WEBSOCKETS

In a regular HTTP connec^n, the foll happens

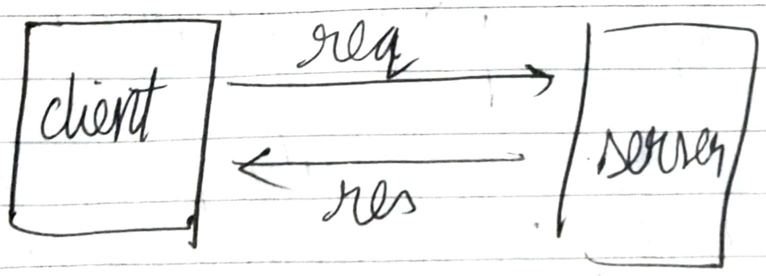
1. Connec^n is established after TLS handshake
2. Req. is sent from client $\xrightarrow{to}$ server
3. Res. is recd by client $\xleftarrow{frm}$ server
4. Connec^n is terminated after reqs. is fulfilled.

eg:- http://abc.com, https://abc.com

If 5 diff reqs. are sent to server, 5 diff connec^{n}s. are established to provide res. and then after gettin back responses, all 5 connec^{n}s are terminated.

5. New connec^n is created every single time on the same

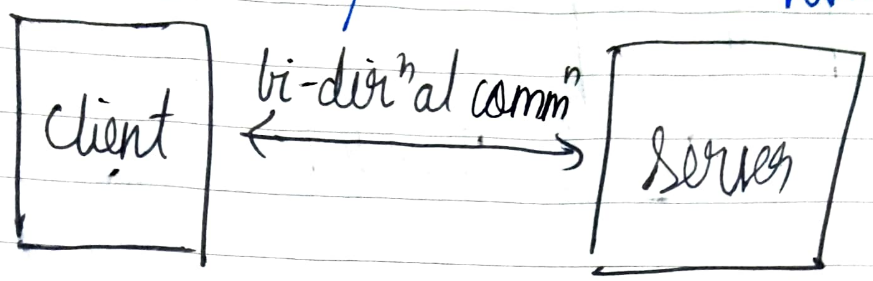
eg:- new connec^n created to hit several REST endpts for issuin reqs.



unidir^{n}al comm^n
(only receiver can send data)

Websockets

client and server can talk on real time w/o cont.ly havin to send reqs.



These sockets are created as ws://abc.com
or wss://abc.com

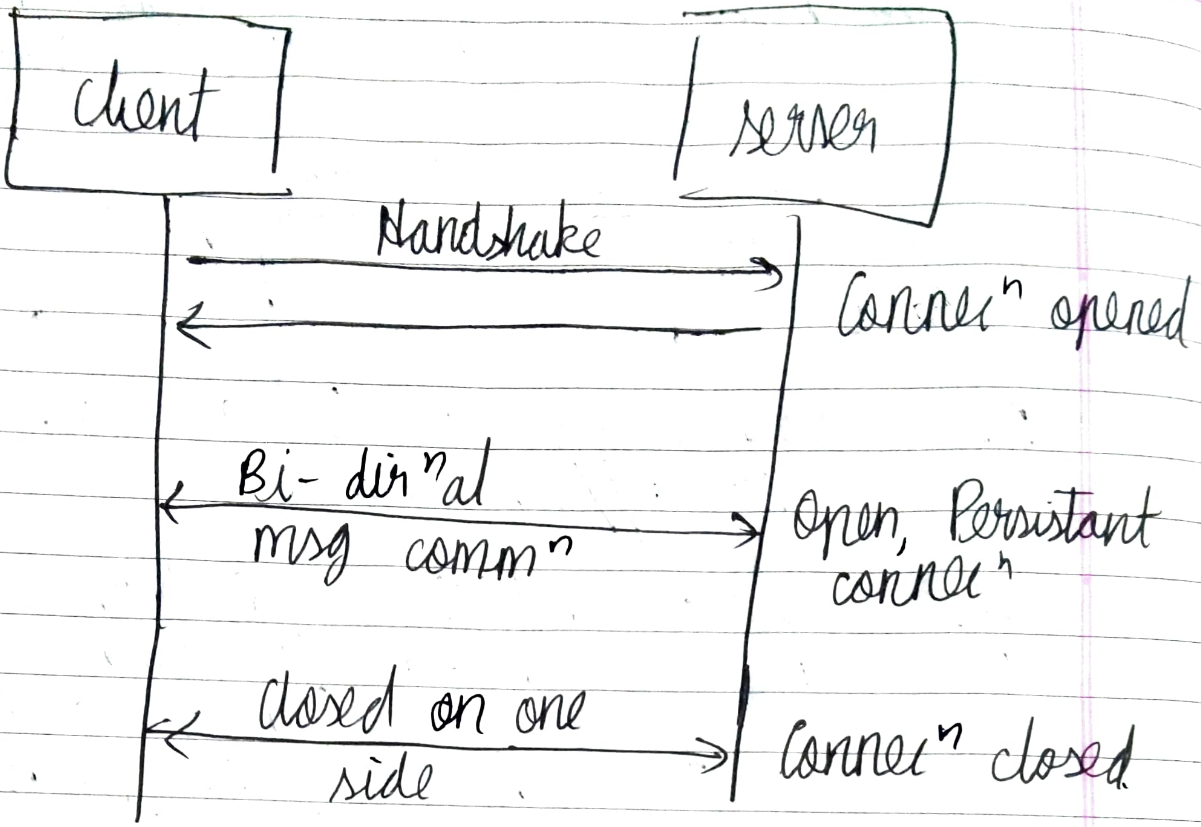
eg:- Send a text to a friend thru a chat window.
Text sender → data pushed frm client (sender's device) to server
Text receiver → data pushed frm server to client (receiver's device) and thus, the UI on client updates to show recd. msg.

