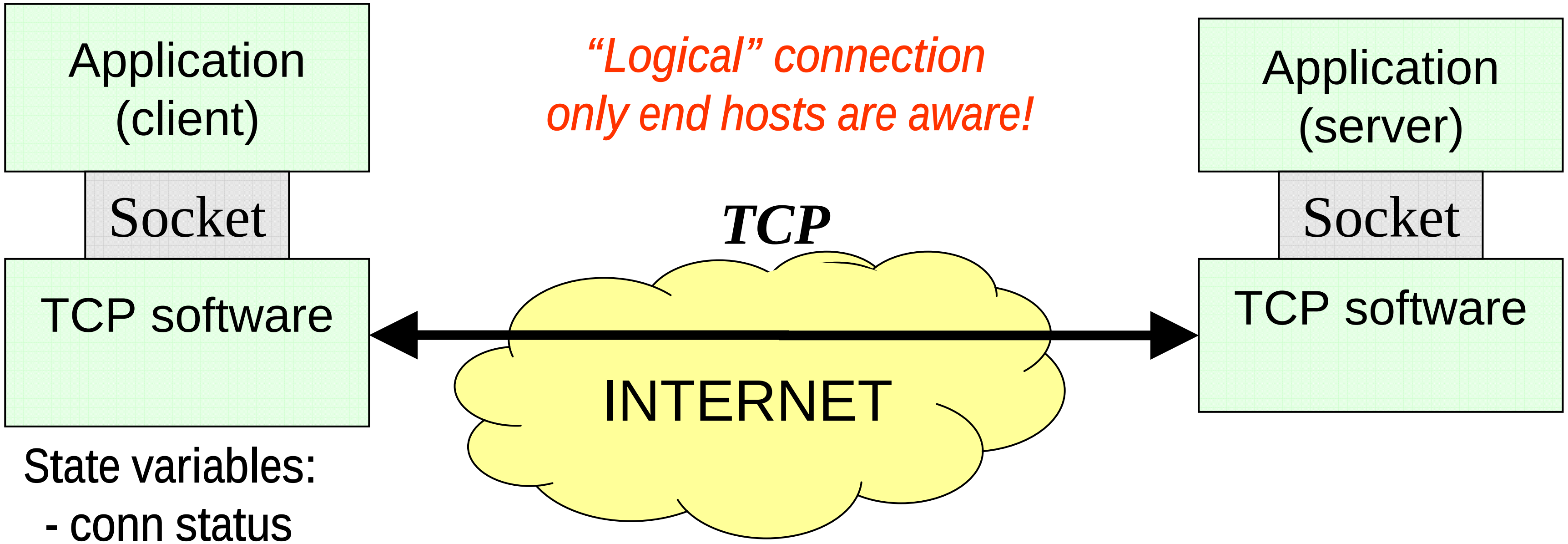
Understanding TCP connection management

G.Bianchi, G.Neglia, V.Mancuso

Università degli Studi di Palermo

TCP connection



State variables:

- conn status
- MSS
- windows
- ...

buffer space

normally 4 to 16 Kbytes

64+ Kbytes possible

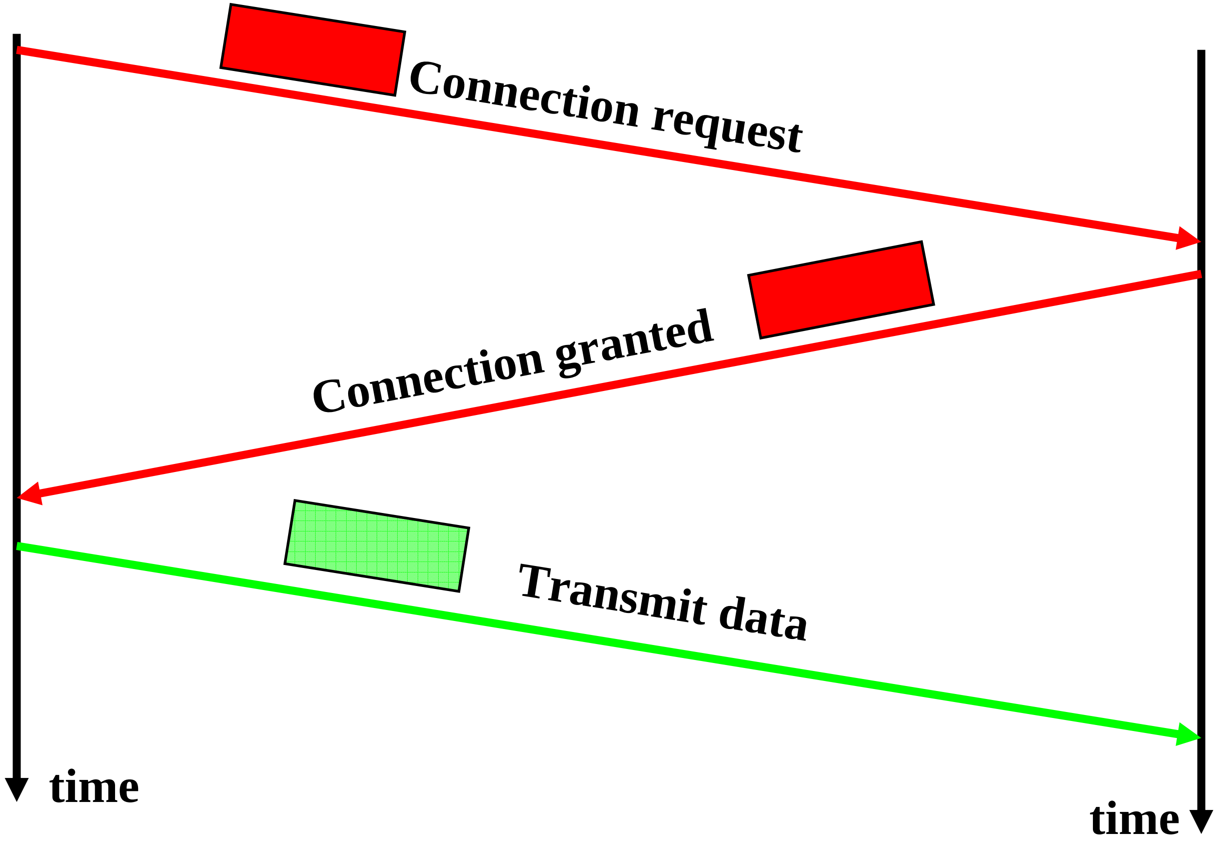
Connection described by client&server <u>status</u>

