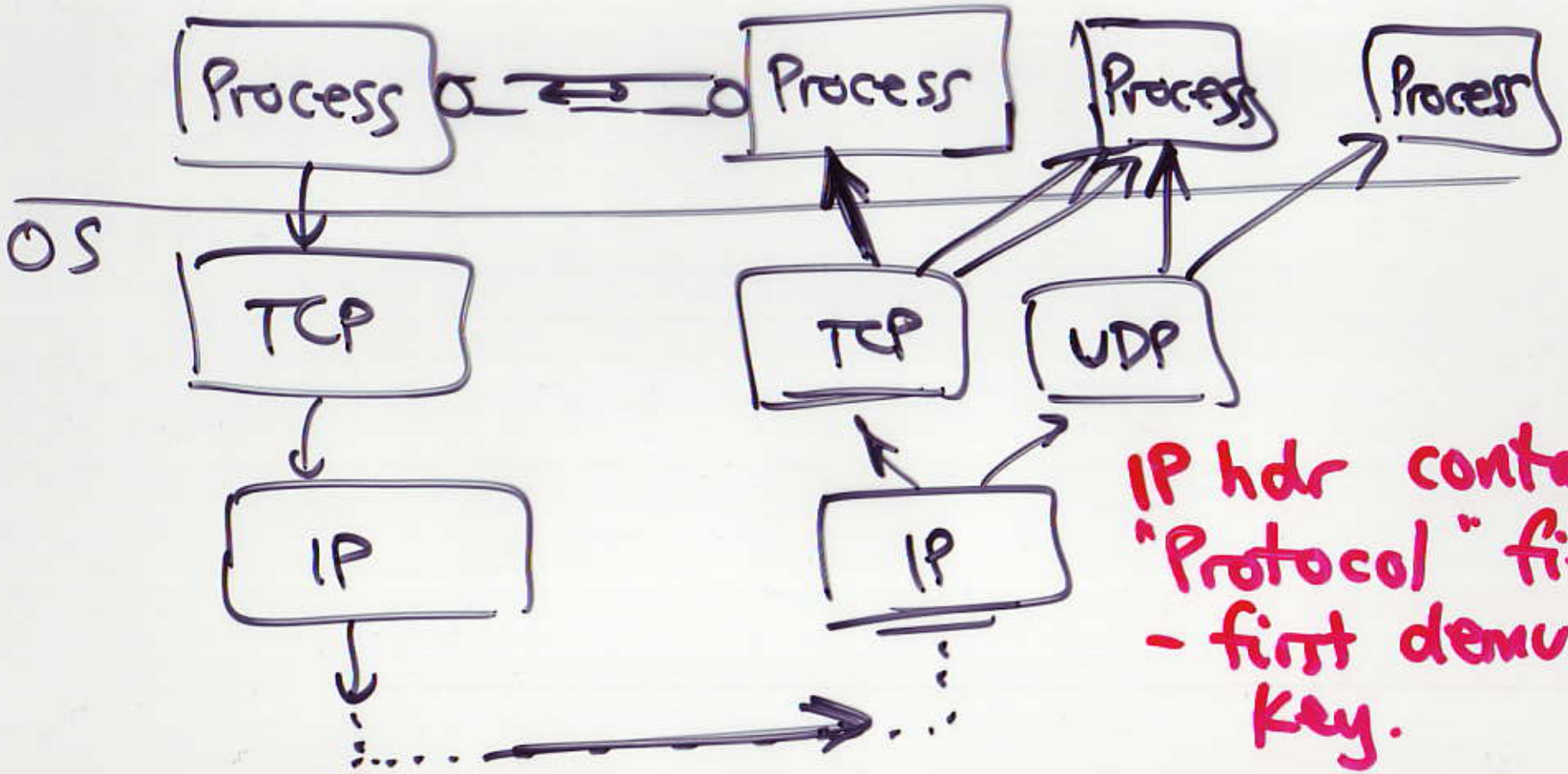
TCP Connections

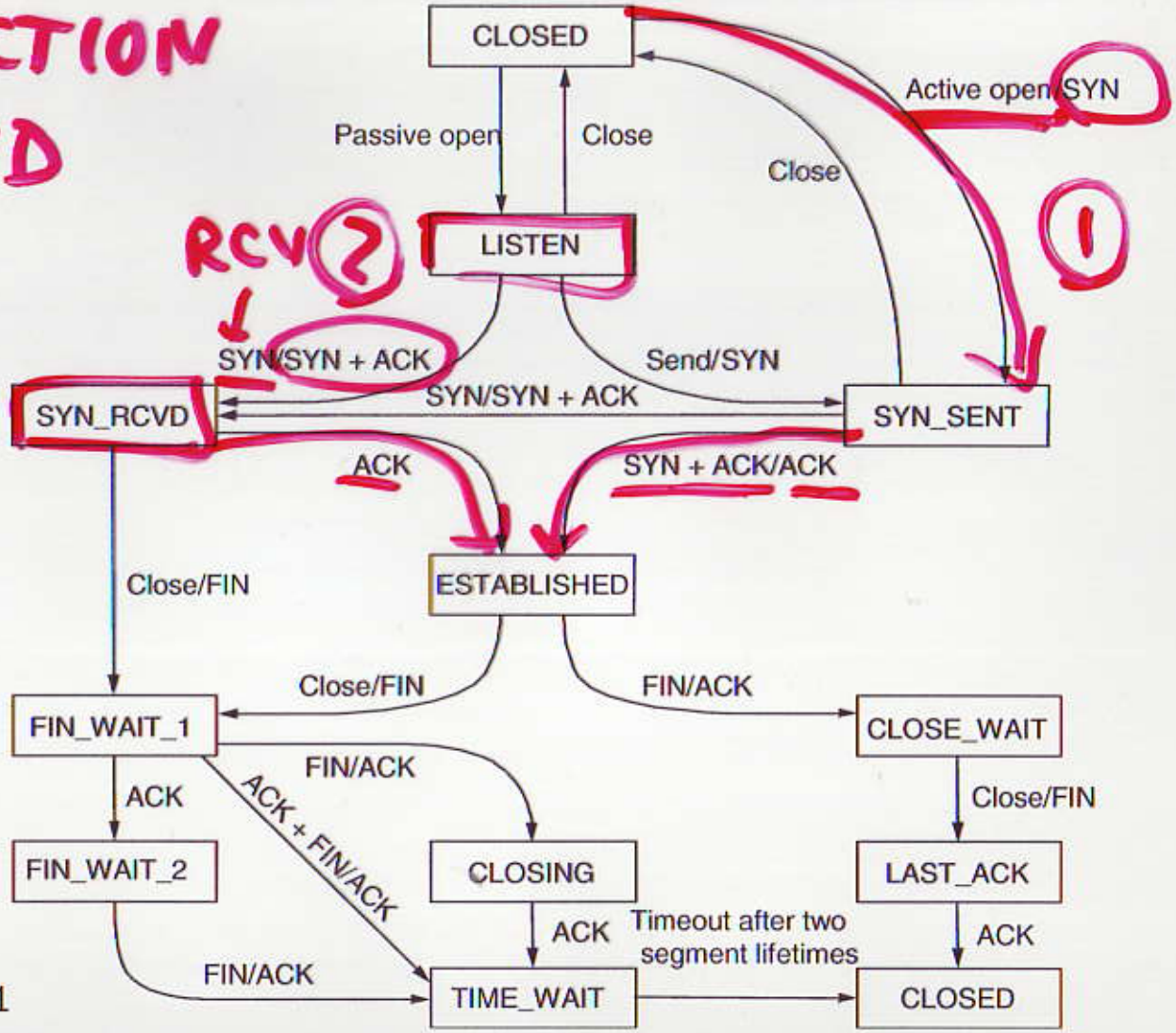
demux = demultiplex



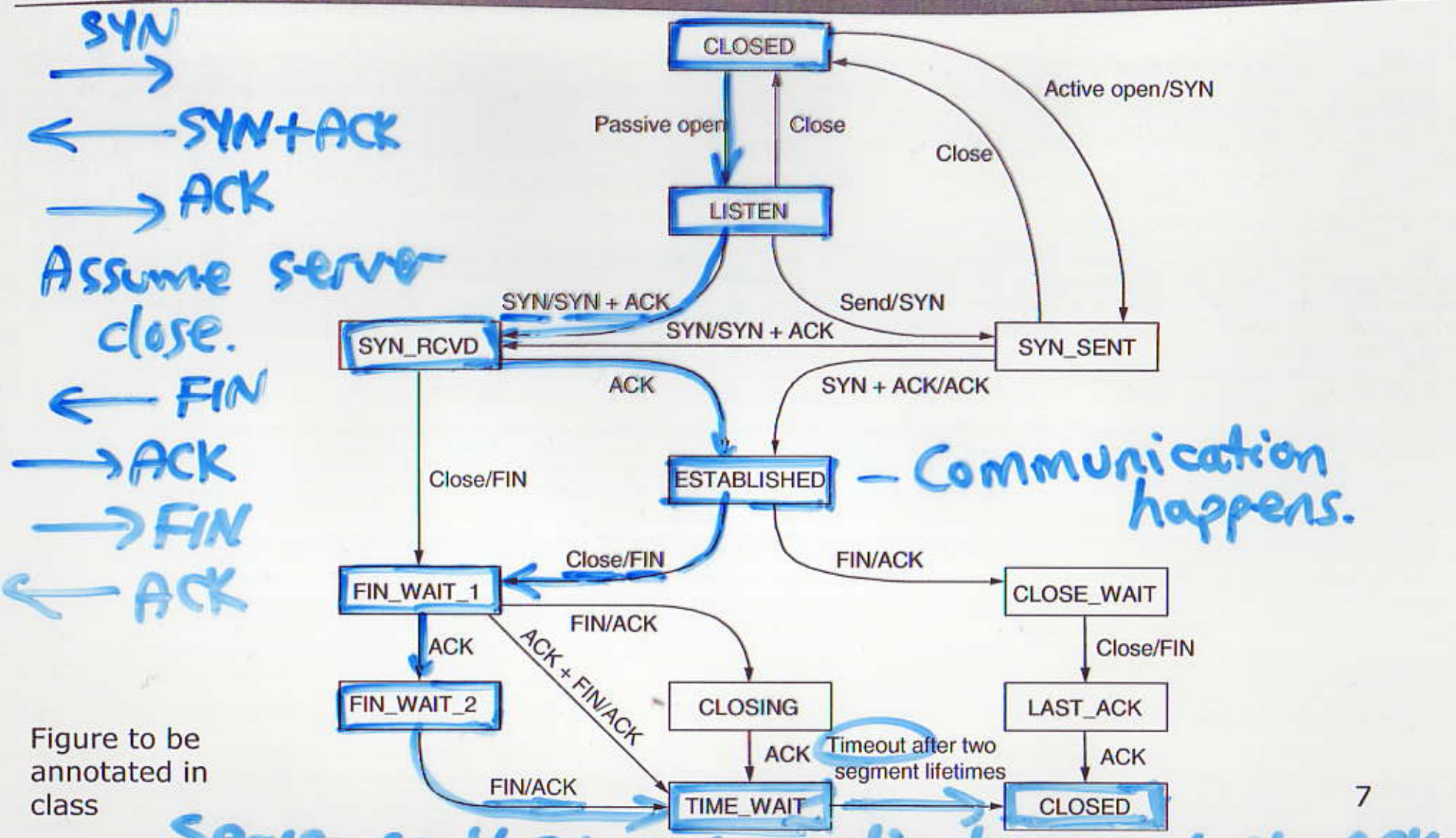
TCP - connection ID is second demux key

