RECAPITULARE RETELE

Def. O retea de calculatoare este o colectie interconectata de calculatoare. O retea poate fi definita in mod recursiv ca doua sau mai multe noduri conectate printr-o legatura fizica, sau prin doua sau mai multe retele conectate de unul sau mai multe noduri.

Tipuri de retele:

Dupa dispunerea spatiala:
– PAN (Personal Area Network)
– LAN (Local Area Network)
– MAN (Metropolitan Area Network)
– WAN (Wide Area Network)
– Internet

In functie de tehnologia de transmisie:
– Retele cu difuzare (un singur canal de comunicare) : broadcast, multicast
– Retele point-to-point : unicast

In functie de elementele componente:
– Omogena: retea de calculatoare folosind configuratii si protocoale similare. Exemplu: O retea folosind Microsoft Windows via TCP/IP
– Eterogena: retea care contine tipuri diferite de calculatoare, sisteme de operare si/sau protocoale diferite. Exemplu: LAN care conecteaza un smart phone cu Android si un computer Apple Machintosh

Componentele unei retele

- Gazda (eng. Host) – este un sistem computational conectat la Internet
- Hub (Hub Network) – dispozitiv (deseori amplificator de semnal) folosit pentru conectarea mai multor dispozitive => segment de retea (network segment)
- Switch (Switch)- dispozitiv care filtreaza si retrimite pachetele in retea
- Ruter (Router) – dispozitiv oferind conectivitatea intre retele individuale, realizind dirijarea pachetelor intre aceste retele 17
- Punte (Bridge) – dispozitiv care conecteaza doua sau mai multe segmente intr-o retea
- Poarta (Gateway) – este punctul de conectare a doua retele incompatibile
- Repeater - este un dispozitiv electronic care primeste semnale pe care le retransmite la un nivel mai inalt sau la o putere mai mare, astfel ca semnalul sa poata acoperi zone mari fara degradare a calitatii sale

Stiva de nivele

– Functionalitate:
    Interfata: asigura comunicarea intre doua nivele consecutive
    Serviciu: furnizeaza functionalitatea unui nivel
– Rezultat: reducerea complexitatii proiectarii
– Principiul de comunicare: ce transmite emitatorul la nivelul n este ceea ce se primeste la destinatar la nivelul n

Def. Protocol – regulile si conventiile prin care se realizeaza comunicarea.

Serviciul

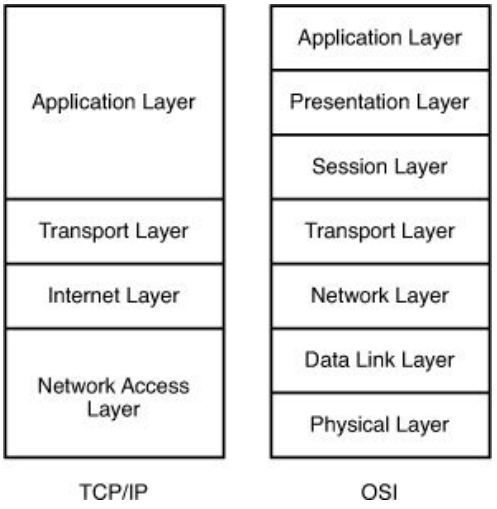
Specificarea serviciului este realizata printr-un set de primitive (operatii) puse la dispozitia celui ce foloseste serviciul(Serviciu <> Protocol).

Tipuri de servicii
– Orientat-conexiune (eng. connection-oriented)
    • Comunicarea necesita stabilirea unei conexiuni
    • Similar serviciului telefonic
– Fara conexiune (eng. connectionless)
    • Comunicarea nu necesita stabilirea unei conexiuni
    • Similar serviciului postal

Def. Arhitectura de retea - multimea de nivele si de protocoale. Specificatia unei arhitecturi trebuie sa ofere suficiente informatii pentru ca programele sau echipamentele destinate unui nivel sa indeplineasca protocoalele corespunzatoare

Def. Stiva de protocoale - lista de protocoale, de pe toate nivelele, utilizate de catre un anumit sistem.

Modele de referinta pentru arhitecturi de retea
- ISO/OSI (International Standard Organization/ Open System Interconnection)
- TCP/IP (Transmission Control Protocol/ Internet Protocol)



MODELUL OSI

- Nivelul Fizic:
  – Rol: asigura faptul ca secventa de biti transmisa de la emitator ajunge la receptor
  – Medii de transmisie:
    • Cu fir (cablu torsadat, cablu coaxial, fibre optice)
    • Fara fir (spectru electromagnetic - radio, microunde, infrarosii,...)
  – Transmiterea datelor:
    • Analogic (valori continue)
    • Digital (valori discrete)
  – Conversia datelor din format analogic în format digital si invers
    • Modem: date în format digital sunt transmise în format analogic

• Codec (coder/decoder): invers

– Aspecte

• Largimea de banda (Bandwidth): numarul de biti care pot fi transmisi pe retea intr-o anumita perioada de timp (viteza transfer de date, bits/secunda)

• Latenta: reprezinta intervalul de timp maxim necesar unui bit de a se propaga de la o extremitate la alta a retelei si se exprima in unitati de timp. RTT (Round Trip Time) este timpul necesar unui bit să traverseze de la un capăt la altul, şi înapoi mediul

– Modificari suferite de semnale in timpul propagarii in mediile de transmisie:

• Atenuarea: pierderea de energie în timpul propagării semnalului printr-un mediu de transmisie

• Zgomotul: modificarea semnalului cauzata de factori externi (e.g. fulgere, alte echipamente electronice etc)

• Distorsiune (engl. Distortion)- este o modificare determinista a semnalului receptionat fata de cel emis

- <u>Nivelul legatura de date:</u>
  – Ofera
    • mecanisme de detectie si corectare a erorilor
    • mecansime de reglementare a fluxului de date
    • mecanism de control al accesului la mediu
    • servicii nivelului retea
  – Datele se incapsuleaza in cadre(frame-uri)
  – Ofera servicii nivelului retea
    • Servicii neconfirmate fara conexiune (Emitatorul transmite cadre independente catre destinatar fara sa astepte confirmare; un cadru pierdut nu este recuperat)
    • Servicii confirmate fara conexiune (Se realizeaza confirmarea cadrelor trimise; transmiterea cadrelor nu se face in ordine)
    • Servicii confirmate orientate-conexiune (Inainte de transmiterea datelor se stabileste o conexiune; cadrele sunt numerotate pentru a se pastra ordinea)
  – Divizat in doua subniveluri:
    • Controlul logic al legaturii – LLC (Logical Link Control) – Rol: Ofera nivelelor superioare o vedere independenta de mediul de comunicare
    • Controlul accesului la mediu – MAC (Medium Access Control)
      – Rol: Folosit pentru a determina cine urmeaza sa transmita intr-un canal multi-acces
      – Strategii:
        – Alocare statica
          » FDM (Frequency Division Multiplexing)
          » TDM (Time Division Multiplexing)
        – Alocare dinamica - Acceptarea posibilitatii coliziunilor si retransmiterea pachetelor afectate de coliziuni
      – Protocoale:
        – ALOHA
          • Pure ALOHA : “transmite oricind doresti”
          • Slotted ALOHA
        – CSMA (Carrier Sense - Multiple Access): protocol cu detectia transmisiei – CSMA/CD (CSMA with Collision Detection)
        – MACA (Multiple Access with Collision Avoidance) - baza pentru retelele wireless
        – MACAW - imbunatateste MACA
  – Echipamente: Punti (bridges)

- Retransmit frame-urile dintre doua retele (LAN)
- Nu modifica continutul frame-urilor si pot schimba doar antetele acestora
- Imbunatatesc siguranta transmiterii si performanta
- Pot oferi controlul fluxului si congestiei datelor
- Retransmiterea datelor se realizeaza via rute statice sau folosind un arbore de acoperire

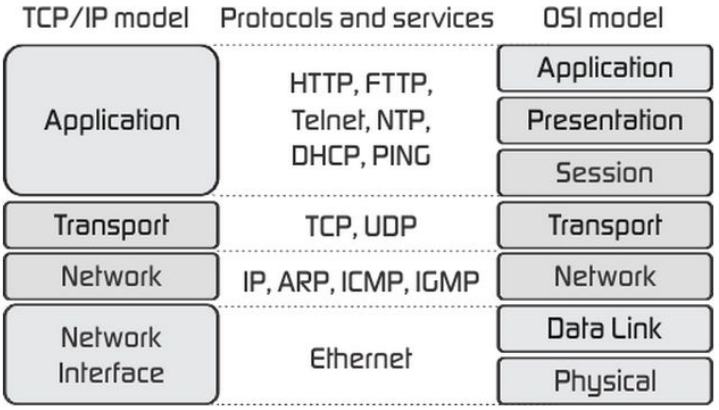
- Nivelul retea:
  – Preia pachetele de la sursa si le transfera catre destinatie
  – Ofera servicii nivelului transport
  –Protocoale folosite
    - X.25 (orientat conexiune)
    - IP
  –Probleme
    - Conversii de protocol si adrese
    - Controlul erorilor (flux, congestie)
    - Divizarea si recompunerea pachetelor
    - Securitatea – criptare, firewall

- Nivelul transport:ofera siguranta si cost-eficient in transportul datelor de la masina sursa la masina destinatie, independent de reteaua fizica sau retelele in prezent in uz
  – Servicii: ofera servicii orientate-conexiune si fara conexiune
  – Primitive:
    - LISTEN – se blocheaza pina cind un proces incearca sa se conecteze
    - CONNECT – incearca sa stabileasca o conexiune
    - SEND – trimite date
    - RECEIVE – se blocheaza pina se primesc datele
    - DISCONNECT – eliberarea conexiunii
  – Performanta – calitatea serviciilor (QoS – Quality of Service): stabilirea/eliberarea conexiunii, rata de eroare, protectia, prioritatea, rezilienta (probabilitatea ca o conexiune sa se inchida din ratiuni interne), duplicarea pachetelor, controlul fluxului

- Nivelul sesiune: se refera la probleme de stabilire de sesiuni (servicii de control al dialogului, de sincronizare etc.)

