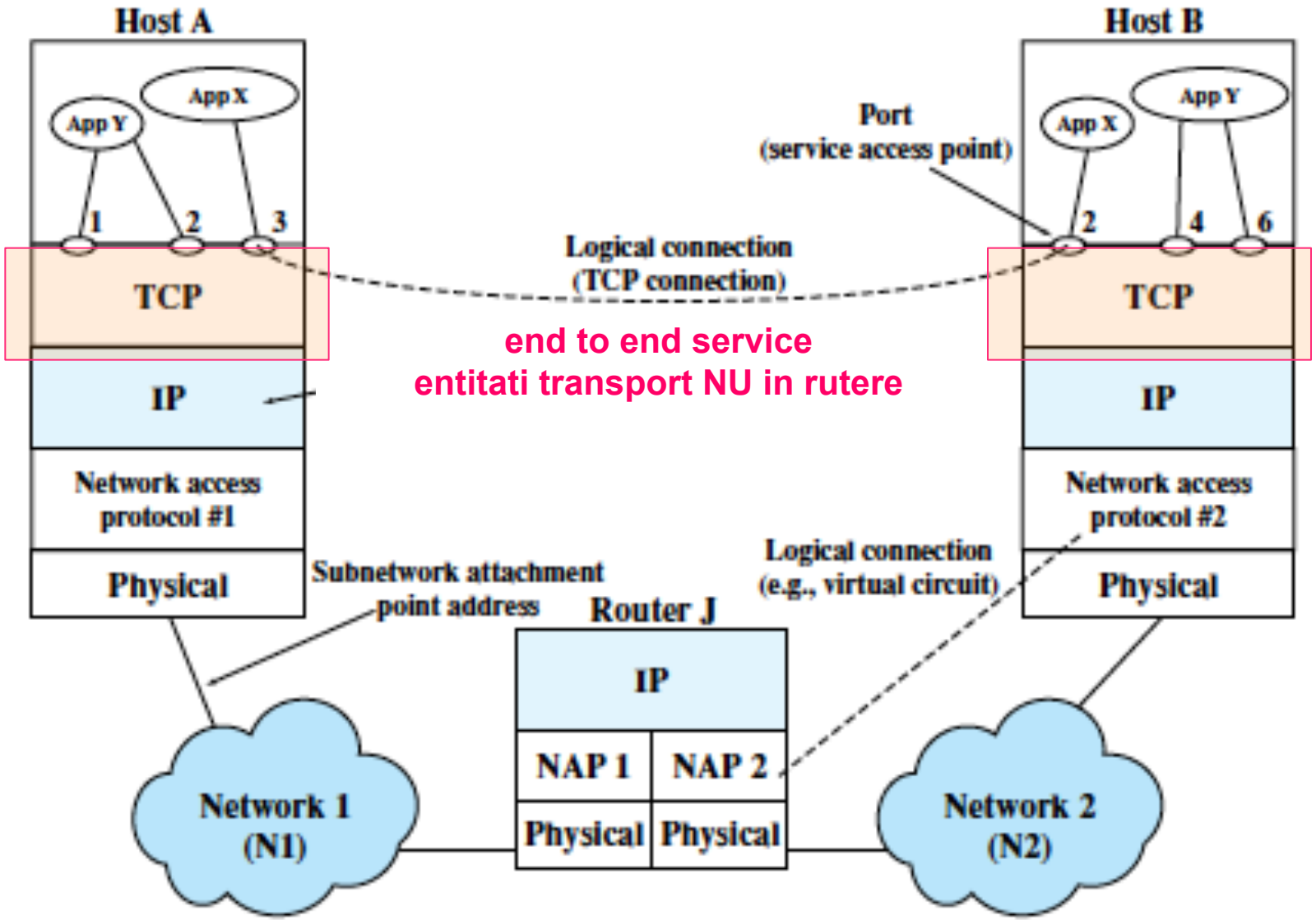
Nivelul transport

Nivelul transport in ierarhia TCP/IP

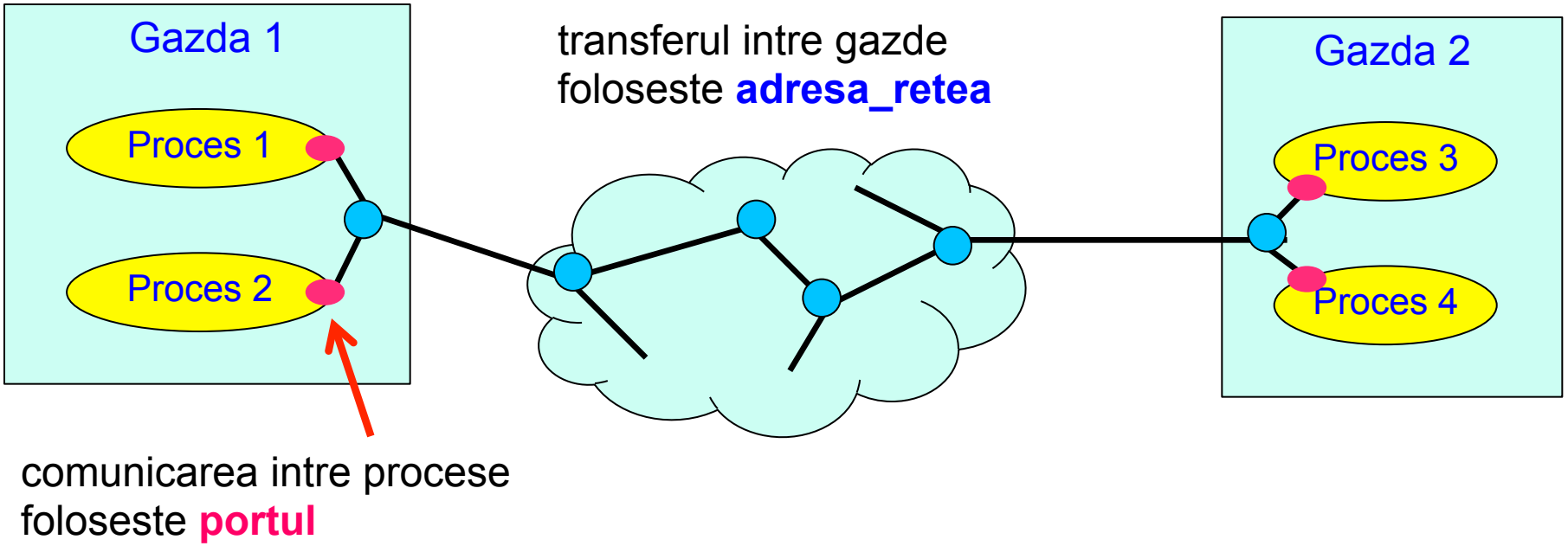


Comunicarea intre aplicatii

Reteaua asigura transmiterea pachetelor (datagrame) intre calculatoare gazda

Transportul asigura comunicarea intre aplicatii

- legatura proceselor de aplicatie cu punctele de acces la retea
- identificarea unica a unui punct de acces prin <**adresa_retea**, **port**>



Servicii ale nivelului Transport

- **Servicii furnizate**
  - transfer de date intre procese de aplicatie, folosind retele de diverse tipuri
  - interfaţa uniforma cu utilizatorii

- **Caracteristici**
  - două tipuri de servicii:
    - orientate pe conexiune (connection oriented) - TCP
    - fără conexiune (connectionless) - UDP

Socket interface

- Serviciile nivelului transport sunt accesibile ca API – Application Programming Interface
- Oferită ca bibliotecă utilizator sau funcții OS
  - API descrie cum se apelează aceste funcții
- *Socket API*
  - Originară din Berkeley BSD UNIX
  - Disponibilă pe Windows, Solaris, etc.
  - socket = punctul în care procesul de aplicație se atașază la rețea
  - identificat prin descriptor - număr (ca la fișiere)
- Nu e standard de jure ci *standard de facto*

Creare socket

int socket(int family, int type, int protocol);

socket_descr = socket (protocol_family, comm_type, protocol)

- deschide un socket
- intoarce socket_descriptor folosit in apelurile urmatoare

protocol_family selecteaza familia de protocoale

PF_INET	- protocoale Internet
PF_APPLETALK	- protocoale AppleTalk etc.

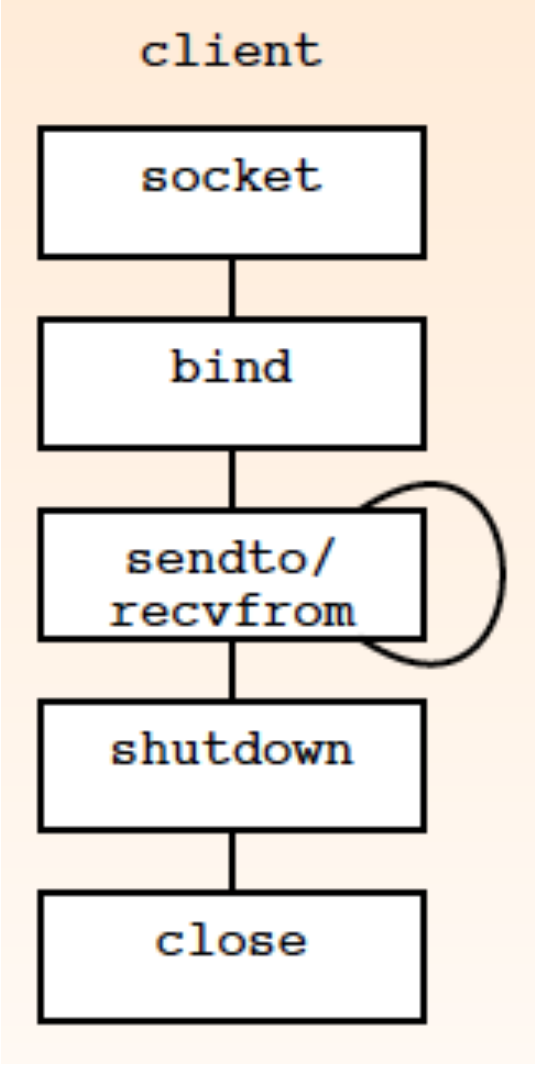
comm_type selecteaza tipul de comunicare

SOCK_DGRAM	- fara conexiune - datagrama
SOCK_STREAM	- orientat pe conexiune – flux de octeti

protocol specifica protocolul

IPPROTO_TCP	- TCP
IPPROTO_UDP	- UDP

Serviciu fara conexiune



`server` este la fel!

bind - optional pentru initiator

Serviciu fara conexiune - server

1. Creaza socket:

int s = ***socket*** *(AF_INET, SOCK_DGRAM, IPPROTO_UDP);*

2. Ataseaza la un port:

int bind (int socket_descriptor, struct sockaddr* local_address, int address_length)

bind(s, &addr, sizeof(addr)); // addr locala - vezi slide urmator !

