Retea: dirijare, adresare / datagrame, CV

- Datagrame: fara conexiune, best effort, nu corectitudine/ ordine, pachet contine IP sursa/ destinatie
- CV : conexiune, control flux, pachetu contine ID-u conexiunii (pentru tabele rutare)

+ **IP**:

- Header 20 bytes
- Clase adrese: 7/14/21 biti prefix, 24/16/8 sufix
- Router primeste pachet si-l livreaza gazdei daca e in aceeasi retea sau il da urmatorului router (nextHop din tabela de dirijare)
- Forwarding IP: <retea, gazda> destinatie => retea apare in tab de dirijare -> retea conectata direct (trimite pachet la gazda) else trimite datagrama la nextHop/ else trimite la ruter implicit
- Livrare directa in retea => se foloseste adresa fizica (pentru mapare se fol tabele de coresp, algoritmi de calcul, schimb de mesaje – ARP)

+ **ARP**:

- W difuzeaza o cerere ARP cu adr IP cunoscuta catre toate hosturile din retea si ala cu adr IP coresp trimite mesaj inapoi cu adr fizica

Spatiu de adrese ocupat inefficient => se fol subretele (o retea de clasa B e impartita in mai multe subretele apropiate geographic) sau adr fara clase

+ **CIDR**:

- Aloca spatiu de adr in blocuri de dif lungimi si are notatie speciala 194.23.0.0/21 unde 21 biti adr retea, 11 biti adr host
- Pentru a afla adr retea se fol masti (in fctie de cati biti sunt indicate dupa slashu ala)
- Reguli pentru a avea o masca: lungime bloc putere a lu 2 si sa inceapa de la un offset multiplu de lungime bloc (toate adresele din bloc au aceeasi adr retea)
- Forwarding: se allege intrarea din tabela pentru care adresa IP & masca = adresa retea/ daca sunt mai multe potriviri o ia pe cea mai lunga
- Reducere dimensiune tabela rutare prin agregare adrese(daca mai multe incep cu aceeasi secventa de biti, tabela dirijare include o singura inregistrare comuna pentru toate)

+ **NAT**:

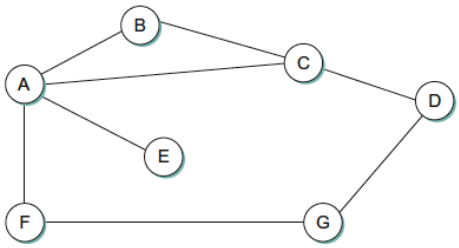
- Adr locala -> adr globala
- Foloseste tabela de translatare index|IP local|global|port
- Transmite: inloc adr locala cu globala, memoreaza in tabela IP-port, inloc nr port cu index, recalc checksum TCP si IP
- Receptie: obt nr port, IP local din pachet fol index din tabela, le inloc in pachet si calculeaza checksum TCP si IP

+ **ICMP**:

- Fol IP pt transmisie/ IP fol ICMP pentru raportare erori
- Ping = ICMP Echo si asteapta rasp/ traceroute trimite o serie de datagrame cu time to live crescatoare si primeste ICMP time exceeded din care extrage adr router)
- **Path MTU** fol ICMP: fol mesaje eroare ICMP frag needed and DF was set (cand datagramele trimise au dimens mai mare decat MTU)

+ **Dirijare**:

- Vectorii distantelor : fiecare nod trimite periodic vecinilor distantele de la el la alte noduri. Tabela cu toate nodurile din retea, iesirea preferata folosita pentru destinatia respective si timpu(determinat prin trimitere pachet ECHO) Bellman-Ford, Ford-Fulkerson



- **Probl nr la infinit**: cade nodu A si celelalte noduri ii spun ca stiu o cale spre E (da caile alea il contin si pe A – punct din care nu se mai poate ajunge laE – iar distantele vor fi incrementate cu 1 mereu => nr la infinit). Solutie: split horizon (noile distante nu se transmit vecinului prin care rec actualele rute). Solutille nu functioneaza pentru bucle cu mai multe noduri.

- **RIP**: fol distante la retele(nu la noduri), transmite vectorii de distante la 30 de secunde, pt retele de mici dimensiuni

- **Starea legaturii**: fiecare nod poate gasi legaturile cu vecinii si costurile. Informatii obtinute prin inundare(trimite un pachet cu ID nod care care creeaza pachet, lista nodurilor conectate + costuri, nr de secventa, lifetime). Cu info primite fiecare nod calculeaza rutile cele mai scurte catre celelalte noduri.

