Networking in Sistemul de Operare

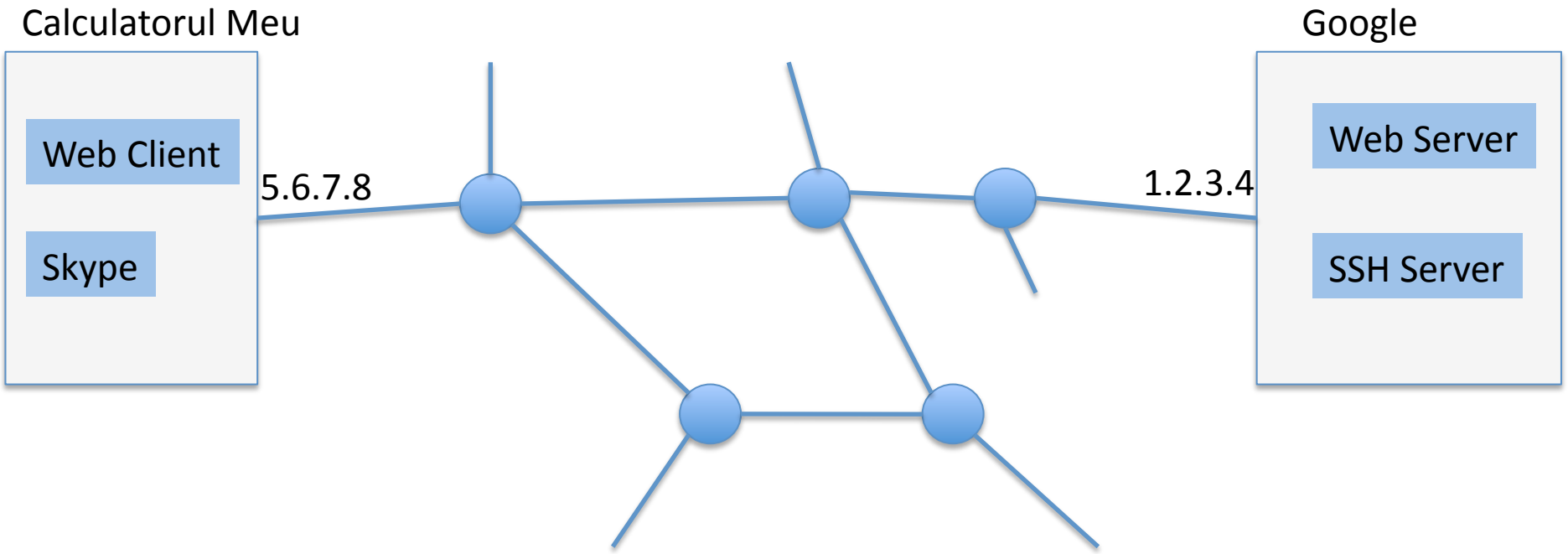
Sisteme de Operare

Curs 11

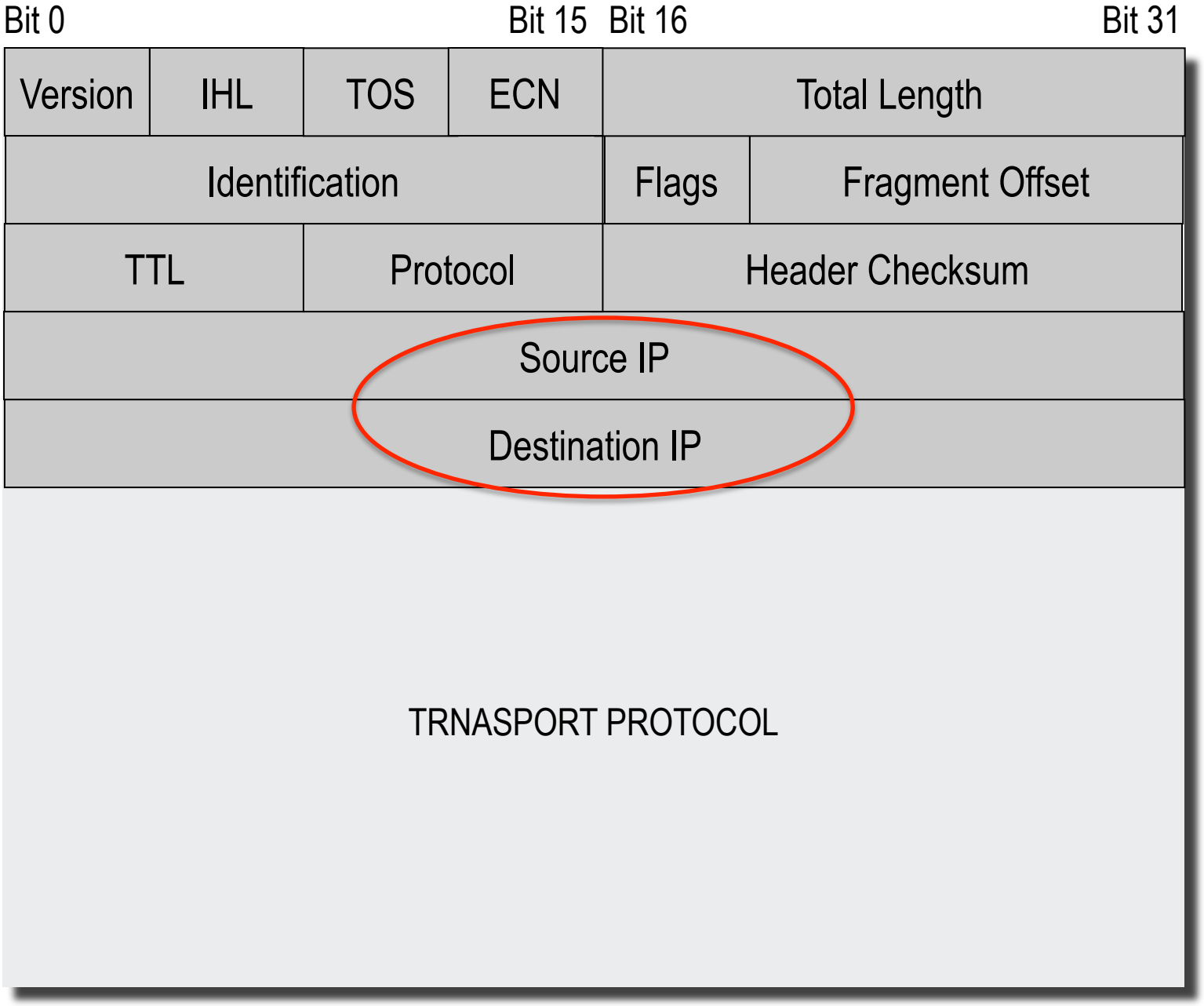
Cum folosim Internetul?

- Aplicatiile doresc sa transmita si sa receptioneze datele
- Cum facem asta folosind modelul best-effort oferit de IP?

