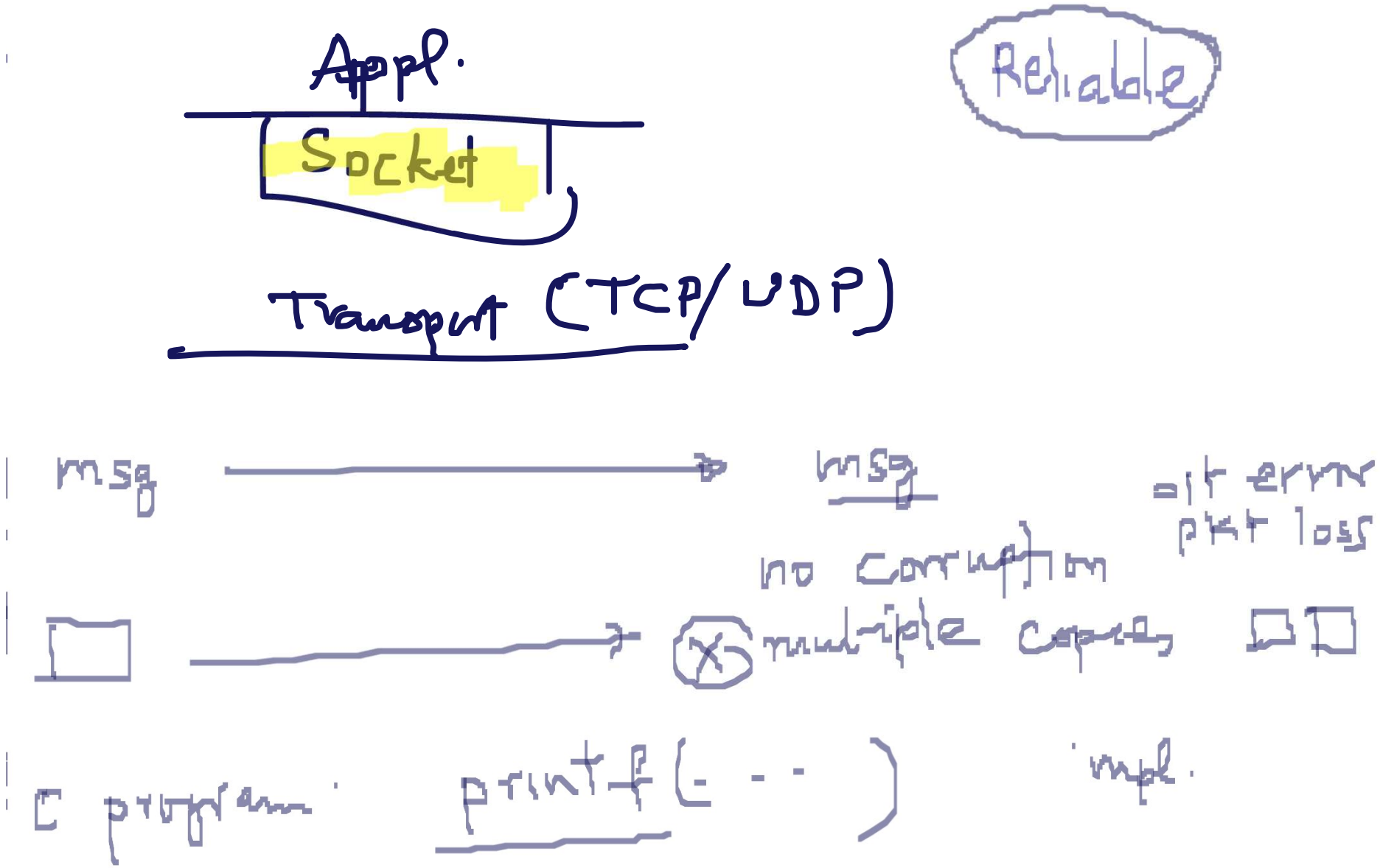

Appl.
Socket
Transport (TCP/UDP)
Reliable
msg
msg
bit error
pkt loss
no corruption
multiple copies
C program: printf(...)
impl.

