# Precizati nivelul de care apartine UDP si enumerati caracteristicile si functiile sale.

# UDP apartine nivelului transport.

# Protocol de transport fara conexiune

# Ofera aplicatiilor o modalitate de a trimite datagrame IP incapsulate si de a le trimite fara a fi nevoie sa stabileasca o conexiune

# Transmite segmente : antet + informatie utila

# Ofera protocolului IP o interfata cu facilitate adaugate de demultiplexare a mai multor procese, folosind porturi

# Nu realizeaza controlul fluxului

# Nu realizeaza controlul erorilor

# Nu realizeaza retransmiterea segmentelor primite gresit

# !!! Util in situatiile client-server : clientul trimite o cerinta scurta serverului si asteapta inapoi un raspuns scurt

# DNS utilizeaza acest protocol

# Antetul UDP

# 32 biti

# Porturi = punctele terminale ale masinilor sursa si destinatie

# Lungime UDP = include antetul de 8 octeti + date

# Suma de control = 0 (initial)

# 

# Explicati motivele pentru care proiectantul unei aplicatii ar alege UDP in loc de TCP.

Proiectantul unei aplicatii ar alege UDP in loc de TCP, daca latenta este foarte importanta, livrarea tuturor datelor nu este necesara, sau daca doreste sa implementeze singur retransmisiile, controlul fluxului si controlul erorilor. El ar putea alege UDP in locul TCP, daca ar dezvolta o aplicatie de tipul: voice-over-ip, un joc online sau o de tipul client- server (cerere-raspuns).

# Descrieti principiul paradigmei client-server.

Clientul si serverul sunt programe.

* Serverul: asteapta pasiv sa fie contactat
* Clientul: activ initializeaza comunicatia
* Un server poate fi client pentru alt server.

Clientul:

* + program obisnuit, invocat de utilizator
  + ruleaza local (PC – utilizator)
  + initializeaza activ contactarea serverului
  + poate accesa mai multe servicii daca este necesar
  + nu necesita hardware sau software special

Server:

* + program privilegiat, cu destinatie speciala
  + ruleaza pe un calculator partajat
  + asteapta pasiv sa fie contactat de clienti
  + necesita hardware si software performant
  + trebuie sa contina cod pentru autentificarea si autorizarea utilizatorilor, securitatea si confidentiaitaeta si protectia datelor

# Precizati operatiile (apeluri de functii) necesare pentru realizarea unei comunicatii client- server.

SERVER: SOCKET → BIND → LISTEN → ACCEPT → SHUTDOWN → CLOSE CLIENT: SOCKET → CONNECT → SEND/RECEIVE → SHUTDOWN → CLOSE

# Caracteristici protocol TCP

# Fiecre octet al unei conexiuni TCP are propriul sau numar de secventa (32biti) – confirmari, mecanismul de secventiere;

# Entitatile TCP de transmisie si de receptive interschimba informatie sub forma de segmente – un segment TCP consta dintr-un antet de 20 octeti urmat de 0 / mai multi octeti de date;

# Fiecare octet are un deplasament unic fata de inceputul mesajului;

# Exista 2 limite care restrictioneaza dimensiunea unui segment:

* + Fiecare segment + antet trebuie sa incapa in cei 65.535 octeti de informatie utila IP;
  + Fiecare retea are o unitate maxima de transfer MTU => fiecare segment trebuie sa incapa in acest MTU (in general 1500 octeti, definind astfel o limita superioara a dimensiunii unui segment).

# Protocolul de baza utilizat este protocolul cu fereastra glisanta:

* + E transmite un segment, el porneste un cronometru
  + Atunci cand segmental ajunge la destinatie, R trimite inapoi un segment (contine numarul de secventa).
  + Daca cronometrul E depaseste timpul inaintea primirii confirmarii, E transmite segmentul neconfirmat.

# Segmentele pot ajunge in ordine arbitrara, pot intarzia pe drum un interval de timp sufficient de mare pentru ca E sa detecteze o depasire a cronometrului si sa le retransmita

# proiectat pentru a asigura flux de octeti sigur de la un capat la celalat chiar si intr-o retea nesigura

# realizeaza controlul fluxului

# realizeaza controlul congestiei

# se asigura transmisie in ordine si sigura

# duplex integral – traficul se poate desfasura in ambele sensuri in acelasi timp

# punct-la-punct – fiecare conexiune are exact 2 puncte finale

# nu suporta difuzarea partiala sau totala

# flux de octeti, nu flux de mesaje

# atunci cand o aplicatie trimite date catre TCP, el le poate expedia imediat sau le poate retine intr-un tampon

# Informatia urgenta

# foloseste “intelegerea in 3 pasi” pentru a stabili o conexiune sigura

# Rolul + Functii TCP

Transmission Control Protocol este un [protocol](https://ro.wikipedia.org/wiki/Protocol_(informatic%C4%83)) folosit de obicei de aplicații care au nevoie de confirmare de primire a [datelor](https://ro.wikipedia.org/wiki/Dat%C4%83). Efectuează o conectare virtuală full duplex între două puncte terminale, fiecare punct fiind definit de către o [adresă IP](https://ro.wikipedia.org/wiki/Adres%C4%83_IP) și de către un port TCP.