3. Primeste mesaje:

int recvfrom (int socket_descriptor, char* buffer_address, int buffer_length, int flags,
struct sockaddr* sender_address, unsigned int sendaddress_length)

recvfrom(s, buf, BUFLEN, 0, NULL, NULL);

3. Opreste trimitere / receptie de date

shutdown (s, SHUT_RD / SHUT_RDWR / SHUT_WR)

4. Inchide socket - termina utilizarea socket

close (s)

Setare adresa

Format TCP/IP:

```
struct sockaddr_in { u_char  sin_len;      /* total length of address */
    u_char  sin_family;                    /* family of the address   */
    u_short sin_port;                      /* protocol port number    */
    struct  in_addr sin_addr;              /* IP address              */
    char    sin_zero[8]                    /* unused                  */
}
```

```
struct sockaddr_in serv_addr
memset ((char *) &serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;                  // adrese pentru Internet
serv_addr.sin_addr.s_addr = INADDR_ANY;
                                                 // foloseste adresa IP a masinii
serv_addr.sin_port = htons(portno);
                                 // converteste de la host la network byte order
```

Serviciu fara conexiune - client

1. Creaza socket:

*int s = **socket** (AF_INET, SOCK_DGRAM, PPROTO_UDP);*

2. Trimite mesaj:

int sendto (int socket_descriptor, char* data_address, int data_length, int flags, struct sockaddr* dest_address, int destaddress_length)

sendto (s, buf, BUFLEN, 0, &addr, sizeof(addr));

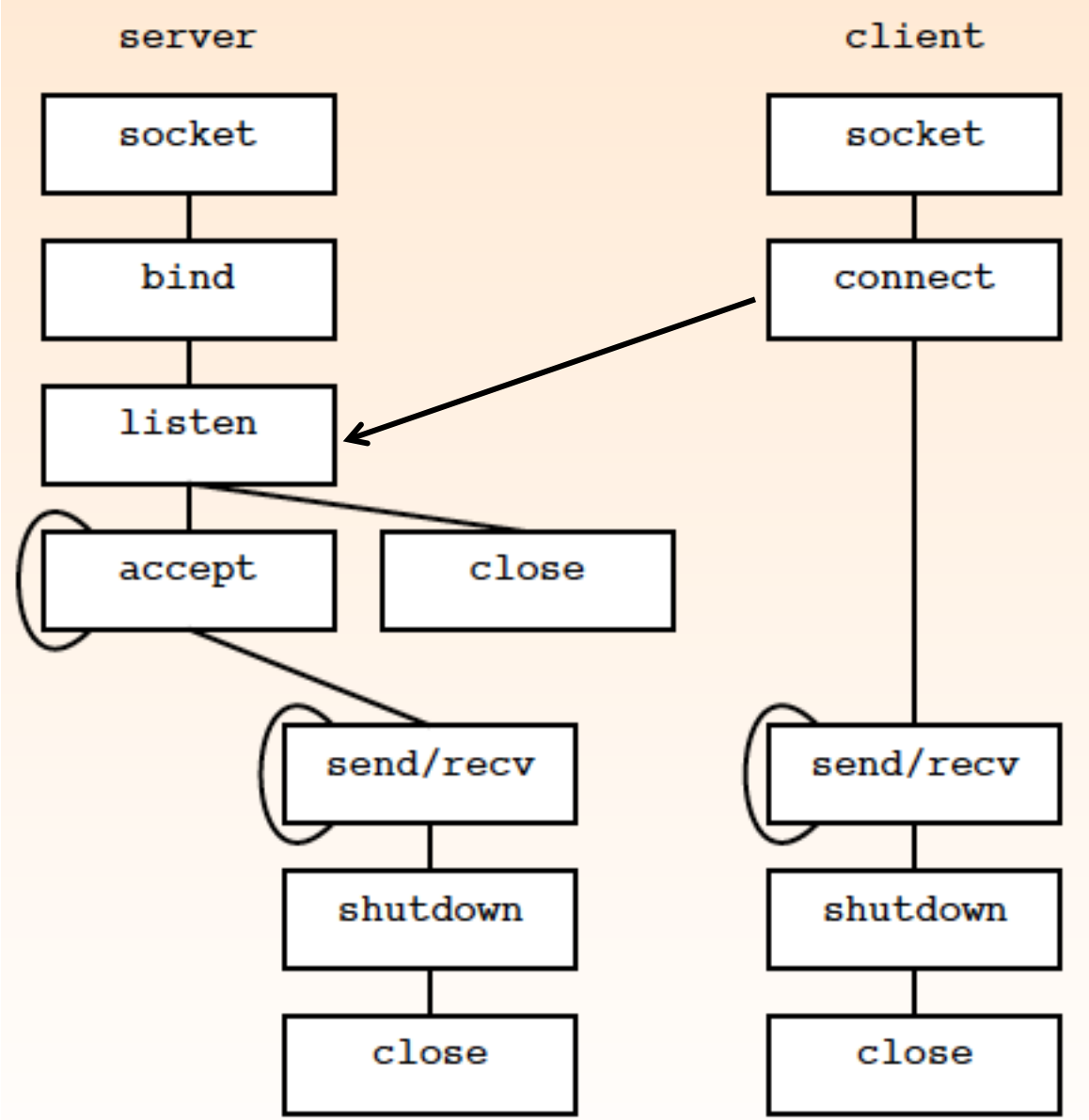
2. Opreste trimitere / receptie de date

shutdown (s, SHUT_RD / SHUT_RDWR / SHUT_WR)

3. Termina utilizarea socket

close (s)

Serviciu orientat pe conexiune



Serviciu orientat pe conexiune - server

1. Creaza socket:

*int ls = **socket** (AF_INET, SOCK_STREAM, IPPROTO_TCP);*

2. Ataseaza la un port:

bind(ls, &addr, sizeof(addr));

3. Asteapta cereri de conectare (declara nr max cereri asteapta):

int listen (int socket_descriptor, int queue_size)

listen(ls, 5);

4. repetat - Accepta o cerere de conectare:

int accept (int listen_socket_descriptor, struct sockaddr* client_socket_addr, int* client_addrlen)

int s = **accept**(ls,NULL,NULL);

5. repetat - Trimite / primeste pe s

Serviciu orientat pe conexiune - client

1. Creaza socket:

int s = ***socket*** *(AF_INET, SOCK_STREAM, IPPROTO_TCP);*

2. Conecteaza la server (uzual ptr. servicii cu conectare):

int connect (int client_socket_descriptor, struct sockaddr* server_socket_address, int server_sockaddress_length)

connect(**s**, &addr, sizeof(addr));

3. repetat - Trimite / primeste

Transmisia de date cu TCP

send (s, buf, len)

int send(int socket, const char* buf, int len, int flags);

- Intoarce numarul de octeti trimisi
  - **Poate fi mai mic decat len !**

recv (s, buf, max_len)

int recv(int socket, char* buf, int len, int flags);

- Intoarce numarul de octeti primiti
  - **Poate fi mai mic decat max_len!**

flags indica optiuni speciale

MSG_OOB – trimite/primeste date out-of-band

MSG_PEEK – livreaza date primite, dar trateaza ca necitite

Inchiderea conexiunii TCP

- Elibereaza resursele asociate conexiunii
- Informeaza capatul celalalt de inchiderea conexiunii
- API
  - shutdown (s,SHUT_RD/SHUT_RDWR/SHUT_WR)

    int shutdown (int socket, int how)
    - opreste primirea – rejecteaza datele care sosesc
    - opreste transmiterea – ignora datele inca netrimise
    - ambele
  - close (s)

    int close (int socket);
    - inchide socket (elibereaza structurile de date din kernel)

Protocoale de Transport

Doua potocoale majore

- UDP
- TCP

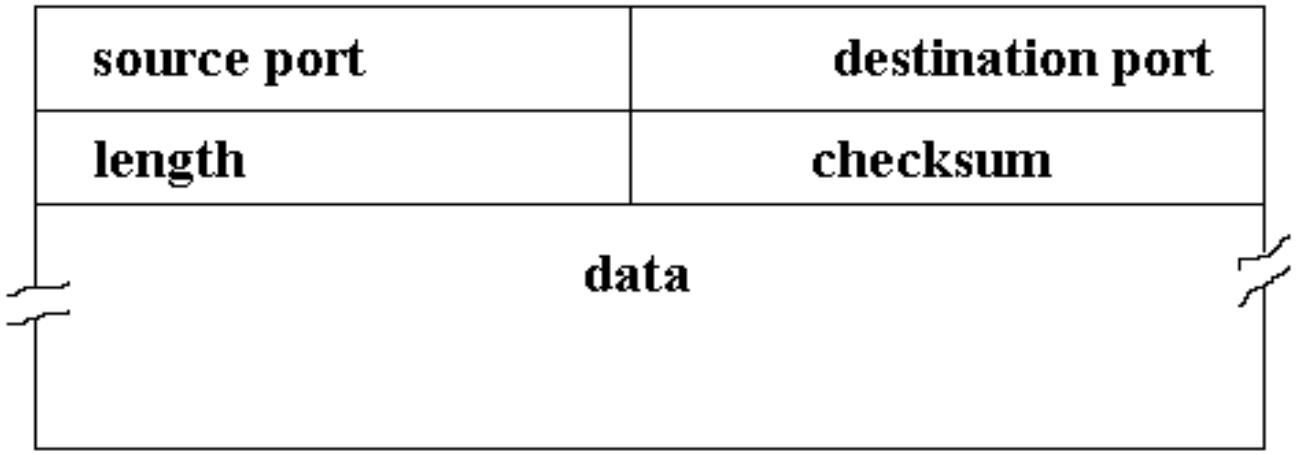
Aspecte discutate

- Formatul datelor
- Adresare
- Functionare

UDP - User Datagram Protocol

- UDP livreaza datagrame utilizator - *user datagrams*
  - Livrare “Best effort” – datagramele pot fi pierdute, primite in alta ordine etc.
  - Sume de control pentru integritate
- Puncte de capat UDP = *protocol ports* sau *ports*
- UDP identifica adresa Internet si numar port pentru sursa si destinatie
- *Destination port* si *source port* pot diferi.

Antet UDP



checksum (calculat la fel ca la TCP) dar nefolosit

Nu control flux

Nu control erori

Nu retransmisie

Aplicatii care folosesc UDP

- DNS
- Voice over IP
- Online Games
- Se foloseste atunci cand:
  - Latenta foarte importanta
  - Livrarea tuturor datelor nu e necesara
  - Retransmisiile sunt implementate de aplicatii cand e nevoie (DNS)

TCP - Transmission Control Protocol

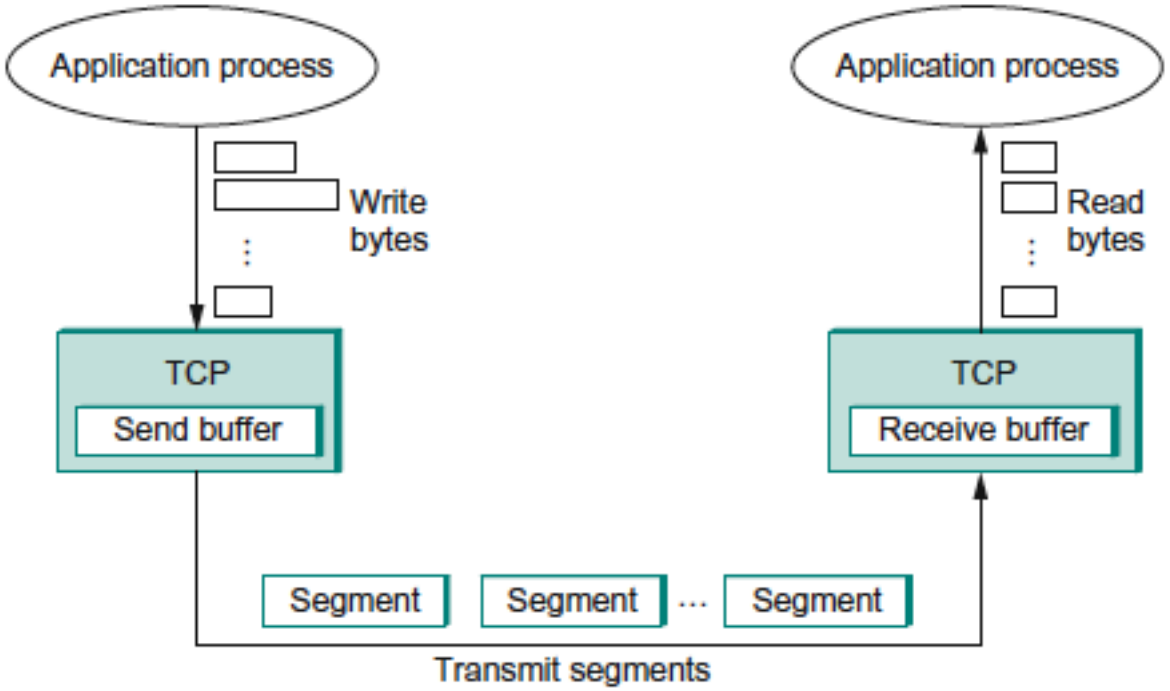
Livrare sigura pe retea nesigura (datagrame)