Bi-dir^{n}al comm^n is achieved by not closin/terminatin the connec^n that they originally opened

Whenever a client establishes a rela^{n}ship with the server its connec^n stays until the client or the server decides to terminate the connec^n

eg:- used in all real time services like stockbrokers platforms, chat apps, maps app.

NOTE: Websockets is an HTTP upgrade → uses the same TCP connec^n over ws:// or wss://



Where, When are Websockets used

low latency

① Real-time apps (load the UI w/o refreshin the UI)

→ asyncly sends req.s w/o refreshin.

NOTE : AJAX → an implem^{n} over HTTP where we do HTTP polling/streaming
AJAX uses HTTP so everytime it's gotta open a new socket & then poll the data

② Gaming apps (UI refreshed w/o hvin to re-establish new connec^{n})

③ Chat apps.

Can go to any real-time website (say :- a stockbrokin service) and open Chrome Dev Tools → Network → click on each channel URL loaded as part of website → see if req URL is → http or https

→ ws or wss

Most feeds are updated via ws only.

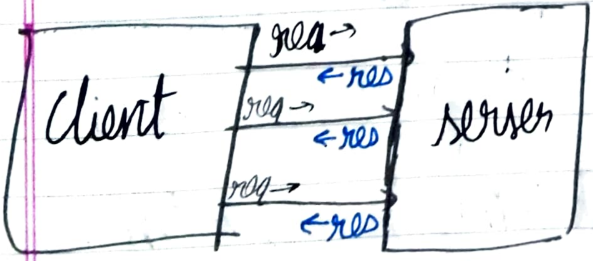
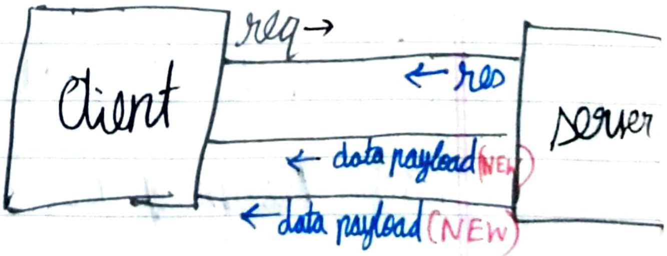
Can go in a ws channel → Frames and observe the diff msgs gettin loaded cont.ly

When not to use Websocket

① Whenever we don't wanna hv ONLY the new data (want old info also → here HTTP REST endpts. provide a 1 stop soln.)

② Loading the data only once.

AJAX vs websockets

AJAX	Websocket
 Client req → ← res req → ← res req → ← res server	 Client req → ← res ← data payload (NEW) ← data payload (NEW) server
Polling → send AJAX req ever 'x' amt of secs. for new data (not true real time)	ⓘ 2.5x faster than AJAX ~~as~~ ~~unlike~~ ⓘⓘ Unlike AJAX, here req. header is sent only once.

Long Polling → send req. to server to keep connec^n open until new data

er

Server Events → use the eventSource API to send msgs from server

→ Not truly bi-direcⁿall as it is based from the server sendin to the client.

