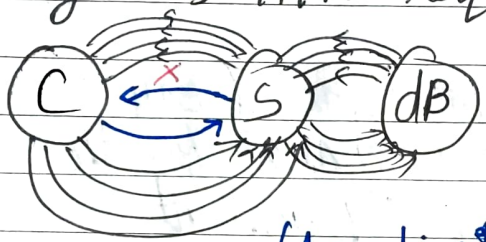
RATE LIMITING

Rate limiting:

1. Settin a threshold of sorts on certain opns. past which these opns return errs. Limitin the amt. of opns. that can be performed in given time

eg:- Only 5 HTTP req. from client every 10 secs.

In IDs



— : for opn 1, 2, 3, 4
— : for 5th opn

→ (1 machine * many reqs) = many reqs

2. Denial of Service (DoS) attack ⇒ Malicious client floods the sys with too many req and effectively clogs the sys with way more traffic that it can handle and thus, sys is brought down

3. Rate limitin prevents cloggin of servers by DoS attacks as past a certain no. of reqs, it'll throw errs. and stop takin in reqs.

eg:- code exec^n engine on AlgoExp. Run code op^n is rate ltd. to prev. someone from spammin the op^n i.e. cloggin the dB with reqs.

4. Rate limiting can also be dependent on other factors:

i) Based on user: Identify user issuin req.

Date:
M T W T F S S

by lookin at req. header or authentica^{n} credential and ~~we~~ once identified we can rate limit no of opns. by user eg:- 3 opns/min for a student to access his/her grades on insti. website

ii) Based on IP address
iii) Based on region of the world
iv) Based on whole sys. eg:- a sys as a whole should never allow servers to handle $> 10^4$ reqs per min.
v) Based on a certain org. fallin under same IP address.

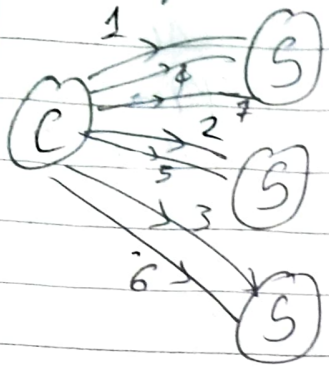
(5) DDoS attack → Distributed DoS attack

(many machines so diff IP address) * (some reqs by each machine) = (many reqs to servers)

might not be identifiable as part of same org/grp.

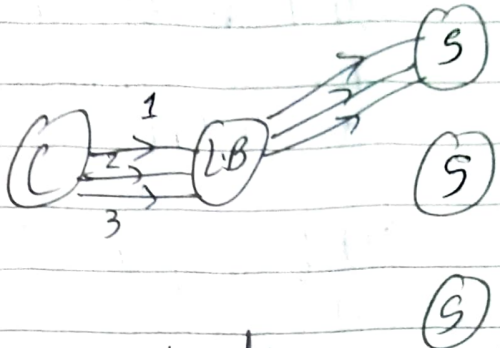
(6) In large-scale distri sys → we gotta hv. v. powerful rate limiting ~~to prot~~ → i.e. we need to hv really powerful load balancin to ensure that a client's req are always routed to the same server (amongst a set of servers cuz its a big sys) to ensure that we have some prev access records of this clients reqs. to the sys, stored in-memory in the same server place so we can compare and say whether ant. of reqs. this client is issuin is within a safe boundary or not

W/o L.B. in



Client can flood all servers as rate lim. is met/server but not across servers (sys as a whole) so sys is flooded with traffic

With L.B

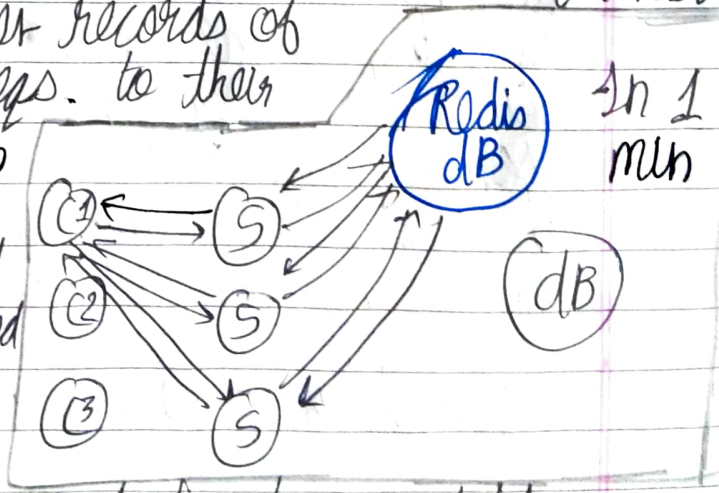


Client always routed to same server → it has prev access records of this client so it'll rate lim. client if it's floodin with too many reqs

∴ It is imp. to config load balancing keepin rate lim. in in mind

(7) ∴ In most cases, large scale distri. sys's use handle rate lim. in in a separated service or dB instead of in-mem. in servers all your servers speak to this separate dB and realise if they gotta rate lim. client or not

which stores prev records of all clients' reqs. to their resp. servers

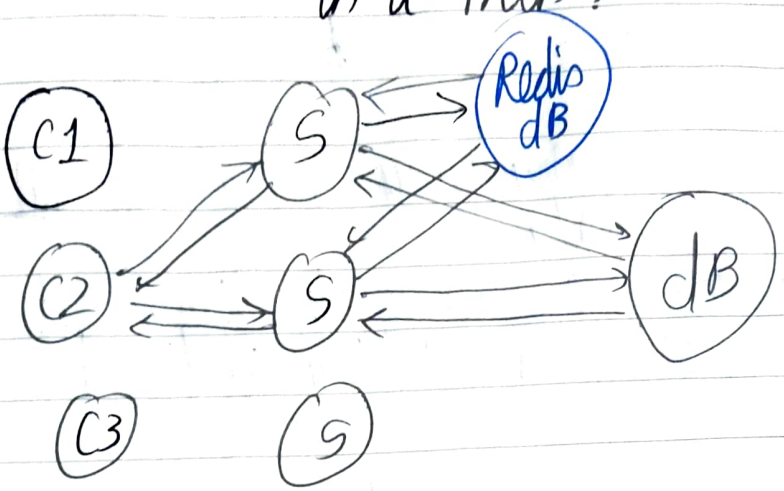


eg! - Redis. → eg 3 req/min allowed

↓ separate dB that stores prev client access records and does rate lim logic and likewise, tells a server to rate lim a client or not

on its 3rd req.

In our eg:- C1 is blocked / rate ltd from all our servers as it issued >2 reqs. in a min.



C2 has ≤ 2 reqs per min so it get backs data both times from main dB

(8) Tier based Rate Limiting

eg - HITP reqs:

∴ req./0.5s but only 3 reqs/10sec but only 10 reqs/min

Thus, we ~~dont~~ give users bit more freedom whilst still maintaining a strong grasp over their power to keep req. in

Complex to code this out tho