Connection SET-UP duty:

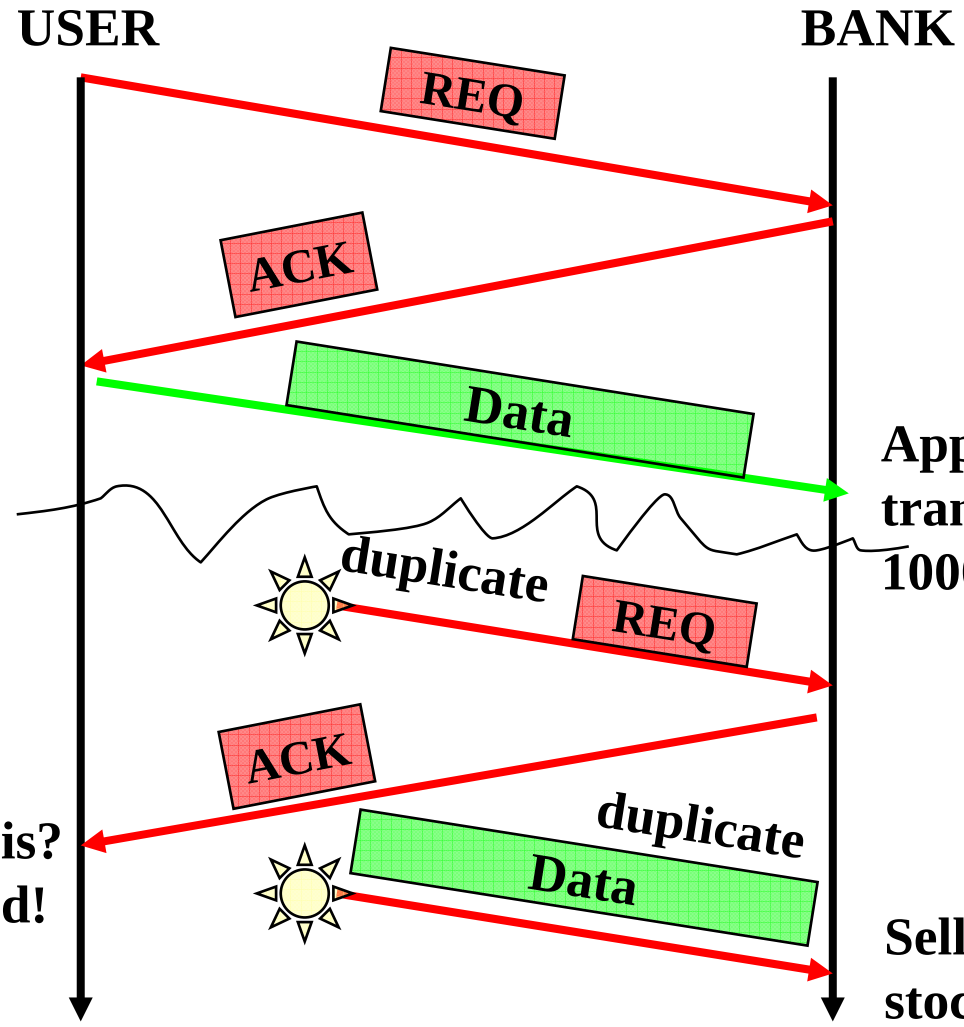
1) initializes state variables

2) reserves buffer space

Connection establishment: simplest approach (non TCP)



Delayed duplicate problem



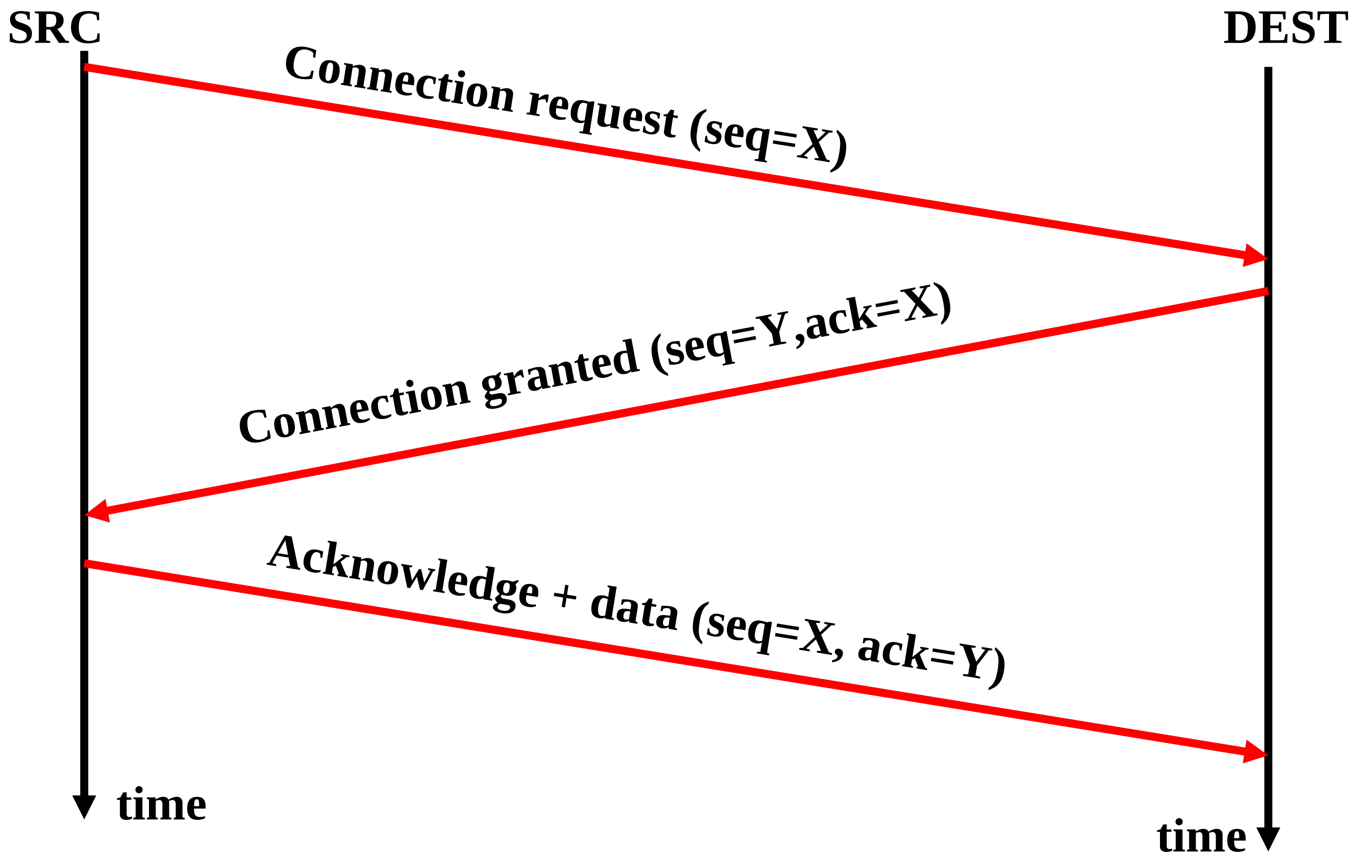
Application: transactional (sell 100000$ stocks)

What is this?
Oh my God!
Too late!!!