→ Requires an async ~~loop~~ server or event loop

→ No binary msg capability (which websockets support)

Websockets == not replacement of HTTP

i) HTTP provides auto-cachin
ii) WS needs special config. for load balancing
iii) WS can't comm. with REST API

socket IO

1 JS lib for manipulatin websockets (includes fallback mech. & reconnecⁿs)

2 Handles disconnecⁿ & connecⁿ events

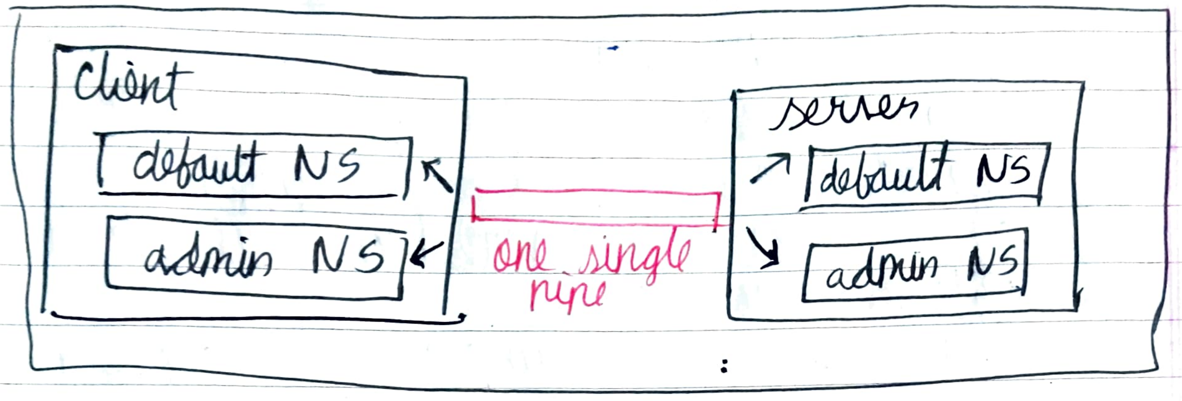
3 Namespaces and Room broadcastin
~~grp of clients~~ → grp of clients

namespace (NS) → comm channel that allows

WS = web socket

Date:
M T W T F S S

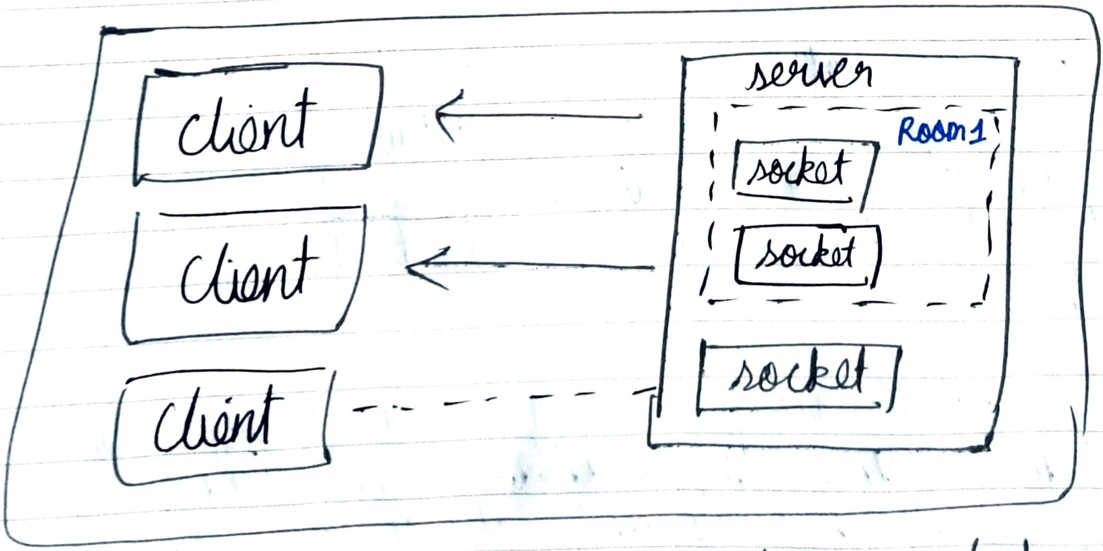
you to split the logic of your app over a single shared connec^n



In this eg :- you created an ~~[illegible]~~ admin NS that only auth.ed users have access to so the logic related to those users is separated from the rest of the app.

Rooms → Within each NS, you can define arbitrary channels a.k.a. "Rooms" that sockets can join or leave.

1 client, 1 socket.



→ this eg is what is used in grp chats

NOTE:- Synchronous servers don't general have support for WSs ~~[illegible]~~ (as they use blockin f's) so we use libs such as Eventlet, gevent to monkey patch (i.e. unblock these blockin f's) deployment of WS onto server