- ***Cel mai folosit protocol de transport***
  - Web
  - Email
  - SSH
  - Chat
  - Video streaming
  - Peer-to-peer

Cateva porturi standard

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial File Transfer Protocol
79	Finger	Lookup info about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

TCP este orientat pe flux de octeti

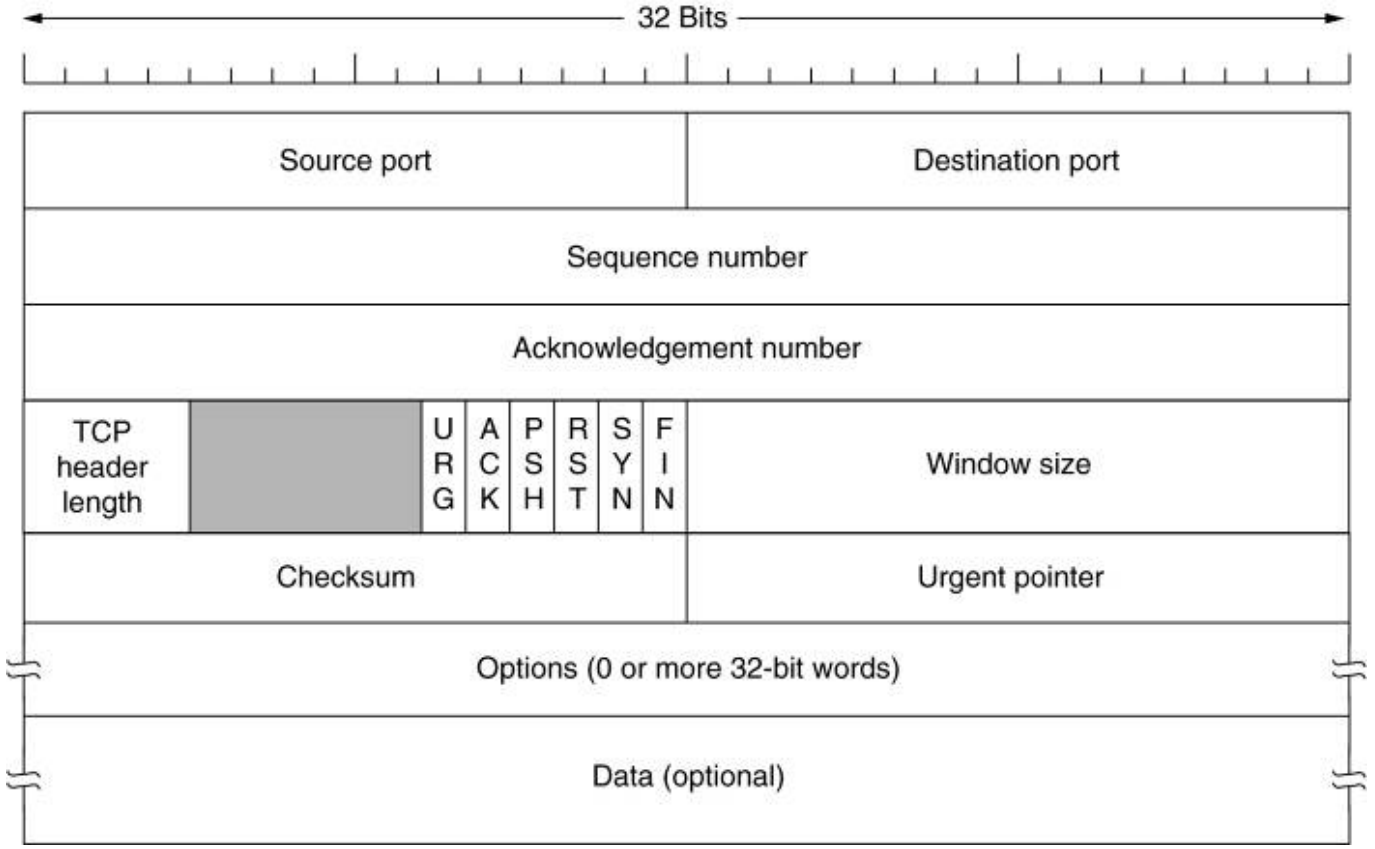


- Aplicația transmițătoare scrie octeți în conexiunea TCP
- TCP la sursă memorează octeții într-un buffer și transmite segmente
- TCP la destinație memorează segmentele într-un buffer
- Aplicația receptoare citește octeți (câți vrea!) din conexiunea TCP

Caracteristici

- Orientat pe conexiune
- Interfață flux (Stream)
  - transmisie si receptie siruri de octeti
- Face controlul congestiei adaptând viteza de transmisie la condițiile rețelei
- Garantează transmisie în ordine și sigură a datelor pe o conexiune
- Full duplex
- Stabilire sigură a conexiunii - three-way handshake
- Eliberare lină a conexiunii – fără pierdere de date

Antet segment TCP



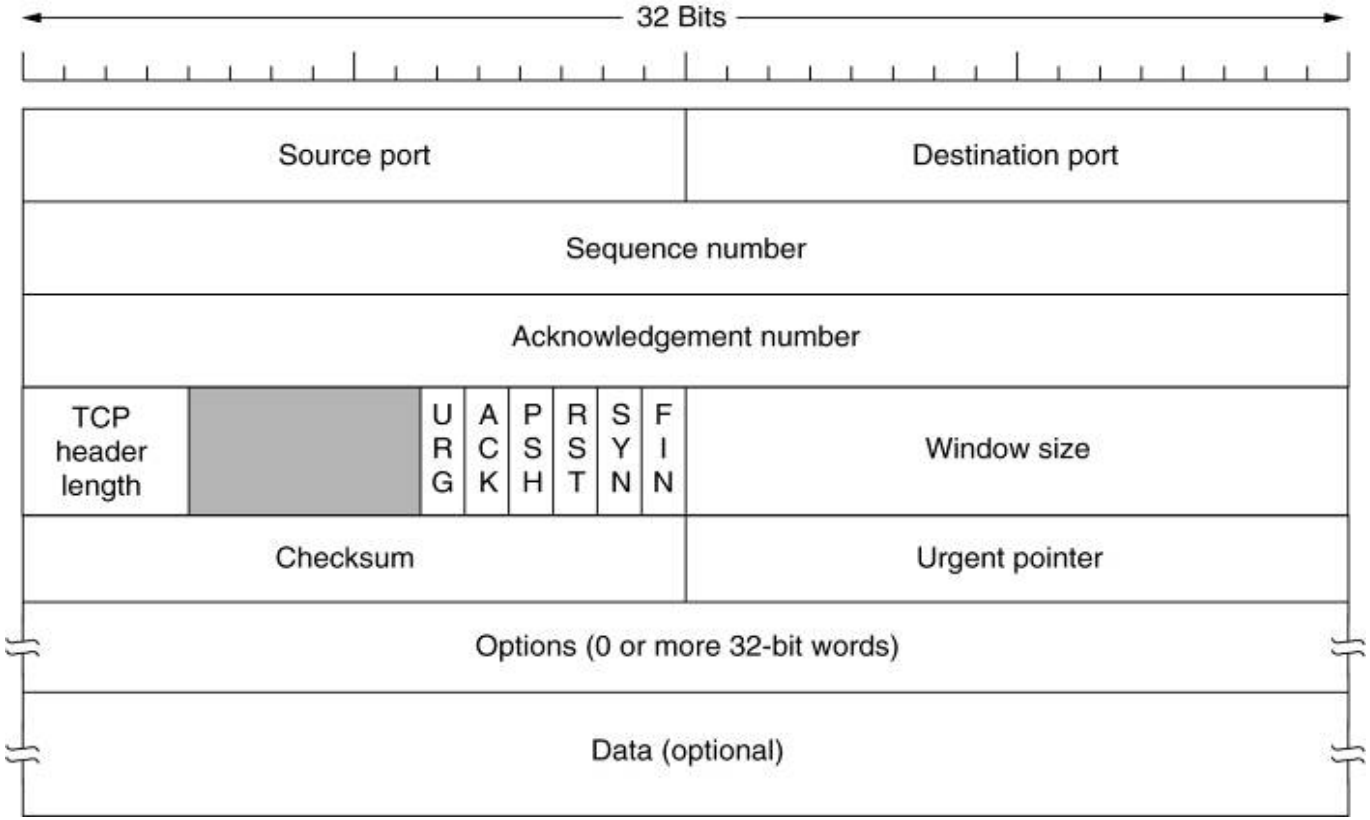
Sequence number – numarul primului octet din segment

Acknoledgement number – numarul urmatorului octet asteptat

Window size - numărul de octeţi care pot fi trimişi, începând cu octetul confirmat

Urgent Pointer – deplasamentul, faţă de *Sequence number,* ptr. info. urgentă

Antet segment TCP



URG Urgent pointer valid

ACK Acknowledge Number valid

PSH - push information to user

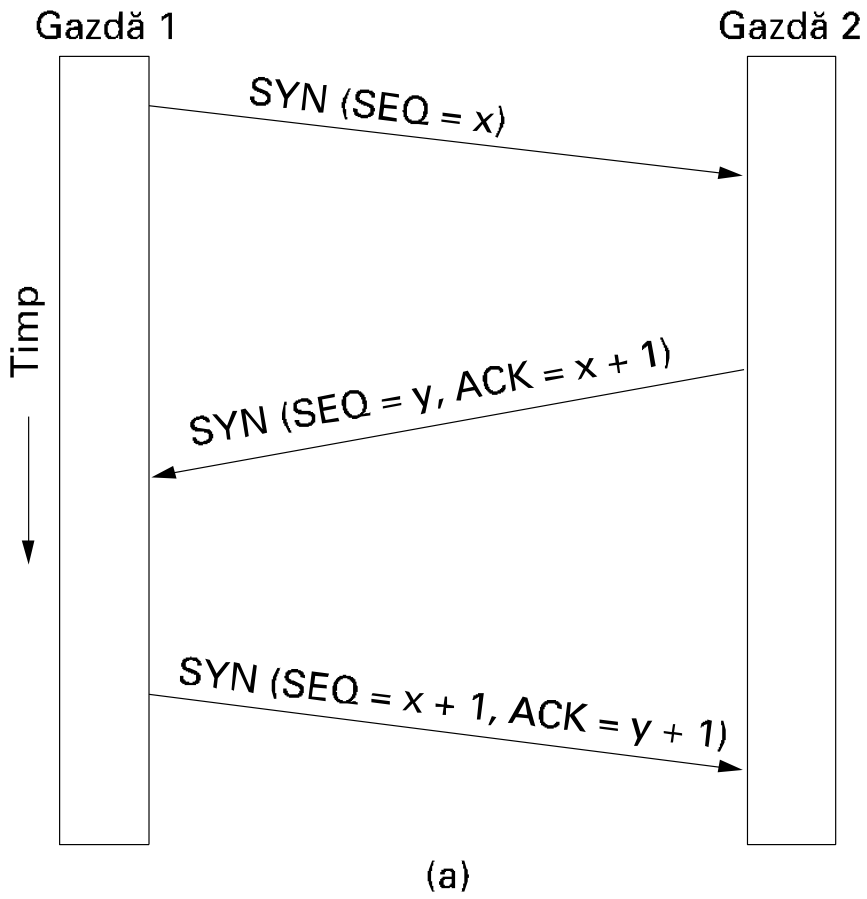
RST - close a connection due to an error

SYN - open connection

FIN - close a connection

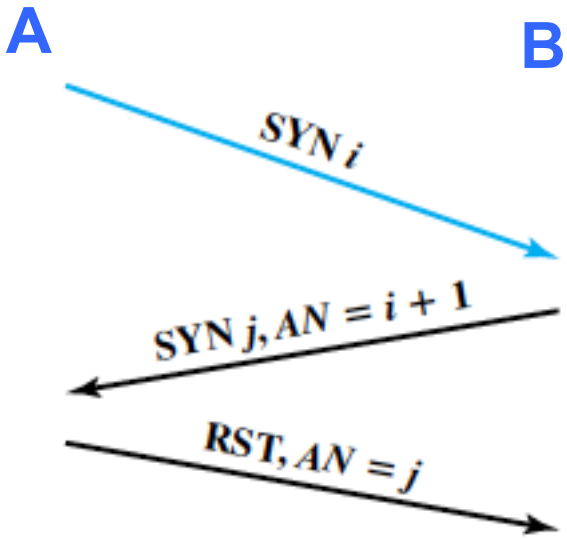
Options: e.g. max TCP payload (implicit **536** octeti), selective repeat

Stabilirea conexiunii - Three way handshaking



(a) Cazul normal.