  ID si nr secv => nodu are o copie a pachetului sau pachetu e vechi => ignorant
  Lifetime => decrementat la fiecare hop -> eliminare pachete vechi
- **Structura ierarhica a internetului:** internet partajat in domenii num AS. Intr-un AS se fol acelasi alg de rutare(OSPF), la fel si intre ASuri (BGP)
- **Struct ierarhica AS**: AS partitionat in mai multe zone (backbone, zone stub), rutile intre noduri din zone stub trec prin backbone.
  Ierarhizarea creste scalabilitate: un nod din un AS nu tre sa stie cum se ajunge la fiecare retea, e sufficient sa stie cum se ajunge la zona ce contine reteaua.
  Tip rutere: backbone, interne, de granite zonala, de granita AS
- **OSPF** : pachete cu 89 nr de protocol – Hello, link state request/ update/ ack

Calcul rute(zona): inundare (fiecare ruter informeaza celelalte rutere din zona despre leg sale si costurile lor). Fiecare ruter calc rutele cele mai scurte catre ruterele din aceeasi zona(inclusive alea de granite zonala)

Calcul rute(AS) : calea unu pachet sursa –> backbone -> destinatie. Ruterele backbone primesc info de la ruterele de granite zonala si calc cele mai bune rute la retelele din orice zona. Rezultatele sunt difuzate de la backbone inapoi la zonele stub pentru a stii care sunt rutile cele mai bune catre retele din alte zone.

- **BGP** : reteaua e formata din noduri(ASes si conexiuni), protocol vector distantelor, tabelele de dirijare contin si ruta efectiva spre destinatii si le comunica vecinilor(ocoleste numararea la infinit) => daca se defecteaza un nod sunt eliminate toate caile ce il contin din tabela
- **Dirijare in retele AD Hoc (in pdf)**

Pachete ROUTE REQUEST

A difuzează un pachet ROUTE REQUEST
- Identificat unic prin Source address + Request ID
- Foloseşte Sequence # pentru a deosebi rutele noi de cele vechi

Prelucrarea ROUTE REQUEST în fiecare nod
- Verifica duplicat în tabela history locală (Source address + Request ID)
- Transmite ROUTE REPLY dacă găsit ruta nouă, adică
  - Dest sequence # în routing table > Dest sequence # în packet
- Altfel,
  - incrementează Hop count şi re-difuzează ROUTE REQUEST
  - memorează informaţia în reverse route table
- Source sequence # folosit pentru actualizare tabela dirijare locala

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count

Pachete ROUTE REPLY

I construieşte ROUTE REPLY şi-l trimite pe legătura inversă
- Source address, Destination address sunt copiate
- Hop count pus pe zero
- Destination sequence # luat din contorul propriu
- Lifetime = cât timp rămâne valid

Prelucrarea la alte noduri
- Actualizează tabela dirijare locală
- Transmite pe legătura inversă
- Trece prin anumite noduri – celelalte şterg intrarea în reverse route table

Source address	Destination address	Destination sequence #	Hop count	Lifetime

Transport : reteaua e pt comunic intre hosturi, transport pt comunic intre aplicatii

DNS: spatiu de nume(org ierarhic, fiecare nod are asociat un set de info pastrat in baze de date DNS), servere DNS(administreaza zone DNS, pastreaza BD cu info necesare clientilor in inregistrari de resurse RR)

Info primara despre zone e pastrata in fisiere master aflate in sistemu local de fisiere al serverului DNS. Server de nume primar => raspunde cererilor resolverului, foloseste Master files pentru a actualize BD pentru una sau mai multe zone.

Nume > IP : program(gethostbyname) > resolver(trimite un DNS REQUEST) > cache(time to live din RR = atat ramane inregistrarea in cache) /name server local (da DNS REPLY)

Format mesaje: Header|Question|Answer(RR answering the question)|Authority(RR pointing to authority)|Additional(RR holding additional info). Header(sectiuni prezente in mesaj, tip mesaj – intrebare/ rasp sau alta op), question(intrebare – tuplu nume domeniu, tip, clasa)

Un server DNS e server autoritate pt numele gestionate. Daca cererea contine nume gestionat de alt srv, ala autoritate trite request mai departe.

Exista mai multe servere root, adresele lor fiind copiate din fisieru de config in cache DNS la pornirea serverului.

Rezolvare recursive: cererea e pasata de la un server la altu si se intoarce pe calea inversa.

Rezolvare iterativa: daca server DNS nu poate afla intregu nume, trimite clientului partea nerezolvata si numele serverului acre o poate rezolva + client trimite cerere noua serverului primit

Cereri inverse: cauta nume pentru un IP => domeniu special in-addr.arpa(nodurile sunt numite dupa numerele din adr IP) contine inregistrari PTR in care numele sunt adrese IP. Clientu face cerere PTR pentru numele <adresa>.in-addr.arpa, se efectueaza cautare in inregistrarile PTR si se intoarce numele domeniului. Aplicatie in tracert pt afisare nume rutere.