Socket Appl. Design

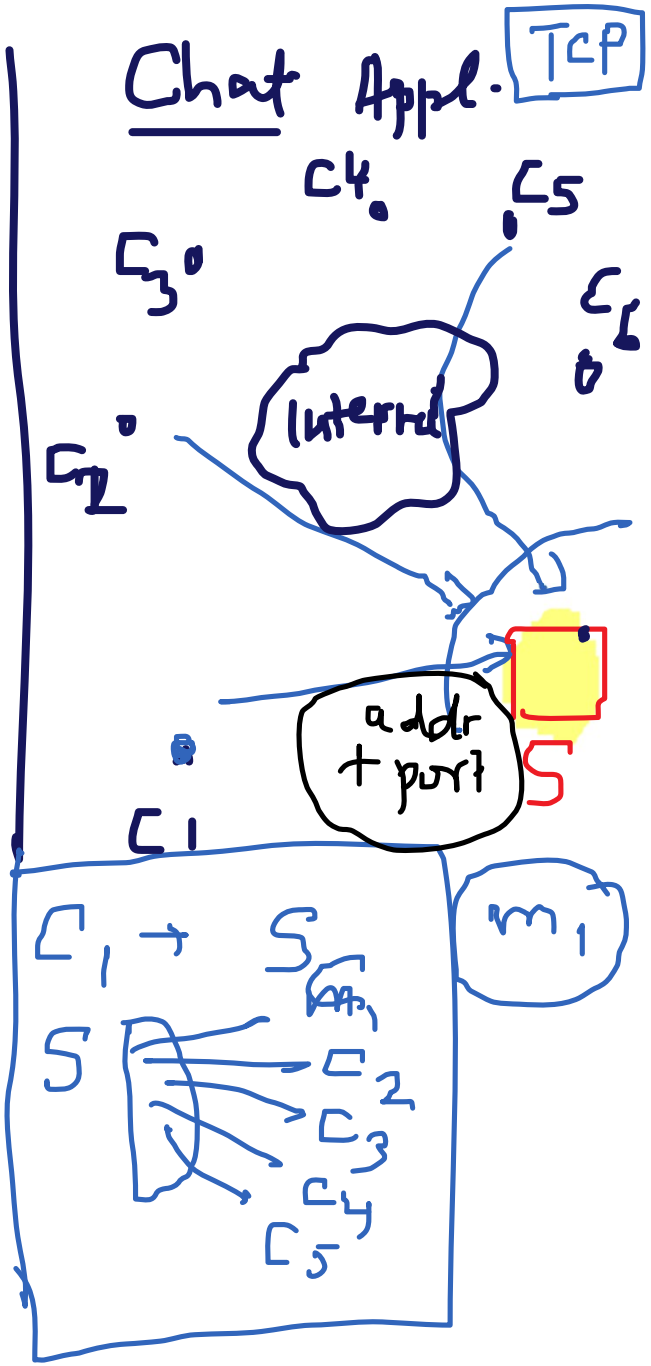
- Client → msg → server
- server → msg → All clients.