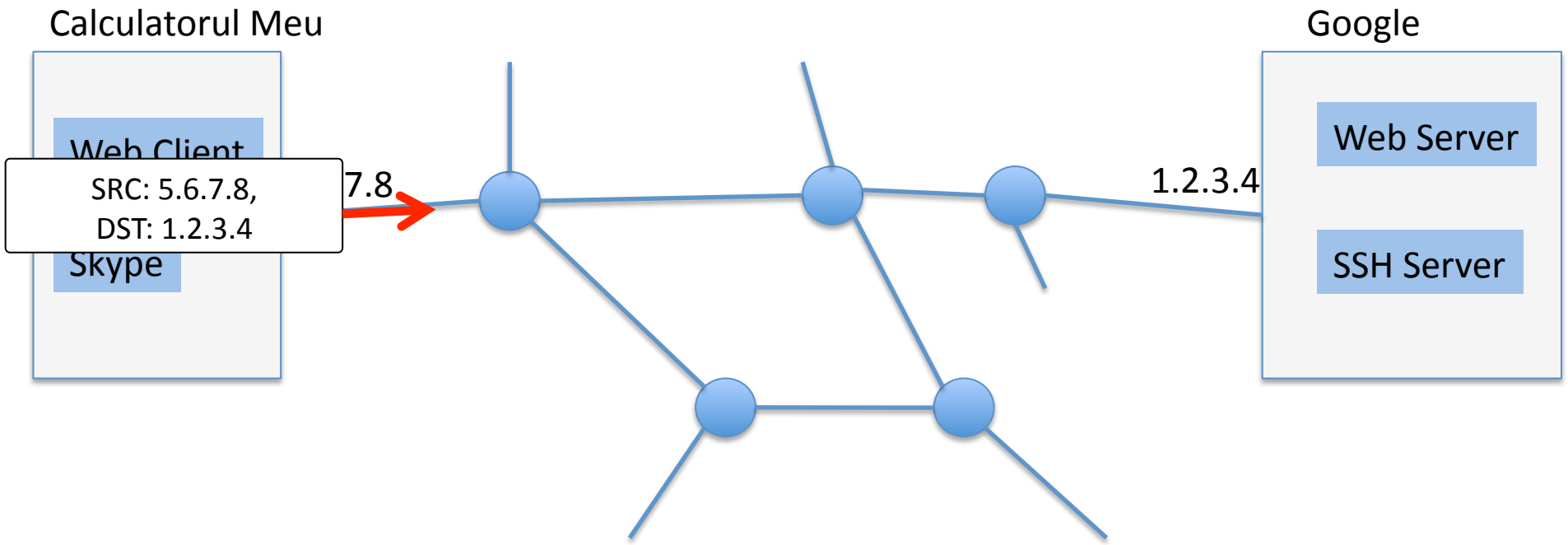
Transmisia de date in Internet



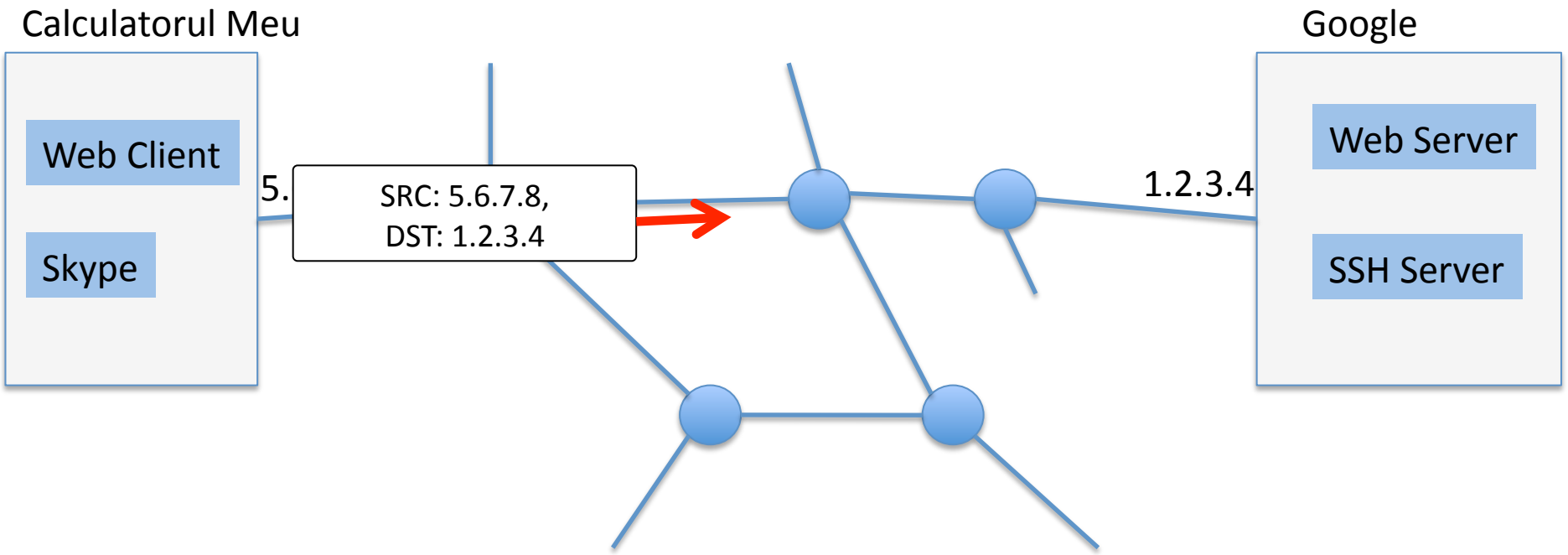
Packet IP


Bit 0
Bit 15
Bit 16
Bit 31
Version
IHL
TOS
ECN
Total Length
Identification
Flags
Fragment Offset
TTL
Protocol
Header Checksum
Source IP
Destination IP
TRNASPORT PROTOCOL