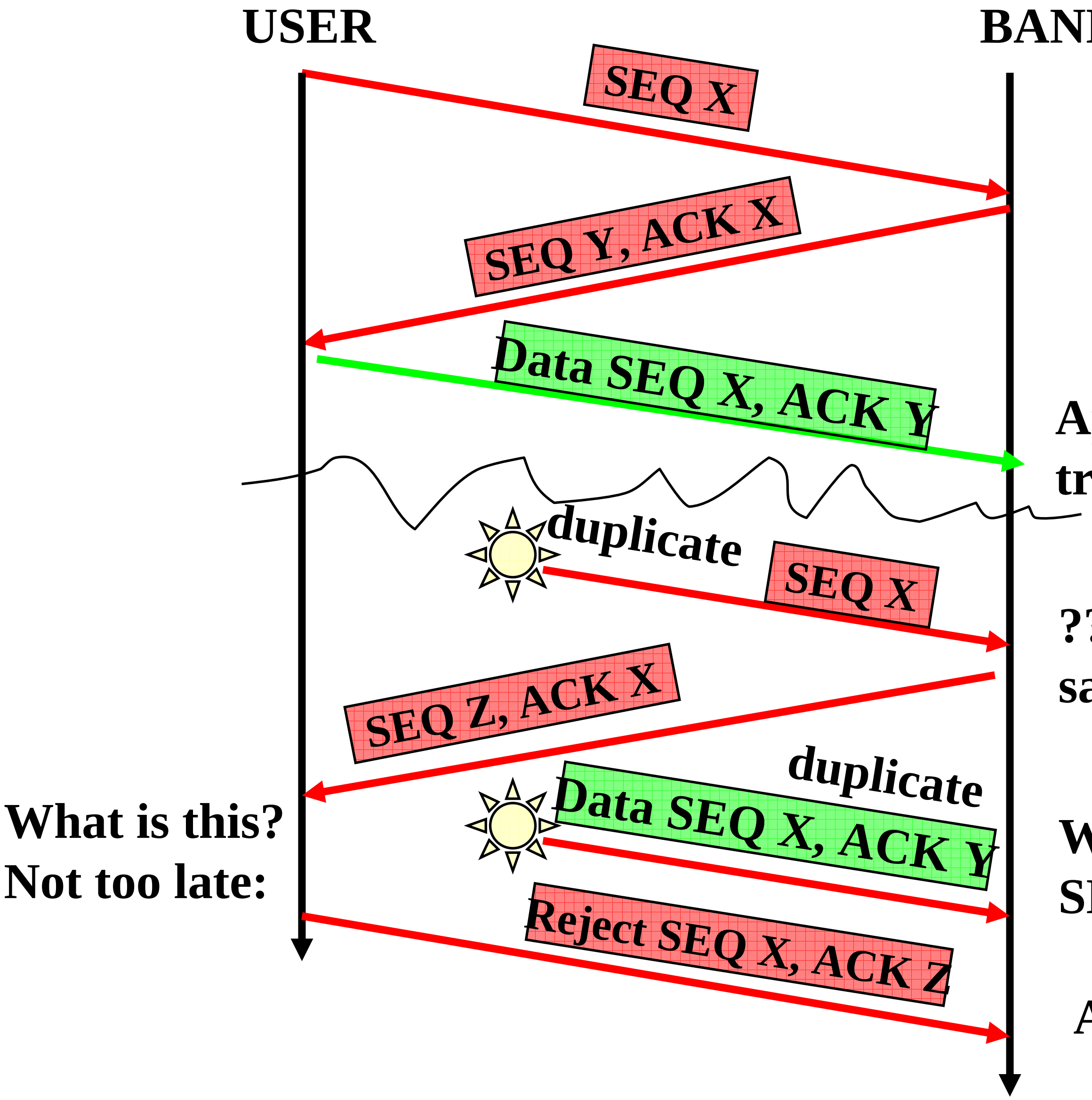
Selling other 100000$ stocks!!!!!

Solution: three way handshake

Tomlinson 1975


SRC
DEST
Connection request (seq=X)
Connection granted (seq=Y,ack=X)
Acknowledge + data (seq=X, ack=Y)
time
time

Delayed duplicate detection



What is this?
Not too late:

Application:
transactional (selling stocks)

??? What a case: request with same indicator X? anyway...

What is this??? Should be SEQ X, ACK Z!!!! STOP...