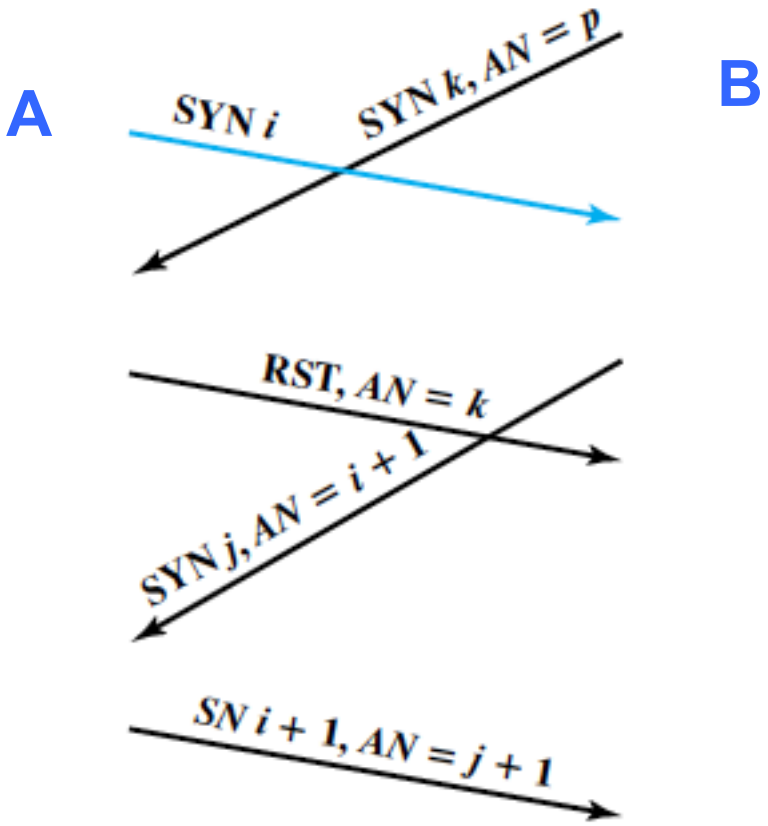
Rejectarea conexiunii



SYN intarziat

B accepta

A rejecteaza

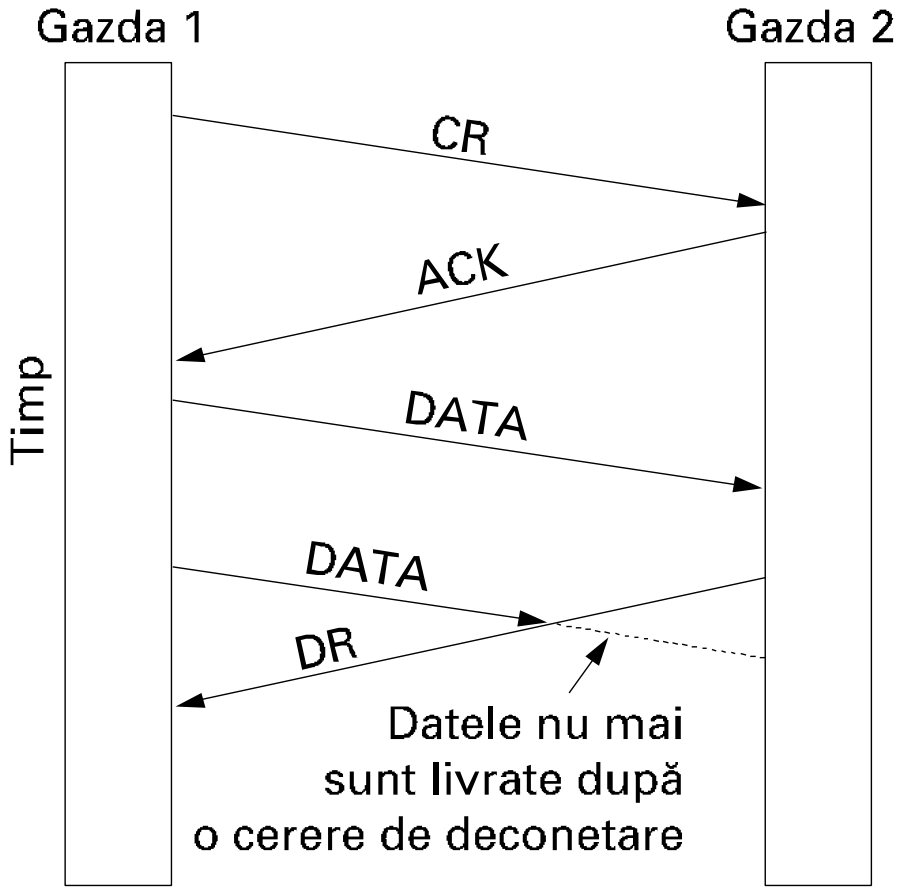


A initiaza conexiune (SYN i)

SYN k intarziat - A refuza

B accepta SYN i – raspunde cu SYN j

Deconectare abrupta cu pierdere de date



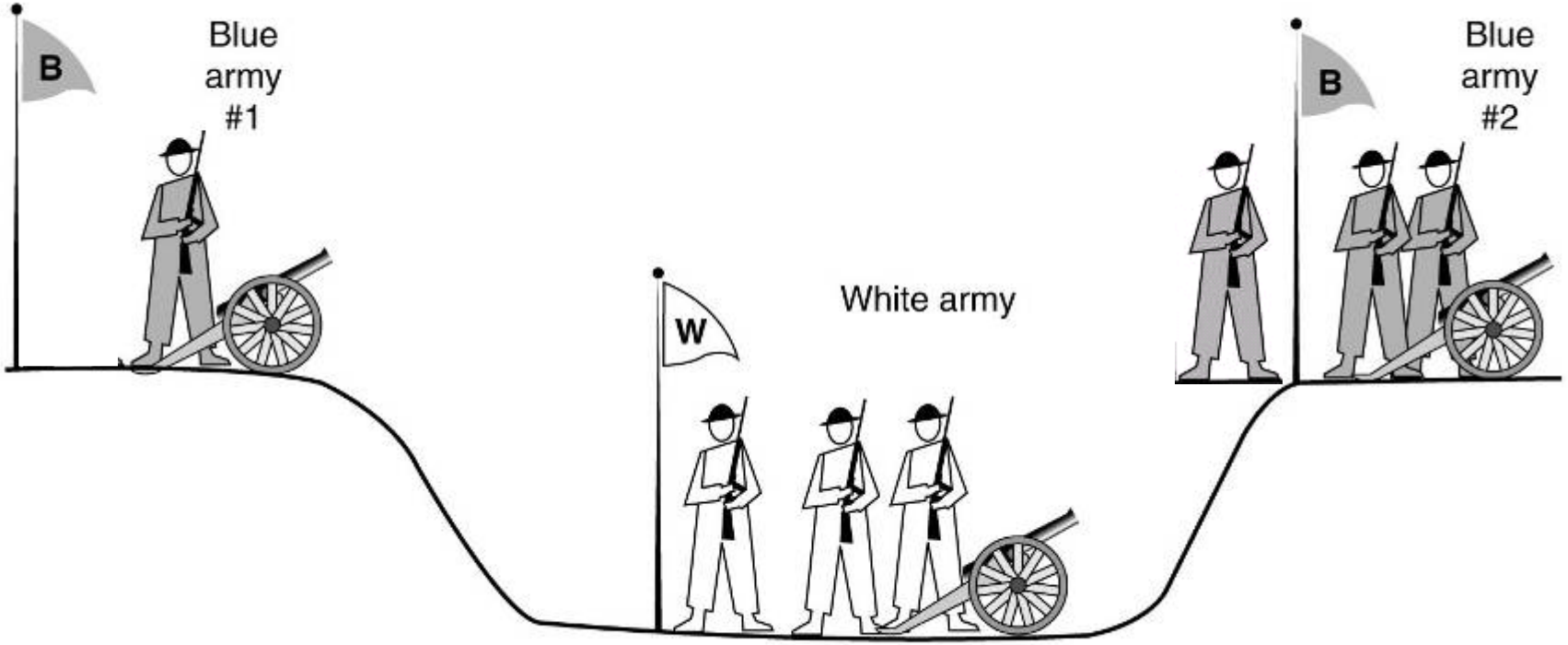
CR – Connection request

echivalent cu SYN

DR – Disconnect request

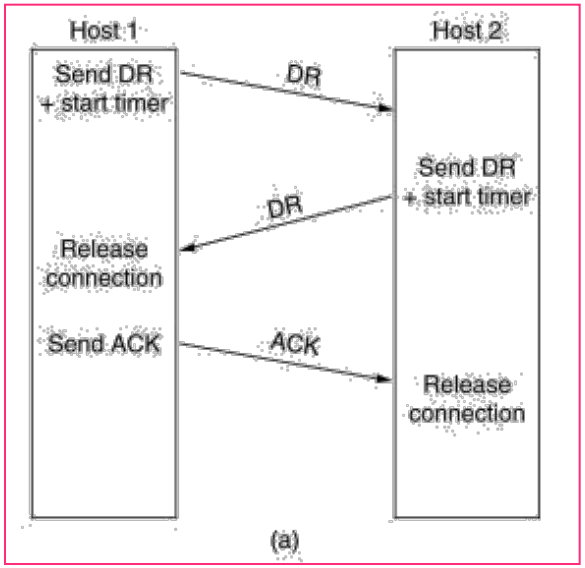
echivalent cu FIN (TCP)