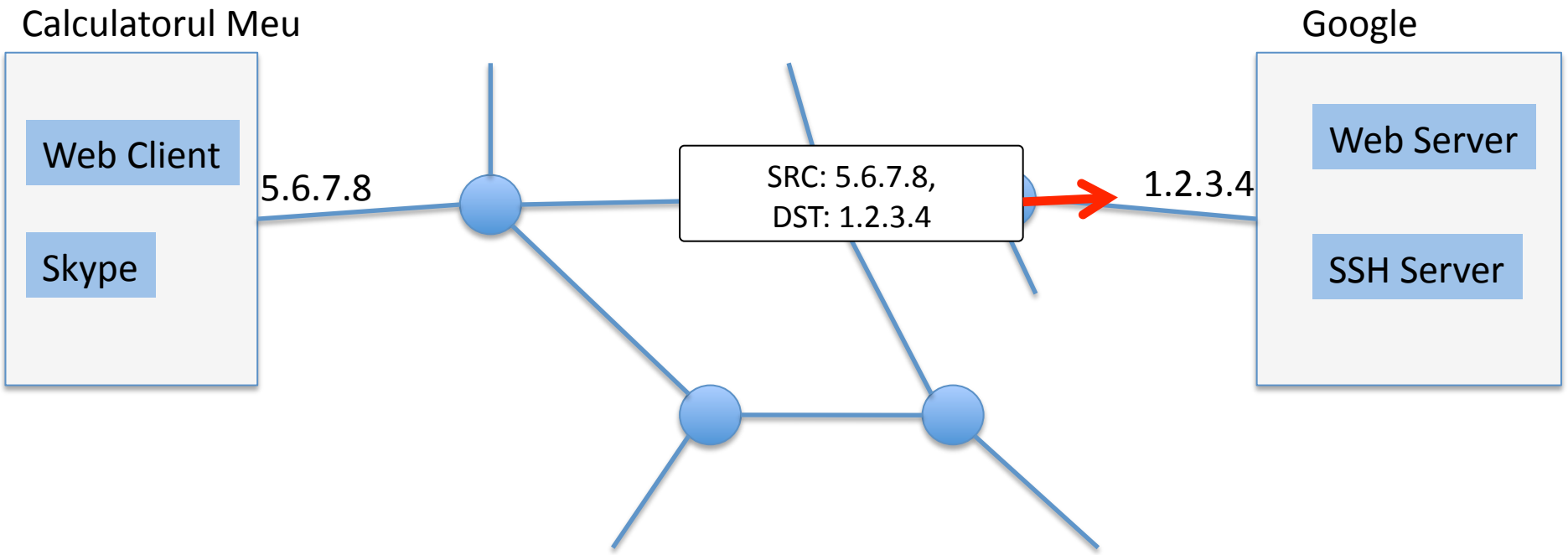
Transmisia de date in Internet



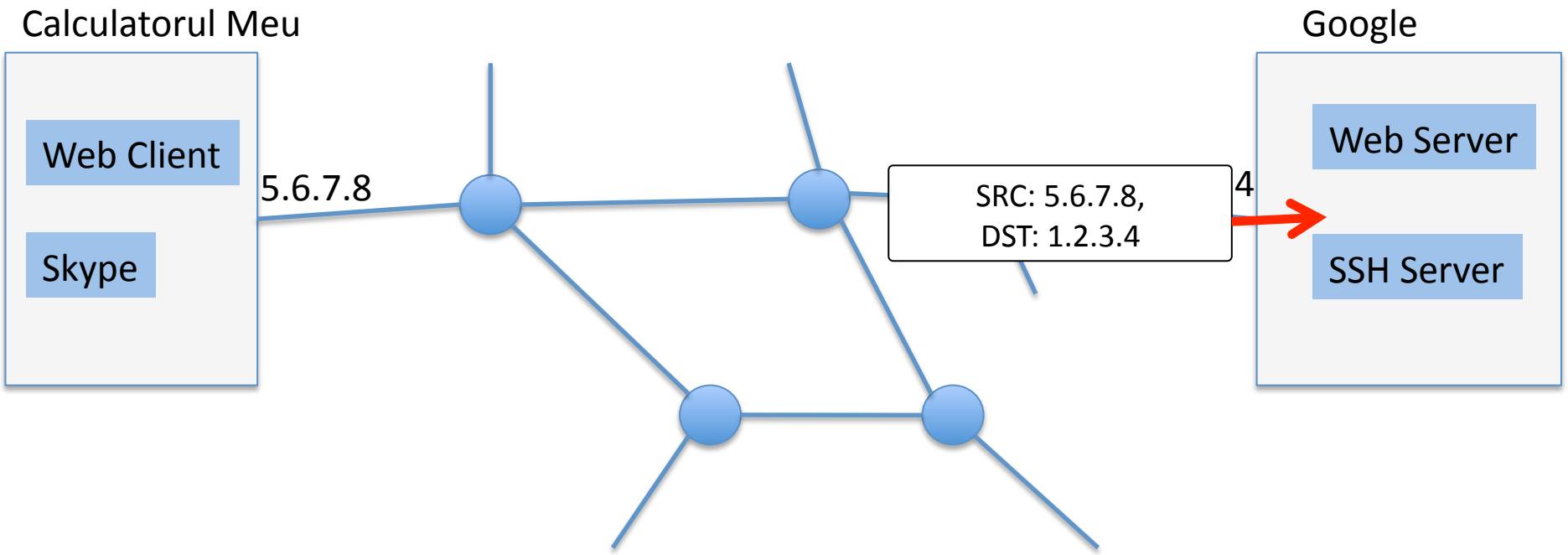
Transmisia de date in Internet



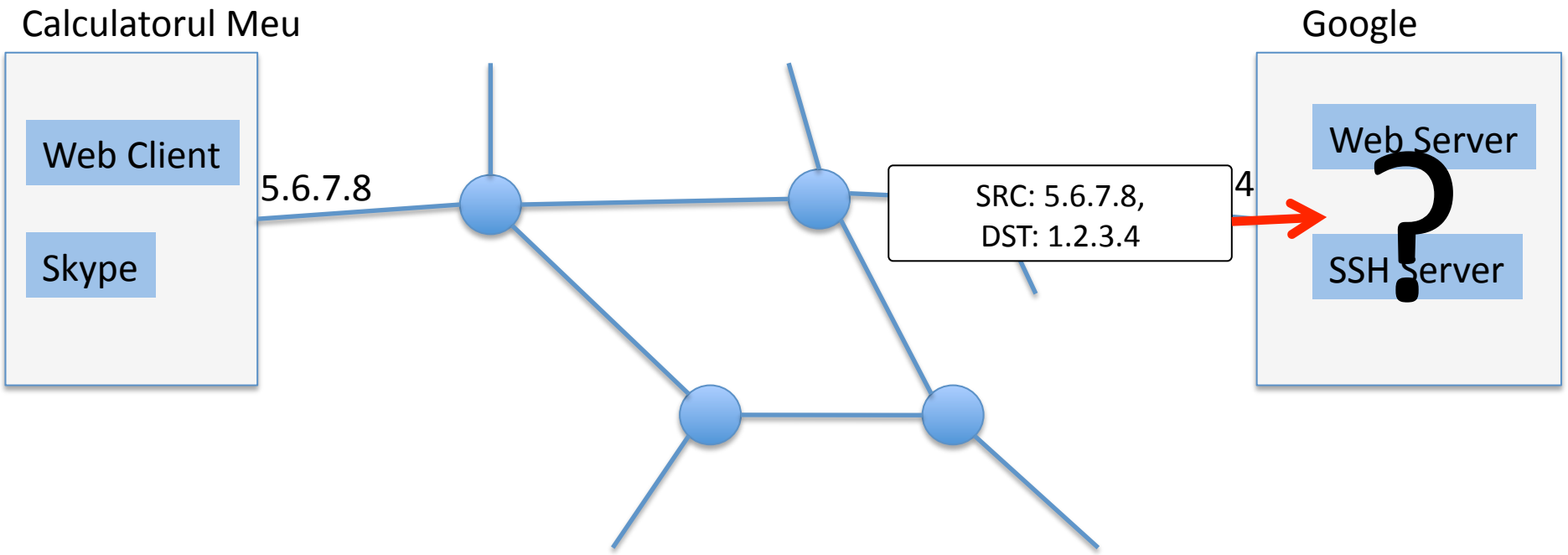
Transmisia de date in Internet



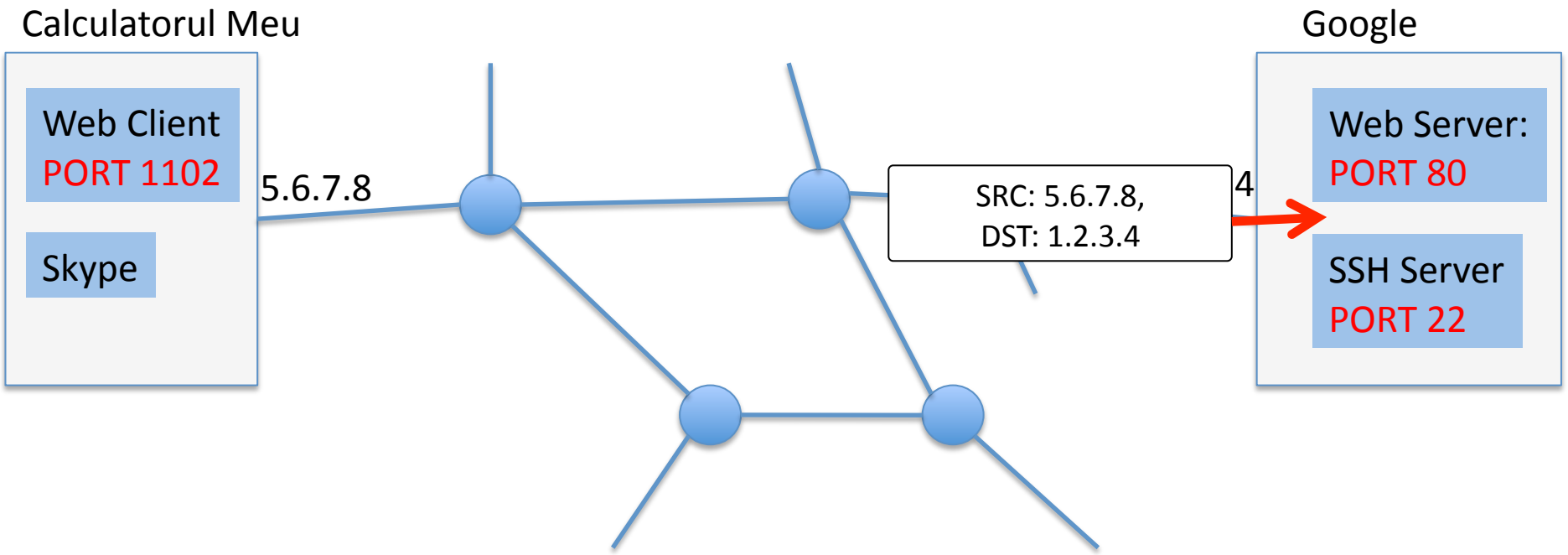
Cum demultiplexam pachetele?

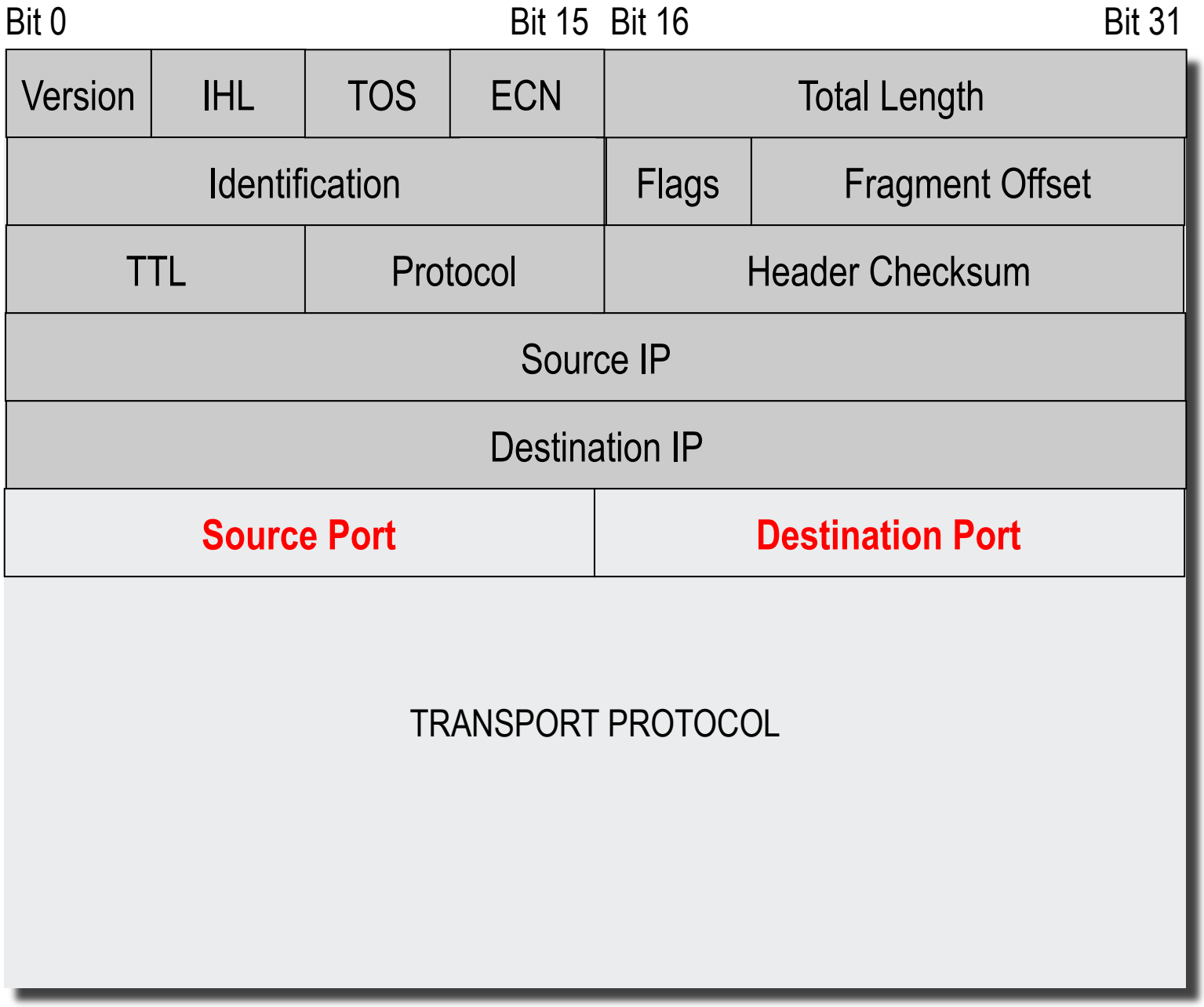


Cum demultiplexam pachetele?

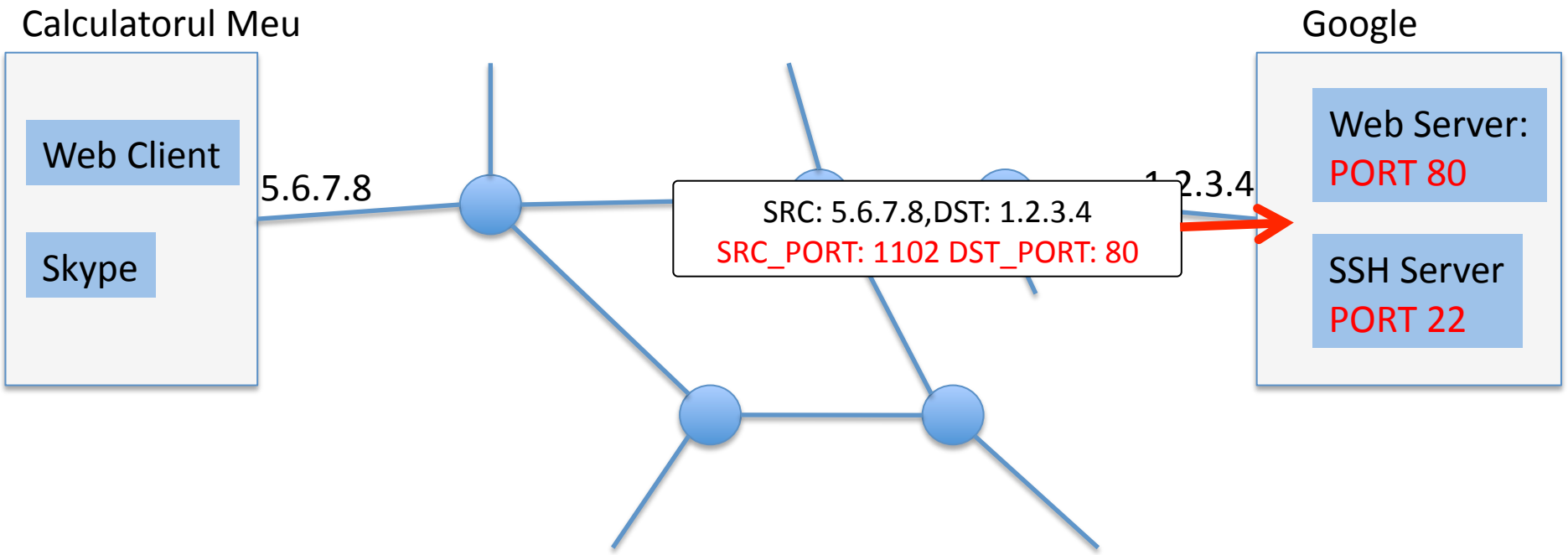


Porturi: adresele nivelului transport




Bit 0
Bit 15
Bit 16
Bit 31
Version
IHL
TOS
ECN
Total Length
Identification
Flags
Fragment Offset
TTL
Protocol
Header Checksum
Source IP
Destination IP
Source Port
Destination Port
TRANSPORT PROTOCOL

Porturi: adresele nivelului transport



Socket API

- Interfata pentru servicii de transport
- Oferit ca biblioteca utilizator sau functii OS
- Foloseste descriptori (ca la fisiere)

- *Socket API* este
  - Originar din Berkeley BSD UNIX
  - Disponibil pe Windows, Solaris, etc.