TCP State Diagram

EVENT/ACTION
RCV /SEND



TCP State Transitions – Typical Server



SYN
SYN+ACK
ACK
Assume server close.
FIN
ACK
FIN
ACK

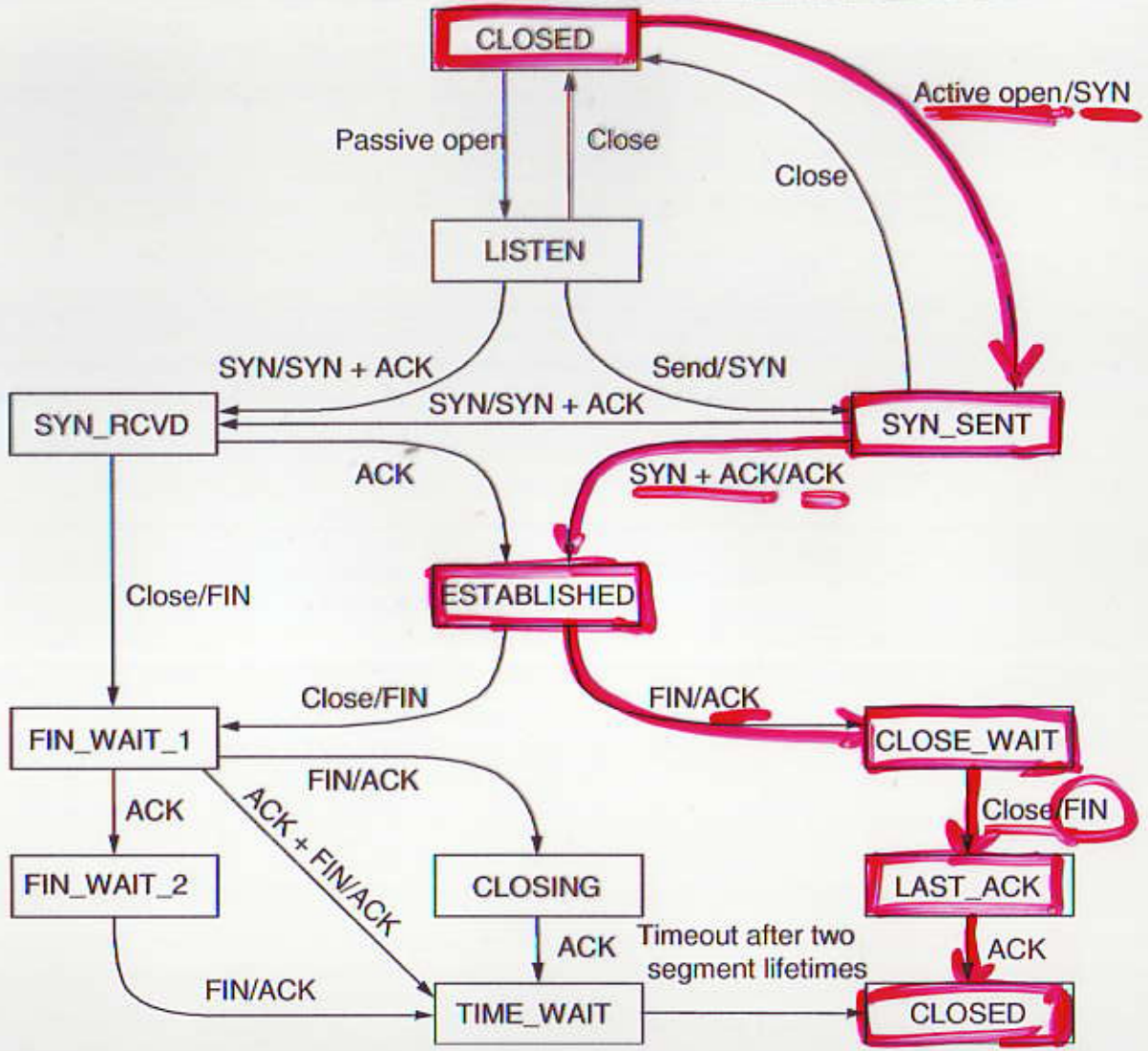