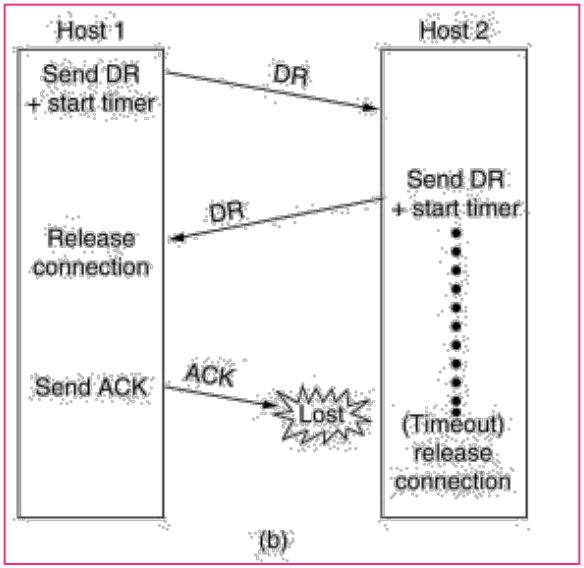
Problema celor doua armate



Patru scenarii de eliberarea conexiunii

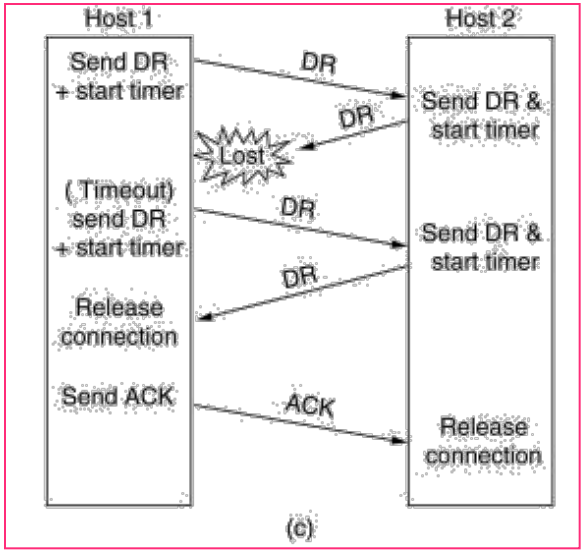
normal

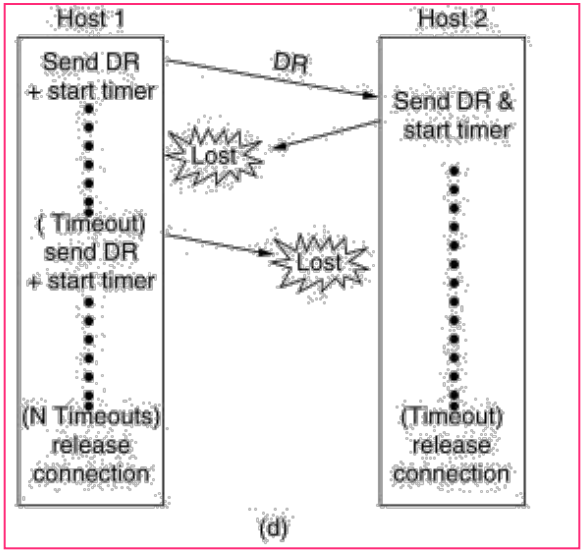




ACK final pierdut

raspuns pierdut





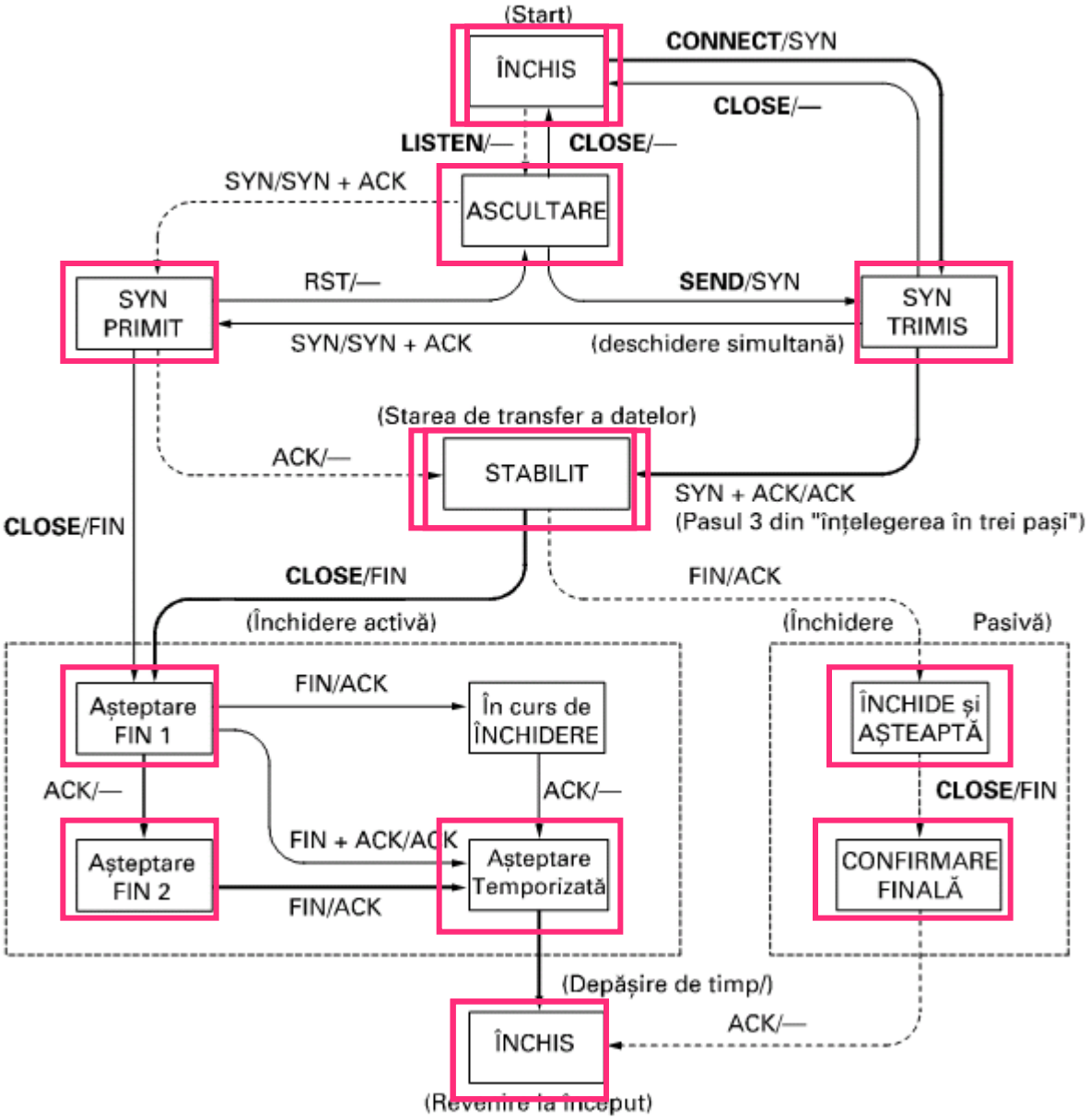
Raspuns si DR pierdute

Starile folosite in managementul conexiunii TCP

Stare	Descriere
CLOSED (ÎNCHIS)	Nici o conexiune activă sau în aşteptare
LISTEN (ASCULTARE)	Serverul aşteaptă recepţionarea unui apel
SYN RCVD (Recepţie SYN)	Recepţie o cerere de conexiune; aştept ACK
SYN SENT (Transmisie SYN)	Aplicaţia începe deschiderea unei conexiuni
ESTABLISHED (STABILIT)	Starea normală de transfer a datelor
FIN WAIT 1 (Aşteptare FIN 1)	Aplicaţia a anunţat că termină
FIN WAIT 2 (Aşteptare FIN 2)	Partenerul este de acord cu eliberarea conexiunii
TIMED WAIT (Aşteptare Temporizată)	Se aşteaptă „moartea” tuturor pachetelor
CLOSING (În curs de INCHIDERE)	Ambele părţi încearcă simultan închiderea
CLOSE WAIT (Închide şi Aşteaptă)	Partenerul a iniţiat eliberarea conexiunii
LAST ACK (Confirmare Finală)	Se aşteaptă ACK de la client

Gestiunea Conexiunilor TCP (mașina de stări)

- client (activ) cere conectarea
- server (pasiv) raspunde
- Linie groasa continua = cale normala pentru client
- Linie intrerupta = cale normala server
- Linii subtiri = evenimente exceptionale
- Fiecare tranzitie are eticheta eveniment / actiune



Corectitudinea segmentelor TCP

1 - Suma de control (checksum) din antet – include

1. antet
2. incarcatura segment TCP (date)
3. un pseudo-antet

adresele IP sursa si destinatie

protocolul (6 pentru TCP)

lungime segment TCP (include antetul)

Algoritm: la **transmisie**

aduna cuvinte de 16 biti in complement fata de 1

complementeaza rezultatul

scrie rezultatul in antet

la **receptie**

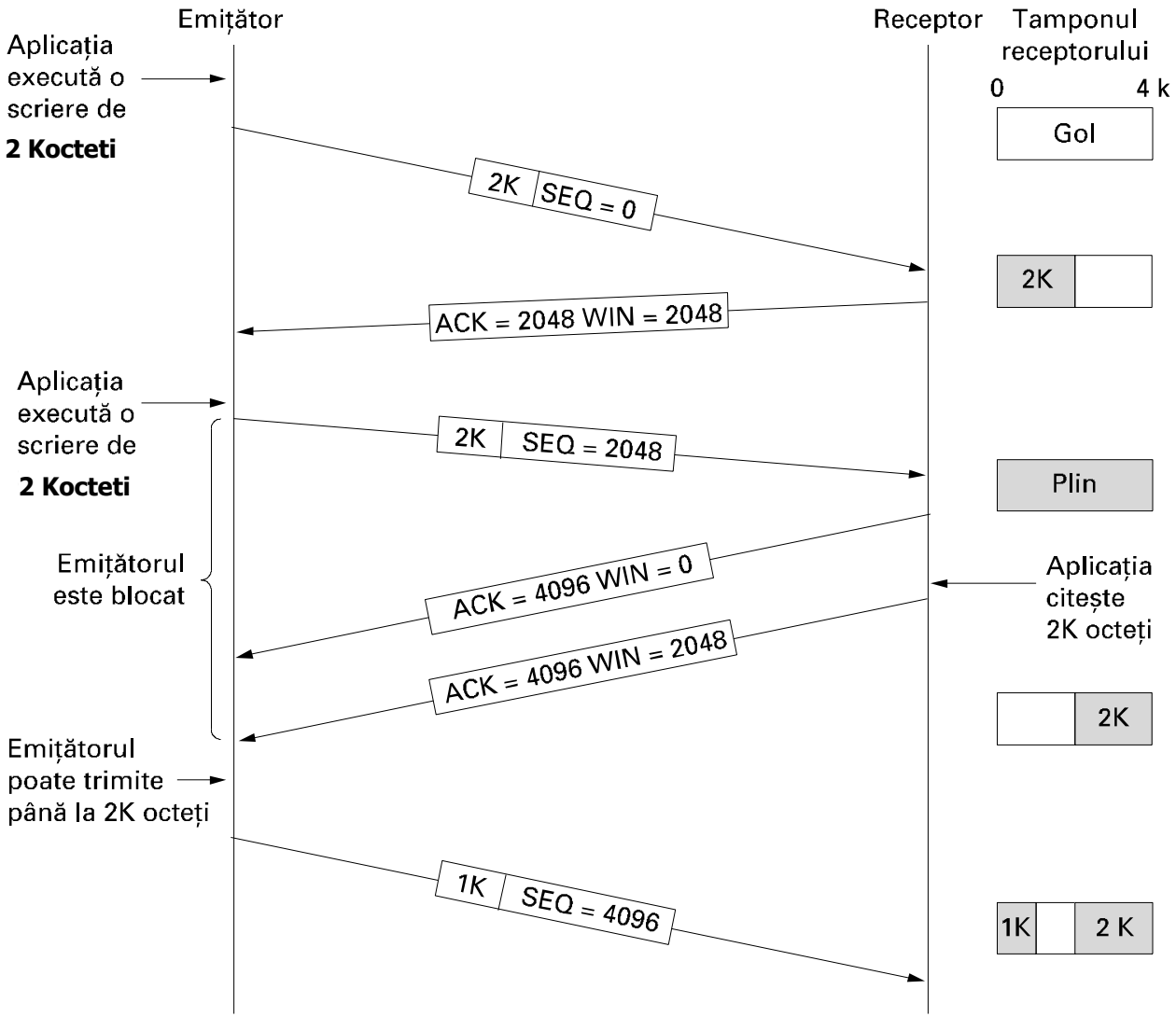
aduna cuvinte de 16 biti – rezultatul trebuie sa fie zero

2 - Acknowledgement number

Corectia se face prin retransmisie

Controlul fluxului de date

Receptorul specifica fereastra de receptie



Un emitator blocat poate trimite:

- date urgente
- un segment de 1 octet ptr a afla fereastra (daca anuntul precedent al receptorului s-a pierdut)

Probleme dimensiuni câmpuri antet

Numere secvență de 32-biti

- Durata ciclu de numarare – depinde de viteza transmisie
  - 1 saptamana pentru 56kbps
  - 57 min pentru 10Mbps
  - 34 sec pentru 1Gb **(sub 120 sec care este timp viata maxim in Internet)**

Problema: cum se diferentiaza segmentele cu acelasi numar de secventa?