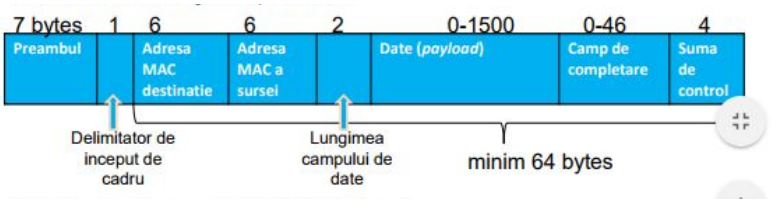
- Nivelul prezentare: se ocupa de prezentarea datelor, codificindu-le intr-un format standard
  – Pentru a se asigura comunicarea intre calculatoare cu reprezentari diferite, nivelul prezentare asigura conversia reprezentarilor interne a structurilor de date in reprezentare standardizata din retea si invers

- Nivelul aplicatie: gestioneaza servicii ale retelei: terminal virtual abstract, transfer de fisiere, posta electronica, executia la distanta a aplicatiilor, etc.



MODELUL TCP/IP

- Nivelul “fizic”
  – Asigura conectarea host-ului la retea
  – Fiecare interfata(placa) de retea are o adresa hardware (MAC) unica (48 de biti)
  e.g. C0:B3:44:17:21:17
  – Primii 24 de biti identifica producatorul
  – Forma unui cadru (frame) de date:



  – Broadcast: adresa are toti bitii setati pe 1
  – Fiecare interfata de retea inspecteaza pentru orice cadru adresa de destinatie
  – Daca adresa destinatie nu se potriveste cu adresa hardware sau cea de broadcast, atunci cadrul este ignorat
- Nivelul retea
  – Permite gazdelor sa emita pachete in orice retea; pachetele circula independent pina la destinatie
  – Aspecte principale:
    • Dirijarea pachetelor
    • Evitarea congestiei
  – Proiectarea nivelului a urmarit atingerea urmatoarelor obiective:
    • Serviciile oferite sunt independente de tehnologia utilizata (e.g. routere)
    • Asigura nivelului transport servicii, care ii permit acestuia sa functioneze in mod independent de numarul, tipul si topologia retelei
    • Furnizeaza un mecanism de adresare unic in LAN-uri si WAN-uri
  – IPv4
  – IPv6
  – Dirijare (routing):
    • OSPF(Open Shortest Path First) – RFC 1131
    • BGP(Border Gateway Protocol) – RFC 1105
  – Multicast:
    • IGMP (Internet Group Management Protocol) – RFC 1112, 1054
  – Control:
    • ICMP (Internet Control Messages Protocol) - RFC 792,777
    • SNMP (Simple Network Management Protocol) – RFC 1157
    • ICMPv6

- Nivelul transport
  – Asigura realizarea comunicarii intre gazda sursa si gazda destinatie
  – Protocoale
    • TCP (Transmission Control Protocol) - RFC 793,761
    • UDP (User Datagram Protocol) – RFC 768
    • Alte protocoale: SCTP (Stream Control Transmission Protocol),;DCCP (Datagram Congestion Control Protocol);

- Nivelul aplicatie:
  – Contine protocoale de nivel inalt
    – SMTP (Simple Mail Transfer Protocol)
    – POP3(Post Office Protocol)
    – TELNET
    – FTP (File Transfer Protocol)
    – NFS (Network File System)
    – DNS (Domain Name System)
    – HTTP (HyperText Transfer Protocol)
    – RTP (Real-time Transport Protocol)
    – SIP (Session Initiation Protocol)...etc

OSI VS TCP/IP

Asemanari:

• Ambele se bazeaza pe o stiva de protocoale
• Functionalitatile straturilor este oarecum asemanatoare
• Ambele au nivelul aplicatie ca nivel superior
• Se bazeaza (direct sau indirect) pe nivelul transport

Deosebiri:

• ISO/OSI este indicat ca model teoretic; TCP/IP este eficient in implementare
• OSI face explicita distinctia intre serviciu, interfata si protocol; TCP/IP nu
• ISO/OSI pune la dispozitie protocoale care asigura o comunicare fiabila (detectarea si tratare de erori la fiecare nivel); TCP/IP face verificarea comunicarii la nivelul transport
• OSI suporta ambele tipuri de comunicatii la nivel retea (fara conexiune si orientate conexiune); TCP/IP suporta la nivelul retea comunicatii fara conexiune si la nivelul transport ambele moduri

PROTOCOLUL IP

- Rol: ofera servicii neorientate-conexiune pentru a transporta datagrame de la sursa la destinatie; sursa si destinatia pot fi in retele diferite
- Fiecare datagrama este independenta de celelalte
- Nu se garanteaza trimiterea corecta a datagramelor (pierdere, multiplicare,...)

- Datagrama IPv4

Version: 4 pentru IPv4, 6 pentru IPv6, etc
Type of Service: permite gazdei sa comunice subretelei ce tip de serviciu doreste
Identification: permite gazdei destinatie sa identifice apartenenta la o datagrama a noului fragment primit
Flagurile: DF (Don't Fragment) - indica routerelor sa nu fragmenteze datagrama
MF (More Fragments) - semnalizeaza ca pachetul este un fragment, urmat de altele, ultimul fragment are acest bit 0.
Fragment Offset : locul fragmentului in datagrama
TTL(Time to Live): specifica durata de viata a pachetului, numarul este decrementat de fiecare router prin care trece pachetul
Protocol: specifica protocolul (de nivel superior) caruia ii este destinata informatia inclusa in datagrama
Header Checksum: folosit pentru detectarea erorilor, daca apare o eroare, datagrama este distrusa.
Source/Destination address: indica adresa sursei/destinatiei

- Rolul si arhitectura unui proxy:
  – Acces indirect la alte retele (Internet) pentru gazdele dintr-o retea locala via proxy
  – Proxy-ul poate fi software sau hardware
  – Rol de poarta (gateway), de firewall sau de server de cache
  – Proxy-ul ofera partajarea unei conexiuni Internet
  – Utilizat la imbunatatirea performantei (e.g., caching, controlul fluxului), filtrarea cererilor, asigurarea anonimitatii
- Fiecare adresa IP include:
  – NetID - identificator de retea
  – HostID - identificator de gazda
- Fiecare interfata(placa) de retea are o adresa IPv4 unica
- O gazda poate avea mai multe placi de retea, deci mai multe adrese IP
- Gazdele unei aceleiasi retele vor avea acelasi identificator de retea(NetID)
- Adresele de broadcast au HostID cu toti bitii 1
- Adresa IP care are HostID cu toti bitii 0 se numeste adresa retelei
- Din spatiul de adrese ce pot fi alocate efectiv sunt rezervate urmatoarele (RFC 1918):
  • 0.0.0.0 – 0.255.255.255
  • 10.0.0.0 – 10.255.255.255 (adrese private)
  • 127.0.0.0 – 127.255.255.255 (pentru loopback)
  • 172.16.0.0 - 172.31.255.255 (adrese private)
  • 192.168.0.0 - 192.168.255.255 (adrese private)
- Adrese private: adrese care nu sunt accesibile spre exterior (Internetul "real"), ci doar in intranetul organizatiei

PROTOCOLUL ICMP

- Utilizat pentru schimbul de mesaje de control
- Foloseste IP
- Mesajele ICMP sunt procesate de software-ul IP, nu de procesele utilizatorului
- Utilizat de:
  • comanda ping (Packet Internet Gropher)

• comanda traceroute

Def. Rezolutia adresei este procesul de a gasi adresa hardware a unei gazde, stiind adresa IP - protocolul ARP(address resolution protocol)

ICMPv6

– Ofera functiile ICMP (raportarea transmiterii datelor, erorilor, etc.) plus:
  • Descoperirea vecinilor (Neighbor Discovery Protocol – NDP) - Inlocuieste ARP
  • Descoperirea ascultatorilor multicast (Multicast Listener Discovery) – inlocuieste IGMP (Internet Group Management Protocol)

MODELUL CLIENT/SERVER

- Proces server
  – Ofera servicii in retea
  – Accepta cereri de la un proces client
  – Realizeaza un anumit serviciu si returneaza rezultatul
- Proces client
  – Initializeaza comunicarea cu serverul
  – Solicita un serviciu, apoi asteapta raspunsul serverului
- Moduri de interactiune
  – Orientat-conexiune – bazat pe TCP
  – Neorientat-conexiune – bazat pe UDP
- Implementare:
  – Iterativa - fiecare client e tratat pe rand, secvential
  – Concurenta - ceretile sunt procesate concurent; procese copil pentru fiecare cerere de procesat, multiplexarea conexiunii, tehnici combinate
- Se pot utiliza mai multe API-uri(Application Programming Interface) pentru programarea aplicatiilor internet
  – Socketuri BSD(Berkeley System Distribution)
  – TLI (Transport Layer Interface)
  – Winsock
  – MacTCP
- TCP/IP nu include definirea unui API

API BAZAT PE SOCKET-URI BSD

Socket

Def. Un socket este o facilitate generala, independenta de arhitectura hardware, de protocol si de tipul de transmisie a datelor, pentru comunicare intre procese aflate pe masini diferite, in retea.