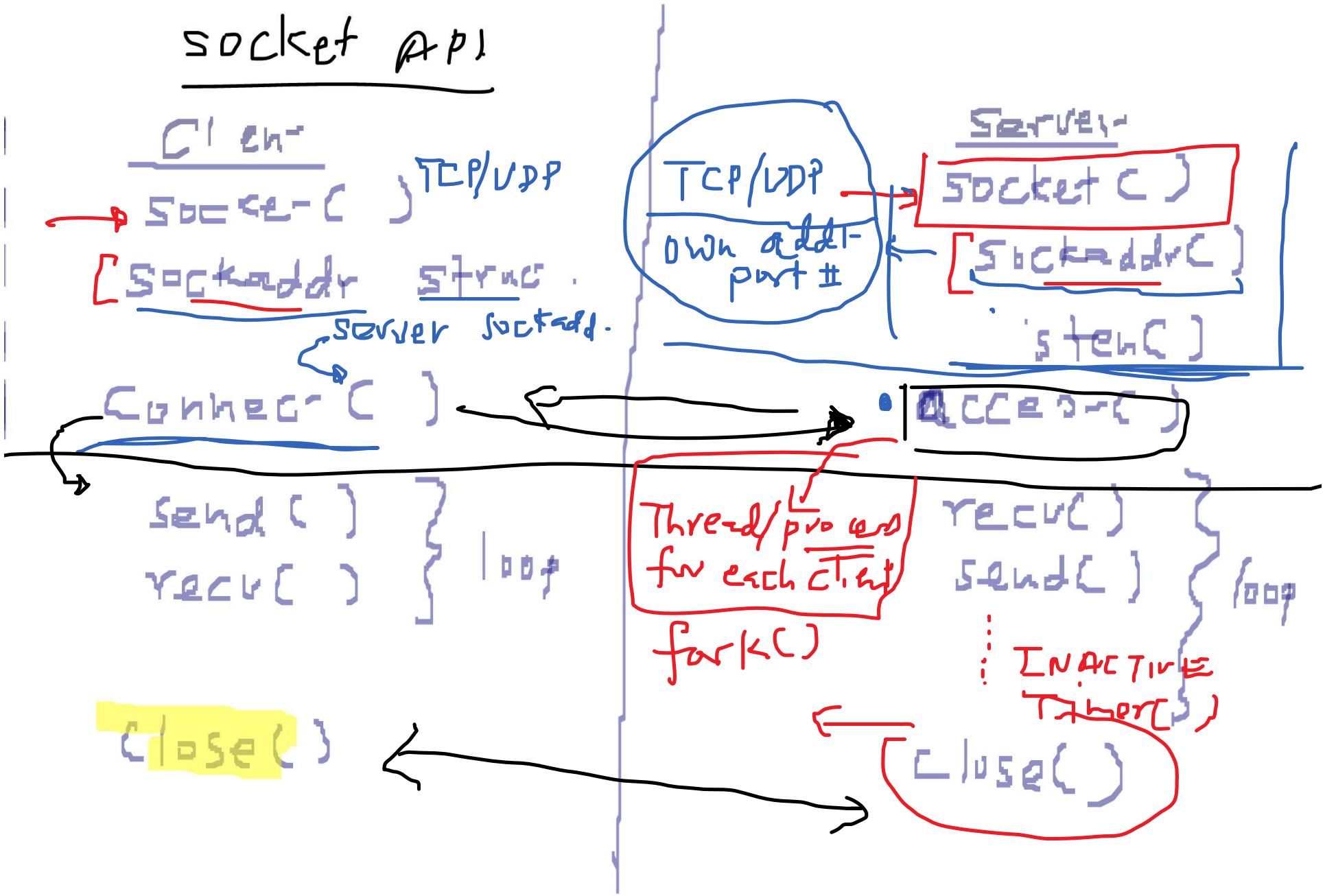
- Client → registration → Server
- Client → termination → Server