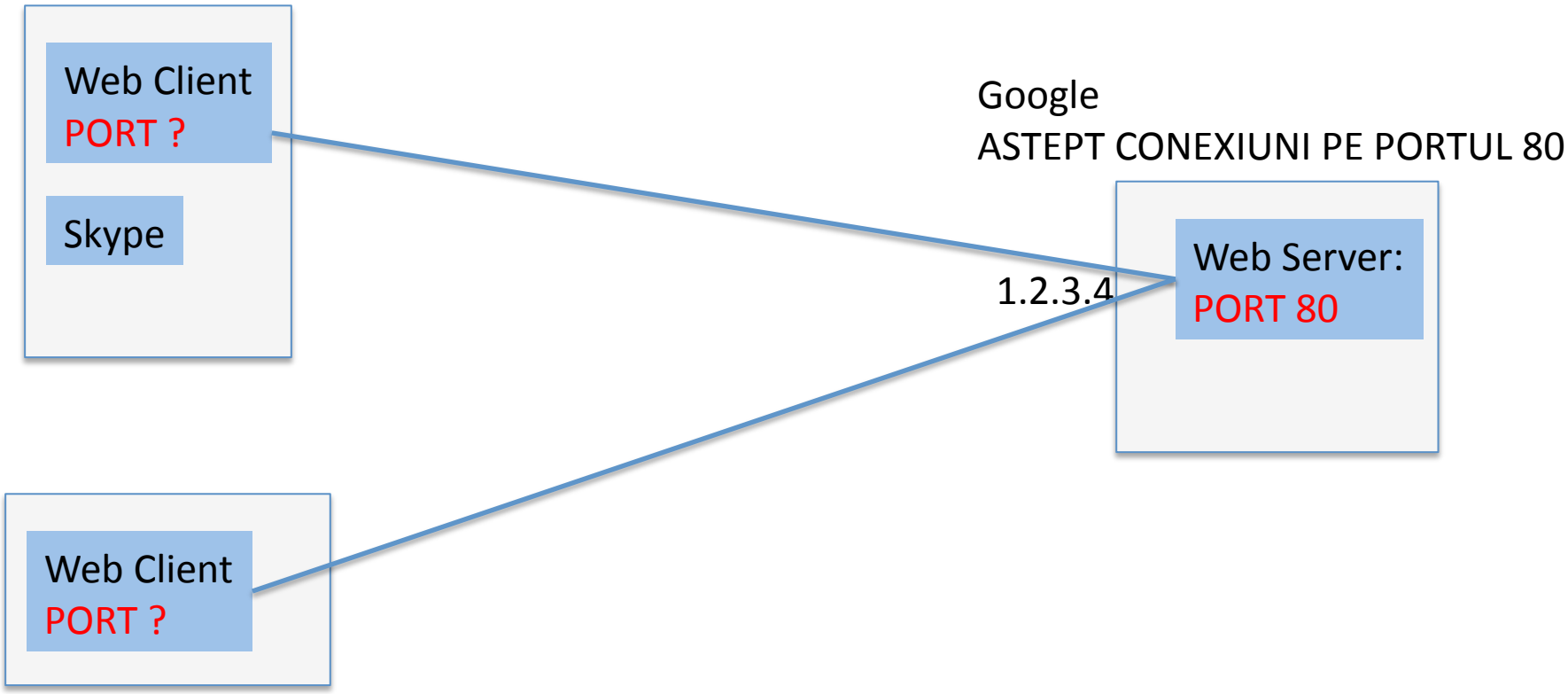
- S*tandard de facto*

Transmission Control Protocol

- **Protocol orientat conexiune**
- In cadrul unei conexiuni **garanteaza transmisie in ordine si sigura a datelor**
- Realizeaza **controlul congestiei** adaptand viteza de transmisie la conditiile retelei
- Interfata oferita:

  transmisie si receptie de sir de octeti

API Conexiuni TCP: Sumar



Transmisie/receptie date cu TCP

Pasi necesari server:

1. Creaza socket:

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

2. Seteaza port pentru socket:

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

3. Declara nr. de clienti:

listen(ls, 5);

4. Asteapta conexiune:

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

Pasi necesari client:

1. Creaza socket:

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

2. Conecteaza-te la server:

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

Inchiderea conexiunii TCP

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

TCP asigura transmisie **sigura, in ordine a unui sir-de-octeti**:

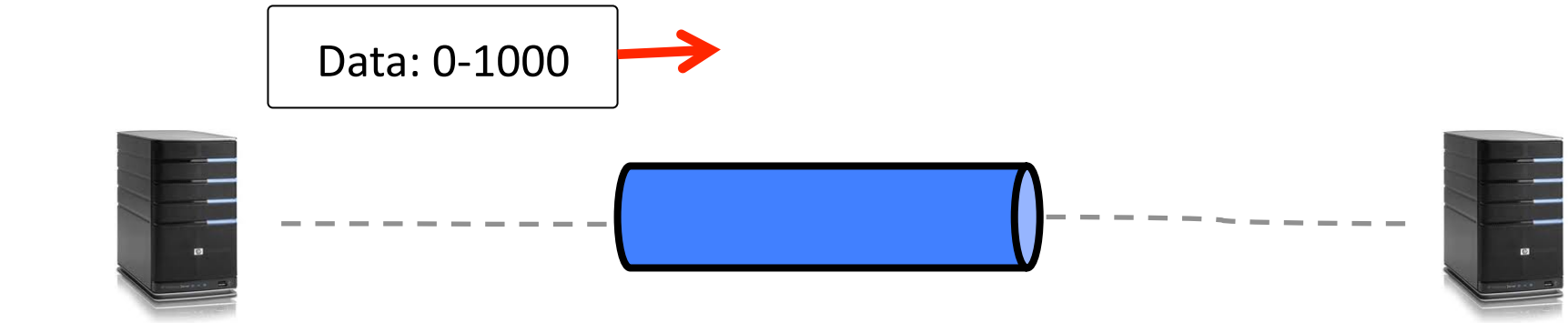
- Applicatiile trimit un numar arbitrar de octeti
  - Sa zicem 100.000B
- TCP imparte octetii in segmente
  - Pentru ca reteaua functioneaza cu pachete de dimensiune fixa
- Le trimite in retea
  - Segmentele pot fi pierdute sau reordonate
- Receptorul TCP **trebuie** sa receptioneze datele in ordine