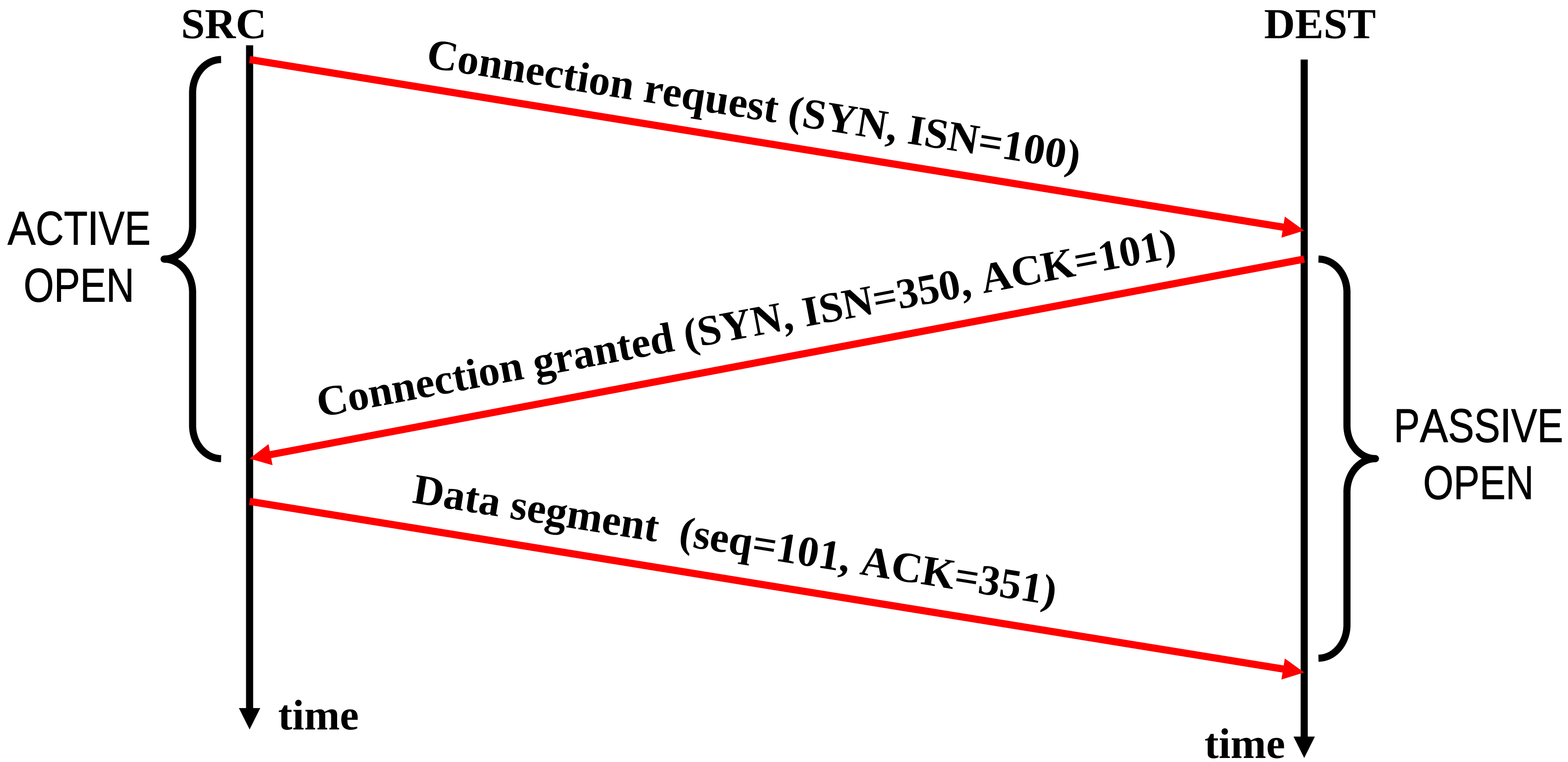
Ah ah! Got the problem!

<table>
<tr><td colspan="8">Source port</td><td>Destination port</td></tr>
<tr><td colspan="9">32 bit Sequence number</td></tr>
<tr><td colspan="9">32 bit acknowledgement number</td></tr>
<tr><td>Header length</td><td>6 bit Reserved</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Window size</td></tr>
<tr><td colspan="8">checksum</td><td>Urgent pointer</td></tr>
</table>

- ➔ **SYN (synchronize sequence numbers): used to open connection**
  - ⇨ SYN present: this host is setting up a connection
  - ⇨ SEQ with SYN: means initial sequence number (ISN)
  - ⇨ data bytes numbered from ISN+1.
- ➔ **FIN: no more data to send**
  - ⇨ used to close connection

...more later about connection closing...

Three way handshake in TCP


SRC
DEST
Connection request (SYN, ISN=100)
ACTIVE OPEN
Connection granted (SYN, ISN=350, ACK=101)
PASSIVE OPEN
Data segment (seq=101, ACK=351)
time
time

