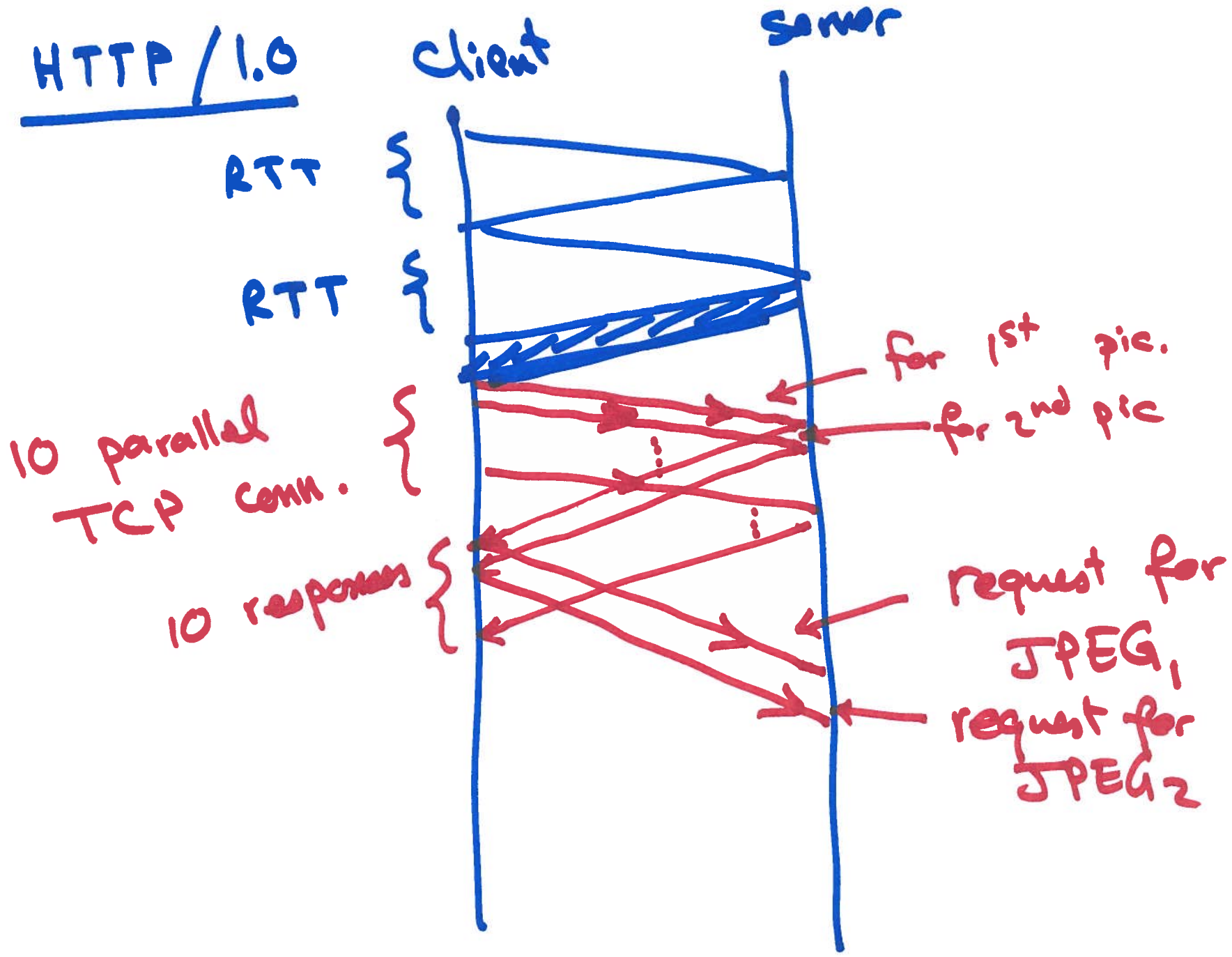
HTTP/1.0


client
server
RTT
RTT
10 parallel TCP conn.
10 responses
for 1st pic.
for 2nd pic
request for JPEG1
request for JPEG2

- Ncat (nc, netcat)
  - Client: ncat remotehost port #
  - Server: ncat -l " "