Transmisia de date TCP

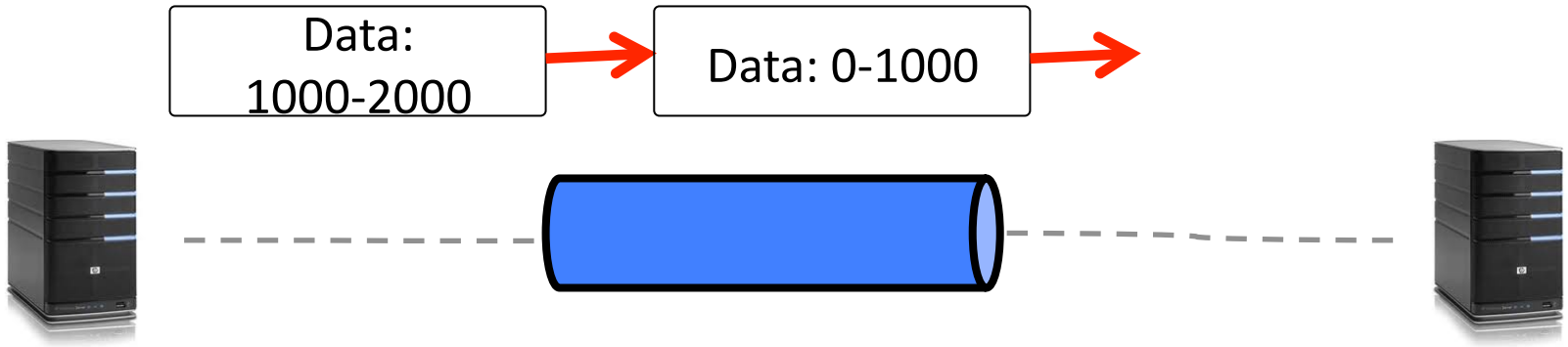
Browser: Send 3KB of data

HTTP Server:
Read Request

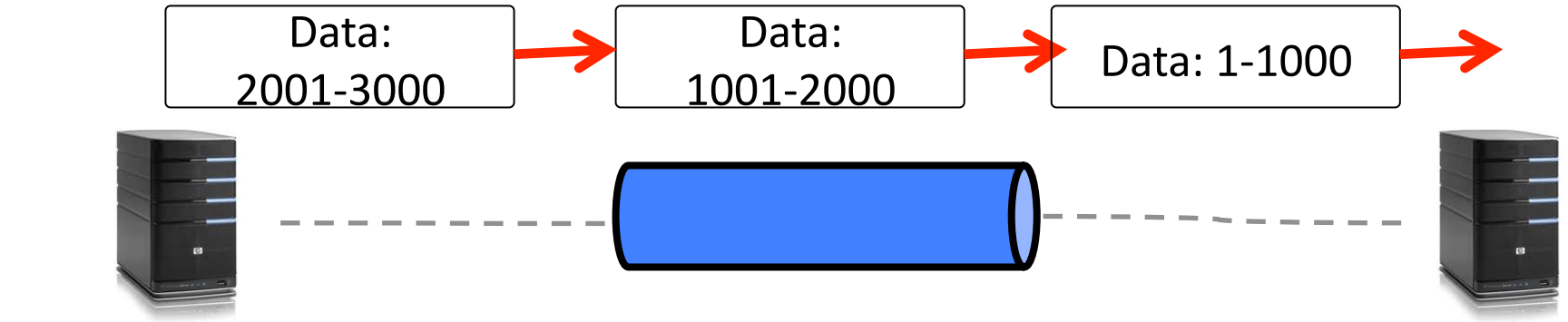
Transmisia de date TCP



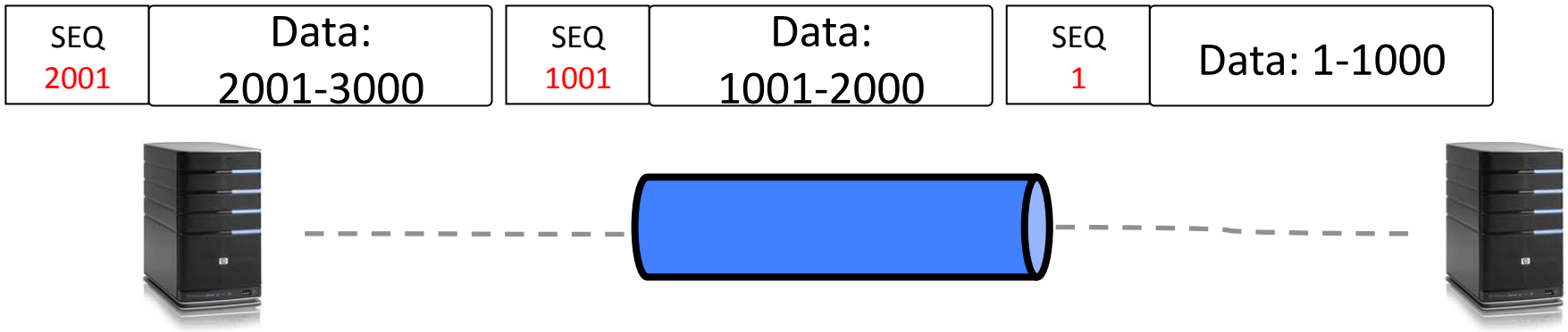
Transmisia de date TCP



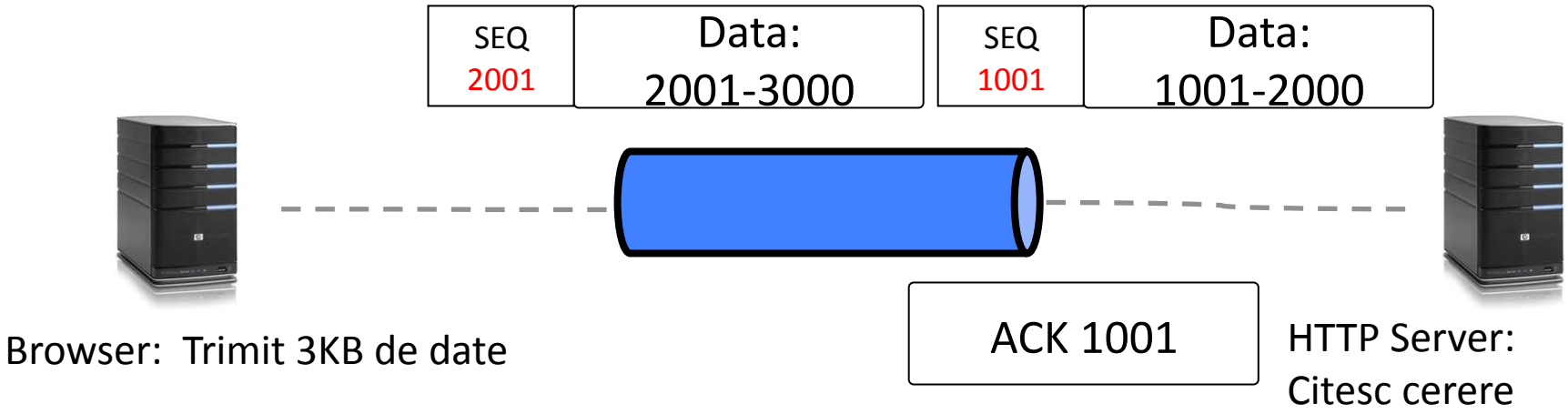
Transmisia de date TCP



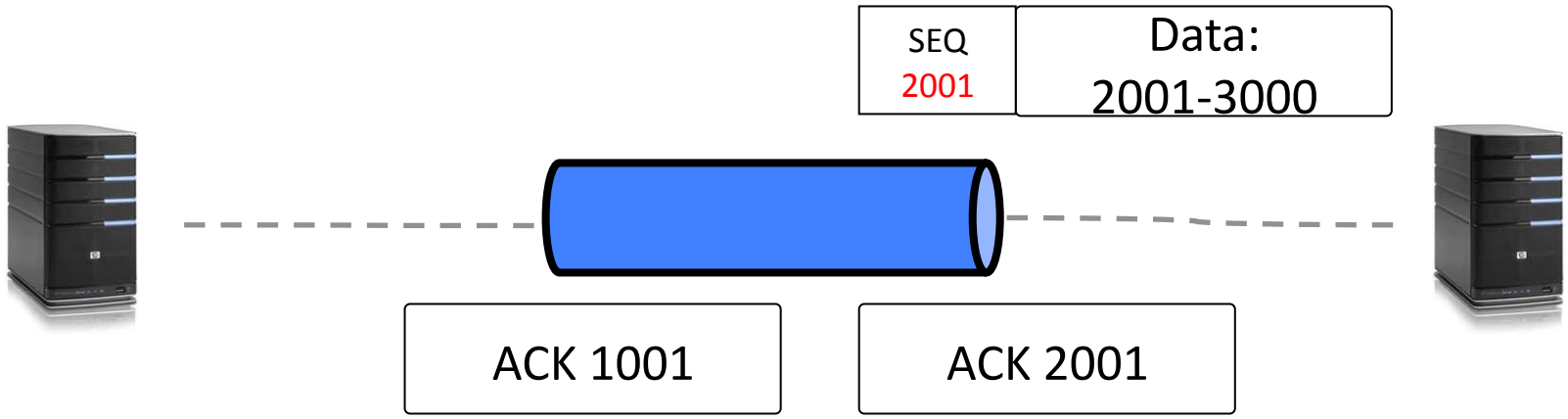
Transmisia de date TCP: Numere de secventa si confirmari



Transmisia de date TCP: Numere de secventa si confirmari



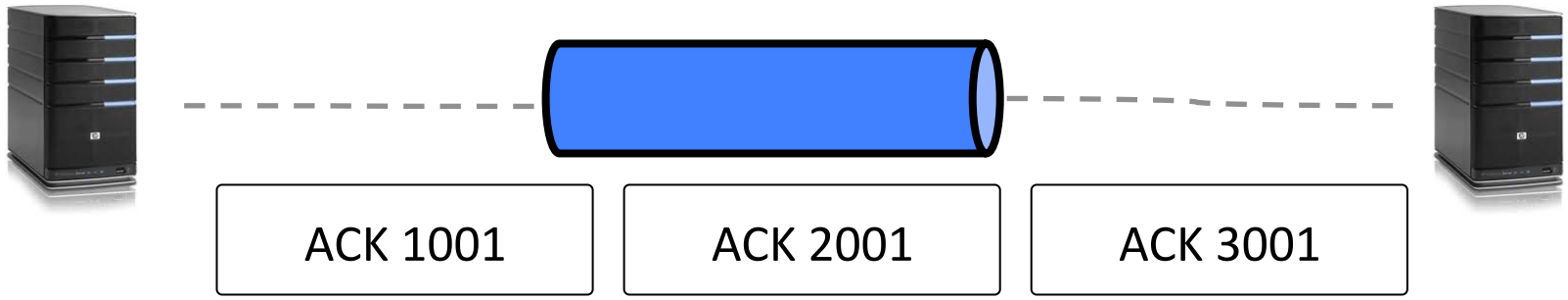
Transmisia de date TCP: Numere de secventa si confirmari



Browser: Trimit 3KB de date

HTTP Server:
Citesc cerere

Transmisia de date TCP: Numere de secventa si confirmari



Browser: Trimit 3KB de date

HTTP Server: Citesc cerere

Receptia de date cu TCP

recv (s, buf, max_len, flags)

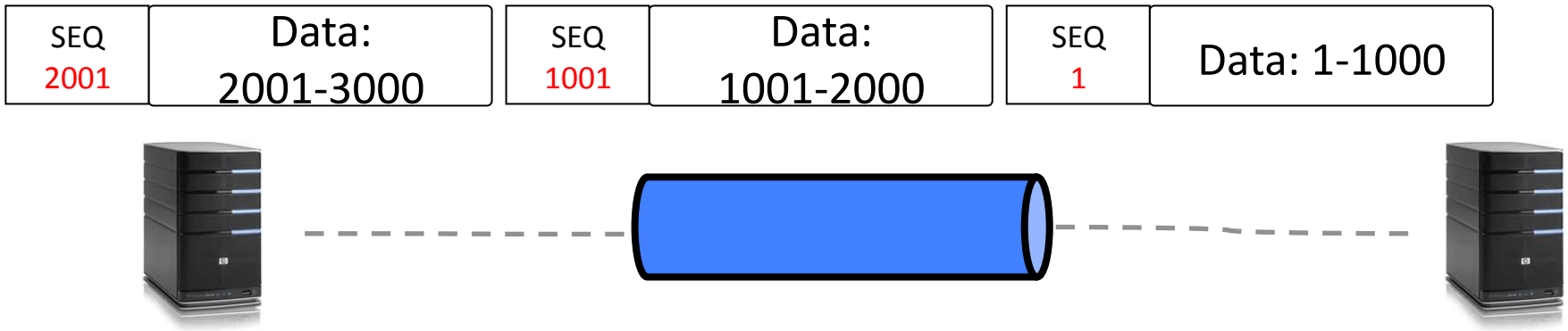
- Intoarce numarul de octeti trimisi
  - **Poate fi mai mic decat max_len!**
- Cod corect:

```
int t = 0;
while (t<len){
   d = recv(s, buf+t, len-t,0);
   if (d<0) break;
   t += d;
}
```

Cum implementam recv in SO?

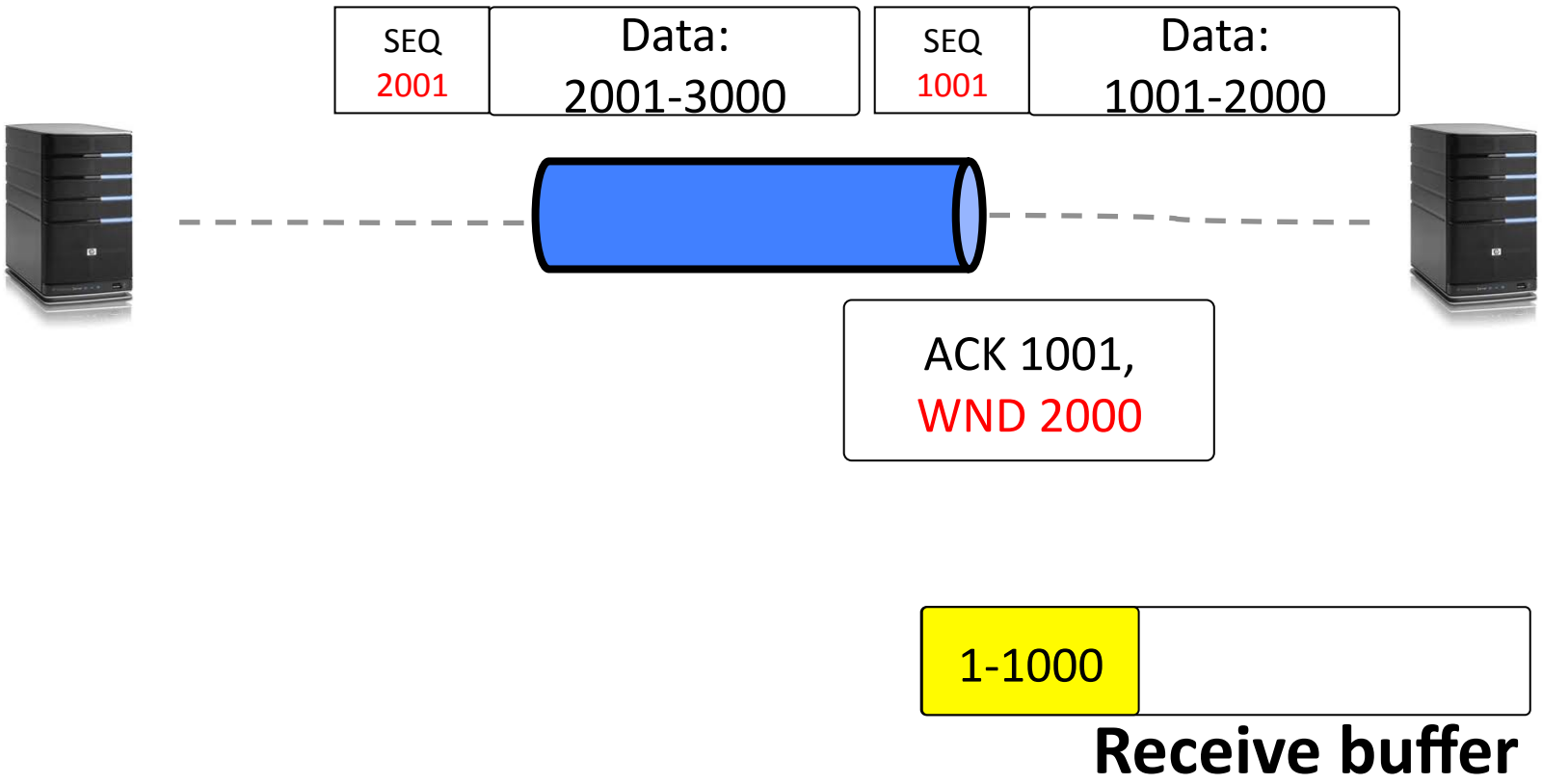
- recv() – intoarce datele in ordine
- Ce se intampla daca se pierd segmente?
  - Stiva trebuie sa pastreze pachetele cu numar de secventa mai mare
  - Este nevoie de un **receive buffer**
  - Ce se intampla daca se umple bufferul?
    - Flow control, sender-ul se opreste din transmisie