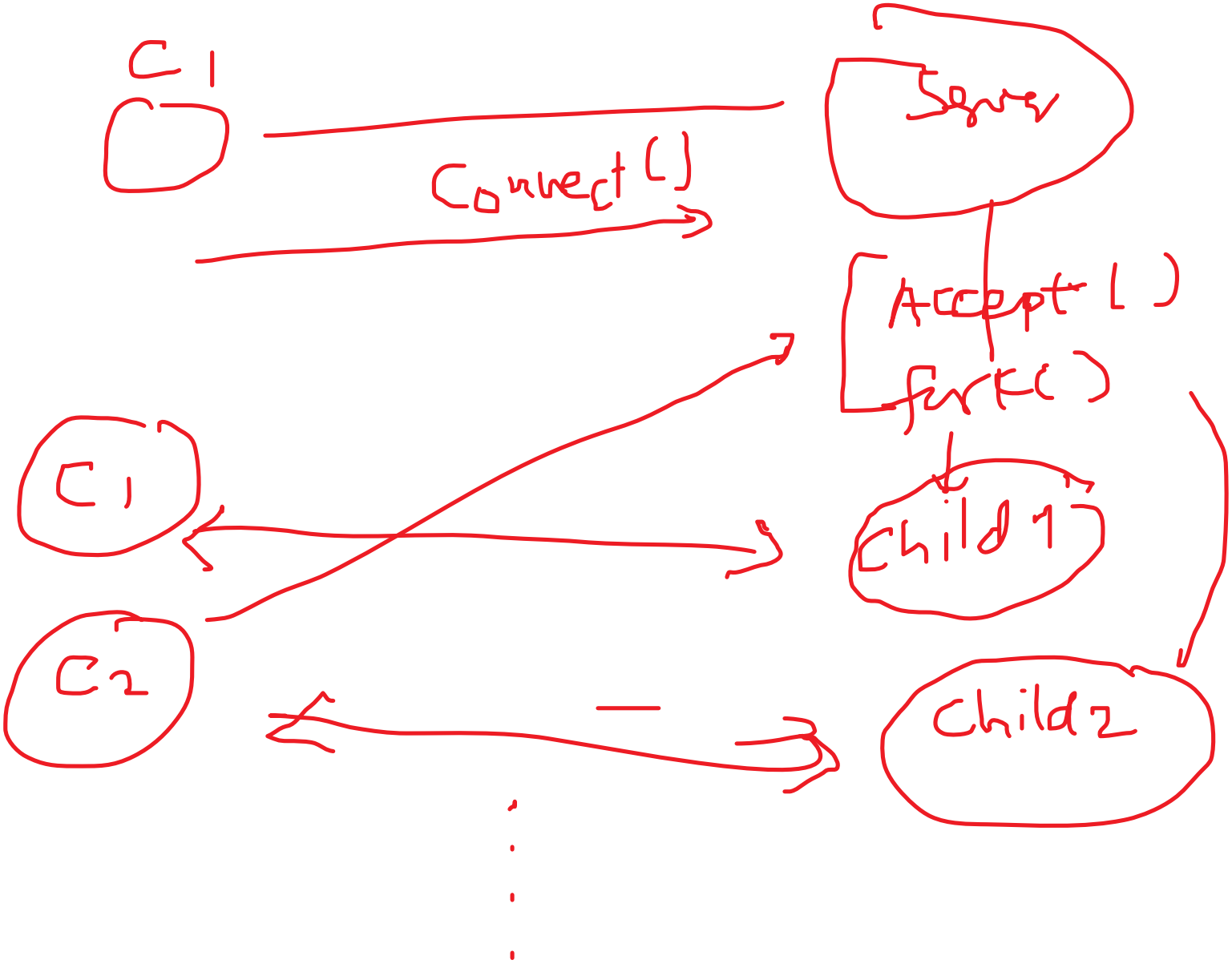
API

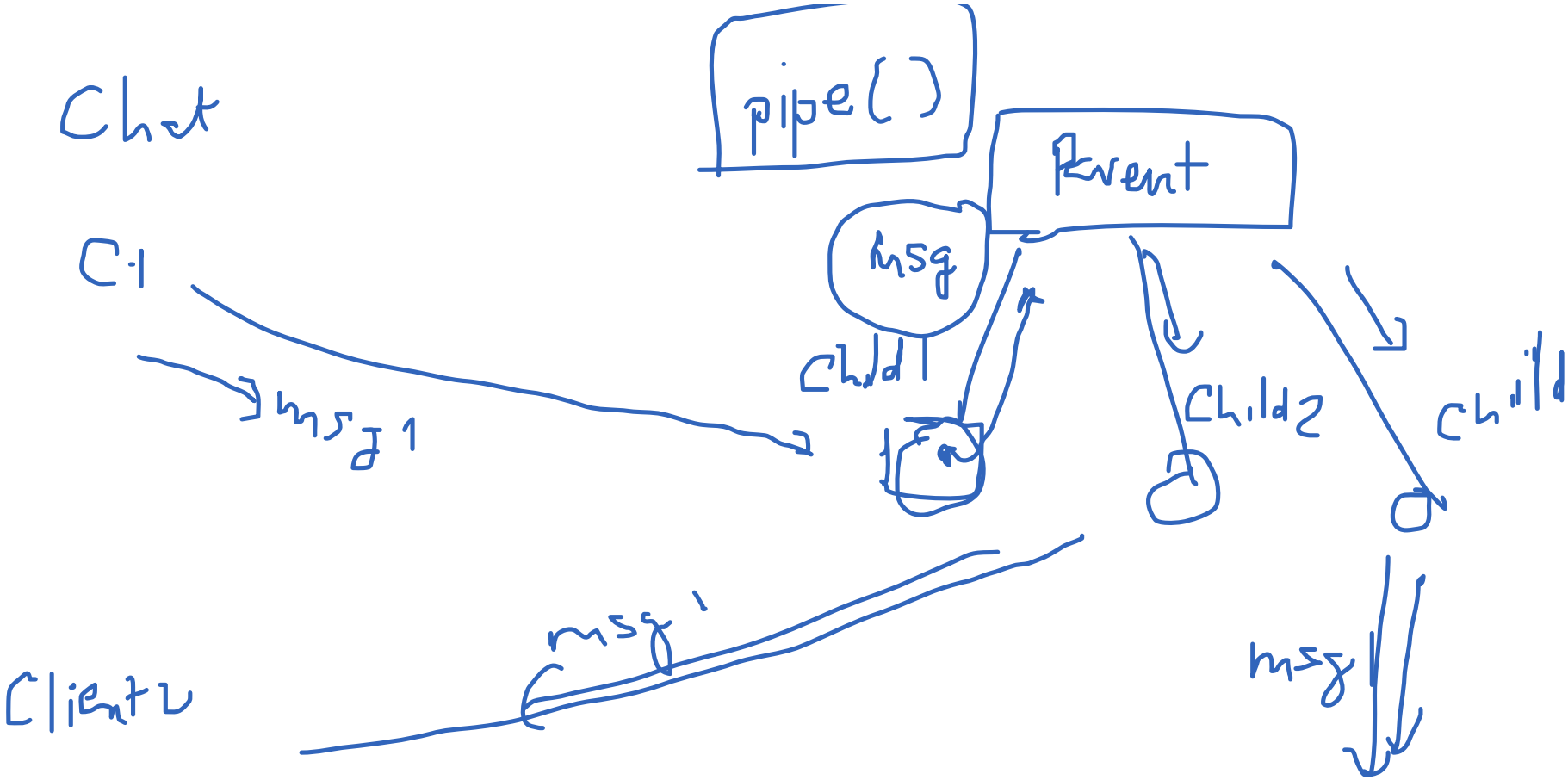
- Connect ( )
- disconnect ( )
- send ( )
- receive ( )

Chat Appl. TCP




socket API
Client
Socket( )
TCP/UDP
[Sockaddr struc.
server sockadd.
Connect( )
send( )
recv( )
loop
close( )
TCP/UDP
own addr-
port #
Server
socket( )
[Sockaddr(
listen( )
Accept( )
Thread/proc
for each client
fork( )
recv( )
send( )
loop
INACTIVE
TIMER( )
close( )


C1
Server
Connect()
Accept()
fork()
C1
child 1
C2
child 2


Chat
pipe()
Parent
msg
C1
msg1
Child1
Child2
Child
msg'
Client
msg

What transport service does an app need?

data integrity

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

security

- encryption, data integrity, …

Transport service requirements: common apps

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	
interactive games	loss-tolerant	few kbps up	yes, few secs
text messaging	no loss	elastic	yes, 100's msec

interactive

yes and no

Internet transport protocols services

TCP service:

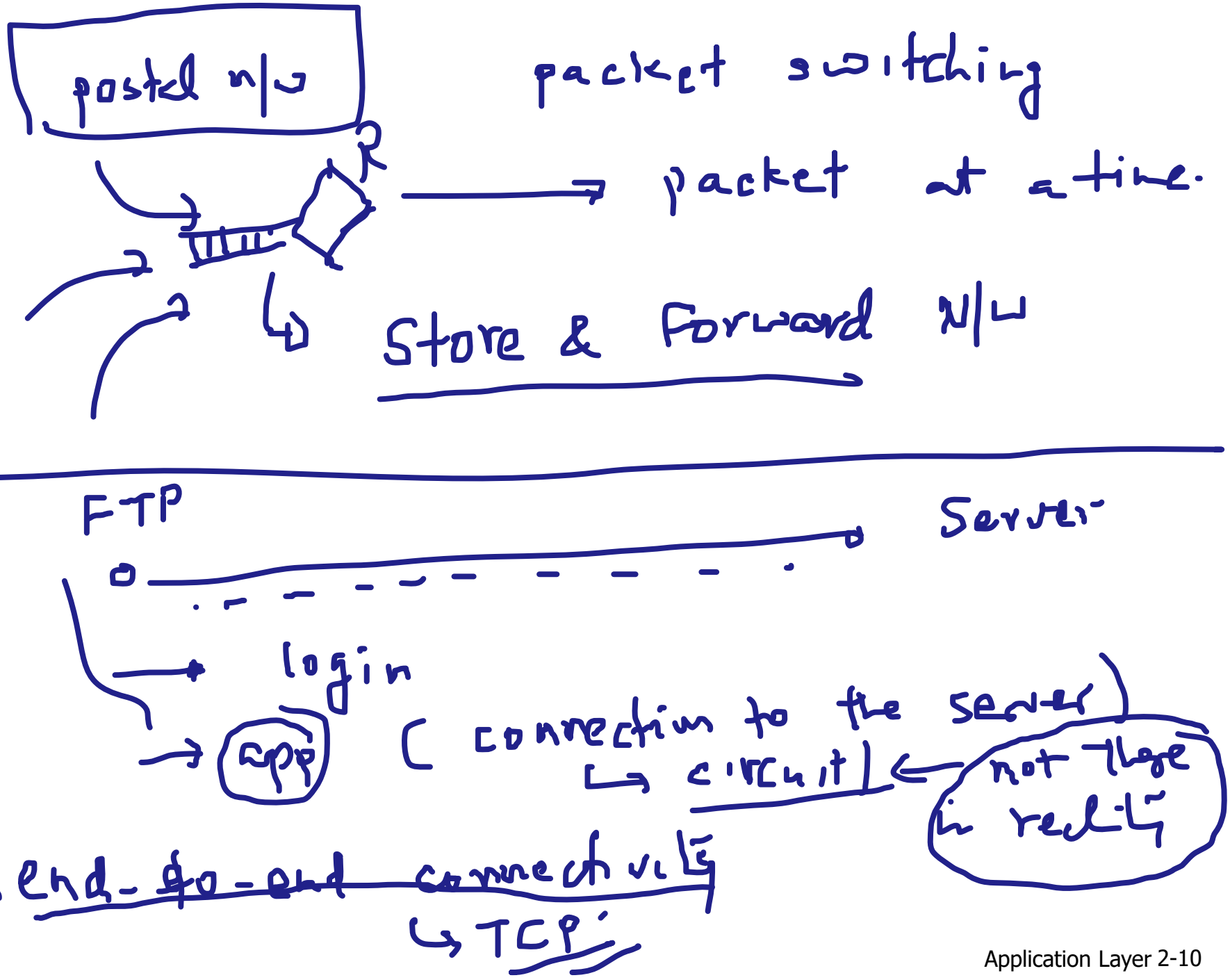
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