- Utilizeaza interfata de programare I/O existenta (similar fisierelor, pipe-urilor, FIFO-urilor)
- Poate fi asociat cu unul/mai multe procese, existind in cadrul unui domeniu de comunicatie
- Ofera un API pentru programarea in retea, avind implementari multiple
- Din punctul de vedere al programatorului, un socket este similar unui descriptor de fisier; diferente apar la creare si la diferite optiuni de control al socketurilor

Primitive de baza:

- socket() – creaza un nou punct terminal al conexiunii
- bind() ataseaza o adresa locala la un socket
- listen() permite unui socket sa accepte conexiuni
- accept() blocheaza apelantul pina la sosirea unei cereri de conectare(utilizata de serverul TCP)
- connect() tentativa (activa) de stabilire a conexiunii (folosita de clientul TCP)
- send() trimitere de date via socket
- recv() receptarea de date via socket
- close() elibereaza conexiunea (inchide un socket)
- shutdown() inchide directional un socket

Alte primitive:

• Citire de date – read() / readv() / recvfrom() / recvmsg()
• Trimitere de date – write() / writev() / sendto() / sendmsg()
• Multiplexare I/O – select()
• Administrarea conexiunii – fcntl() / ioctl() / setsockopt() / getsockopt() / getsockname() / getpeername()

Socket(): – int socket (int domain, int type, int protocol)

#include <sys/types.h>
#include <sys/socket.h>

Valoare retur: -1 pentru eroare, altfel descriptorul de socket este creat

- Primitiva socket() aloca resursele necesare unui punct terminal de comunicare, dar nu stabileste modul de adresare

int socket (int domain, int type, int protocol)

Domeniu	Tip	Protocol
AF_INET	SOCK_STREAM	TCP
	SOCK_DGRAM	UDP
	SOCK_RAW	IP
AF_INET6	SOCK_STREAM	TCP
	SOCK_DGRAM	UDP
	SOCK_RAW	IPv6
AF_LOCAL	SOCK_STREAM	Mecanism intern de comunicare
	SOCK_DGRAM	

- Socket-urile ofera un mod generic de adresare; pentru TCP/IP trebuie specificate (adresa IP, port)

Bind(): – int bind (int sockfd, const struct sockaddr *addr, int addrlen)

- Asignarea unei adrese la un socket existent se realizeaza cu bind()
  – returneaza -1 pentru eroare, 0 pentru succes.
- Utilizari ale lui bind():
  – Serverul doreste sa ataseze un socket la un port prestabilit (pentru a oferi servicii via acel port)
  – Clientul vrea sa ataseze un socket la un port specificat
  – Clientul cere sistemului de operare sa asigneze orice port disponibil
- In mod normal, clientul nu necesita atasarea la un port specificat
- Alegerea oricarui port liber: adresa.sin_port = htons(0);
- Pentru a atasa un socket la adresa IP locala, se va utiliza in locul unei adrese IP constanta INADDR_ANY

Listen():

- Alegerea valorii backlog depinde de aplicatie
- Serverele HTTP trebuie sa specifice o valoare backlog cit mai mare (din cauza incarcarii cu cereri multiple)

Accept(): – int accept (int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);

- Acceptarea propriu-zisa a conexiunilor din partea clientilor
  – Cand aplicatia este pregatia pentru a trata o noua conexiune, va trebui sa interogam sistemul asupra unei alte conexiuni cu un client
  – addrlen trebuie initial sa fie egal cu lungimea structurii cliaddr; se va returna numarul de bytes folositi in cliaddr

Connect(): – int connect (int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen)

- Incercarea de a stabili o conexiune cu serverul
  – se returneaza 0->succes, -1->eroare

Read(): – int read(int sockfd, void *buf, int max);

- Apelul este blocant in mod normal, read() returneaza doar cind exista date disponibile
- Citirea de la un socket TCP poate returna mai putini octeti decat numarul maxim dorit – Trebuie sa fim pregatiti sa citim cate 1 byte la un moment dat
- Daca partenerul a inchis conexiunea si nu mai sunt date de primit, se returneaza 0 (EOF)
- Erori: EINTR – un semnal a intrerupt citirea, EIO – eroare I/O, EWOULDBLOCK – socket-ul nu are date intr-o citire neblocanta

Write(): – int write(int sockfd, const void *buf, int count);

- Apelul este blocant in mod normal
- Erori:
  – EPIPE – scriere la un socket neconectat
  – EWOULDBLOCK – nu se pot accepta date fara blocare, insa operatiunea este setata ca find blocanta

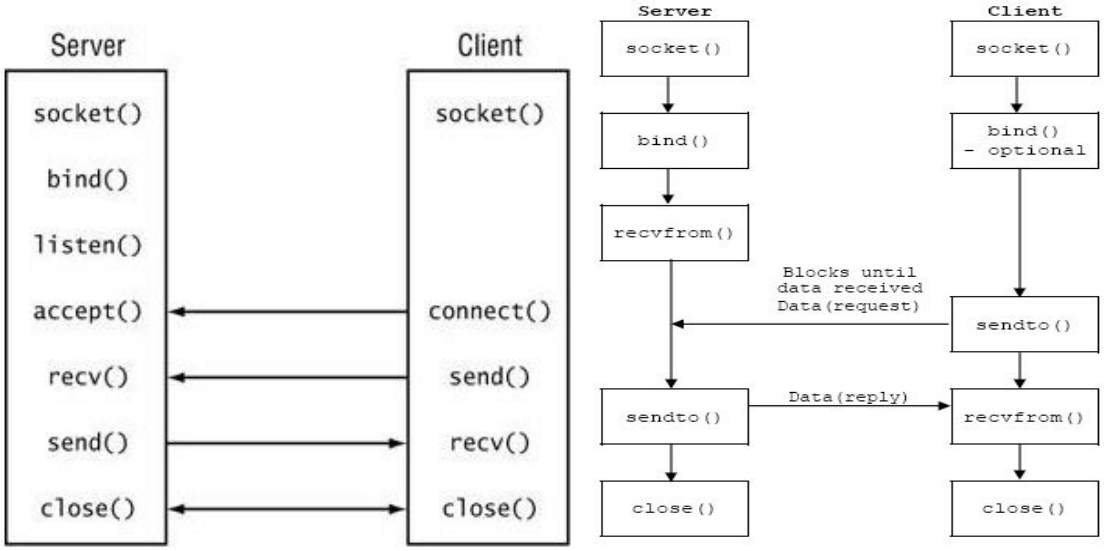
Close(): – int close(int sockfd)

- Efect:
  – terminarea conexiunii;
  – dealocarea memoriei alocate socket-ului

- Probleme:
  – serverul nu poate termina conexiunea, nu stie daca si cind clientul va mai trimite si alte cereri
  – clientul nu poate sti daca datele au ajuns la server

Shutdown(): – int shutdown (int sockfd, int how);

- Inchidere unidirectionala
  – Cind un client termina de trimis cererile, poate apela shutdown() pentru a specifica faptul ca nu va mai trimite date pe socket, fara a dealoca socket-ul
  – Serverul va primi EOF si, dupa expedierea catre client a ultimului raspuns, va putea inchide conexiunea



TCP **UDP**

CLIENT/SERVER UDP

- Pentru socket() se va folosi SOCK_DGRAM
- Apelurile listen(), accept(), connect() nu vor mai fi utilizate in mod uzual
- Pentru scriere de datagrame se pot folosi sendto() sau send() (mai general)
- Pentru citire de datagrame se pot folosi recvfrom() sau recv()
- Nimeni nu garanteaza ca datele expediate au ajuns la destinatar sau nu sunt duplicate
- Socket-urile UDP pot fi “conectate”: clientul poate folosi connect() pentru a specifica adresa (IP, port) a punctului terminal (serverul) – pseudo-conexiuni:
  – Utilitate: trimiterea mai multor datagrame la acelasi server, fara a mai specifica adresa serverului pentru fiecare datagrama in parte
  – Pentru UDP, connect() va retine doar informatiile despre punctul terminal, fara a se initia nici un schimb de date
  – Desi connect() raporteaza succes, nu inseamna ca adresa punctului terminal e valida sau serverul este disponibil
  – Se poate utiliza shutdown() pentru a opri directionat transmiterea de date, dar nu se va trimite nici un mesaj partenerului de conversatie
  – Primitiva close() poate fi apelata si pentru a elimina o pseudo-conexiune

int send (int sockfd, char *buff, int nbytes, int flags);
int recv (int sockfd, char *buff, int nbytes, int flags);

• Primele 3 argumente sunt similare cu cele de la write(), respectiv read()
• Argumentul a patrulea este de regulă 0, dar poate avea si alte valori care precizează condiții de efectuare a apelului
• Cele 2 apeluri returnează la execuție normală lungimea transferului în octeți

int sendto (int sockfd, char *buff, int nbytes, int flags, struct sockaddr *to, int addrlen);
int recvfrom (int sockfd, char *buff, int nbytes, int flags, struct sockaddr *from, int *addrlen);

• Sunt folosite pentru comunicatii neorientate conexiune
• La sendto() si recvfrom() elementele pentru identificarea nodului la distanță se specifică în apel, prin ultimele 2 argumente
• Cele 2 apeluri returnează la execuție normală lungimea transferului în octeți

MULTIPLEXAREA I/O

- Posibilitatea de a monitoriza mai multi descriptori I/O
  – Un client TCP generic (e.g., telnet) – Un client interactiv (e.g., ftp, scp, browser Web,...)
  – Un server care poate manipula mai multe protocoale (TCP si UDP) simultan
  – Rezolvarea unor situatii neasteptate (i.e. caderea unui server in mijlocul comunicarii)
- Exemplu: datele citite de la intrarea standard trebuie scrise la un socket, iar datele receptionate prin retea trebuie afisate la iesirea standard
- Utilizarea mecanismului neblocant folosind primitivele fnctl()
  – Se seteaza apelurile I/O ca neblocante

```
int flags;
flags = fcntl ( sd, F_GETFL, 0 );
fcntl( sd, F_SETFL, flags | O_NONBLOCK);
```

  – Daca nu sunt date disponibile un apel read() va intoarce -1 sau daca nu este suficient spatiu in buffer un apel write() va intoarce -1 (cu eroarea EAGAIN)
- Utilizarea mecanismului neblocant folosind primitivele ioctl()
  #include <sys/ioctl.h>
  ioctl (sd, FIOSNBIO, &arg)
    -arg este un pointer la un int
    -Daca int are valoare 0, socketul este setat in mod blocant
    -Daca int are valoare 1, socketul este setat in mod neblocant
- Utilizarea mecanismului asincron
  – Socket-urile asincrone permit trimiterea unui semnal (SIGIO) procesului
  – Socket-urile asincrone permit utilizatorului separarea “procesarilor socket” de alte procesari
  – Generarea semnalului SIGIO este dependenta de protocol
- Folosirea alarm() pentru a intrerupe apelurile de sistem lente
- Utilizarea unor procese/thread-uri multiple (multitasking)