Receptia datelor: receive buffer

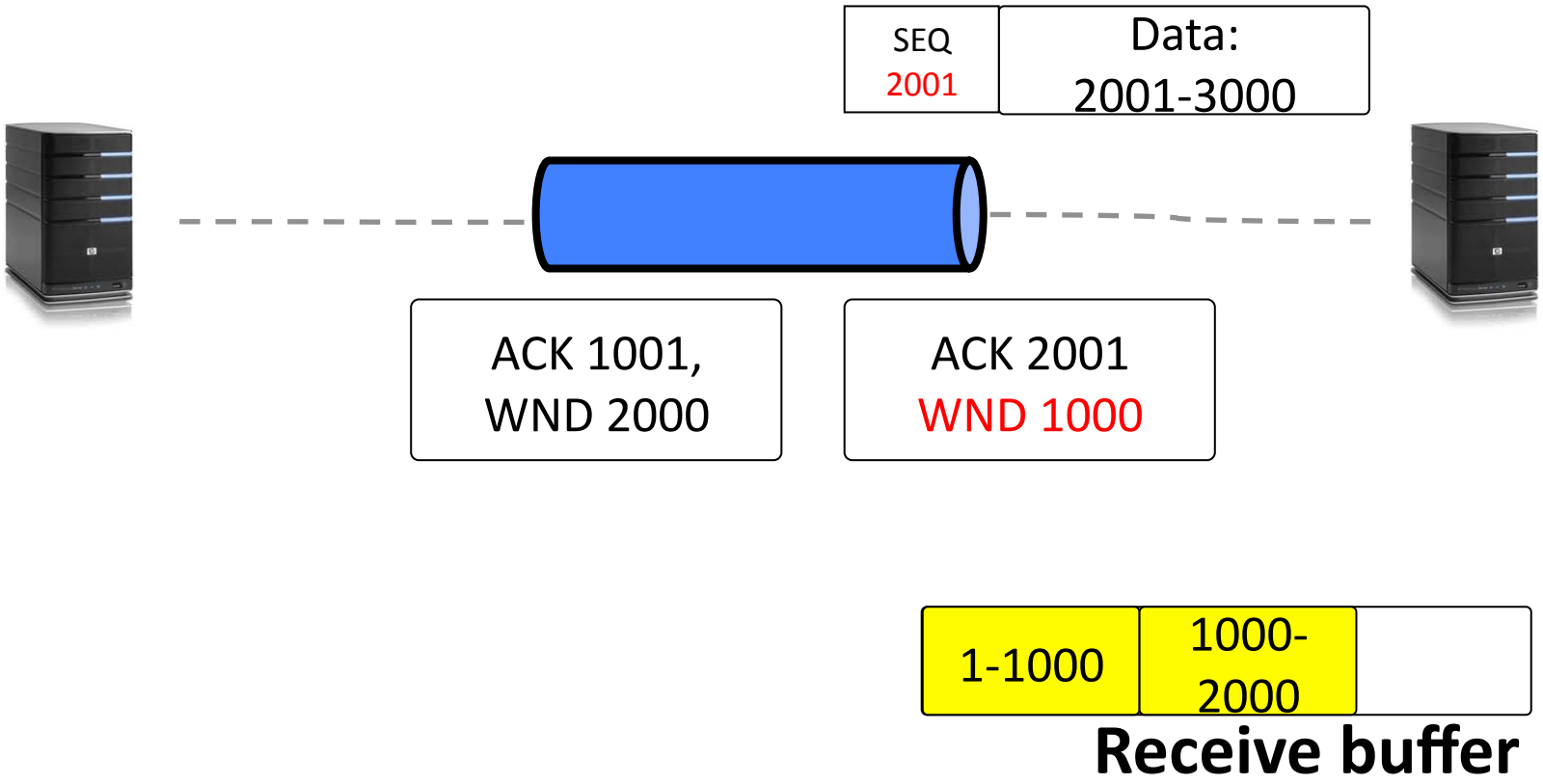


Receive buffer

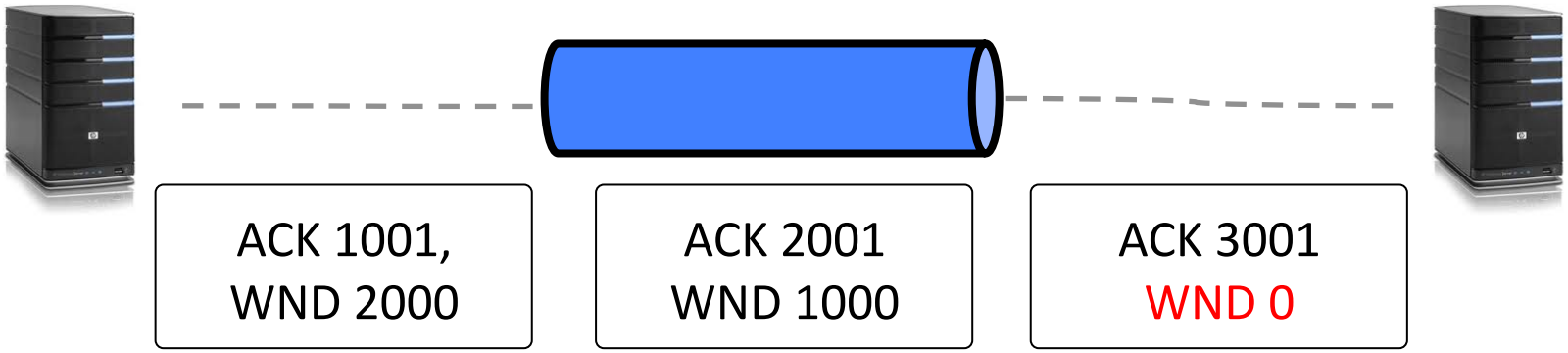
Receptia datelor: receive buffer

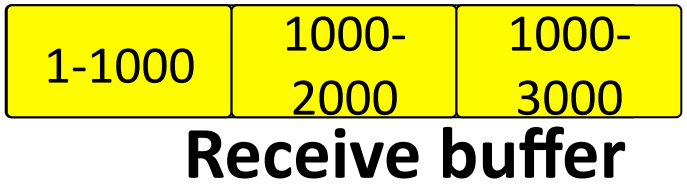


Receptia datelor: receive buffer

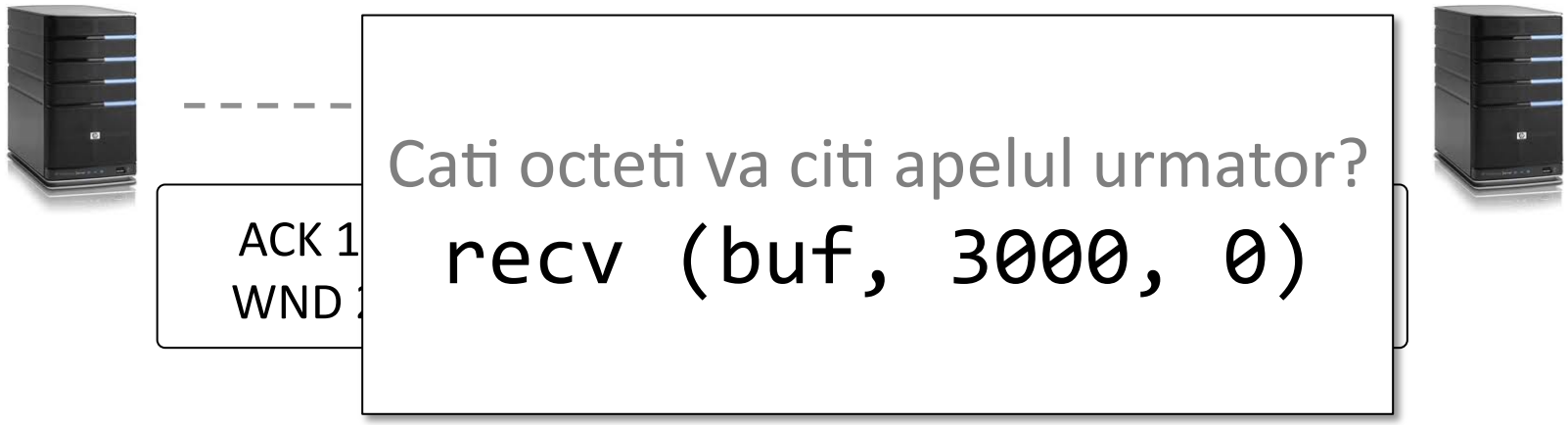


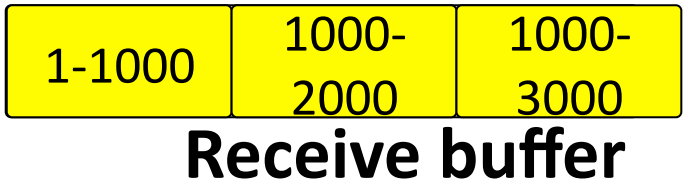
Receptia datelor: receive buffer





Receptia datelor: receive buffer





Transmisia de date cu TCP

send (s, buf, len)

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

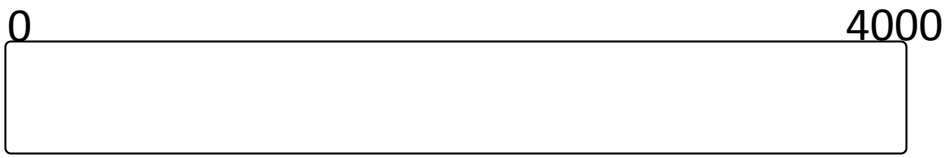
```
int t = 0;
while (t<len){
   d = send(s, buf+t, len-t);
   if (d<0) break;
   t += d;
}
```

Cum implementam send in SO?

