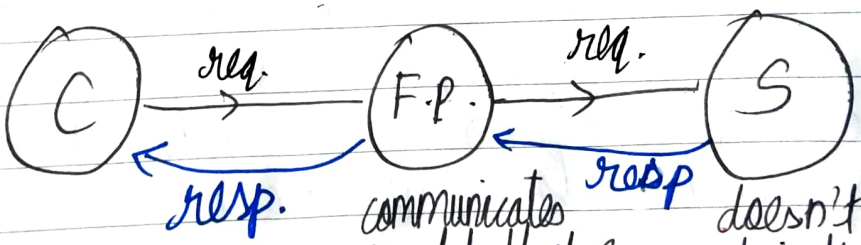
PROXIES

PROXY:
Just a machine-server set up to hide the IP address and other details of an interacting machine (eg:- client, server, etc.)
2 types (primary) :-
i) Foward proxy (usually ref-ed to as - 'proxy')
ii) Reverse proxy

Foward Proxy (F.P) and Reverse Proxy (R.P)

1] **Foward Proxy (F.P.)** → on client's team
i) It is a server located b/w client/set of clients & server/set of servers
ii) It acts on behalf on clients/set of clients by hiding clients IP address from server and instead supplying its own IP address.
iii) When a client sends a req. to the server and this F.P. has been properly confg-ed by client, the req. goes as



server has no idea abt the client and that an F.P exists

iv) F.P. serves as a way to hide the identity of client by issuing req. from server by changing source IP address sent in req. to its own IP address (i.e. IPA) instead of clients IPA eg:- VPN s

v) Some types of ~~proxy~~ F.P.s make client IPA visib. to server in some way, but, typically original source IPA is replaced

NOTE: VPN → a fwd proxy b/w client and server that hides the client's identity
∴ client can access servers restricted to it. Thus, client can, for eg, access a website not availa. in his/her country but avail. in other.

2] Reverse Proxy (R.P.) → on server's team.
i) A server b/w client / set of clients and server / set of servers
ii) When a client interacts with a server. eg:- sendin a req.

client sends req. client thinks its sendin req. to destina server (i.e. doesn't know abt R.P.

req → R.P. req → server
resp ← resp ←

Sends its IPA to client who thinks that this IPA is server's IPA

Client has no idea abt server or the fact that an R.P. exists in middle

eg:- say google.com sets up an R.P.

google.com —req + DNS query to get back IPA of Google server→ R.P

client's req → Google server

client browser ← returns its own IPA + resp. ← resp.

Use of R.P.s → Powerful tool for sys. des.

1) eg:- Config. R.P. to filter out reqs. you want your sys to ignore

2) eg:- Log stuff, gather metrics → can be done by R.P.

3) eg 3:- R.P. can also cache stuff (eg:- HTML page) Thus, server doesn't get bothered a lot

4) eg 4:- As a load balancer → a server that can distri req. load amongst a bunch of servers

NOTE: 1] ~~Malicious~~ Malicious client → sends a fuckload of reqs. to a server to bring it down.
Now R.P. acts as load balancer will distri these reqs across all servers, thereby safeguarding sys. from malicious clients, viruses etc.

2] NginX is a popular web server that can be used as an R.P.

eg:- Code on app page:

1) We set up an R.P. for any req. comin to port 8081 of our web service
2) Each time a req. is directed to the endpt='/' at port 8081, the req. header on the req. known as 'systemexpert-tutorial' and set it to 'true'
3) Then we gonna prwd this req. to the server that pts. to localhost:3000 (its name is 'nodejs-backend')