Functii:

* Ofera livrarea sigura a datelor pe o retea nesigura
* Stabileste o conexiune intre client si server
* Realizeaza controlul congestiei, adaptand viteza de transmisie
* Este full duplex, are confirmare
* Protocolul de baza utilizat de TCP = cu fereastra glisanta
* In TCP, conexiunile sunt stabilite utilizant “Three Way Handshake”.

# Antetul segmentului TCP

# Port sursa, Port destinatie = identifica punctele finale ale conexiunii

# Numar de secventa, Numar de confirmare =

# semnificatia functiilor lor uzuale (32biti)

# Lungimea antetului = numar de cuvinte de 32 biti

# Care sunt continute in antetul TCP

# 6 biti neutilizati

# 6 indicatori de cate 1 bit

# Dimensiunea fereastra = numarul de octeti care pot fi trimisi incepand de la octetul confirmat

# Suma de control

# Optiuni = adaugarea unor facilitti suplimentare

# Cum se face initializarea conexiunii in TCP

SYN – bitul din antet folosit pentru stabilirea unei conexiuni Cerere de conexiune: SYN = 1, ACK = 0

Raspuns: SYN = 1, ACK = 1

→ bitul SYN este folosit pentru a indica o **cerere de conexiune**, **conexiune acceptata**, bitul ACK facand diferenta intre cele 2 posibilitati.

# Foloseste three way handshake: SYN→ SYN → ACK

# Controlul fluxului TCP

Atunci cand se stabilieste o conexiune, receptorul poate specifica dimensiunea ferestrei proprii. Astfel fiecare emitator mentine 2 ferestre: fereastra receptorului si fereastra de congestie. Numarul de octeti ce poate fi transmis este minimul dintre cele 2 dimensiuni.

# Cum asigura TCP corectitudinea datelor

La transmise:

* aduna cuvinte de 16 biti in complement fata de 1
* complementeaza rezultatul
* scrie rezultatul in antent La receptie:
* aduna cuvinte de 16 biti → rezultatul trebuie sa fie 0

→ Acknowledgement number: urmatorul octet asteptat

→ Corectarea sa face prin retransmisie

# Protocolul RTP

RTP este un protocol de transfer. Prin intermediul sa se pot prelucra date multimedia. Este un protocol nesigur, nu asigura corectitudinea datelor. Functia sa principala este de multiplexare a fluxurilor RTP in fluxuri UDP.

# Comparatie RTP, UDP, TCP (d.p.d.v al erorilor)

UDP si RTP nu asigura corectitudinea datelor, asa ca au aceleasi erori (duplicate, modificate, pierdute). TCP este un protocol sigur asa ca are al tip de erori decat protocoalele fara conexiune.

# Servicii transport neorientate vs servicii de confirmare

Serviciile de transport neorientate pe conexiune

* SERVERUL: deschide un socket → asculta la o adresa → primeste/trimite mesaje

→ inchide trimiterea/primirea → inchide socketul

* CLIENTUL: creaza un socket → trimite/primeste → inchide trimiterea/primirea → inchide socketul

Serviciile de transport de confirmare

* SERVERUL: creaza un socket → asculta la o adresa → stabileste un numar maxim de cereri pe care le poate avea (listen) → accepta → repreat → primeste date
* CLIENTUL: creaza un socket → se conecteaza la server → trimite date →

elibereaza resursele conexiunii → informeaza serverul despre inchiderea conexiunii.

# Ce este un socket?

Un socket este punctul in care procesul de aplicatie se ataseaza la retea (foloseste descriptori pentru identificare si este bazat pe modeul fisierelor open-read-write-close).

# Restrictionarea segmentelor la TCP

Exista doua limite care restrictioneaza dimensiunea unui segmenet.