- Folosirea unor primitive care suporta verificarea existentei datelor de intrare de la descriptori de citire multipli: select() si poll()
- select():
  • Permite utilizarea apelurilor blocante pentru un set de descriptori (fisiere, pipeuri, socket-uri,...)
  • Suspenda programul pana cand descriptori din liste sunt pregatiti de operatii de I/O
  #include <sys/time.h>
  #include <sys/types.h>
  #include <unistd.h>
  int select (int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout);
  Nfds - valoarea maxima a descriptorului plus 1
  fd_set(toate) - multimea descriptorilor de citire, scriere, exceptie
  Timeout - timpul de asteptare
  Un descriptor de socket este gata de citire daca:
  • Exista octeti receptionati in buffer-ul de intrare (se poate face read() care va returna >0)
  • O conexiune TCP a receptionat FIN (read() returneaza 0)
  • Socket-ul e socket de ascultare si nr. de conexiuni complete este nenul (se poate utiliza accept())
  • A aparut o eroare la socket (read() returneaza –1, cu errno setat) – erorile pot filtratate via getsockopt() cu optiunea SO_ERROR
  Un descriptor de socket este gata de scriere daca:
  – Exista un numar de octeti disponibili in buffer-ul de scriere (write() va returna > 0)
  – Conexiunea in sensul scrierii este inchisa (incercarea de write() va duce la generarea SIGPIPE)
  – A aparut o eroare de scriere (write() returneaza –1, cu errno setat) – erorile pot fitratate via getsockopt() cu optiunea SO_ERROR

  select() poate returna
  – Numarul descriptorilor pregatiti pentru o operatiune de citire, scriere sau exceptie
  – Valoarea 0 – timpul s-a scurs, nici un descriptor nu este gata
  – Valoarea –1 in caz de eroare
  Utilizarea lui select() – pasii generali:
  – Declararea unei variabile de tip fd_set
  – Initializarea multimii cu FD_ZERO()
  – Adaugarea cu FD_SET() a fiecarui descriptor dorit a fi monitorizat
  – Apelarea primitivei select()
  – La intoarcerea cu succes, verificarea cu FD_ISSET() a descriptorilor pregatiti pentru I/O

ALTE PRIMITIVE I/O

#include <sys/uio.h>
ssize_t readv (int filedes, const struct iovec *iov, int iovcnt);
ssize_t writev (int filedes, const struct iovec *iov, int iovcnt);

struct iovec {

```
        void *iov_base; /* adresa de start a bufferului */
        size_t iov_len; /* dimensiunea bufferului */
};
```

Mai generale decit read()/write(), ofera posibilitatea de a transmite date aflate in zone necontigue de memorie Cele 2 apeluri returneaza la execuție normala lungimea transferului in octeti.

```
#include <sys/socket.h>

ssize_t recvmsg (int sockfd, struct msghdr *msg, int flags);
ssize_t sendmsg (int sockfd, struct msghdr *msg, int flags);
```

Ambele functii au majoritatea optiunilor incorporate in structura msghrd Cele mai generale functii I/O; apelurile read/readv/recv/recvfrom pot fi inlocuite de recvmsg Cele 2 apeluri returnează la execuție normală lungimea transferului în octeți; -1 in caz de eroare.

Function	Orice descriptor	Doar descriptor de socket	Un singur read/write buffer	Scatter/ gather read/write	Flag-uri optionale	Adresa nodului *peer*
read, write	●		●			
readv, writev	●			●		
recv, send		●	●		●	
recvfrom, sendto		●	●		●	●
recvmsg, sendmsg		●		●	●	●

SERVER UDP CONCURENT

Majoritatea serverelor UDP sunt iterative – Server UDP care citeste cererea clientului, proceseaza cererea, trimite raspunsul si termina cu acel client.

Server UDP concurent
- daca elaborarea raspunsului ia mult timp serverul poate crea (fork()) un proces copil care va rezolva cererea.
- Server UDP care schimba datagrame multiple cu un client
– Problema: Doar un numar de port este cunoscut de client ca fiind un port “wellknown”
– Solutia: serverul creaza un socket nou pentru fiecare client, si il ataseaza la un port “efemer”, si utilizeaza acest socket pentru toate raspunsurile.
    • Obligatoriu clientul trebuie sa preia din primul raspuns al serverului noul numar de port si sa faca urmatoarele cereri la acel port
– Exemplu: TFTP - Trivial File Transfer Protocol

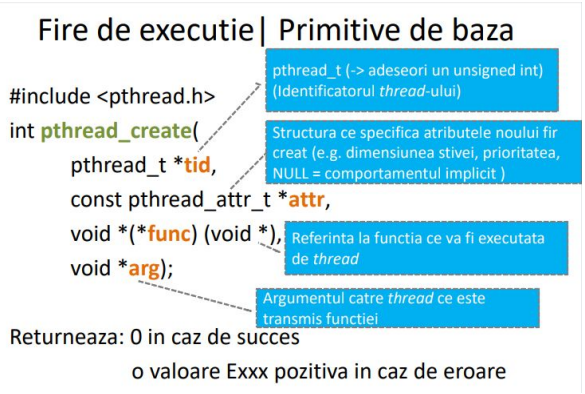
TCP VS UDP

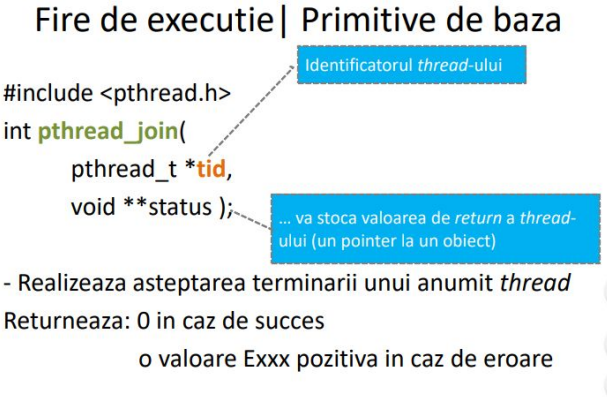
Aspecte privind utilizarea UDP:
- UDP suporta broadcasting si multicasting
- UDP nu are nevoie de un mecanism de stabilire a conexiunii
- Minimul de timp necesar unei tranzactii UDP cerere-raspuns este: RRT(Round Trip Time) + SPT (server processing time)

Aspecte privind utilizarea TCP:
- TCP suporta point-to-point
- TCP este orientat conexiune
- Ofera siguranta si asigura transmiterea in ordine a datelor;
- Ofera mecanisme de control al fluxului si control al congestiei
- Minimul de timp necesar unei tranzactii TCP cerere-raspuns daca se creaza o noua conexiune este: 2 *RRT + SPT

Alternative de proiectare si implementare al modelului client/server TCP

<u>Fire de executie:</u>
• fork() poate fi un mecanism costisitor – implementarile curente folosesc mecanismul copyon-write
• IPC (Inter-Process Comunication) necesita trimiterea informatiei intre parinte si copil dupa fork()
• Firele de executie (threads) sunt numite si lightweight processes (LWP)
• Pot fi vazute ca un program aflat in executie fara spatiu de adresa proprie
• Pthreads (POSIX Threads) standard ce defineste un API pentru crearea si manipularea firelor de executie





Fire de executie| Primitive de baza

```
#include <pthread.h>
pthread_t pthread_self();
```

Identificatorul *thread*-ului

Returneaza: ID-ul *thread*-ului care a apelat primitiva

Fire de executie| Primitive de baza

```
#include <pthread.h>
int pthread_detach(pthread_t tid);
```

Identificatorul *thread*-ului

Thread-urile pot fi:

- *joinable*: cind thread-ul se termina, ID-ul si codul de iesire sunt pastrate pina cand se apeleaza pthread_join() <– comportament implicit
- *detached:* cand thread-ul se termina toate resursele sunt eliberate

Returneaza: 0 in caz de succes
o valoare Exxx pozitiva in caz de eroare

Exemplu: pthread_detach(pthread_self());

Fire de executie| Primitive de baza

```
#include <pthread.h>
void pthread_exit(void*status);
```

- Terminarea unui *thread*

Thread-urile se pot termina:

- Functia executata de *thread* returneaza (Obs. Valoarea de return este void * si va reprezenta codul de iesire a *thread*-ului)
- Daca functia *main* a procesului returneaza sau oricare din *thread*-uri a apelat *exit*(), procesul se termina

1. Client TCP – utilizind select() – Clientul este notificat de evenimentele din retea in timp ce asteapta date de intrare de la utilizator 30
• Apelul write() poate fi blocant daca buffer-ul de la socket-ul emitator este plin
• Daca peer-ul trimite date, read() returneaza o valoare >0;
• Daca peer-ul TCP trimite FIN, socket-ul devine “citibil” si read() intoarce 0;
• Daca peer-ul trimite RST (peer-ul a cazut sau a rebootat), socket-ul devine “citibil” si read() intoarce -1;
2. Client TCP – utilizind select() si operatii I/O neblocante
• Implementare complexa => cand sunt necesare operatii I/O neblocante se recomanda utilizare de procese (fork()) sau de thread-uri
3. Client TCP – utilizind fork()
– Mecanismul de functionare:
• exista doua procese
– Un proces face managementul datelor client-server
– Un proces face managementul datelor server-client
4. Client TCP – utilizind pthread()
– Mecanismul de functionare:
• exista doua fire de executie
– Un fir de executie face managementul datelor clientserver
– Un fir de executie face managementul datelor server-client