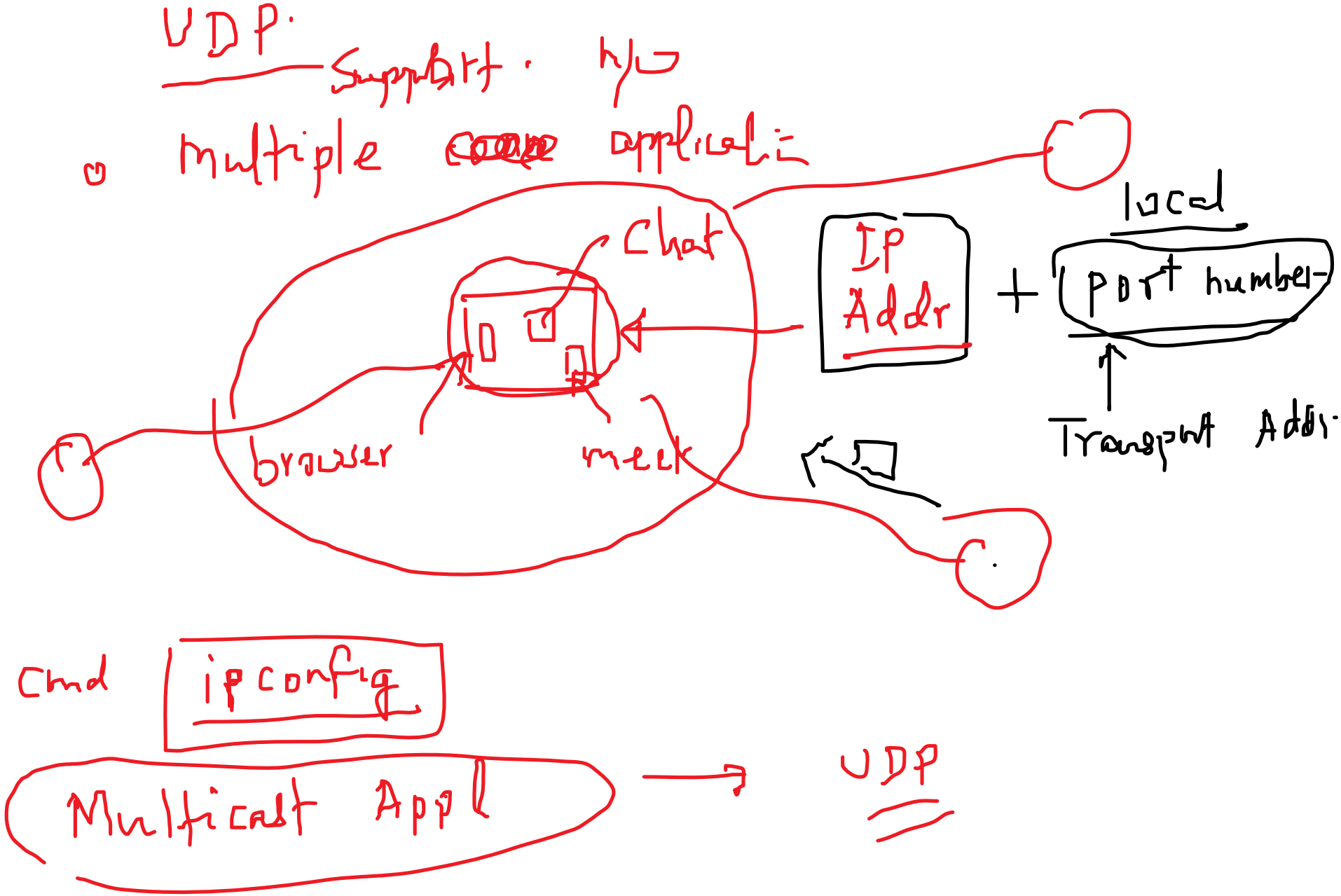
→ one to one appl.

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup,

Q: why bother? Why is there a UDP?


postal n/w
packet switching
packet at a time.
Store & Forward N/W
FTP
Server
login
copy
(connection to the server
↳ circuit
not there in reality
→ end-to-end connectivity
↳ TCP


UDP.
Support. h/w
multiple applicat
Chat
IP Addr
local
port number
Transport Addr.
browser
meek
Cmd
ipconfig
Multicast Appl
UDP

Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Securing TCP

TCP & UDP

- no encryption
- cleartext passwds sent into socket traverse Internet in cleartext

SSL

- provides encrypted TCP connection
- data integrity
- end-point authentication

SSL is at app layer

- apps use SSL libraries, that "talk" to TCP

SSL socket API

- cleartext passwords sent into socket traverse Internet encrypted
- see Chapter 8