**

TCP este un protocol de transport pentru transmisia de date în rețelele TCP / IP care stabilește o conexiune la rețea.

UDP este un protocol de transmisie care transmite mesaje datagrame fără a fi necesară stabilirea unei conexiuni în rețeaua IP.

Diferența dintre TCP și UDP - „garanția de livrare“, așa-numitul TCP necesită un răspuns de la clientul căruia este livrat pachetul de date, confirmarea livrării și pentru aceasta are nevoie de o conexiune prestabilită. De asemenea, protocolul TCP este considerat de încredere, în timp ce UDP a primit chiar și numele "protocol de datagrame nesigure”. TCP elimină pierderea de date, duplicarea și amestecarea pachetelor, întârzieri. UDP permite toate acestea, și nu are nevoie de o conexiune pentru lucru. Procesele la care datele sunt transmise prin UDP trebuie să gestioneze primite, chiar și cu pierderile.TCP controlează congestia conexiunii, UDP nu controlează decât integritatea datagramei recepționate.

Pe de altă parte, din cauza lipsei de control și non-selectivității, UDP oferă pachete de date (datagramele) sunt mult mai rapid, deoarece aplicațiile care sunt proiectate pentru lățime de bandă largă și de schimb rapid, UDP poate fi considerat ca protocolul optim. Acestea includ jocuri de rețea și de browser, precum și programe și aplicații video streaming pentru video (sau voce) a pierderii de pachete, totală sau parțială, nimic nu sa schimbat, se repetă cererea nu este în mod necesar, dar descărcarea este mult mai rapid. Protocolul TCP, ca și mai fiabil, este folosit cu succes chiar și în programele de e-mail, permițându-vă să monitorizați nu numai traficul, dar și lungimea mesajului și viteza schimbului de trafic.

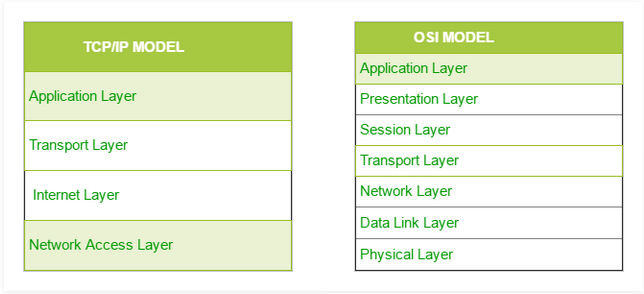
**5. Ordinea în care octeții sunt secvențializați în rețea**

Un octet este o secvență de [biți](https://ro.eyewated.com/ce-este-un-pic-in-retea-de-calculatoare/) . În rețelele de calculatoare, unele protocoale de rețea trimit și primesc date sub formă de secvențe de octeți. Acestea sunt numite protocoale orientate de octeți . Exemple de protocoale orientate de byte includ TCP/ IP și telnet .

Adresele IPv4 au o lungime de 32 de biți (4 octeți). Fiecare adresă identifică o rețea (*network*) și o stație de lucru (*work station*) din cadrul rețelei.

IPv6 este un protocol dezvoltat pentru a înlocui IPv4 în Internet. Adresele au o lungime de 128 [biț](https://ro.wikipedia.org/wiki/Bit)i (16 [octeți](https://ro.wikipedia.org/wiki/Octet)), ceea ce este considerat suficient pentru o perioadă îndelungată. Teoretic există 2128, sau aproximativ 3,403 × 1038 adrese unice. Lungimea mare a adresei permite împărțirea în blocuri de dimensiuni mari și implicit devine posibilă introducerea unor informații suplimentare de rutare în adresă.

**6. Cele patru nivele ale stivei TCP/IP și șapte nivele ale modelului OSI**

**TCP/IP Model**

1. Network Access Layer

Acest strat corespunde combinației dintre Data Link Layer și Physical Layer al modelului OSI. El caută adrese hardware, iar protocoalele prezentate la acest nivel permit transferul de date fizice.

2. Internet Layer

Acest layer este paralel cu funcțiile Network layer al OSI. El definește protocoalele care sunt responsabile pentru transmiterea logică a datelor pe întreaga rețea. Principalele protocoale care stau la acest nivel sunt:

- IP – Internet Protocol.