1. Server TCP – iterativ
   – Se realizeaza procesarea completa a cererii clientului inainte de a deservi urmatorul client
   – Sunt destul de rar intilnite in implementarile reale
   – Un astfel de server serveste foarte rapid un client
2. Server TCP – cate un proces copil pentru fiecare client
   • Serverul deserveste clintii in mod simultan
   • Este des intilnit in practica
      – Exemplu de mecanism folosit pentru distribuirea cererilor: DNS round robin
      – Crearea fiecarui copil (fork()) pentru fiecare client consuma mult timp de CPU
3. Server TCP – preforking; fara protectie pe accept()
   • Serverul creaza un numar de procese copil cand este pornit, si apoi acestia sunt gata sa serveasca clientii
      – Daca numarul de clienti este mai mare decat numarul de procese copil disponibile, clientul va resimti o “degradare” a raspunsului in raport cu factorul timp
      – Acest timp de implementare merge pe sisteme ce au accept() primitiva de sistem
4. Server TCP – preforking; cu blocare pentru protectia accept()
   – Serverul creaza un numar de procese copil cand este pornit, si apoi acestia sunt gata sa serveasca clientii
   – Se foloseste un mecansim de blocare (e.g. fcntl()) a apelului primitivei accept(), si doar un singur proces la un moment dat va putea apela accept(); procesele ramase vor fi blocate pina vor putea obtine accesul
5. Server TCP – preforking; cu “transmiterea” socket-ului conectat Implementare:
   • Serverul creaza un numar de procese copil cand este pornit, si apoi acestia sunt gata sa serveasca clientii
      – Procesul parinte trebuie sa aiba evidenta actiunilor proceselor fii => o complexitate mai mare a implementarii
6. Server TCP – cate un thread pentru fiecare client Implementare:
   – Thread-ul principal este blocat la apelul lui accept() si de fiecare data cind este acceptat un client se creaza (pthread_create()) un thread care il va servi
   – Aceasta implementare este in general mai rapida decat cea mai rapida versiune de server TCP preforked
7. Server TCP – prethreaded; cu blocare pentru protectia accept()
   – Serverul creaza un numar de thread-uri cand este pornit, si apoi acestea sunt gata sa serveasca clientii
   – Se foloseste un mecansim de blocare (e.g. mutex lock) a apelului primitivei accept(), si doar un singur thread la un moment dat va apela accept();
   Obs. Thread-urile nu vor fi blocate in apelul accept()
8. Server TCP – prethreaded; cu “transmiterea” socket-ului conectat Implementare:
   – Serverul creaza un numar de thread-uri cand este pornit, si apoi acestia sunt gata sa serveasca clientii
   – Procesul parinte este cel care apeleaza accept() si “transmite” socket-ul conectat la un thread disponibil
   Obs. Deoarece thread-urile si descriptorii sunt in cadrul aceluiasi proces ,“transmiterea” socket-ului conectat inseamna de fapt ca thread-ul vizat sa stie numarul descriptorului

DNS(Domain Name System)

• Adresele IP (ex. 85.122.23.145) sunt dificil de memorat
• Se utilizeaza un sistem al numelor de domenii pentru a translata adresele IP in nume de domenii si invers
• Numele de domenii se organizeaza in ierarhii

Organizare:
• Initial: /etc/hosts – perechi (nume, IP) – Probleme de scalabilitate
• Actual: DNS consta dintr-o schema ierarhica de nume de domenii si a unui sistem de baze de date distribuite pentru implementarea acestei scheme de nume

Tipuri de domenii:
• Primare (Top Level Domains – TLD)
  – pentru Infrastructura Internet – un singur domeniu .arpa ARPA (Address and Routing Parameter Area)
  – State (ccTLD) – coduri de state: .ro, .fr, .jp, ...
  – IDN ccTLD (Internationalized Country Code Top-Level Domains)
  – Generice: .biz, .com, .info, .name, .net, .org, .pro
  – Sponsorizate: .aero, .edu, .gov, .int, .jobs, .mil, .tel
  – Rezervate: .example, .invalid, .localhost, .test
  – Pseudo-domenii: .bitnet, .local, .root, .uucp etc
• Domeniu de nume
  – Subarbore al arborelui de domenii
  – Nu trebuie sa respecte topologia retelei fizice
• Sub-domenii:
  – intreaga cale de nume nu depaseste 255 de caractere
• Nume de calculatoare (gazde)

Organizare:
• Reguli de alocare a numelor de domenii:
  – Fiecare domeniu controleaza cum sunt alocate subdomeniile sale
  – Pentru a crea un nou subdomeniu, se cere permisiunea domeniului in care va fi inclus (un domeniu de la un anumit nivel va avea o autoritate)
  – Atribuirea de nume de domenii respecta granitele organizationale, nu pe cele ale retelelor
  – Un anumit nivel din ierarhia de niveluri poate fi controlat de mai multe servere
• Servere de nume (name servers)
  – Teoretic, un singur server de nume poate contine intreaga baza de date DNS si poate raspunde tuturor cererilor
    • Probleme: incarcarea si “single point of failure”
  – Spatiul de nume DNS se divide in zone nesuprapuse
  – Exista un server primar (primary/authoritative name server) care deserveste un anumit domeniu si, eventual, mai multe servere secundare continind baze de date replicate
  – TCP se utilizeaza pentru replicarea DNS
  – UDP pentru interogari (lookups)
• Client DNS

– Denumit resolver, trimite un pachet UDP serverului DNS care cauta numele si returneaza adresa IP

• Exemplu de implementari ale serverului de nume: BIND (Berkeley Internet Name Domain), MSDNS, PowerDNS etc.

• Ca resolver (client) interactiv, se poate folosi de exemplu una dintre comenzile: nslookup, host sau dig.

Interogari:

– Recursiva – daca un server DNS nu cunoaste adresa pentru numele solicitat, atunci va interoga alt server DNS

– Incrementala – daca serverul DNS nu stie sa raspunda, returneaza eroare si adresa altui server DNS (numit si referral) care ar putea cunoaste raspunsul la interogare

• Fiecarui domeniu ii este asociata o multime de inregistrari de resurse (resource record – RR)

• Mecanismul:

– Cererea: resolver-ul trimite un nume de domeniu

– Raspunsul: inregistrarile de resurse asociate acelui nume (stocate in bazele de date DNS

• Forma generala RR este:

Nume_domeniu Timp_de_viata Tip Clasa Valoare

Nume_domeniu (domain name) – precizeaza domeniul caruia i se aplica aceasta inregistrare

Timp_de_viata (time-to-live) – da o indicatie despre cat de stabila este inregistrarea

Tip - precizeaza tipul inregistrarii

Clasa: pentru Internet valoarea acestuia este IN Valoare: acest camp poate fi un numar, un nume de domeniu sau un sir ASCII; semantica depinde de tipul de inregistrare

Optimizari:

Proximitatea spatiala: serverele locale vor fi interogate mai des decat cele la distanta

Proximitatea temporala: daca un set de domenii sunt referentiate repetat atunci se apeleaza la caching DNS Fiecare intrare DNS va avea stabilita o valoare TTL (time to live) Se va utiliza si replicarea (servere multiple, servere root multiple) – se va interoga cel mai apropiat (geografic) server

Comenzi:

Ca resolver interactiv se pot folosi comenzile: – nslookup – dig – host –whois

Primitive:

• Nu trebuie scris un resolver pentru a afla adresa IP a unei gazde

• Functii principale:

• gethostbyname(); getaddrinfo();

• gethostbyaddr() ; getnameinfo();

• La unele sisteme de operare (e.g., Solaris) va trebui la compilare sa folosim biblioteca nsl (Name Server Library): gcc ... -lnsl

Una din structurile folosite: hostent

```
struct hostent {
    char *h_name; /* nume oficial (canonical) */
    char **h_aliases; /* alias-uri */
    int h_addrtype; /* AF_INET */
```

```
        int h_length; /* lungimea adresei: 4 sau 6 */
        char **h_addr_list; /*pointeri la adresele IP */
};
```

gethostbyname():

```
#include <netdb.h>
struct hostent *gethostbyname (const char *hostname);
```

• In termenii DNS, gethostbyname() realizeaza o cerere pentru o inregistrare A
• Obs. gethostbyname() se foloseste in special pentru IPv4
• Returneaza:
  – In caz de succes returneaza un pointer la hostent, ce contine adresa IP a host-ului
  – In caz de eroare NULL, iar variabila h_errno indica eroarea aparuta

gethostbyaddress():

```
#include <netdb.h>
struct hostent *gethostbyaddr ( const char *addr, socklen_t len, int family);
```

• In termenii DNS, gethostbyaddr() realizeaza o cerere la serverul de nume pentru o inregistrare PTR in domeniul in-addr.arpa
• Returneaza: In caz de succes returneaza un pointer la hostent, ce contine numele oficial al host-ului ; In caz de eroare NULL, iar variabila h_errno indica eroarea aparuta
Obs. gethostbyaddr() se foloseste in special pentru IPv4

getservbyname():

```
#include <netdb.h>
struct servent *getservbyname (const char *servname, const char *protoname);
```

• Returneaza: un pointer la struct servent in caz de sucess, NULL in caz de eroare

```
struct servent {
        char *s_name; /* numele oficial al serviciului*/
        char **s_aliases; /* alias-uri */
        int s-port; /* portul (network-byte order) */
        char *s_proto; /* protocolul */
};
```

getservbyport():

```
#include <netdb.h>
struct servent *getservbyport (int port, const char *protoname);
```

• Cauta un serviciu dupa un numar de port si dupa protocol (optional)
• Returneaza: un pointer la struct servent in caz de sucess, NULL in caz de eroare
Obs. port este in network byte order

getaddrinfo():

```
#include <netdb.h>
int getaddrinfo ( const char *hostname, const char *service, const struct addrinfo *hints, struct addrinfo **result ) ;
```

• Obs. hostname, service, hints – parametri de intrare
• Returneaza: 0 in caz de sucess, !=0 in caz de eroare
• Se recomanda a fi folosita si pentru IPv4 si pentru IPv6

```
struct addrinfo {
int ai_flags; /* AI_PASSIVE, AI_CANONNAME */
int ai_family; /* AF_INET, AF_INET6, AF_UNSPEC */
int ai_socktype; /* SOCK_STREAM sau SOCK_DGRAM */
int ai_protocol; /* 0 (auto) sau IPPROTO_TCP, IPPROTO_UDP */
socklen_t ai_addrlen; /* lungimea lui ai_addr */
char *ai_canonname; /* numele canonic al host-ului */
struct sockaddr *ai_addr; /* adresa binara a socket-ului */
struct addrinfo *ai_next; /* pointer la urmatoarea structura din lista */
};
```

getnameinfo():

```
#include <netdb.h>

int getnameinfo ( const struct sockaddr *sockaddr, socklen_t addrlen, char *host, socklen_t
hostlen, char *serv, socklen_t servlen, int flags) ;
```

• Inlocuieste gethostbyaddr() si getservbyport()
• Returneaza: 0 in caz de sucess, !=0 in caz de eroare

NIVELUL APLICATIE

Tipuri de protocoale in functie de natura datelor transferate:
– Fluxuri de caractere generate de utilizator
– Mesaje intrebare/raspuns ASCII
– Formate binare
– Protocoale ad-hoc folosite de aplicatiile (nestandard) scrise de utilizatori