– Communication happens.

Figure to be annotated in class

Server can't guarantee client rcvs last ACK

Both sides must close connections

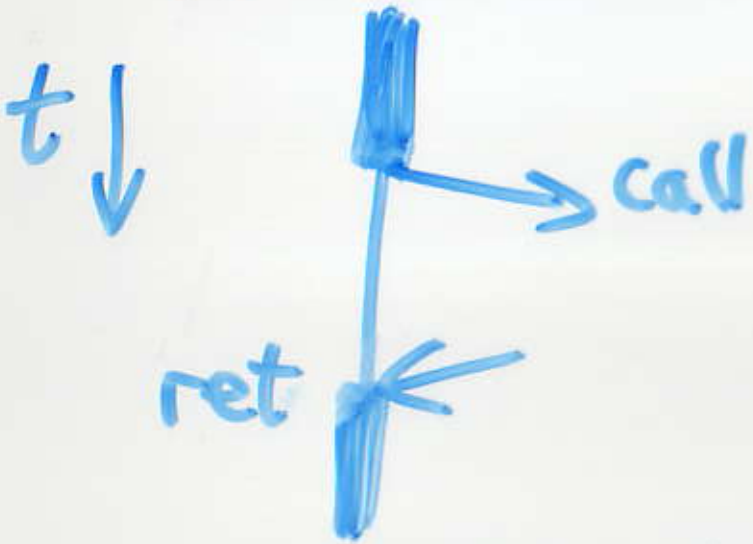
TCP State Transitions – Typical Client



SYN
SYN+ACK
ACK
FIN
ACK
FIN
ACK

Figure to be annotated in class

RPC Client



- no work to be done whilst awaiting reply
- no incoming comm'n from other than server
- reply is expected