- ICMP – stands for Internet Control Message Protocol.

- ARP – Address Resolution Protocol

3.Host-to-Host Layer

Acest strat este analog stratului de transport al modelului OSI. Este responsabil pentru comunicarea end-to-end și livrarea fără erori a datelor. Protejează aplicațiile din stratul superior de complexitățile datelor. Cele două protocoale principale prezente în acest strat sunt:

1. Transmission Control Protocol (TCP)
2. User Datagram Protocol (UDP)

### 4. Application Layer

Acest strat îndeplinește funcțiile primelor trei straturi ale modelului OSI: Application, Presentation and Session Layer. Este responsabil pentru comunicarea node-to-node și controlează specificațiile interfeței utilizatorului. Unele dintre protocoalele prezente în acest strat sunt: HTTP, HTTPS, FTP, TFTP, Telnet, SSH, SMTP, SNMP, NTP, DNS, DHCP, NFS, X Window, LPD.

**OSI**

1. Physical Layer (Layer 1) :

Cel mai jos strat al modelului de referință OSI este stratul fizic. Este responsabil pentru conexiunea fizică reală între dispozitive. Stratul fizic conține informații sub formă de biți.

2. Data Link Layer(Layer 2) :

Data Link Layer este responsabil pentru livrarea mesajului de la nod la nod. Funcția principală a acestui strat este de a asigura că transferul de date este fără erori de la un nod la altul, peste stratul fizic.  
Data Link Layer este împărțit în două sub-straturi:  
 Logical Link Control (LLC)  
 Control acces media (MAC)

3. Network Layer (Layer 3) :  
Network Layer funcționează pentru a transfera date de la un host la altul, localizate pe diferite rețele. De asemenea, asigură dirijarea pachetelor, adică selectarea celei mai scurte căi de transmitere a pachetelor dintre rutele disponibile.

4. Transport Layer (Layer 4):

Stratul de transport oferă servicii pentru stratul de aplicații și preia serviciile din stratul de rețea. Datele din stratul de transport sunt denumite Segmente. Este responsabil pentru End to End Delivery a mesajului complet. Stratul de transport oferă, de asemenea, recunoașterea transmisiei de succes a datelor și transmite din nou datele dacă se constată o eroare.

### 5. Session Layer (Layer 5) : Acest strat este responsabil pentru stabilirea conexiunii, întreținerea sesiunilor, autentificare și, de asemenea, asigură securitatea.

### 6. Presentation Layer (Layer 6) :

Presentation Layer se numește și Translation layer. Datele din stratul de aplicație sunt extrase aici și manipulate conform formatului necesar pentru a fi transmis în rețea.

7. Application Layer (Layer 7) :

În partea de sus a stivei de straturi de model de referință OSI, găsim stratul de aplicații care este implementat de aplicațiile de rețea. Aceste aplicații produc date, care trebuie transferate prin rețea. Acest strat servește, de asemenea, ca o fereastră pentru accesul la rețea al serviciilor de aplicații și pentru afișarea informațiilor primite utilizatorului.

**7. Din ce cauză nu puteți transmite cereri HTTP website-ului utm.md utilizînd portul 80 ?**



Website-ul utm.md, lucreaza pe browserul HTTPS, requesturile pot fi trimise pe website-ul dat, insa cu portul 443.

**8. Cum se creează o conexiune Socket ?**

Un socket este interfata pe care sistemul de operare o pune la dispozitia aplicatiei pentru ca aceasta sa poata comunica prin intermediul retelei cu o alta aplicatie de pe un alt sistem. Un socket identifica in mod unic un capat (endpoit) dintr-o conexiune. Socket-ul client si socket-ul server formeaza o conexiune. Din acest punct de vedere un socket poate fi considerat un tuplu:

**<protocol nivel 3, adresa nivel 3, protocol nivel 4, adresa nivel 4>**

O conexiune este o pereche formata din doi sockets: **<socket client, socket server>**.