Full *duplex connection: opened in both ways*

SRC: performs ACTIVE OPEN

DEST: Performs PASSIVE OPEN

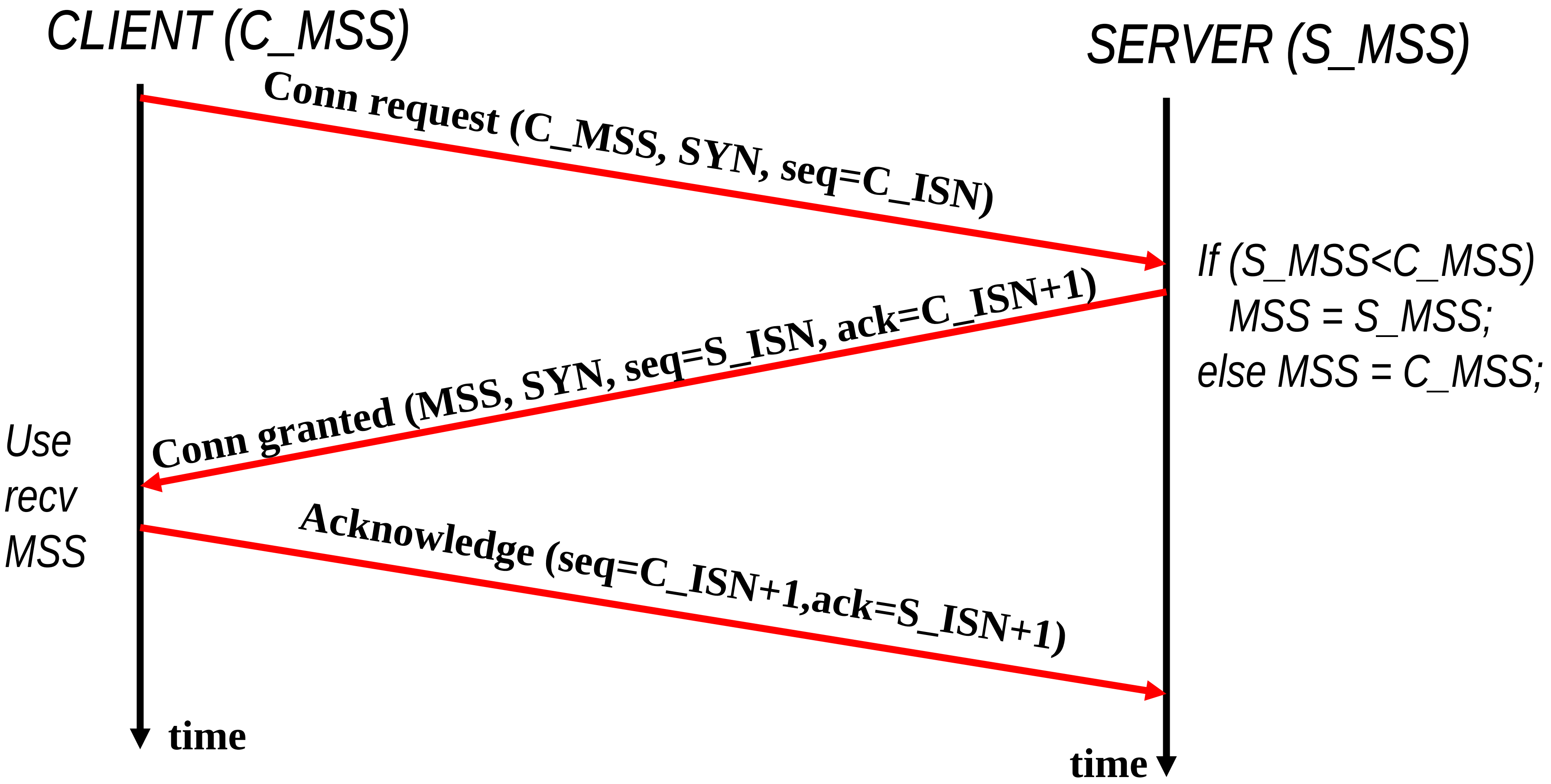
Initial Sequence Number

- ➔ **Should change in time**
  - ⇨ RFC 793 (but not all implementations are conforming) suggests to generate ISN as a sample of a 32 bit counter incrementing at 4us rate
- ➔ **transmitted whenever SYN (Synchronize sequence numbers) flag active**
  - ⇨ note that both src and dest transmit THEIR initial sequence number (remember: full duplex)
- ➔ **Data Bytes numbered from ISN+1**
  - ⇨ necessary to allow SYN segment ack

Maximum Segment Size - MSS

- ➔ **Announced at setup by both ends.**
- ➔ **Lower value selected.**
- ➔ **MSS sent in the Options header of the SYN segment**
  - ⇨ clearly cannot (=ignored if happens) send MSS in a non SYN segment, as connection has been already setup
  - ⇨ when SYN has no MSS, default value 536 used
- ➔ **goal: the larger the MSS, the better...**
  - ⇨ until fragmentation occurs
  - ⇨ e.g. if host is on ethernet, sets MSS=1460
    - →1500 max ethernet size - 20 IP header - 20 TCP header

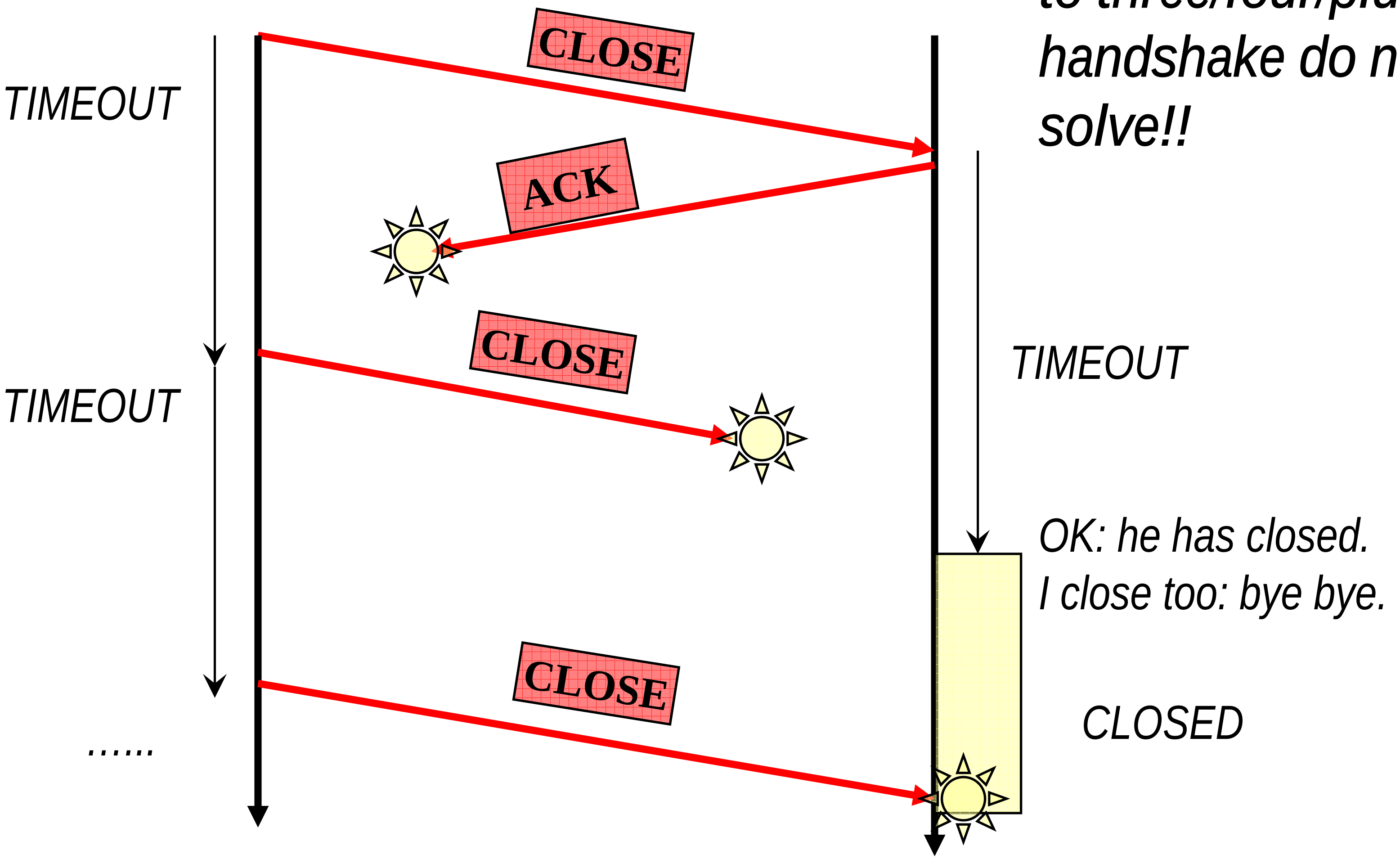
MSS advertise


CLIENT (C_MSS)
SERVER (S_MSS)
Conn request (C_MSS, SYN, seq=C_ISN)
Conn granted (MSS, SYN, seq=S_ISN, ack=C_ISN+1)
Acknowledge (seq=C_ISN+1,ack=S_ISN+1)
Use
recv
MSS
time
time

```
If (S_MSS<C_MSS)
  MSS = S_MSS;
else MSS = C_MSS;
```

Does not avoid fragmentation to occur WITHIN the network!!