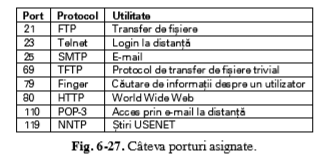
In primul rand fiecare segment, inclusiv antetul TCP trebuie sa incapa in cei 65535 (2^16 – 1) de octeti de informatie utila IP.

In al doilea rand, fiecare retea are o unitate maxima de transfer sau MTU, deci fiecare segment trebuie sa incapa in aceste MTU, in realitate, MTU este in general de cateva mii de octeti, definind astfel o limita superioara a dimensiunii unui segment. Daca un segment parcurge o secventa de retele fara a fi fragmentatat si apoi ajunge la o reta ai carui MTU este mai mica deat dimensiunea segmentului, ruterul de la frotiera acelei retele fragmenteaza segmentul in doua sau mai multe segmente mai mici.

# UDP vs TCP

UDP-ul este un protocol simplu şi are anumite nişe de utilizare, cum ar fi interacţiunile client-server şi cele multimedia, dar pentru cele mai multe aplicaţii de Internet este necesar un transport de încredere, secvenţial al informaţiei. UDP-ul nu poate oferi acest lucru, deci este nevoie de un alt protocol. Acesta este TCP şi este pionul principal de lucru al Internet-ului.

# Port

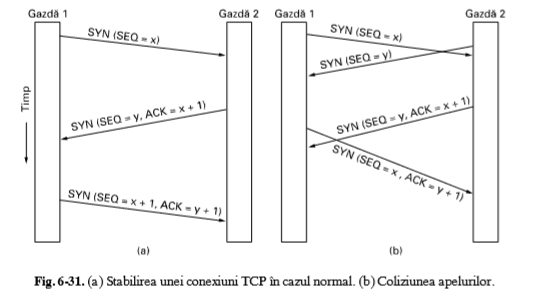
Serviciul TCP este obţinut prin crearea atât de către emiţător, cât şi de către receptor, a unor puncte finale, numite socluri (sockets). Fiecare soclu are un număr de soclu (adresă) format din adresa IP a maşinii gazdă şi un număr de 16 biţi, local gazdei respective, numit port. Pentru a obţine o conexiune TCP, trebuie stabilită explicit o conexiune între un soclu de pe maşina emiţătoare şi un soclu de pe maşina receptoare.

Numerele de port mai mici decât 256 se numesc porturi general cunoscute şi sunt rezervate serviciilor standard.

# Conexiunea TCP - Caracteristici

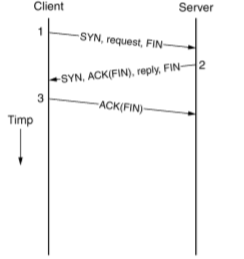
* + duplex integral – traficul se poate desfasura in ambele sensuri in acelasi timp
  + punct-la-punct – fiecare conexiune are exact 2 puncte finale
  + nu suporta difuzarea partiala sau totala
  + flux de octeti, nu flux de mesaje
  + atunci cand o aplicatie trimite date catre TCP, el le poate expedia imediat sau le poate retine intr-un tampon
  + Informatia urgenta

# Stabilirea conexiunii TCP

* Intelegerea in 3 pasi:
  + - Pentru a stabili o conexiune, una din parti – serverul – asteapta in mod pasiv o cerere de conexiune prin executia primitivelor LISTEN si ACCEPT, putand specifica o sursa anume sau nicio sursa in particular;
    - Cealalta parte – clientul - executa CONNECT, indicand adresa IP si numarul de port la care doreste sa se conecteze, dim maxima a segmentului TCP pe care este dispusa sa o accepte (optional: o informatie utilizator). CONNECT trimite un segment TCP ( SYN =1, ACK=0) si asteapta raspuns;
    - Atunci cand ajunge la destinatie un segment, R verifica daca nu cumvaexista un proces care a executat LISTEN pe numarul de port specificat in campul Port destinatie. Altfel, trimite TCP cu RST =1, pentru a refuza conexiunea.

# Eliberarea conexiunii TCP

* Pentru eliberarea unei conexiuni, orice partener poate expedia un segment TCP avand FIN =1.
* Atunci cand FIN este confirmat, sensul de comunicare este efectiv oprit pentru noi date
* Conexiunea = desfiintata : ambele directii au fost oprite
* Pentru a elibera o conexiune sunt necesare: FIN si ACK pentru fiecare sens.
* Pentru a evita problema celor 2 armate : folosim cronometre. Daca un raspuns la un FIN nu este receptionat pe durata a cel mult 2 cicluri, emitatorul FIN elibereaza conexiunea.



# TCP Tranzactional

# Permite transfer de date odata cu stabilirea conexiunii.