Soluție

- Folosire opțiuni TCP (RFC 1323)
  - TCP Timestamps – rezolva numere de secventa duplicate

Probleme dimensiuni câmpuri antet (2)

Fereastra receptor

Transmitere 500 Kb pe legatura 1 Gbps ocupa 500 µsec

La intarziere de 20 ms pe sens confirmarea se primeste dupa 40 ms =>
ocupare canal pe un ciclu complet este mică - 1.25%

Ocupare completă in ambele direcții: produs bandwidth*delay =
= 1 Gbps * 40 ms = 40 milioane biti

Condiție - fereastra receptor >= bandwidth*delay

Soluție

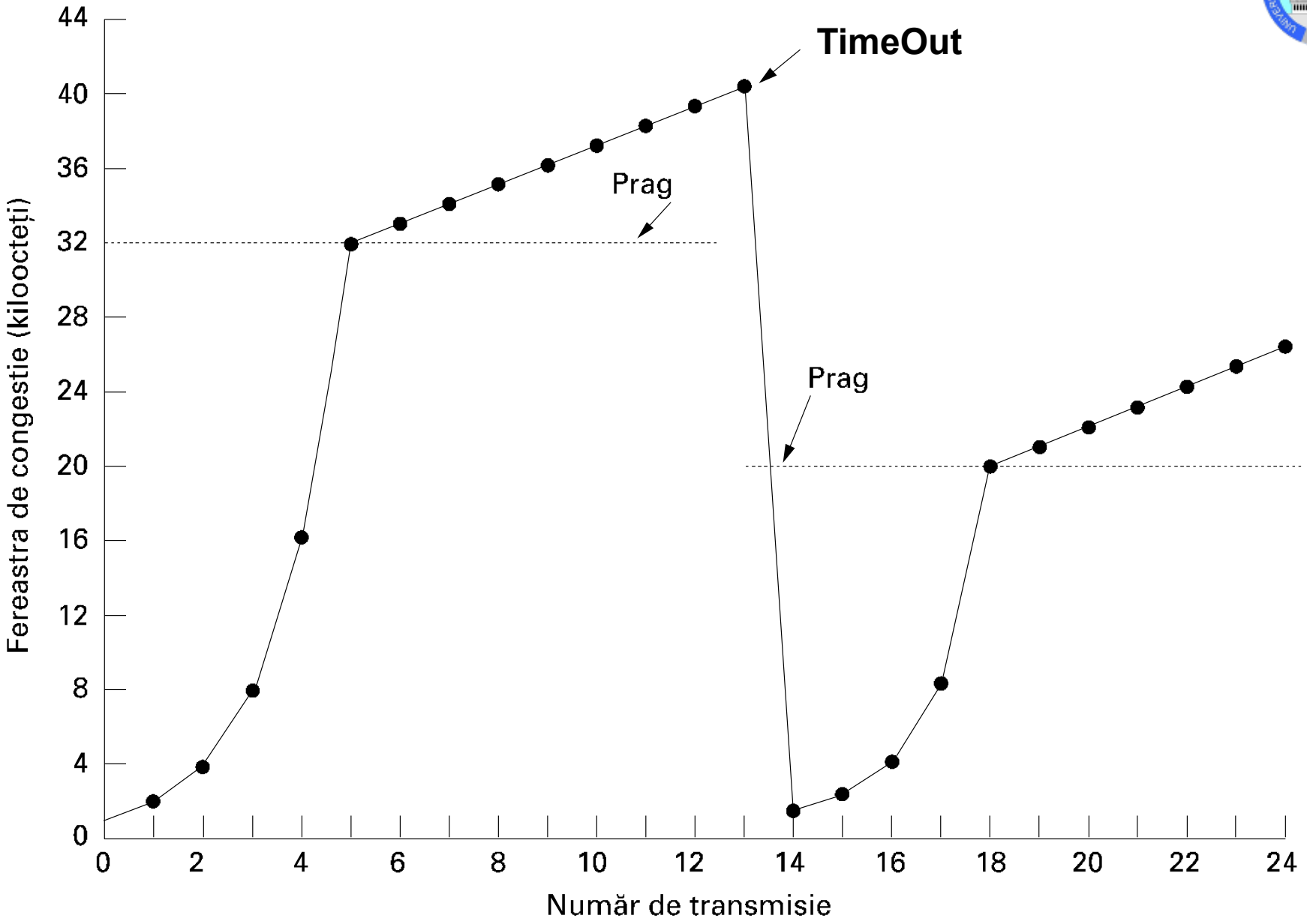
- Folosire opțiuni TCP (RFC 1323)
  - Window Scale – factor de scalare a câmpului Window size de până la 2**14 ori
    ➔ ferestre de până la 2**30 octeți

Controlul congestiei

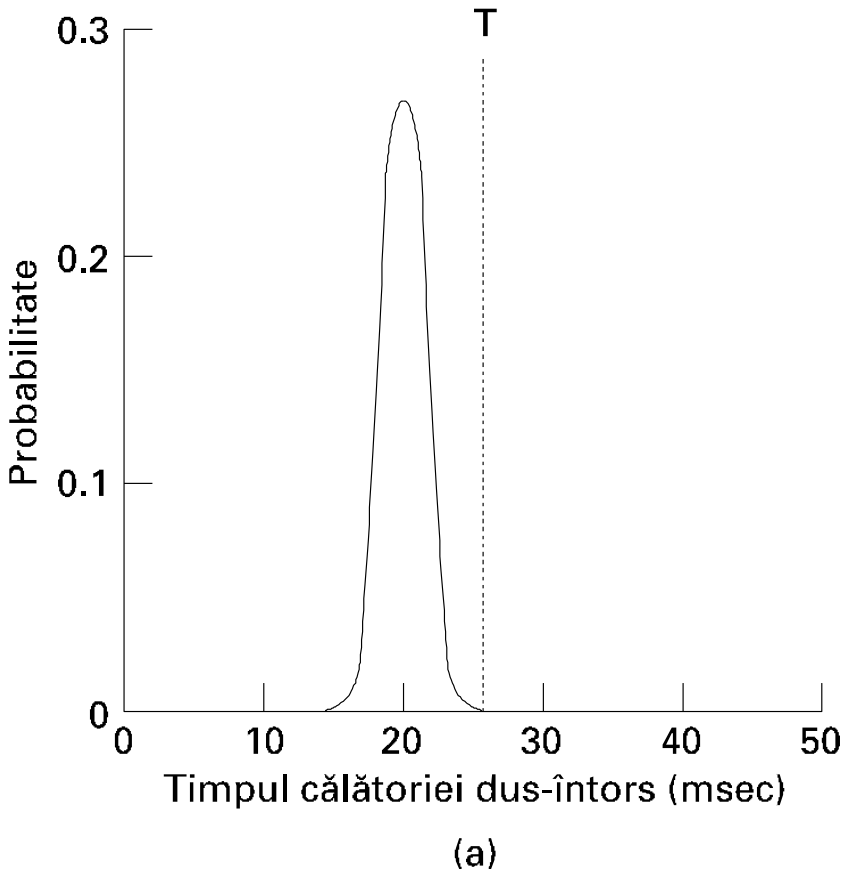
- Fluxul de date transmis pe o conex. TCP limitat de minimul dintre:
  - dimensiunea fereastrei receptorului
  - capacitatea retelei (**fereastra de congestie**)
- Algoritm de stabilire fereastra de congestie
  - transmite un segment de dimensiune maximă pe conexiunea stabilita
  - se dubleaza volumul de date – rafală de segmente - (creștere exponențială) la fiecare transmisie confirmată la timp
  - la primul timeout se opreste procedeul si fereastra ramane la valoarea ultimei transmisii confirmate la timp (fara timeout)

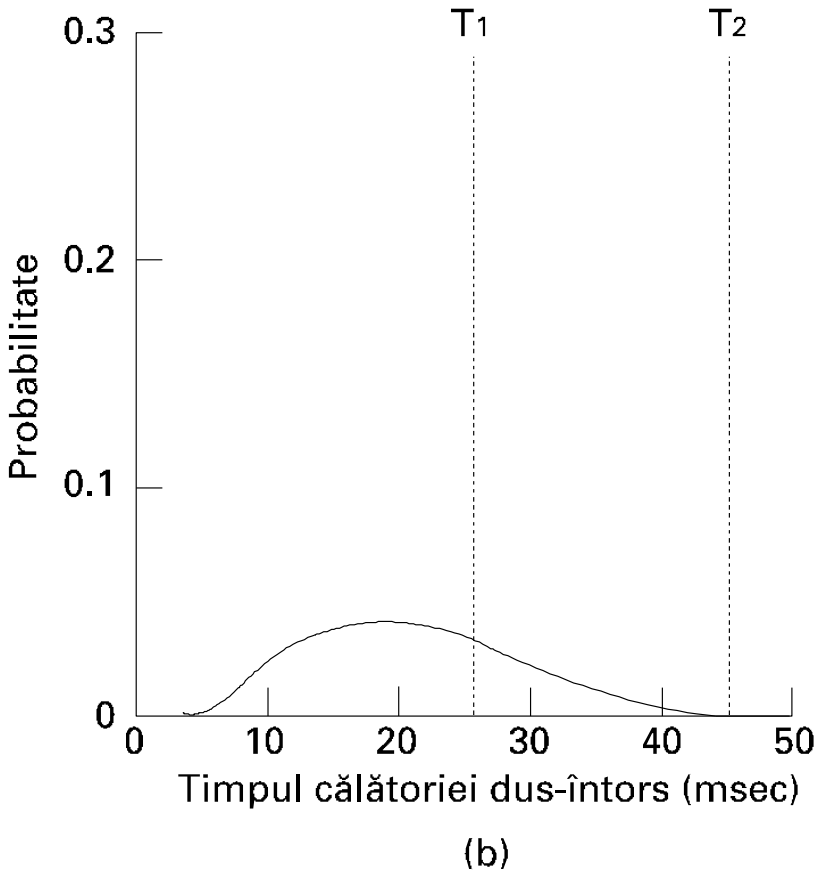
Controlul congestiei

- Algoritmul de control al congestiei
  - foloseste un **prag** (threshold)
  - la un timeout pragul setat la jumatate din fereastra de congestie
  - se aplica procedeul de crestere (exponentiala) a fereastrei de congestie pana se atinge pragul
  - peste prag se aplica o crestere liniara (cu cate un segment de dimensiune maximă o dată)


TimeOut
Prag
Prag
Fereastra de congestie (kiloocteți)
Număr de transmisie
0
4
8
12
16
20
24
28
32
36
40
44
0
2
4
6
8
10
12
14
16
18
20
22
24

Gestiunea ceasurilor in TCP





(a) Densitatea de probabilitate a timpilor de sosire ACK in nivelul legatura de date.