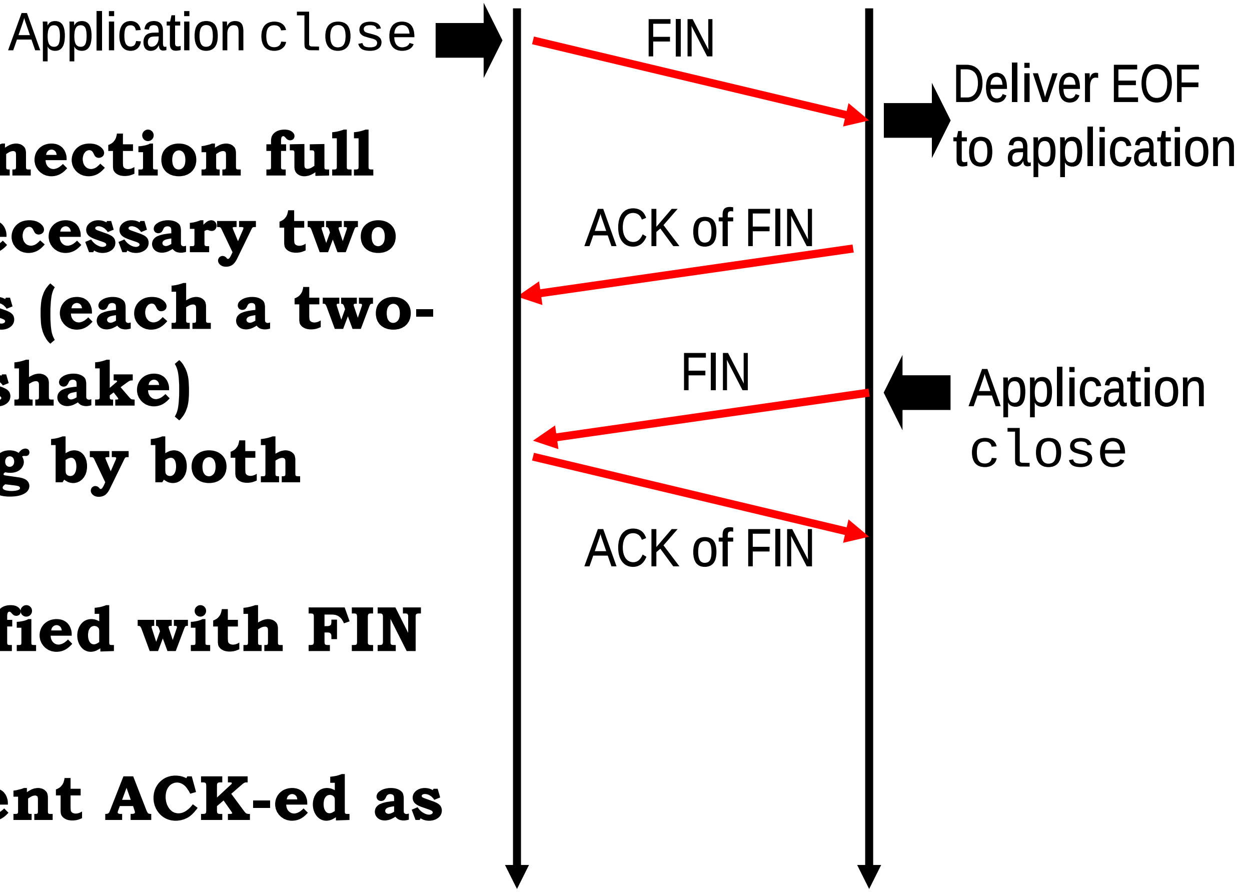
connection closing:
an impossible problem!



Suitable Timeout settings & Extension to three/four/plus way handshake do not solve!!

Connection closing in TCP

since it is impossible problem, use simples solution (two way handshake)