- send() – garanteaza ca datele acceptate vor fi transmise, cat timp conexiunea merge
- Ce se intampla daca se pierd segmente?
  - Sender-ul trebuie sa pastreze mesajele pana sunt confirmate
  - Este nevoie de un **send buffer**
  - Ce se intampla daca se umple bufferul?
    - Send se blocheaza

Transmisia datelor: send buffer


```
send (buf, 5000, 0);
```



Send buffer

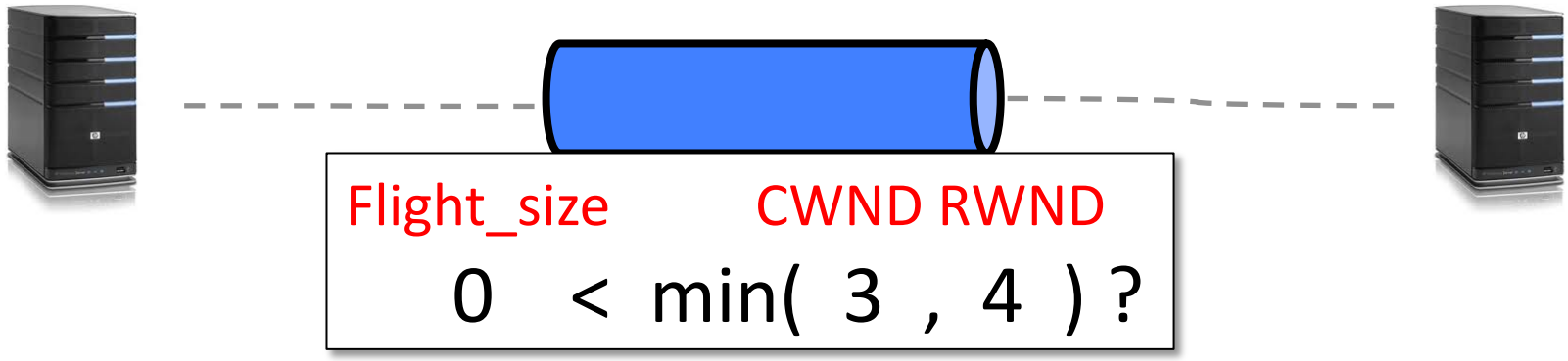
Transmisia datelor: send buffer

1-1000	1000-2000	2000-3000	3000-4000

Send buffer

CWND=3, RWND=4

Transmisia datelor: send buffer

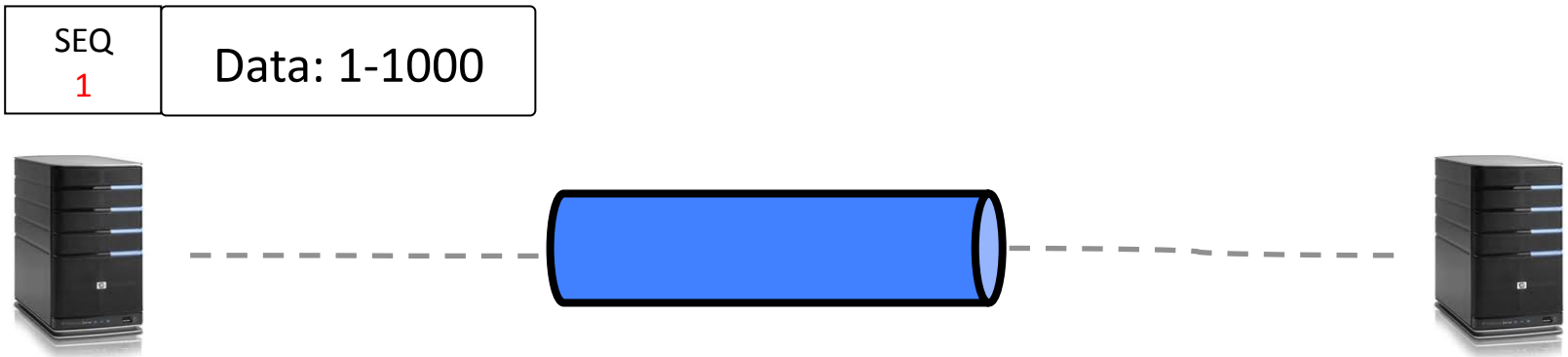


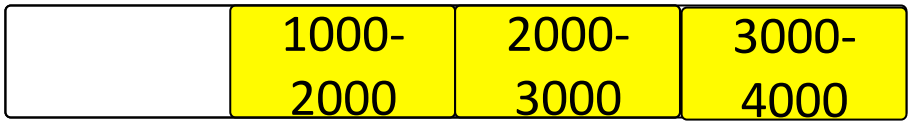
1-1000	1000-2000	2000-3000	3000-4000

Send buffer

CWND=3, RWND=4

Transmisia datelor: send buffer

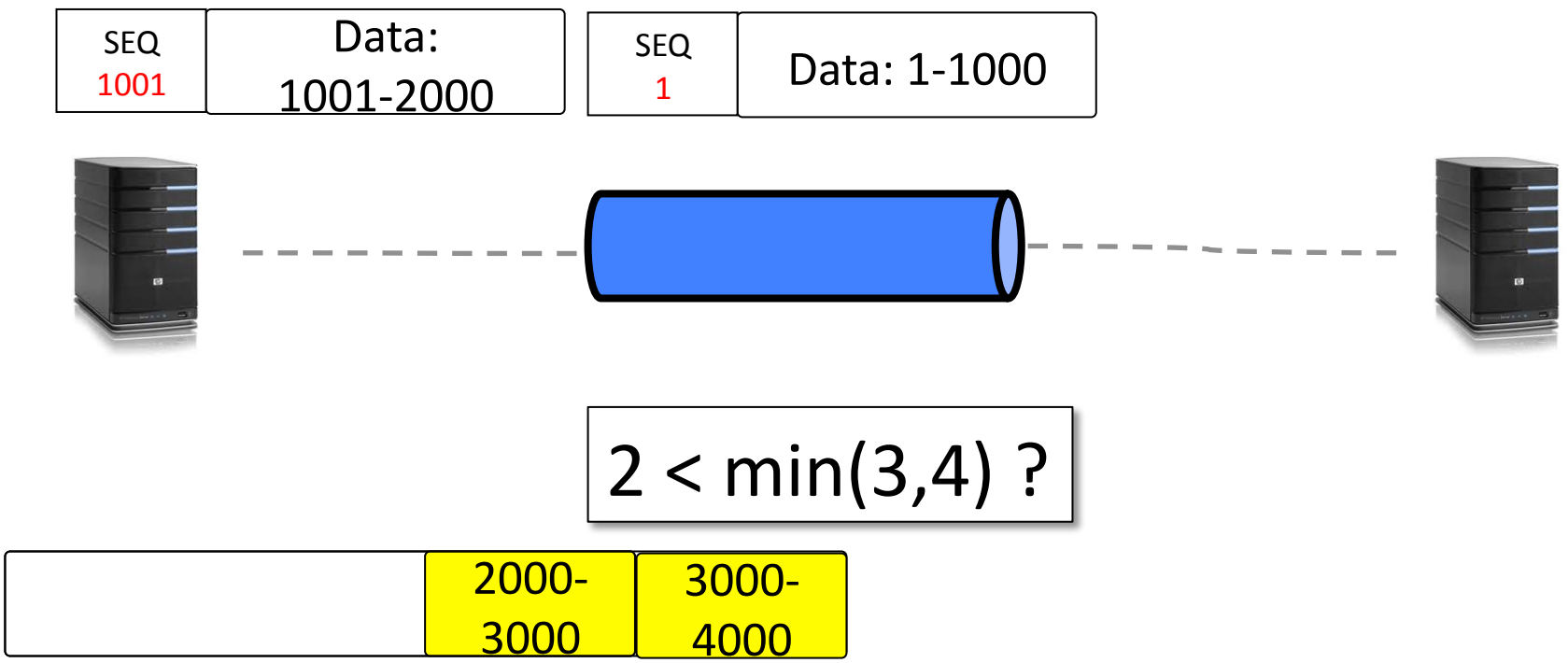




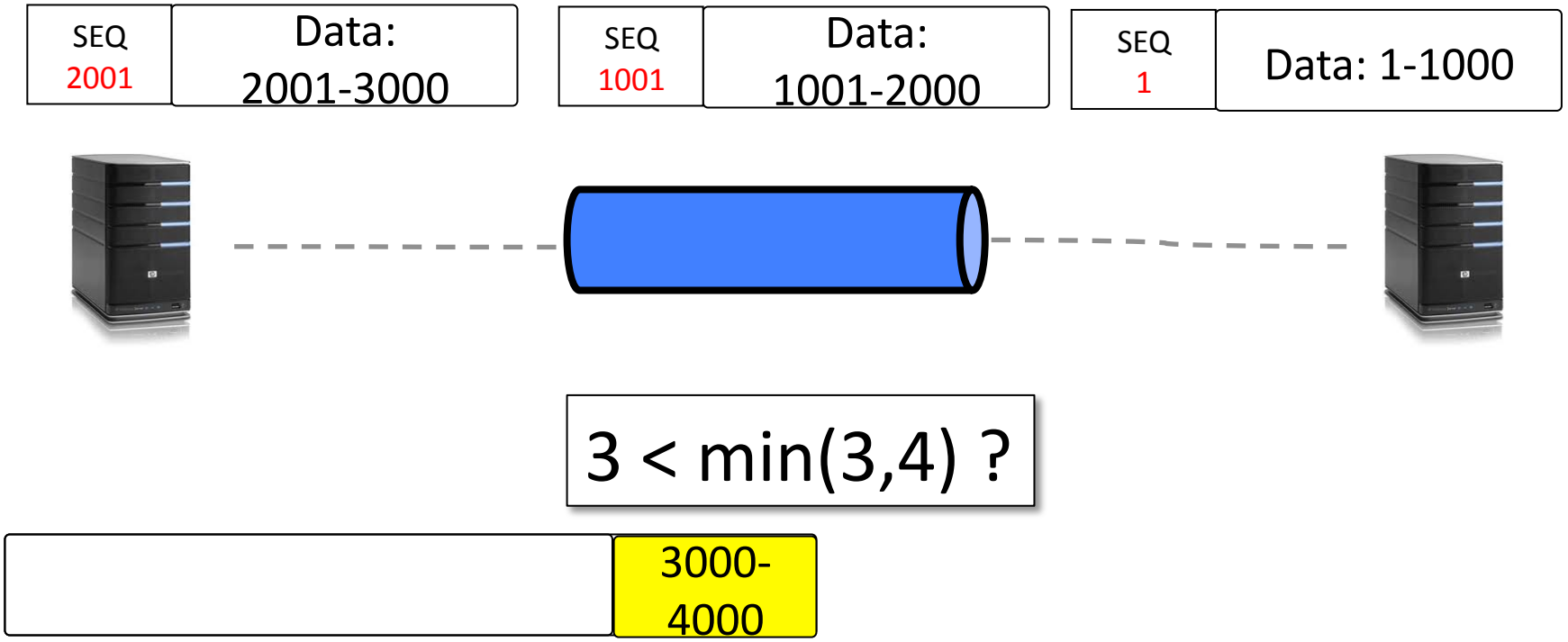
Send buffer

CWND=3, RWND=4

Transmisia datelor: send buffer

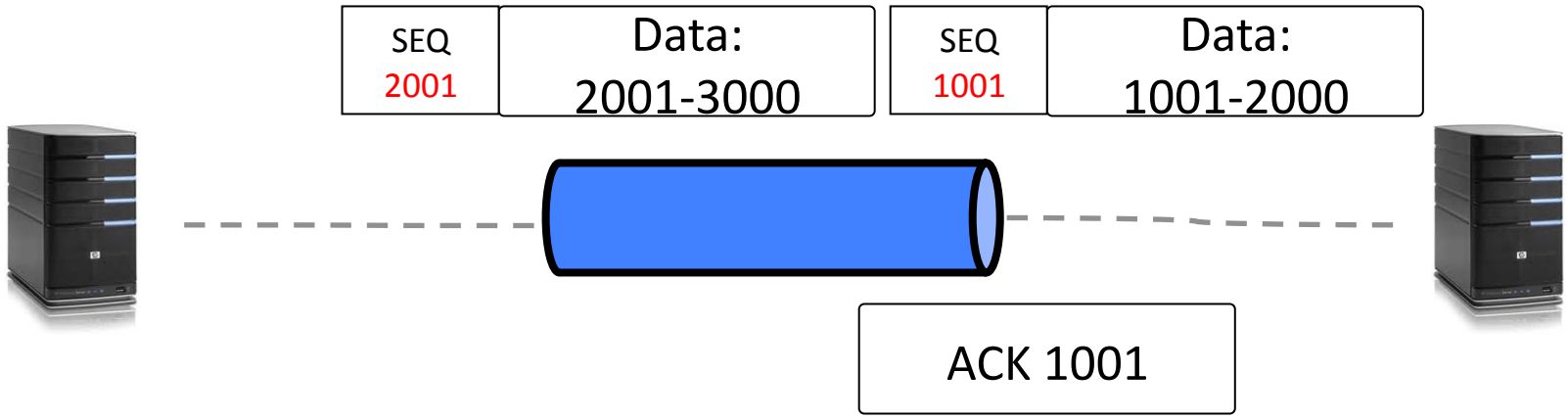


Transmisia datelor: send buffer



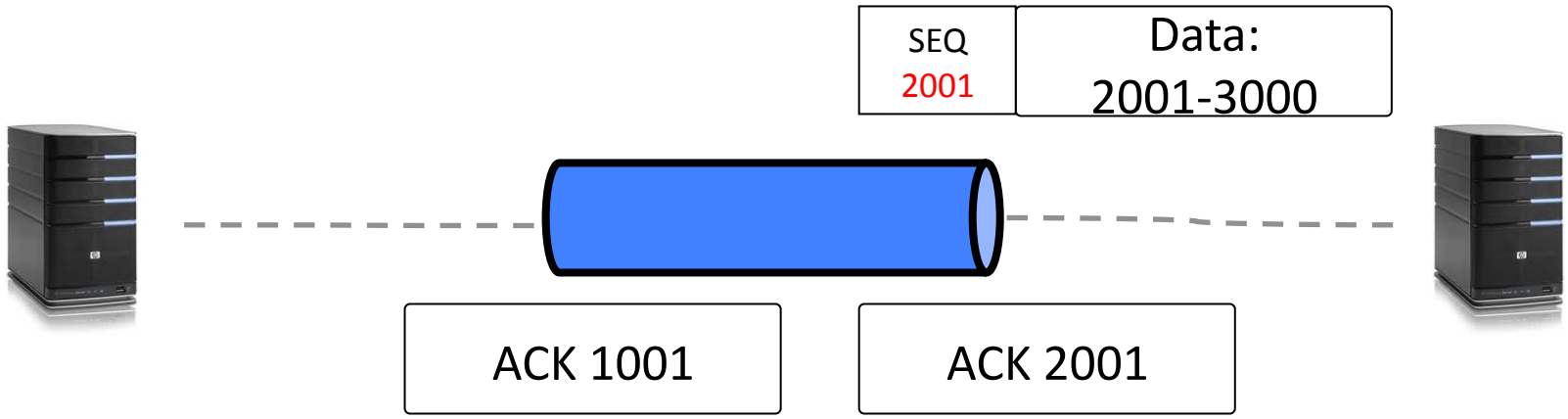
Send buffer

Transmisia de date TCP: Numere de secventa si confirmari



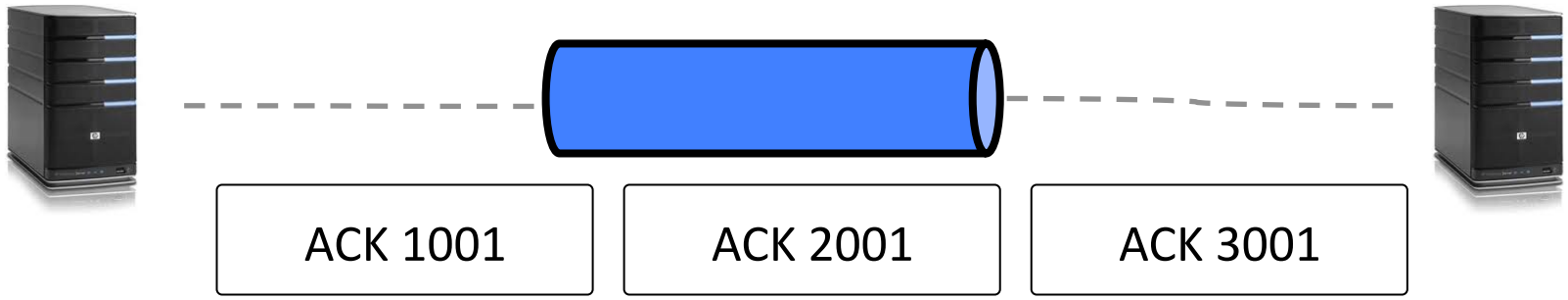
Send buffer

Transmisia de date TCP: Numere de secventa si confirmari