In marea majoritatea cazurilor, protoculul e nivel 3 este IP, iar adresa 3 este o adresa IO pe 32 de biti. Protoculul de nivel 4 poate fi UDP sau TCP, iar adresa de nivel 4 este portul asociat conexiunii.

**9. Cum se transmit date intr-un Socket ?**

La nivelul interfetei socket conexiunea este stabilita in momentul in care socket-ul server se gaseste in starea *listening* dupa apelul **accept**, iar clientul initiaza conexiunea folosind **connect**. In momentul in care conexiunea este realizata (protocolul de initiere a conexiuenii se incheie cu succes), apelul **accept** intoarce descriptorul unui nou socket. Acest nou socket este cel care va coordona transferul de date intre client si server. Cu alte cuvinte, socketul aflat in starea listening este utilizat numai pentru asculatea de cereri de initiere de conexiune, urmand ca, ulterior stabilirii conexiunii, transferul de date sa fie intermediat de un alt socket. Iar o alta conexiune presupune crearea unui alt socket...

**10. Cum se citesc datele dintr-un Socket ?**

In precedentele doua raspunsuri, este inclus si raspunsul la aceasta intrebare ^

**11. Care sunt metodele HTTP?**

Metoda GET solicită o reprezentare a resursei. Interogările care utilizează această metodă pot prelua doar date.  
HEAD solicită resursa în același mod ca metoda GET, dar fără corpul de răspuns.  
POST este utilizat pentru a trimite entități la o resursă specifică. Adesea provoacă o schimbare de stare sau unele efecte secundare asupra serverului.  
PUT înlocuiește toate reprezentările actuale ale resurselor cu datele solicitării.  
DELETE șterge resursa specificată.  
CONNECT stabilește un "tunel"/conexiune către serverul identificat de resursă.  
OPTIONS este utilizat pentru a descrie parametrii pentru conectarea la resursă.  
TRACE apelează mesajul de test returnat din resursă.  
PATCH este utilizat pentru a modifica parțial resursa.

**12. Codurile de stare HTTP (200, 301, 302, 401, 404, 405, 500)**

**200 – OK –** cod de raspuns, care indica faptul ca solicitarea a fost finalizata cu succes.

**301 – Moved Permanently –** cod de redirectionare, care indica faptul ca resursa solicitata a fost mutata permanent, la adresa URL specificta in antetul Locatie.

**302 – Found –** cod de redirectionare, inseamna ca resursa solicitata a fost mutata temporar la adresa specificata in antetul de Locatie. Dupa ce a primit un astfel de raspuns, browserul este redirectionat catre o noua pagina, iar motorul de cautare nu isi actualizeaza legaturile catre resursa nu se schimba si nu est trimisa la o noua adresa URL

**401 – Unauthorized –** client error status, indica faptul ca soliciatrea nu a fost aplicata, deoarece nu esti autorizat pe resursa data si lipsescacreditari valide pentru resursa data. Este similara cu **403**, doar ca in cazul dat autorizarea este posibila.

**404 – Not Found –** client error status, indica faptul ca serverul nu poate gasi resursa ceruta. Acest stats code, cel mai probabil cel mai cunoscut in retea.

**405 – Method Not Allowed –** status code HTTP, indica faptul ca metoda solicitarii este cunoscuta de server, DAR a fost dezactivata si nu poatefi utilizata. Doua metode GET si HEAD nu trebuie niciodata dezactivate si nu ar trebuie sa returneze acest cod de eroare. Serverul trebuie sa genereze un camp cu antet **Allow** in raspunsul cu codul 405, care contine lista metodelor de resurse disponibile.

**500 – Internal Server Error –** server error response code, indica faptul ca serverul a intalnit o conditie neasteptata care l-a impiedicat sa execute cererea.

**13. Ce este HTTP securizat ?**

HTTPS este o extensie a protocolului HTTP, care asigură o conexiune criptată. Acesta generează, la fiecare sesiune de comunicare, o cheie criptografică unică. Cheia respectivă poate fi decriptată doar de browserul utilizatorului și serverul pe care sunt stocate informațiile solicitate de utilizator. Astfel, loginurile și parolele utilizatorului, mesajele lui personale, datele cardurilor bancare și orice alte informații introduse de utilizator devin inaccesibile pentru terți.