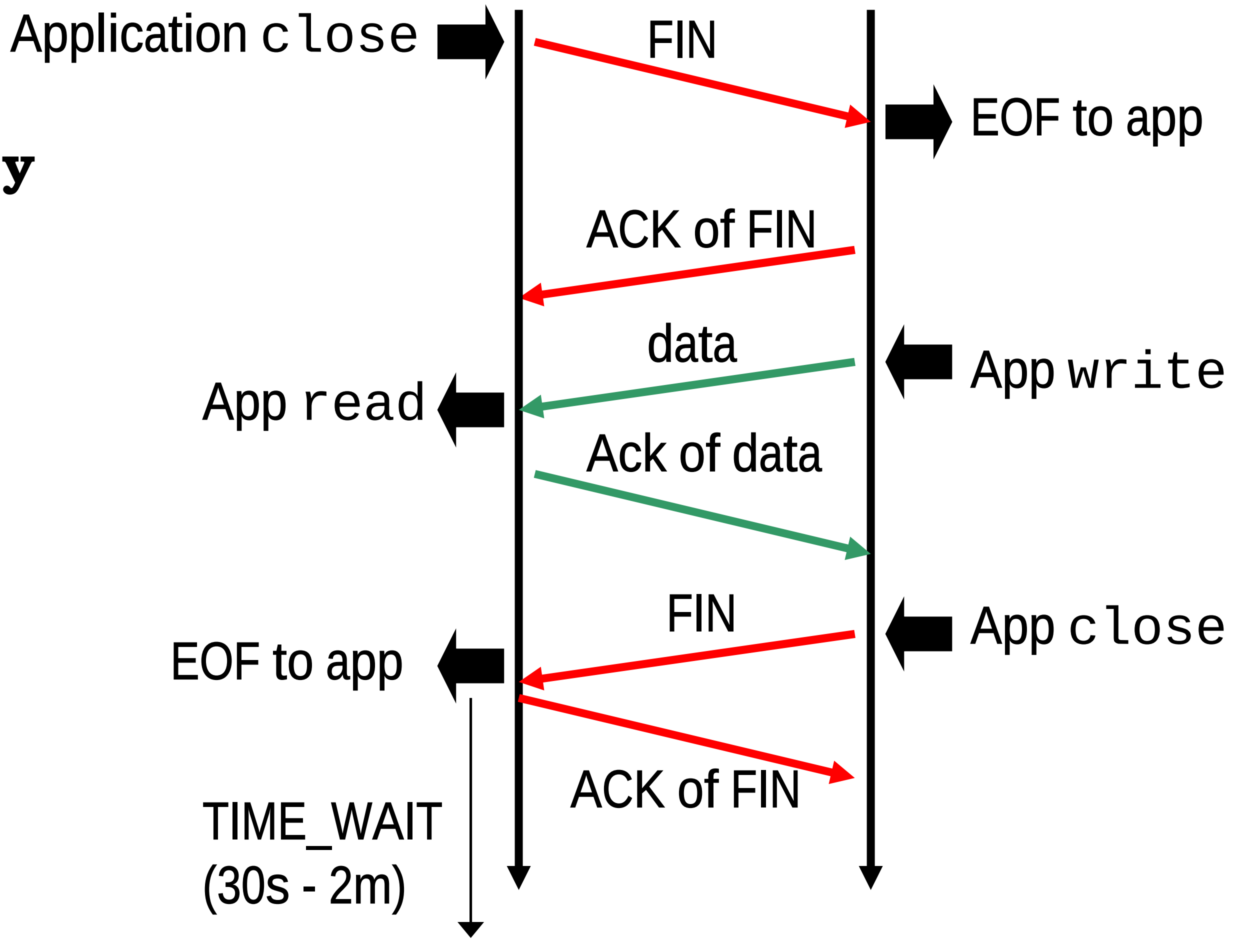


- ➔ **Since connection full duplex, necessary two half-closes (each a two-way handshake) originating by both sides**
- ➔ **close notified with FIN flag on**
- ➔ **FIN segment ACK-ed as usual**