If process matches these

⇒ use rpc call

(at some point in time)

RPC server

accept = blocking receive

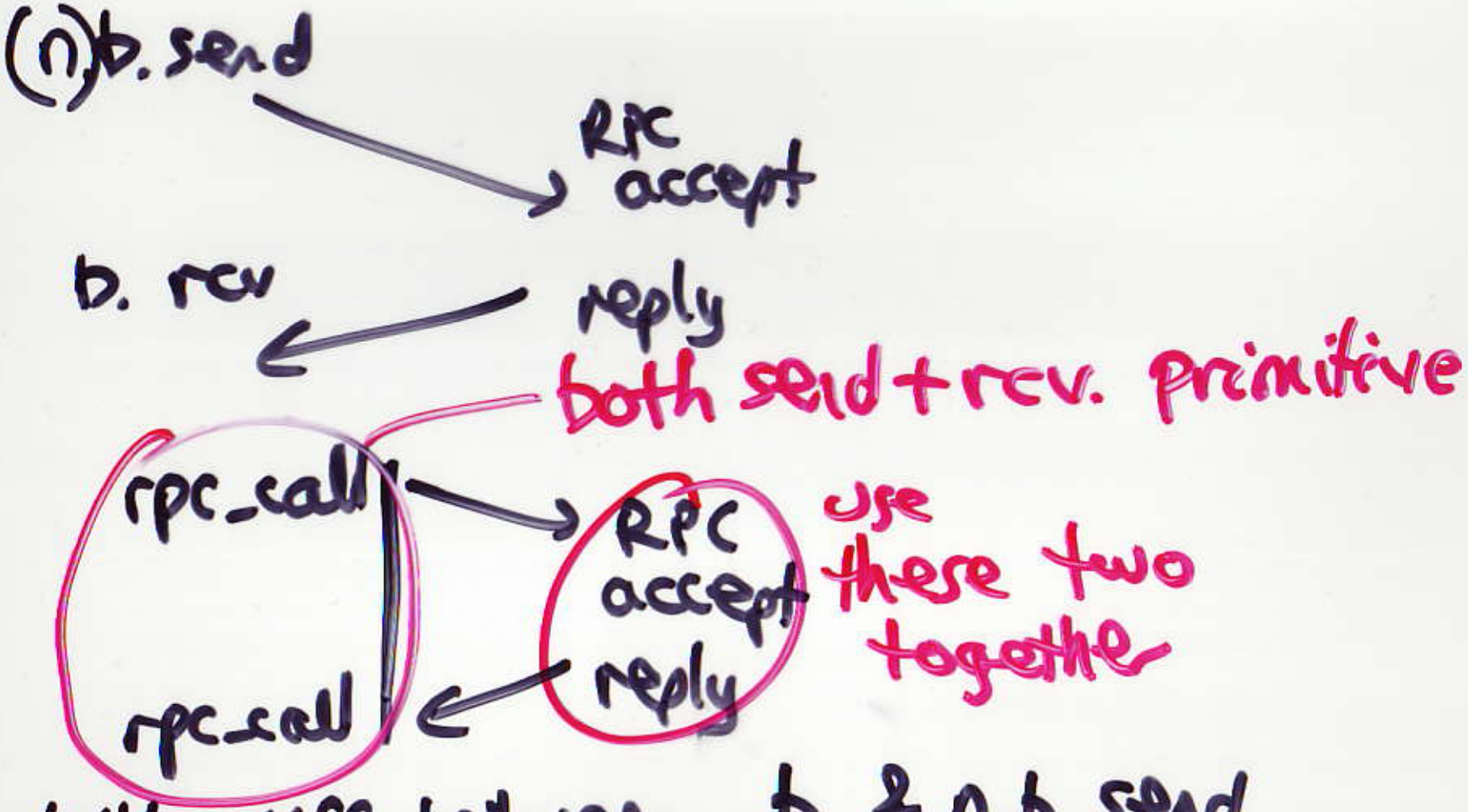
reply ← can send messages as part of processing

— Can't receive messages

If matches these ⇒ RPC accept reply

- only exists to process client requests
- all incoming messages are service requests
- each message/request gets single reply
- no other computation to do when idle

Each process' communication primitives can be chosen independently of other processes



Little diff between b. & n.b send (except when other processing to do)

Choose n.b. rcv only if other processing to do.