Replicarea serverelor: Fiecare zona tre sa aiba mai multe servere DNS, unu primar pe care se fac modificarile inregistrarilor folosind Master Files si secundar care preia info de la ala primar => facilitati : transfer toata zona: server secundar cere SOA, verifica daca serial nr e mai amre decat cel local, daca da cere toata zona si info aferente(cerere AXFR)

FTP: are doua conexiuni, date(pt transmisie/ receptive simultana, User/Server DTP) si control(pt comenzi si raspunzuri, User PI initiaza conexiunea de control si genereaza comenzi FTP, Server PI genereaza raspunsuri FTP).

Comenzile au 3-4 caractere + parametri. Raspunsuri = cod nr [blanc] text.

Conexiunea de date:

- Active: serveru se conecteaza la client => client specifica IP si port, server initiaza conexiunea de date
- Pasiva: client initiaza conexiunea la server => client cere serverului sa asculte pe IP si port pe conexiunea de control, server comunica adresa si port

Transf intre doua server: client se conecteaza pasiv la un server si la altu active si le puna sa vb intre ele.

Securitate: FTP(user/ pass clar), FTP over SSH(pass criptat), SFTP(securitate pt date, server autentifica useru), bbFTP(secure user/pass, pt vol mari, pt streamuri paralele de date)

Mail:

- Agent utilizator pentru scriere/ citire mesaje (e interfata cu utilizatoru)
- Agent de transfer suporta transmiterea mesajelor de la sursa la destinatie (agentu client preia un mesaj, stabileste conexiune cu agentu server si transmite mesaju, agentu server primeste mesaju si il plaseaza in cutia postala). Agenti = demoni care lucreaza in fundal
- Mesaju e de forma plic(adr destinatar, prioritate, nivel securitate – fol de protocol)|header(fol de agentu utilizator, perechi nume – valoare referitoare la utilizatori si la continutu mesajului)|body(info destinata utilizatorului)
- Adrese email: user@nume_server_mail – user are specific local, fol de server pentru livrarea mesajelor/ nume_server_mail e num de domeniu fol de client care rezolva numele destinatarului folosind DNS, contacteaza server si transmite mesaju
- Continut header: To, From, Sender, Received, Return-Path fol de agentu de transfer
- Antete adaugate de MIME: MIME-Version, Content-Description/ID/Transfer-Encoding/Type. Encoding indica ce codificare a fost fol pt a converti datele din MIME in ASCII(SMTP: base64, 7bit, quoted-printable)
- **SMTP**:fol TCP si un schimb de mesaje text intre client si server(nume comanda + parametri/ raspunsuri = nr de 3 cifre + text), livrare sigura a mesajelor, port 25

  Serveru trimite clientului un mesaj prin care-l anunta ca e gata sa primeasca mesaje, client anunta de la cine e mesaju sip t cine. Daca receptoru exista server ii da permisiunea sa trimita mesaju.

  Problema: daca nu sunt ambii conectati nu se trimite mesaju => mesajele sunt salvate de agentu de transfer si trimise clientului cand vine online

Porti de email(mail gateways) asigura o adresa unica pentru mesajele trimise unui grup, un exploder cerand transmiterea unei copii a mesajului fiecarei adrese din lista

- **IMAP**: pastreaza info la server
- **POP3**: descarca mesaj in PC si sterge de pe server

HTTP:

- Cache: server/ proxy – partajat, client – privat; control caching antet Cache-Control public/ private/ no-cache
- Consistenta cache: HEAD si verific last modified/ GET cu antet if-modified-since/ Performanta: raspunsu serverului include antet Expires si client memoreaza(client verifica existenta cache, nu exista – download pagina, exista expirata – da cu if modified since si daca primeste 304 not modif fol ce e in cache, exista neexpirata fol din cache)
- Autentificare: permite accesu la pagini protejate prin antet de Authorize(user si pass ttrimise codat base64). Clientu cere pag protejata, server rasp cu 4019 WWW-Authenticate), browser trimite cerere cu antet Authorization, server verifica cererea si satisfice sau refuza cu 403. Dupa ce trimit credentials browser le trimite mereu in antet (in cadru domeniului respective)
- Cookie: intelegere initiate de server prin antet Set-Cookie: <nume>=<valoare> cu path si domain. Intelegerea e acceptata prin antet Cookie de catre client