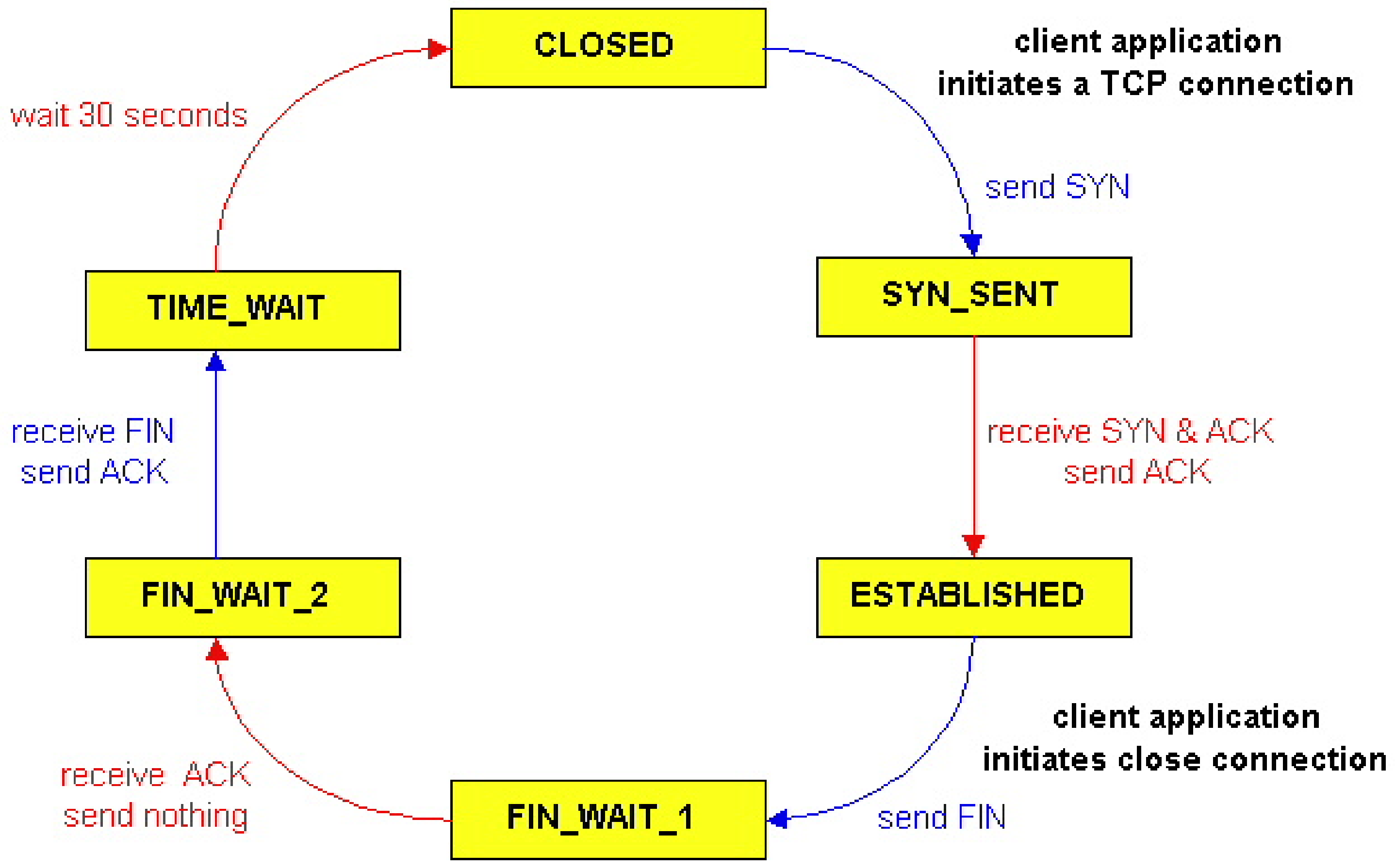
Half close

may close one direction only - seldomly used

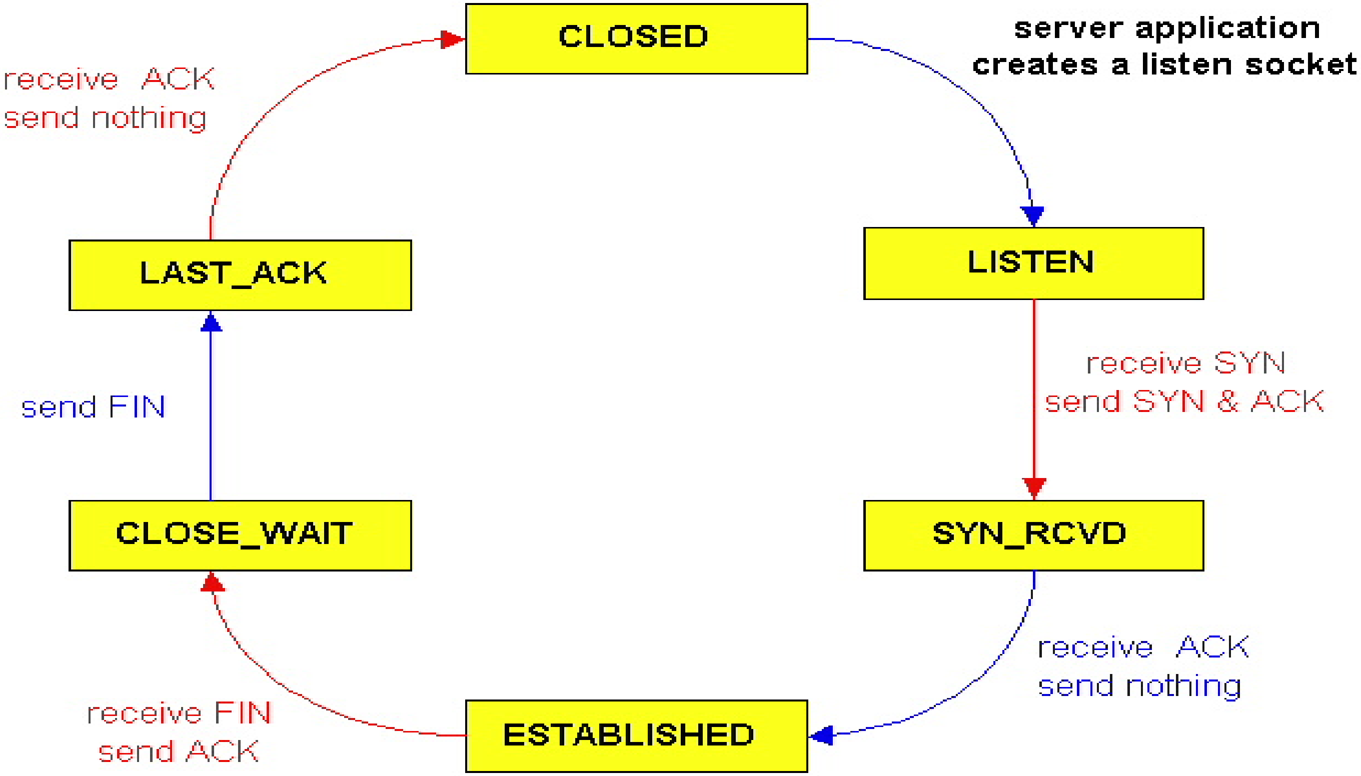
- ➔ **Supported by system call** shutdown **instead of** close



Connection states - Client

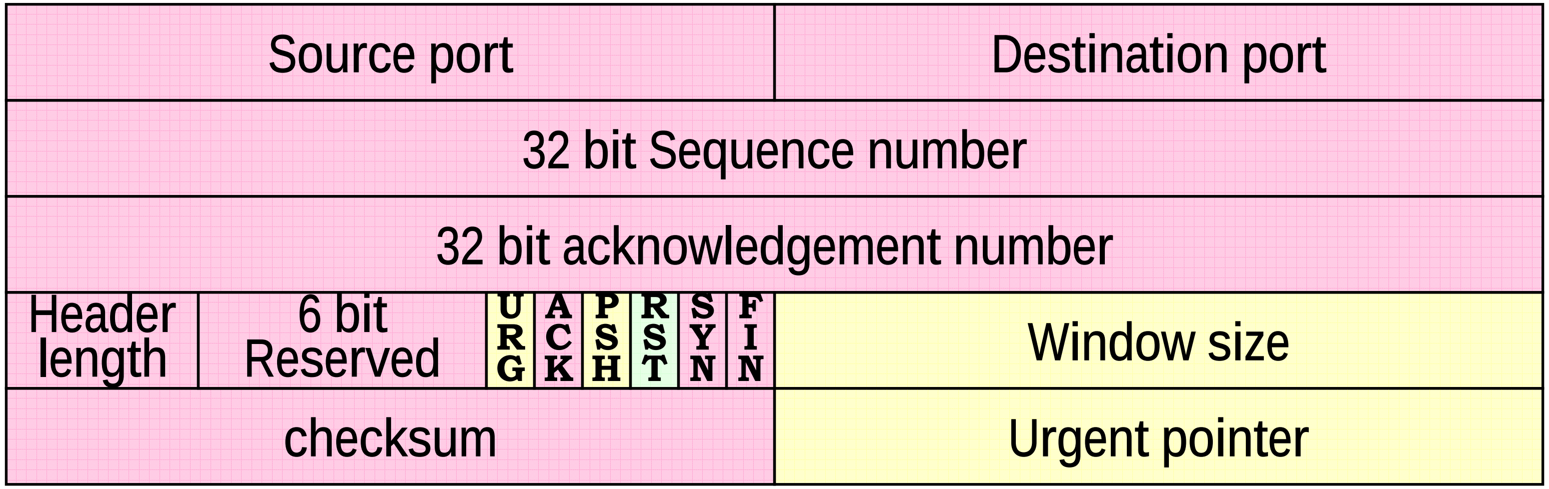


Connection States - Server



Why TIME_WAIT?

- ➔ **MSL (Maximum Segment Lifetime):** maximum time a segment can live in the Internet
  - →no timers on IP packets! Only hop counter
  - →RFC 793 specifies MSL=120s, but each implementation has its own value (from 30s to 120s)
- ➔ **TIME_WAIT state:** 2 x MSL
  - ⇨ **allows to "clean" the network of delayed packets belonging to the connection**
  - ⇨ **2xMSL because a lost FIN_ACK implies a new FIN from server**
- ➔ **during TIME_WAIT conn sock pair reserved**
  - ⇨ **many implementations even more restictive (local port non reusable)**
  - ⇨ **clearly this may be a serious problem when restarting server daemon (must pause from 1 to 4 minutes…)**



Source port
Destination port
32 bit Sequence number
32 bit acknowledgement number
Header length
6 bit Reserved
URG
ACK
PSH
RST
SYN
FIN
Window size
checksum
Urgent pointer

- ➔ **RST (Reset)**
  - ⇨ sent whenever a segment arrives and does not apparently belong to the connection
  - ⇨ typical RST case: connection request arriving to port not in use
- ➔ **Sending RST within an active connection:**
  - ⇨ allows *aborting release* of connection (versus *orderly release*)
    - → any queued data thrown away
    - → receiver of RST can notify app that abort was performed at other end