Accesul la terminal:
• rlogin
  – protocol simplu de acces la distanta
  – rlogin comunica cu un daemon rlogind de pe gazda remote
  – autentificarea se face prin apelarea la gazde “de incredere”
• telnet (terminal network)
  – Protocol standard TCP/IP de acces la distanta (RFC 854,855)
  – Utilizat indiferent de platforma
  – Poate fi utilizat drept client generic fara a sti detalii despre server
  – Autentificarea clientilor nu se face de catre protocol, ci de catre aplicatie
  – Protocolul se bazeaza pe conceptul NVT (Network Virtual Terminal): un dispozitiv virtual cu o structura generala comuna cu o gama larga de terminale; fiecare host face maparea caracteristicilor propriului terminal cu cele ale NVT. Odata ce a fost stabilita o conexiune prin TELNET, ambele capete ale comunicarii sunt tratate simetric. Ambele parti ale comunicarii pot sa negocieze utilizarea de optiuni aditionale care sa reflecte partea hardware utilizata
  – Comunicare dintre client si server se realizeaza prin comenzi de tipul:

- IP (Interrupt Process; 244) -> terminarea programului care ruleaza
- AO (Abort output; 245) -> elibereaza orice buffer de iesire
- AYT (Are you there; 246) -> permite clientului trimiterea unei interogari OOB pentru verificarea faptului ca partea remote este activa
- EC (Erase character; 247) -> stergerea caracterului anterior
- EL (Erase Line; 248) -> stergerea intregii linii curente

• SSH (secure shell)
  – Fata de telnet, furnizeaza o comunicare sigura (bazata pe TCP) prin mesaje criptate si mesaje de autentificare
  – SSH foloseste modelul client/server
    - Un program client SSH este utilizat pentru stabilirea unei conexiuni cu un daemon SSH
  – Utilizari:
    - logarea pe o masina la distanta si executarea de comenzi
    - suport pentru tunneling
    - permite si transfer de fisiere in asociere cu protocoalele SFTP sau SCP
  – Are suport in majoritatea sistemelor de operare moderne

EMAIL

• Protocoale bazate pe TCP:
  – SMTP (Simple Mail Transfer Protocol)
  – POP (Post Office Protocol)
  – POP3S – varianta securizata a POP

<u>Terminologie</u>
– Agent utilizator (MUA – Mail User Agent): client (local) pentru posta electronica Ex: alpine, mutt, Mozilla Thunderbird, Kmail, Outlook etc.
– Agent de transfer (MTA – Mail Transport Agent) responsabil cu comunicarea cu gazdele la distanta si cu trimiterea/receptionarea de posta (client & server) - sendmail, qmail
– Agent de distributie (MDA - Mail Distribution Agent sau LDA – Local Delivery Agent) - directioneaza mesajele primite catre casuta postala a utilizatorului; Ex: maildrop, Sieve, procmail
– Mail exchanger (MX) – gazda responsabila cu e-mail-urile unui domeniu (masina intermediara)

<u>Caracteristici</u>
– Distinctia dintre plic si continut
  - Plicul incapsuleaza mesajul, contine date necesare pentru transportul mesajului: destinatar, adresa, prioritate, securitate, ...
  - Plicul este folosit pentru dirijarea mesajului la destinatar
  - Mesajul din plic contine un antet (date de control pentru MUA) si un corp (date pentru utilizator)

EMAIL | SMTP

Componente:
– Plic (envelope)– folosit de MTA pentru livrare Exemplu: MAIL From: RCPT to:
– Anteturi (headers) – folositi de MUA Exemplu: Received, Message-ID, From, Date
– Continut –ul mesajului (body) -
• Mecanism: MUA preia continutul, adauga anteturi si il transmite la MTA; MTA adauga anteturi, adauga plicul si il trimite la un alt MTA

Comunicarea:
– Se realizeaza o conexiune TCP intre Sender SMTP si Receiver SMTP (intre MTA-uri).
Obs. Receiver SMTP poate fi destinatia finala sau un intermediar (mail gateway)
– Clientul trimite comenzi SMTP, iar serverul raspunde cu coduri de stare
– Mesajele de stare include coduri numerice NNN si texte explicative
– Ordinea comenzilor este importanta
– Se utilizeaza portul 25

Comenzi uzuale:
– HELO: identifica gazda expeditoare
– MAIL FROM: porneste o tranzactie si identifica orginea email-ului
– RCPT TO: identifica recipientii individuali ai mesajului (adrese de e-mail); pot exista comenzi RCPT TO: multiple
– DATA desemneaza o serie de linii text terminate cu \r\n, ultima linie continind doar “.”
– QUIT

Alte comenzi:
– VRFY: permite verificarea validitatii unui recipient
– NOOP: forteaza serverul sa raspunda cu un cod de OK (200)
– EXPN: expandeaza un grup de adrese (alias)
– TURN: interschimba destinatarul cu expeditorul fara a fi necesara crearea unei noi conexiuni TCP (sendmail nu suporta aceasta comanda)
– RSET abandoneaza tranzactia curenta

DNS & E-MAIL
Inregistrarea de tip MX din DNS identifica gazda (MX) cu rol de procesare si forwardare a mailurilor pentru respectivul domeniu.

EMAIL | POP

• POP (Post Office Protocol) – RFC 1939
• Utilizat la transferul de mesaje de pe un server de posta la un MUA – portul 110
• Comenzile si raspunsurile sunt mesaje ASCII
• Raspunsurile incep cu +OK sau -ERR

Comenzi uzuale:
– USER specifica numele de cont
– PASS specifica parola
– STAT furnizeaza numarul de mesaje din cutia postala (mailbox)
– LIST afiseaza lista de mesaje si lungimea, cate 1 pe linie
– RETR preia un mesaj
– DELE reseteaza tranzactia, iar orice marcaj de stergere este eliminat
– QUIT sterge mesajele marcate si inchide conexiunea

Caracteristici:
– In general, daca utilizatorul schimba clientul el nu-si mai poate reciti mailurile;
Obs: Clienti cu optiunea: 'keep a copy of the email on the server'
– Foloseste mecanismul "download-and-keep": copierea mesajelor pe clienti diferiti
– POP3 este fara stare intre sesiuni

TFTP(Trivial File Transfer Protocol)

– utilizeaza UDP si portul 69
– utilizat deseori la initializarea statiilor de lucru fara disc sau a altor dispozitive
– nu are mecanisme de autentificare si criptare => este utilizat in retele locale

FTP(File Transfer Protocol)

• Folosit atat interactiv, cat si de programe
• Asigura transferul sigur si eficient al fisierelor
• Se bazeaza pe modelul client/server
• FTP utilizeaza doua conexiuni TCP pentru transferul fisierelor:
– Conexiune de control
• folosita pentru trimiterea comenzilor si receptionarea codurilor de stare
• Conexiunea de control utilizeaza portul 21
– Conexiunea de date
• folosita pentru transferul efectiv
• conexiunea de date foloseste portul 20 sau unul aleator (P > 1023)
• nu este obligatorie intr-o sesiune FTP
Pentru interactivitate se foloseste protocolul TELNET Tipuri de acces:
• Anonim (FTP anonymous) – RFC 1635
– Autentificare cu numele anonymous si drept parola o adresa de e-mail
– Acces public la o serie de resurse (aplicatii, date, multimedia etc.)
• Autentificat
– Necesita un nume de utilizator existent, insotit de o parola valida
– Pentru transferul de date in/din contul personal
Comenzi (protocol)
• Comenzi de control al accesului
– USER username, PASS password, QUIT, ChangeWorkingDir,...
• Comenzi de transfer a parametrilor
– PORT, TYPE, MODE

• Comenzi de realizarea a serviciilor FTP
– RETR filename, ABOR, STOR filename, LIST, PrintWorkingDir

Raspunsul de stare
Linie de text continind: XYZ un cod de stare (utilizat de software) + un mesaj explicativ (destinat oamenilor)

Codul de stare (xyz)
Prima cifra semnifica:
1 replica pozitiva preliminara (“am indeplinit, dar asteapta”)
2 replica pozitiva finala (“succes”)
3 replica pozitiva intermediara (“am nevoie si de alte informatii”)
4 replica negativa tranzitorie (“eroare, incerc iar”)
5 replica negativa finala (“eroare fatala”)

A doua cifra specifica grupuri de functii:
0 erori de sintaxa
1 informare (ajutor, informatii de stare)
2 referitor la conexiuni
3 privitor la autentificarea utilizatorului
4 nespecificat
5 referitor la sistemul de fisiere

A treia cifra da informatii suplimentare asupra mesajelor de eroare

Moduri de transfer
- STREAM - Fisierul este trimis ca un flux de octeti; sfirsitul transmisiei este indicat de inchiderea normala a conexiunii;
- BLOCK - Fisierul este transmis ca o serie de blocuri de date precedate de antete continand contoare si descriptori de bloc (e.g. End of data block)
- COMPRESSED - Fisierele sunt compresate, conform unui algoritm de compresare (e.g., gzip) si sunt trimise ca date binare

HTTP(Hyper Text Transfer Protocol)

– Protocol utilizat in Internet, bazat pe stiva TCP/IP
– Sta la baza comunicarii dintre serverele si clientii Web
• Client: in mod uzual poate fi un browser
• Server: server Web care trimite raspunsuri la cererile primite

Protocolul HTTPS
– asigura comunicatii “sigure” HTTP via TLS (Transport Layer Security): – autentificare pe baza certificatelor digitale + criptare bidirectionala

Protocolul SPDY
– un experiment Google, disponibil ca Internet Draft la care Google a renuntat in 2016
• Reducerea latentei incarcarii si cresterea securitatii

Protocolul HTTP/2.0

• Extinde ideile SPDY, focalizat asupra performante

Mecanism general:

- Clientul initiaza o conexiune TCP cu serverul folosind portul 80
- Serverul accepta conexiunea TCP
- Are loc schimbul de mesaje HTTP intre clientul HTTP (browser) si server-ul Web
- Se inchide conexiunea TCP

Rutare

- Partea software-ului nivelului retea care alege calea pe care un pachet receptionat trebuie trimis pentru a ajunge la destinatie
- Daca se folosesc datagrame, decizia de rutare trebuie luata pentru fiecare pachet
- Daca se utilizeaza circuite virtuale, decizia de rutare se ia la stabilirea unui nou circuit
- Cerintele pentru un algoritm de rutare: corect, simplu, robust, optim, rapid convergent
- Activitati
    - Determinarea caii optime de rutare (routing)
    - Transportarea pachetelor: comutare (packet switching)

Terminologie:

- **end systems** – dispozitive de retea fara capacitati de redirectat pachete catre subretele
- **intermediate systems** – dispozitive de retea avand capacitati de redirectat pachete - Intradomain IS – comunicare in cadrul unui domeniu de rutare - Interdomain IS – comunicare si intre domenii de rutare
- **sistem autonom** – AS (eng. Autonomous system) – colectie de retele care partajeaza aceeasi strategie de dirijare

Comutare

• O gazda (eng. host) are de trimis un pachet la un alt host
• Host-ul sursa trimite pachetul la un router, folosind adresa hardware (MAC) a acestuia, un pachet continand adresa de retea a gazdei destinatie
• Routerul examineaza adresa de retea a destinatarului, iar daca nu cunoaste unde sa trimita pachetul, il va distruge
• Altfel, va modifica adresa continuta de pachet in adresa hardware a urmatorului hop (punct indermediar de transmitere – Intermediate System) si va trimite pachetul spre acesta
• Daca urmatorul hop nu este destinatia finala, atunci procesul se repeta pentru un alt router, s.a.m.d.