(b) Densitatea de probabilitate a timpilor de sosire ACK pentru TCP.

Stabilire time-out

- Setare proasta – performante slabe:
  - Prea lung (T2) – transmitatorul asteapta mult ptr retransmisie
  - Prea scurt (T1) – trafic inutil generat de transmitator
- Timeout diferit la fiecare conexiune - setat dinamic
- Transmitatorul alege *Retransmission TimeOut* (RTO) pe baza *Round Trip Time* (RTT)

  M este timpul masurat pana la primirea ack

  RTT = α*RTT + (1- α)*M cu α = 7/8

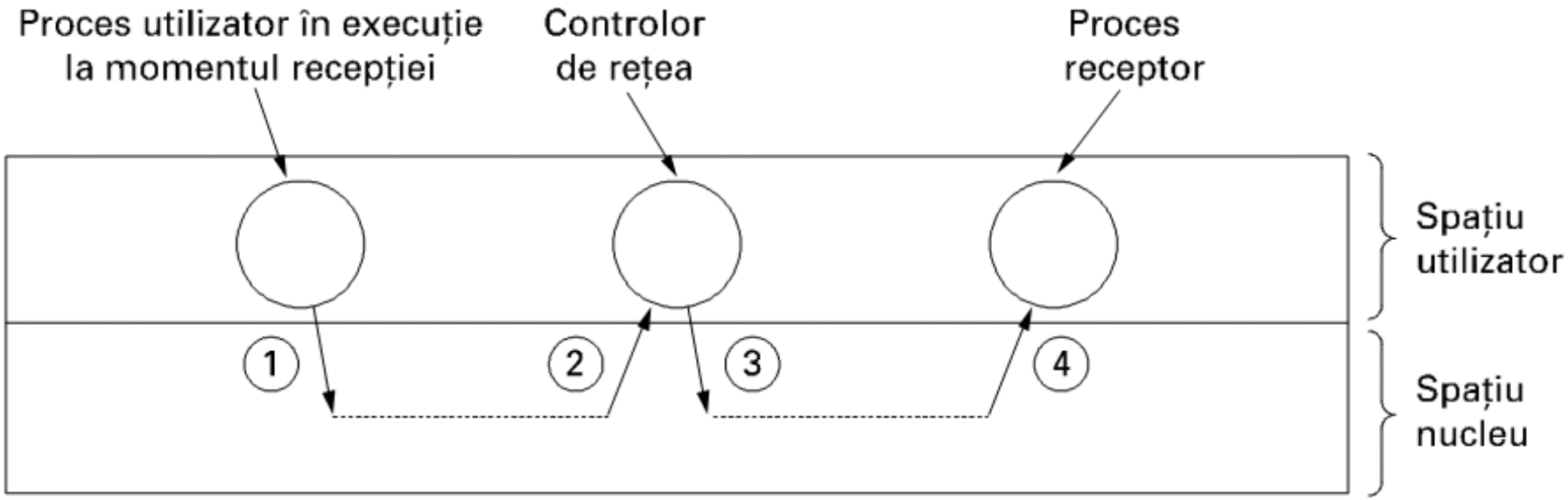
  RTO = β*RTT cu β = 2
- Alegere dupa *deviatia standard (DS);*

  D aproximeaza DS

  D = α*D + (1- α)*|RTT-M|

  RTO = RTT + 4*D

Proiectarea pentru performanta (optional)



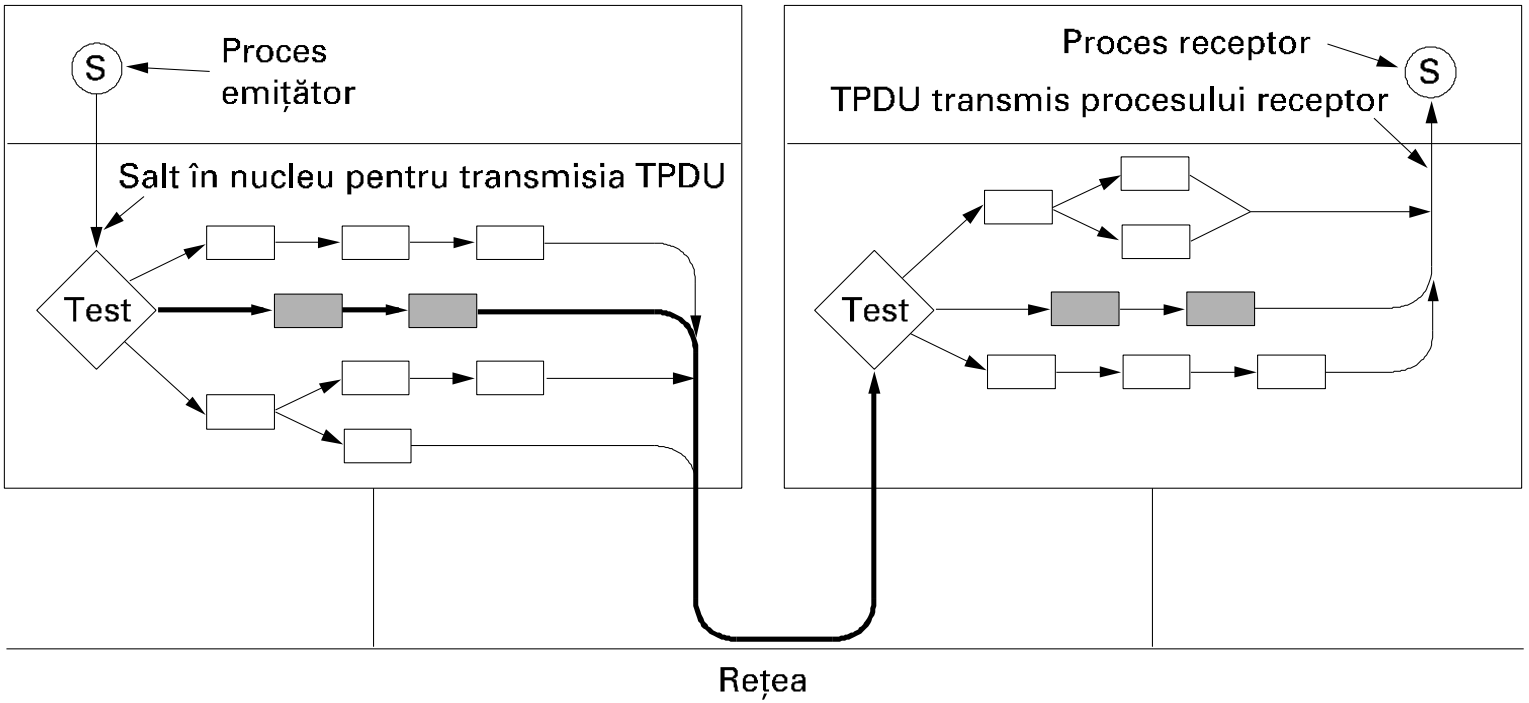
Problema: Schimbari de context suplimentare pentru a manevra pachete cu un Controlor de retea in spatiul utilizatorului.

acesta asambleaza segmentele pentru a le furniza apoi procesului receptor

Solutia: acumularea mesajelor sosite, in memorie tampon si livrarea in grup catre utilizator.

Similar, gruparea mesajelor de transmis, in memorie tampon si trimiterea lor grupata

Prelucrare Fast TPDU (optional)



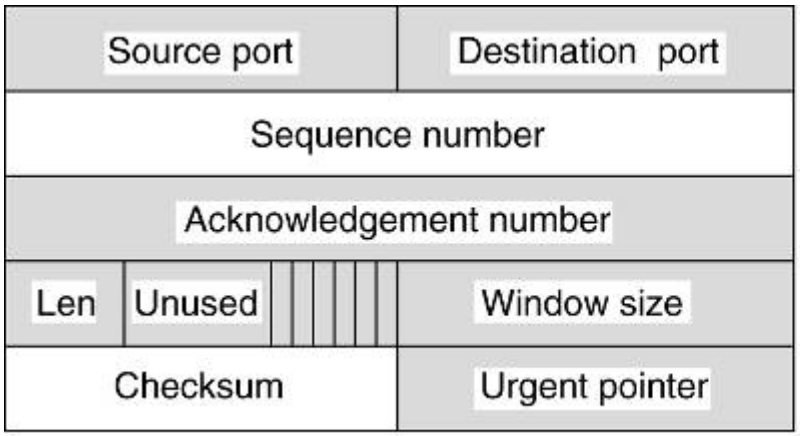
Calea rapida intre transmitator si receptor este cu line groasa. Pasii sunt reprezentati cu gri.

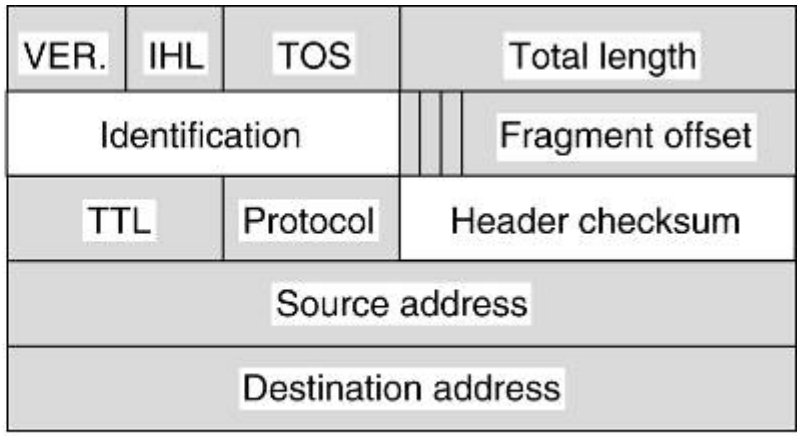
Test caz normal:

starea = ESTABLISHED
nu se incearca inchiderea conexiunii,
TPDU normal (nu out-of-band),
suficient spatiu la receptor

} –> alege fast path

Prelucrare Fast TPDU (2)





(a) Antet TCP.

(b) Antet IP.