HTTP messages

Request

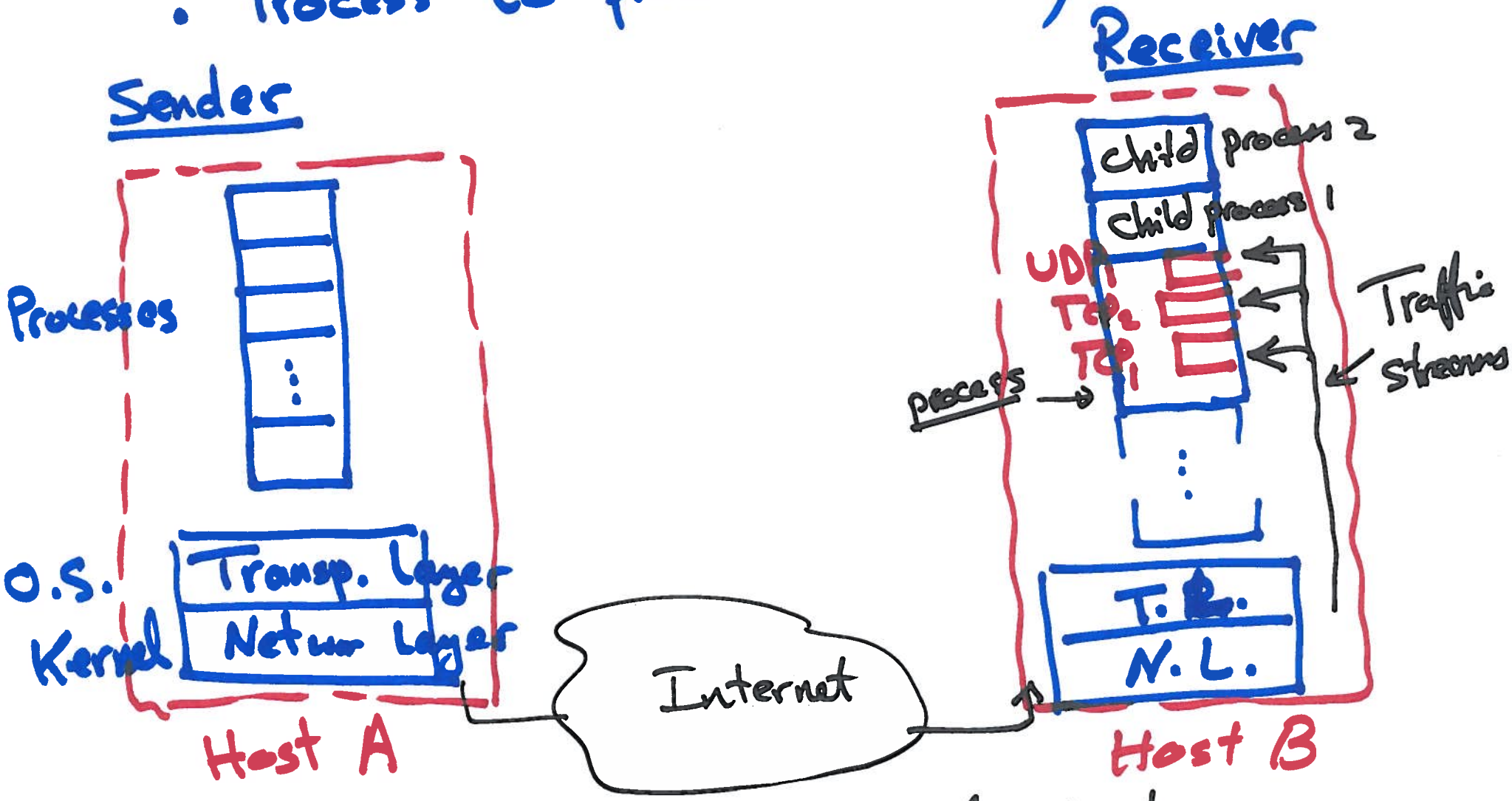
- Method
- [Header lines
- [Entity body

Response

- Status line
- [Header lines
- [Entity body

Transport Layer Demultiplexing Services

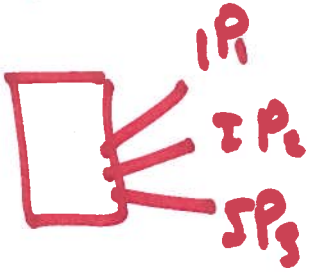
- Process to process connectivity



- Demultiplexing: Which queue of which process receive a packet?