• In cazul in care la nivelul retea avem servicii neorientate conexiune, pachetele (numite si datagrame) sunt trimise individual si sunt rutate in mod independent una de alta
• In cazul in care la nivelul retea avem servicii orientate conexiune se folosesc circuite virtuale si decizia de rutare se ia la stabilirea unui nou circuit

Determinarea caii de rutare

- Pentru fiecare cale de rutare se determina un cost (metrica)

- Lungimea caii, siguranta, intarzierea, largimea de banda, incarcarea, costul comunicarii
- Algoritmii de rutare initializeaza si mentin (pentru fiecare gazda) tabele de rutare continand informatii de dirijare
  – Rute catre gazde specificate
  – Rute spre retele specificate
  – O ruta implicita

Algoritmii de rutare - caracteristici:
– Acuratete (engl. Accuracy) – un algoritm trebuie sa opereze in mod corect si rapid pentru gasirea destinatiei
– Complexitate redusa – important pentru rutere cu resurse fizice (soft) limitate
– Optimalitate – abilitatea de a gasi ruta optima
– Robustete – capacitatea de a functiona corect pentru o perioada lunga de timp, in circumstante diferite
– Adaptabilitate – la aparitia unei erori in retea, algoritmul trebuie sa se adapteze (de ex. caderea nodurilor sau coruperea tabelelor de rutare)
– Convergenta – algoritmii de rutare trebuie sa convearga rapid atunci cand sunt distribuite mesaje de rutare de actualizare
– Load balancing – un algoritm de rutare cantareste diferite posibilitati de rutare pentru evitarea legaturilor incete sau a congestiilor

Tipuri de rutare:
– Centralizata – drumul de cost minim poate fi determinat avand disponibile toate informatiile despre retea <– **algoritmi folosind starea legaturii**
  – Topologia retelei & costurile tuturor legaturilor sunt cunoscute
  – Un nod trebuie sa cunoasca identitatile & costurile nodurilor vecine
  – Fiecare nod difuzeaza prin broadcast identitatile si costurile tuturor legaturilor de la acel nod la altele
– Descentralizata – drumul de cost minim este determinat in mod iterativ, distribuit (nici un nod nu poseda informatii complete despre costurile legaturilor din retea) <- **algoritmi cu vectori distanta**
  – Fiecare nod primeste informatii de la nodurile vecine, realizeaza calcule si distribuie rezultatele inapoi la vecinii directi – algoritmul este distribuit si asincron
  – Fiecare nod mentine o tabela de distanta (distance table)
    – X: nodul dorind sa realizeze o rutare la nodul Y via nodul vecin Z
    – Dx (Y,Z): suma costului legaturii directe intre X si Z (c(X,Z)) plus costul curent al drumului minim de la vecinii lui Z la Y: Dx (Y,Z)=c(X,Z) + minw{Dz (Y,w)}
    – Tabela de rutare a unui nod poate fi construita cunoscand tabela de distanta a nodului

Clasificare:
• **Statici (neadaptivi)**
  – Topologia legaturilor se incarca pentru o perioada de timp in tabelele de rutare a fiecarui nod
  – Dezavantaje:
    • Reteaua trebuie sa aiba o dimensiune optima pentru a putea fi controlabila
    • Daca au loc esuari in retea, nu se poate reactiona imediat

  – Dirijare pe calea cea mai scurta
  – Inundare (eng. flooding)

• Un pachet primit este copiat si transmis prin toate legaturile de comunicare (exceptand cea pe unde a venit)
• Problema: packet reflection (un nod poate primi o copie nedorita a unui pachet)
• Utilizari ale algoritmilor de tip flooding: aplicatii militare, baze de date distribuite, etc.

– Deflecting routing (sau hot-potato routing)
• La fiecare pas un pachet este examinat in raport cu adresa destinatie; daca legatura ceruta este libera pachetul este trimis, altfel este deviat (deflected) catre o alta linie de comunicare aleasa aleator;
• Un pachet are asociat un camp cu o valoare de prioritate care il poate ajuta pe viitor sa castige disputa cu alte pachete

• **Dinamici (adaptivi)**

– Starea retelei este "invatata" din comunicarea ruterelor cu vecinii lor; starea fiecarei regiuni din retea este propagata in retea dupa ce toate nodurile isi actualizaza tabelele de rutare => fiecare ruter poate gasi calea cea mai buna pe baza informatiilor de la nodurile vecine

– Cu vectori distanta
• Fiecare router mentine un tabel (vector) cu distanta si linia de comunicare catre destinatie; tabelele sunt actualizate cu informatiile de la vecini
Algoritmul Bellman-Ford
• Algoritm folosit de protocoalele RIP, BGP, IGRP

– Folosind starea legaturilor
Fiecare router trebuie sa:
– Descopere vecinii si sa le "invete" adresele de retea
– Masoare intarzierea sau costul asociat fiecarui vecin
– Construiasca un pachet prin care anunta pe "toti" ceea ce a invatat Dilema: cand trebuie construite pachetele? (de ex. periodic sau cand apare un eveniment special)
– Trimita pachetul
– Calculeze cea mai scurta cale catre fiecare router

– Dirijare ierarhica
Necesitate: in retele de mari dimensiuni nu este fezabil ca un router sa aiba cate o intrare despre fiecare alt router;
Mecanism: Ruterele stiu detalii asociate unei regiuni , dar nu stiu detalii despre structura interna a altor regiuni

– Prin difuziune (broadcast)
Utilizare: actualizarea stocurilor (de la bursa de valori), streaming multimedia, serviciu de distribuire a rapoartelor despre vreme, etc.
Modalitati:
– Sursa trimite cate un pachet distinct fiecarui destinatar
Obs.: Metoda ineficienta: neutilizarea latimii de banda; Sursa trebuie sa aiba adresele tuturor destinatarilor
– Flooding - util cind alte metode nu pot fi aplicate Problema: Se genereaza prea multe pachete si se consuma multa latime de banda

– Cu trimitere multipla (multicast)
Mecanism: router-ul va face periodic o interogare asupra host-urilor care apartin unui grup; apoi informatia este propagata catre routere

<u>Protocoale de rutare - clasificare</u>

- Intradomain routing protocol – realizeaza rutarea pachetelor intr-un domeniu
– RIP (Routing Information Protocol)

– OSPF (Open Shortest Path First)
– Interdomain routing protocol – realizeaza rutarea pachetelor intre domenii
– BGP (Border Gateway Protocol)
- EGP (Exterior Gateway Protocol)

RIP (Routing Information Protocol):
– Se aplica algoritmul Bellman-Ford (pentru host-uri si routere)
– Pentru fiecare router, se creeaza un vector continand costul rutei si alte informatii
– Daca survin modificari intr-un punct, acestea sunt propagate periodic la routerele si host-urile vecine cu acel punct

• Foloseste mesaje IP
• Fiecare router trimite un broadcast continand intreaga tabela de rutare a router-ului – la fiecare 30 sec.
• O intrare a tabelei de rutare RIP contine: – Adresa IP – Metrica (numarul de hop-uri: 1-15) – Timeout (in secunde)
• Retelele conectate direct au metrica =1 (un hop)
• Daca o ruta da timeout, metrica devine 16 (nu exista conexiune) si ruta e stearsa dupa 1 minut
• Daca o informatie de rutare se modifica (de ex. o legatura sau un router esueaza), propagarea acestei schimbari are loc foarte lent – RIP sufera de convergenta lenta
• RIP – Este un protocol matur, stabil, larg suportat si usor de implementat – Este indicat a fi folosit de sistemele autonome de dimensiuni reduse fara rute redundante – In practica este inlocuit in majoritatea situatiilor de OSPF

OSPF (Open Shortest Path First)
• Fiecare router ce foloseste OSPF cunoaste starea intregii topologii de retea (algoritm folosind starea legaturii) si transmite actualizari la toate routerele
• Conduce la trafic aditional, care poate conduce la congestii OSPF permite ca traficul sa fie distribuit pe rute cu costuri similare (load balancing) OSPF suporta rutarea dupa tipul serviciilor (ToS) – protocolul IP contine campul ToS (in general neutilizat)
• Convergenta mai rapida
• Ofera suport pentru folosirea mai multor tipuri de metrici

Opereaza intr-o ierarhie de entitati de retea
Motivatie: retele de dimensiuni mari => un router nu poate cunoaste intreaga topologie
– **Sistemul autonom (AS)** – colectie de retele care partajeaza aceeasi strategie de dirijare
– Un AS este divizat in **domenii (engl. areas)** – grupuri contigue de retele si gazde; routerele au aceeasi informatie privitoare la topologie si ruleaza acelasi algoritm
– **Coloana vertebrala (backbone sau area 0)** – responsabila cu distributia informatiilor de rutare intre domenii; orice router conectat la doua sau mai multe domenii face parte din backbone (aceste routere vor rula algoritmi corespunzatori pt fiecare domeniu)