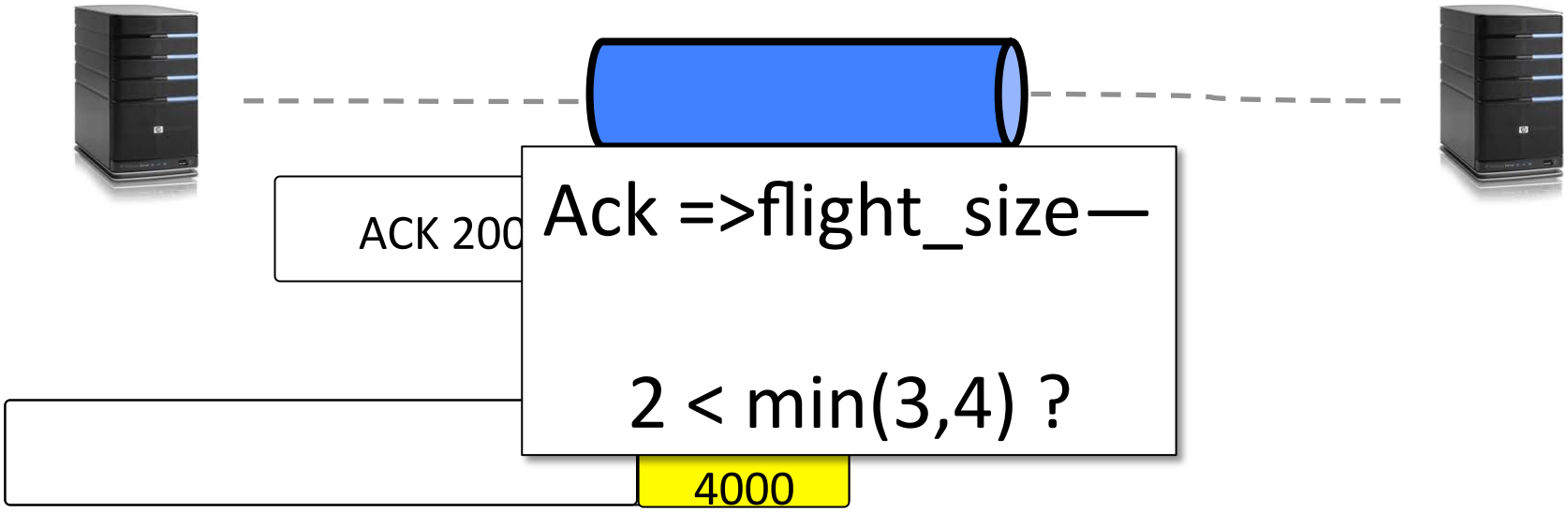
Send buffer

Transmisia de date TCP: Numere de secventa si confirmari

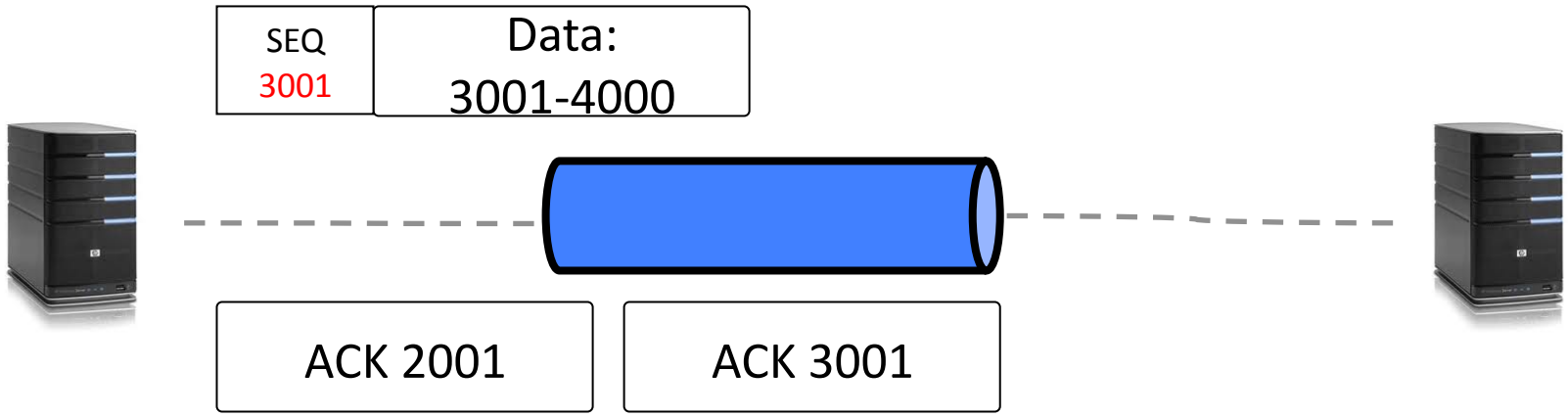


Send buffer

Transmisia de date TCP:
Numere de secventa si confirmari



Transmisia de date TCP: Numere de secventa si confirmari



Send buffer

Analizati codul urmator

Sender

```
char buf[1000]; …
for  (i=0;i<1000;i++){
   send(s, b+i, 1);
}
```

Receiver

```
char  buf[1000];…
recv(s, buf, 1000);
```

Cati octeti va primi recv?

Imbunatatirea performantei: batching

- Dimensiunea datelor pentru send/receive
- Dimensiunea send si receive buffers
  - Recomandat la minim 2 * Bandwidth * Delay
  - Controlabil cu sysctl tcp.rmem / tcp.wmem

Batching (2)

- Batching in nucleul SO
  - Stiva lucreaza cu segmente mari, de 64KB
  - **TCP Segmentation Offload**: Placa de retea fragmenteaza segmentele inainte sa le puna pe fir
  - **Large Receive Offload:** operatia opusa, la receiver
- Exista si variante software: (gso) generic segmentation offload
- Controlabile cu ajutorul utilitarului *ethtool*
- Fara TSO/LRO Linux nu atinge 10Gbps cu TCP!

Consideratii de performanta

- Evitarea copierilor inutile
  - Sendfile

Thread-uri vs. event I/O

- Discutie la tabla!