- Analogy

Postal System	Internet
• Each building has a street address	• Each host has an IP #
	net. interface card
• Each apartment has a #	• Deliver a pkt to a process (identified by its #)
	• Deliver a pkt to a TCP queue or UDP queue (identified by using raw numbers)



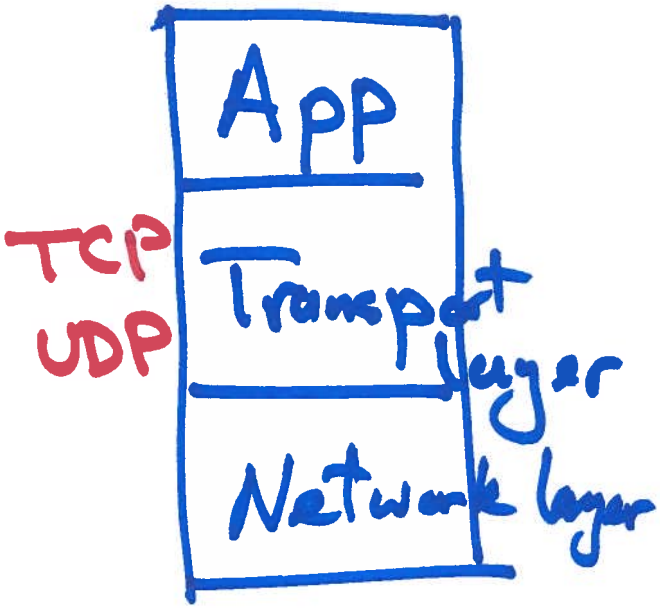
Port numbers

No for using process numbers

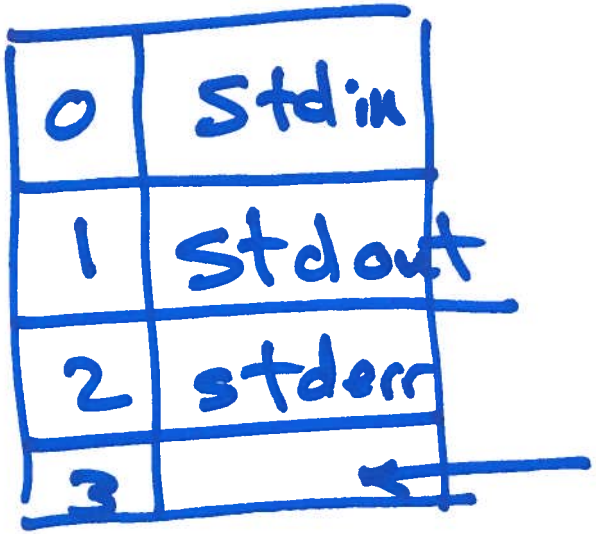
- A process may have multiple "sockets".
- A process providing a "well known service" may not have a "well known process #".
- A process may fork children processes.

Solution: port #s

Socket API



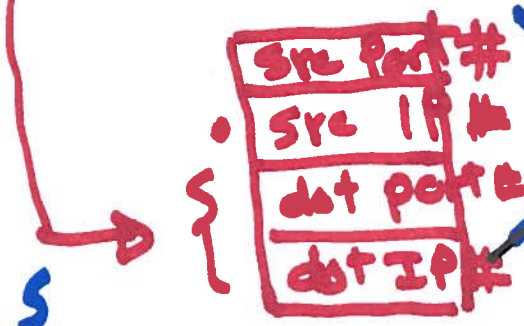
Unix file descriptor



Q: Can the socket of "P_3 on client A" & the socket of another unrelated process "P_3' on client B "A" be assigned the same port #?

Client

- int sfd = socket(...)
  - TCP
- Fill in special C-struct with server's (IP#, Port#)
- Connect(...)
- for(;;){
  - read(sfd, ...)
  - write(sfd, ...)
  - }



Server

- int listenfd = socket(...)
  - TCP
- Fill in special C-struct with server's (IP#, Port#)

- Bind(listenfd,)
- listen(, 10)
  - Can accept 10 connections
- for(;;){
  - int connfd = accept(listenfd, ...)
  - read(connfd, ...)
  - write(connfd, ...)

- UDP: no need for connect() or accept()