Transmitator

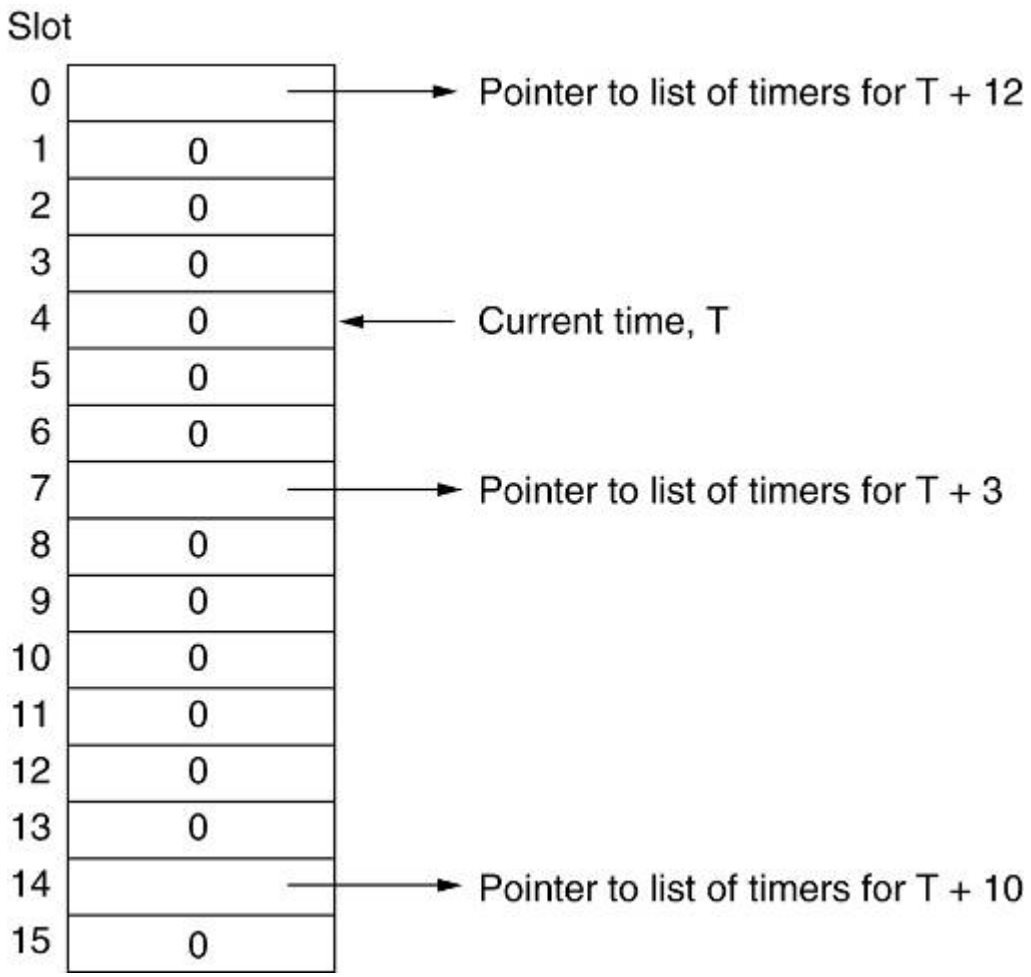
Pastreaza un prototip de mesaj in entitatea de transport – campuri nemodificate in unitati de date consecutive; la fel pentru pachet IP

Campurile gri sunt luate din prototip fara modificari. Celelalte se calculeaza pentru fiecare segment

Receptor

- Localizeaza inregistrarea conexiunii din TPDU intr-o tabela hash
- Testeaza pentru cazul normal (similar cu transmisia)
- Actualizeaza inregistrarea conexiunii (starea curenta)
- Copiaza datele la utilizator si calculeaza suma de control
- Transmite confirmarea

Prelucrare Fast TPDU (timer management)



"Timing wheel."

1 slot = 1 clock tick

Timp curent T=4

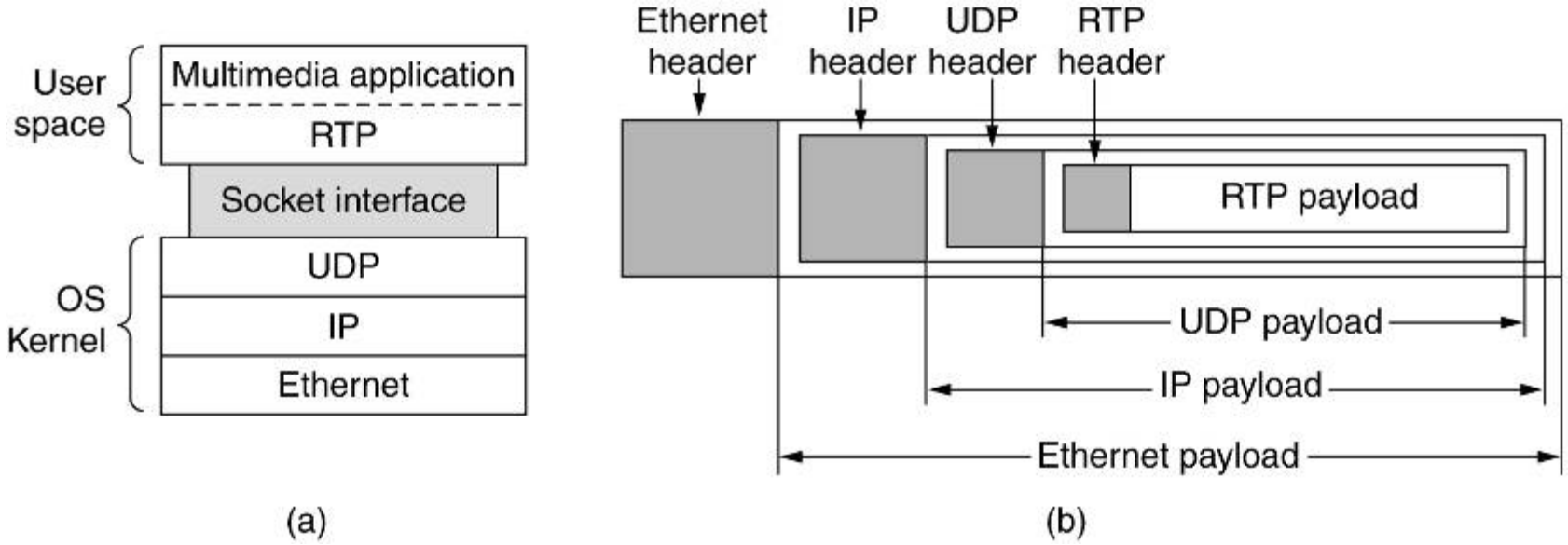
Programare time-out peste 7 tick-uri –> insereaza eveniment in lista de la slot 11

Anulare -> cauta in lista de la slot corespunzator si elimina eveniment

La fiecare Clock tick, un pointer avaneaza cu un slot, circular

Daca slot nevid, proceseaza toate evenimentele

The Real-Time Transport Protocol



(a) Pozitia RTP in stiva de protocoale. (b) Format pachet.

Functie principala: multiplexare fluxuri RTP in flux UDP

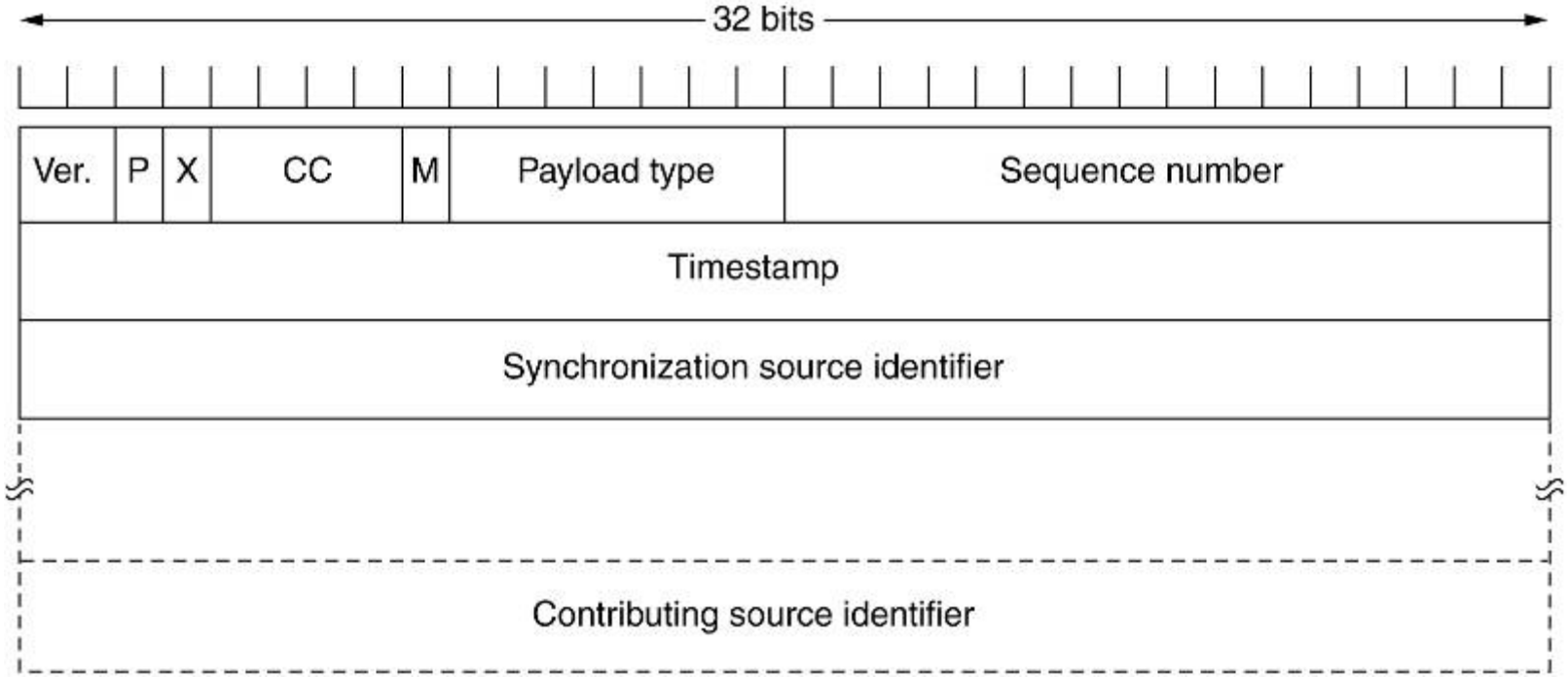
Unicast sau multicast

Fara retransmitere – receptorul “interpoleaza” packetele absente

Formatul datelor din "RTP payload" este specific aplicatiilor

RTP permite definirea unor **profile** si, pentru fiecare profil, a mai multor formate

Antet RTP



P – padding - pachet extins la multiplu de 4 octeti

X – extension - antet extins (primul cuvant da lungimea)

CC – no. antete surse contributoare

M – mark (specific aplicatiei. Ex start video frame)

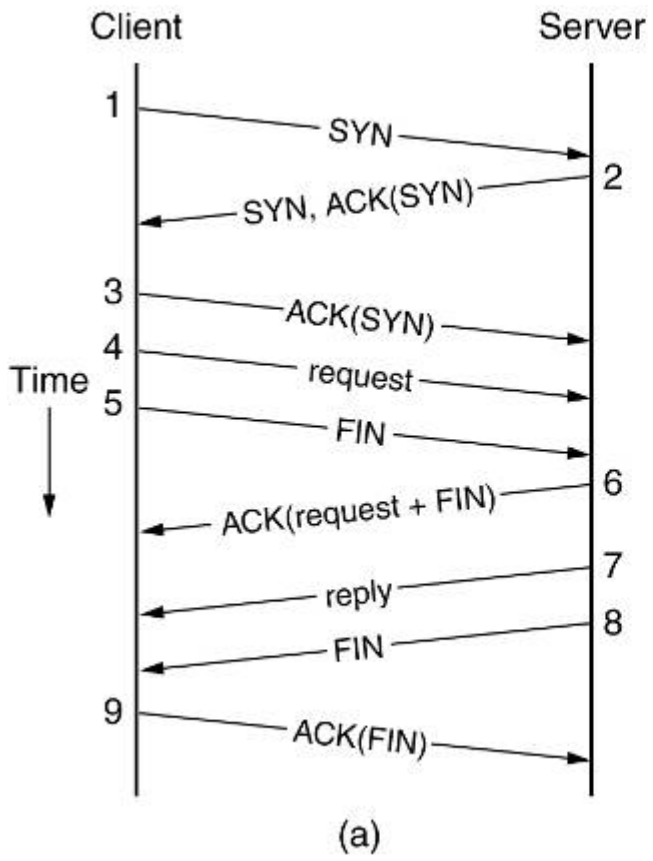
Payload type – e.g. MP3

Timestamp – amprenta de timp relativa la inceputul fluxului (receptorul sa foloseasca proba la timpul potrivit)

Synchro – identifica sursa de sincronizare (fluxul caruia ii apartine pachetul)

Contrib – lista surselor care au contribuit la continutul curent; folosit pentru mixere

TCP Tranzactional



(a) RPC folosind TPC normal.

(b) RPC folosind T/T

Permite transfer de date la "connection setup"