**14. Diferența dintre HTTP și HTTPS**

Hypertext Transfer Protocol (HTTP) : In mod normal este folosit protocolul HTTP pentru a trimite cereri catre server. in acest caz adresa web va incepe cu ‘http://’ si va folosi portul 80 pentru a comunica cu serverul. Acest protocol nu foloseste nici-un fel de criptare a datelor, deci nu este recomandat sa folosim acest protocol pentru trimiterea informatiilor importante (date personale, date card, date cont bancar, etc).

Hypertext Transfer Protocol Secure (HTTPS) :HTTPS este de fapt o combinatie intredoua protocoale diferite, Hypertext Transfer Protocol (HTTPS) si SSL/TLS. LInk-urile incep cu ‘https://’. Acest protocol ne ofera o metoda sigura de a transmite datele importante.Acest protocol foloseste portul 443 pentru a comunica cu serverul. HTTPS are un algoritm de criptare ce asigura siguranta datelor.

**15. Diferența dintre Socket și WebSocket**

Deși realizează (în general) lucruri similare, dar, sunt cu adevărat diferite. WebSockets sunt de obicei lansate din browserele care se conectează la Serverul de aplicații folosind un protocol similar cu HTTP, care funcționează prin TCP / IP. Astfel, sunt destinate în general aplicațiilor web care necesită o conexiune permanentă la serverul lor. Sockets simple, pe de altă parte, sunt mai puternice și mai unversale. Funcționează prin TCP / IP, dar nu sunt limitate la browsere sau HTTP. Pot fi folosite pentru a implementa orice tip de comunicare.

**16. Diferența dintre un proces și un fir de execuție**

Procesul  
Fiecare proces oferă resursele necesare pentru a executa programul. Un proces are un spațiu de adrese virtuale, cod executabil, descriptori deschiși de obiecte de sistem, context de securitate, identificator de proces unic, variabile de mediu, clasă de prioritate, dimensiuni minime și maxime ale setului de lucru și cel puțin un fir de execuție. Fiecare proces pornește de la un fir, adesea numit thread principal, dar poate crea fire suplimentare de la oricare dintre firele sale.  
  
Fir de execuție   
Un thread este un obiect dintr-un proces care poate fi programat pentru execuție. Toate firele unui proces împărtășesc spațiul său virtual de adrese și resursele sistemului. În plus, fiecare flux acceptă gestionări de excepții, prioritate de planificare, stocare de flux local, un identificator unic de flux și un set de structuri pe care sistemul le va folosi pentru a menține contextul fluxului înainte de a-l planifica. Un context de thread include un set de registre de fir de mașină, o stivă de nucleu, un bloc de medii de thread și o stivă de utilizator în spațiul de adrese al unui proces de thread. De asemenea, fluxurile pot avea propriul context de securitate care poate fi folosit pentru a imprima clienților.

**17. Ce este un Deadlock și când el apare ?**

**Deadlock** este o situație în care un set de procese sunt blocate, deoarece fiecare proces deține o resursă și așteaptă o altă resursă dobândită de un alt proces.

Deadlock poate apărea dacă următoarele patru condiții se mențin simultan:  
Mutual Exclusion: Una sau mai multe resurse nu pot fi folosite (un singur proces poate fi utilizat simultan)  
Hold and Wait: un proces reține cel puțin o resursă și așteaptă resurse.  
No Preemption: o resursă nu poate fi preluată dintr-un proces decât dacă procesul eliberează resursa.  
Circular Wait: Un set de procese se așteaptă reciproc în formă ciclica.

**18. Pentru ce este nevoie de metoda join() în programarea multithreading?**

**Join([Timeout])**  
  
E nevoie de așteptat până se termină threadul. Acest lucru blochează threadul apelant până când threadul a cărui metodă join() este apelat, nu se va termina, fie în modul normal, fie printr-o excepție neprelucrata, sau până când intervine un timeout opțional.

Utilizarea principala a join(), este așteptarea unui alt thread și începutul executarii lui, în momenutl în carea threadul curent și-a finisat exeutia sau a murit.