BGP (Border Gateway Protocol)
• Utilizat pentru comunicarea intre routere aflate in sisteme autonome diferite
• Functii majore:
  • Neighbor relationship – se refera la acordul dintre routerele din doua sisteme autonome de a schimba informatii pe baza unor reguli (un router poate refuza stabilirea unei astfel de relatii in functie de: regulile domeniului, supraincarcare etc)
  • Neighbor maintenance – routerele isi vor trimite mesaje de tip keep-alive

    - Network maintenance – fiecare router tine o baza de date cu subretelele existente pentru o rutare eficienta in acea subretea
- Exista patru tipuri de pachete BGP:
    - Open: folosit pentru stabilirea unei relatii dintre doua routere
    - Update: contine informatii actualizate despre rute
    - Keep-alive: folosit pentru confirmarea de relatii stabilite anterior
    - Notification: folosit atunci cand apar erori
- Perechile de routere BGP comunica intre ele folosind conexiuni TCP
- BGP este un protocol bazat pe vectori distanta cu urmatoarele diferente:
    - nu se pastreaza doar costul asociat unei destinatii, ci se mentine si calea catre acea destinatie
    - nu se furnizeaza vecinilor doar costul estimat, ci si calea exact

CONGESTII

Apare atunci cand se realizeaza supraincarcarea resurselor unei retea.
Congestia poate aparea:

- La nivelul legaturii de date: cand latimea de banda nu este suficienta
- La nivelul retea: cand coada de pachete de la noduri nu poate fi controlata
- La nivelul transport: cand legatura logica dintre doua rutere aflate intr-o sesiune de comunicare nu mai poate fi controlata

<u>Controlul congestiei</u>

- Open-loop: rezolvarea inseamna de fapt prevenirea aparitiei congestiilor printr-un design si decizii potrivite
- Close-loop
    - Monitorizarea sistemului pentru detectarea congestiilor Metrici: procentul de pachete eliminate datorita lipsei spatiului in buffer, intarzierea pachetelor, etc.
    - Trimiterea acestei informatii la nodurile care pot lua decizii
    - Ajustarea operatiilor pentru corectarea problemei

Obs.: controlul congestiei != controlul fluxului

- Controlul congestiei asigura faptul ca reteaua are capacitatea de a transporta traficul oferit; implica actiunile tuturor host-urilor si a routerelor
- Controlul fluxului se ocupa de comunicarea point-to-point dintre un emitator si un receptor si se asigura faptul ca un emitator nu transmite date mai repede decat poate receptorul sa le proceseze

RETELE WIRELESS

Fixed versus Mobile

– Sistemele wireless fixed – locatia emitatorului si receptorului nu se schimba (legatura point-to-point). Toata energia este folosita pentru transmiterea semnalului, si nu pentru transmiterea intr-o arie geografica mare
– Sisteme wireless mobile – receptorul poate fi localizat oriunde in aria emitatorului

- **Dispozitive**
  Aspecte de interes
  - Slot-uri pentru extinderea ulterioara (memorie suplimentara, conectivitatea cu alte dispozitive)
  – Timpul de viata al bateriei
  – Caracteristici integrate: camera digitala, tastatura, porturi cu infrarosu, Bluetooth,…
  – Suport software: aplicatii, instrumente de dezvoltare, navigatoare mobile, drivere pentru hardware etc.
- **NIC (Network Interface Card)**
  Asigura interfata dintre dispozitive si infrastructura retelei wireless

Componente:
- Base stations - componente care interfateaza comunicatiile wireless cu cele wired
- Access controllers - autentificaarea si autorizarea utilizatorilor, criptare, etc.
- Soft de conectare - asigura comunicarea eficienta si sigura

Categorii:
- WPAN(Wireless Personal-Area Network)
- WLAN(Wireless Local-Area Network)
- WMAN(Wireless Metropolitan-Area Network)
- WWAN(Wireless Wide-Area Network)

WPAN

Standarde de conectivitate pentru WPAN
• IrDa (Infrared Data Association): comunicatie punctla-punct bidirectionala via porturi cu infrarosu
– Poate fi folosita intr-o arie limitata fara obstacole (line-of-sight)
– Nu este afectata de interferente RF (radio frequency)
Exemplu: sincronizare smartphone - PC
• Bluetooth
– Un dispozitiv poate fi setat ca putind fi descoperit in mod general, limitat sau deloc (descoperirea e automata)
– 8 dispozitive formeaza un piconet (1 master si 7 dispozitive slave). Reteaua ad-hoc formata are suprafata de maxim 10m.

Utilizari:
• Sincronizare
– Ex. Intre telefoane mobile cu un laptop, tableta sau PC
• Streaming multimedia

• Control
    – Ex: mouse wireless, tastatura wireless
• Acces mai usor la dispozitive
    – Ex: conexiunea wireless dintre un PC si o imprimanta
• Conexiune Internet
• Enterprise
    – Folosesc WPAN pentru sincronizari sau acces la dispozitive
    – Pentru conectivitatea la Internet se utilizeaza WLAN

WLAN

• Dispozitive utilizator – PC-uri, laptopuri, PDA-uri, smarthpone-uri -> echipate corespunzator
• Radio NIC (Network Interface Card) sau adaptere sau card radio – Opereaza in cadrul dispozitivului si ofera conectivitatea wireless – Implementeaza si suporta unul sau mai multe versiuni al unui standard (e.g. 802.11a, 802.11b/g, ...)
• Punct de acces (Access point) – Contine un card radio care comunica cu un dispozitiv utilizator folosind o retea wireless

Componente:
• **Routere** • Dispozitiv ce asigura rutarea corespunzatoare a pachetelor
• **Repetor (Repeater)** • Dispozitive care asigura regenerarea semnalului (=>largirea ariei retelei) fara adaugarea de noi puncte de acces
• Determina o reducere a performantei WLAN-urilor; repetorul primeste si retransmite fiecare frame pe acelasi canal radio, ceea ce duce la dublarea cantitatii de trafic in retea

WMAN

Componente:
• Bridges: asigura conectivitatea a doua retele care utilizeaza protocoale similare sau diferite la nivelul legaturii de date
• Antene: pentru retelele WMAN se folosesc in special antene directionale, pentru maximizarea intensitatii undelor radio intr-o directie

Sistemele WMAN asigura conectivitatea intre cladiri si utilizatori in cadrul unui oras folosind cateva configuratii
– point-to-point: utilizeaza RF sau transmisie prin infrarosu, folosind antene (semi)directionale care pot atinge arii de 30 mile pentru sistemele RF.
– point-to-multipoint: utilizeaza o antena omnidirectionala
– Sisteme “packet radio”: utilizeaza routere wireless speciale care trasmit pachetele intre ele

WWAN

Componente

• Radio NIC – utile pentru a integra un laptop sau alte dispozitive intr-un WWAN; Unele telefoane mobile au integrate aceste dispozitive, insa apar probleme de integrare deoarece exista diferite tipuri de WWAN-uri (e.g. in functie de hardware-ul satelitului)

• Base Stations
  – Cell towers
  – Sateliti

• Antene

WWAN bazate pe celule - (celula = zona geografica avind o arie de acoperire a semnalului) Format din:

- *Cell towers* : recepteaza semnale de la utilizator 50 si transmite informatie inapoi la utilizator
- *Voice switches* – realizeaza conectarea dispozitivului utilizator la alt utilizator folosind sistemul de telefonie
- *Gateway-uri* – transforma sistemul intr-un WWAN; face posibil accesul utilizatorilor la Internet

Multiplexarea: semnalul este folosit (partajat) de mai multi utilizatori

– Frequency-division multiple access (FDMA): fiecare semnal din cadrul canalului de comunicatie are o frecventa unica (modelul posturilor radio)

– Time-division multiple acess (TDMA): se asigneaza fiecarui utilizator segmente de timp in care poate comunica

– Code-division multiple access (CDMA): fiecare semnal are atasat un cod, toate semnalele fiind transmise pentru a “umple” intreaga latime de banda; receptorul va procesa doar semnalele avind codul “corect”

MOBILE IP

– Protocol care permite unui dispozitiv mobil deplasarea dintr-o retea in alta si mentinerea adresei IP

– Mobile IPv4 – RFC 3344, RFC 4721

– Mobile IPv6 – RFC 3775, RFC 6275 (Proxy Mobile IPv6 – PMIPv6)

– Protocol de rutare in care dispozitivele terminale (end devices) isi semnalizeaza propriile actualizari de rute si tunelele dinamice de date elimina necesitatea propagarii informatiilor privitoare la rute

– Un utilizator poate folosi (roam) diverse sub-retele IP si legaturi de acces, mentinindu-se o comunicare continua

• Implementari:

– Interactive Protocol for Mobile Networking

– Network Mobility (NEMO)

Componente:
– Nod mobil: dispozitiv folosind IP
– Home agent (HA): nod responsabil cu redirectarea datelor spre locatia curenta a nodului mobil
– Forign Agent(FA): un router atasat la o legatura de acces, aflat la celalt capat al tunelului stabilit cu un nod mobil

Nodul mobil are doua adrese:
– Home Address • Adresa IP a nodului mobil;
– CoA (Care-of Adress) • adresa IP valida si rutabila

Mecanismul general: Un nod care doreste sa comunice cu nodul mobil va utiliza permanent home adress a acestuia - Daca nodul mobil se gaseste in HomeNetwork, atunci daca cineva ii trimite un pachet se foloseste IP forwarding - Daca nodul mobil se afla intr-o retea straina se foloseste CoA - Nodul mobil isi inregistreaza locatia reala la home agent - Pachetele sunt trimise printr-un tunel de la home agent la CoA (capatul tunelului) Obs. Daca nodul nu primeste mesaje de tip agent advertisment atunci incearca sa obtina o adresa prin tehnici precum DHCP pentru a-si cunoaste locatia curenta

Tuneluri: legaturi logice la distanta de 1 hop, aflate la marginile Foreign Network la care sunt atasate nodurile mobile • Pot transporta orice pachete IP intre punctele finale ale comunicatiei • Incapsularea datelor se face via IP-in-IP – RFC 2003 (20 de bytes suplimentari)

Rutarea: • Cind nodul mobil doreste sa trimita pachete, el nu le va trimite la home agent, ci le va trimite direct la nodul cu care doreste sa comunice, folosind home adress ca valoare a cimpului source adress a pachetului IP (folosind desigur foreign agent) – triangle routing