

### Rate limiting



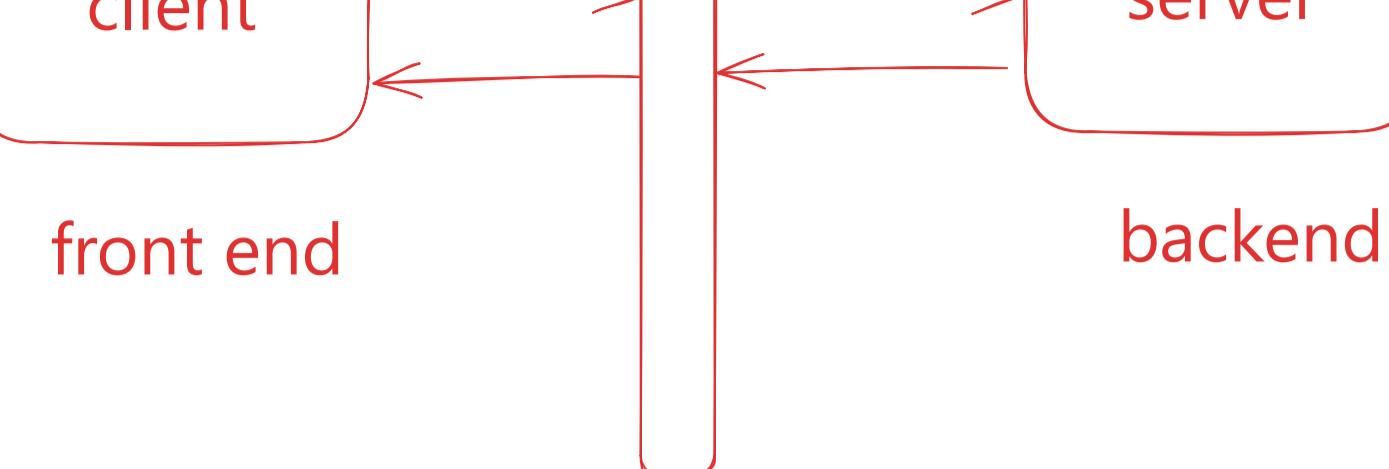
problems:

1. Client can send too many requests within a timeframe.

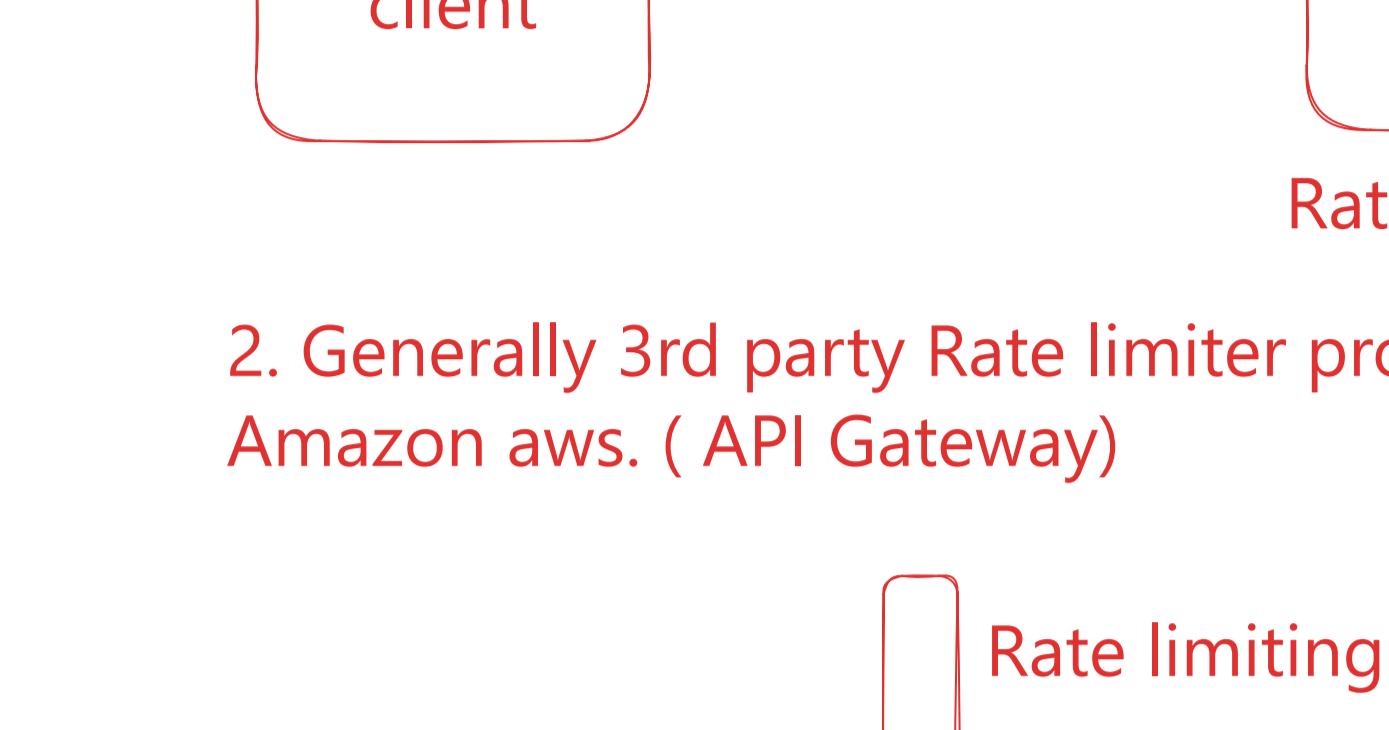
DDOS/DOS

2. Server cost expensive

Each req is expensive.

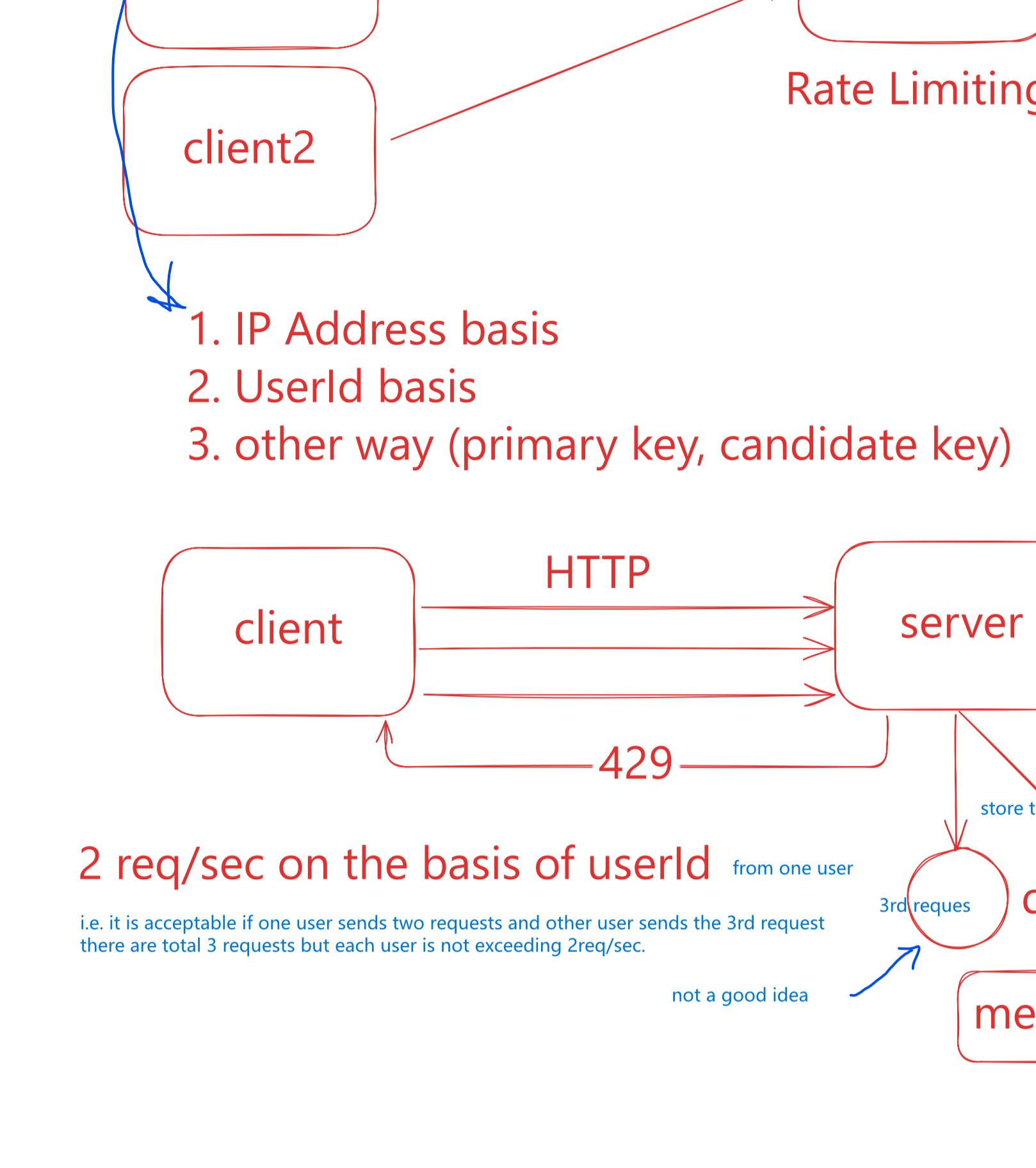


How many req client can send in any timeframe



Rate limiting can be implemented by

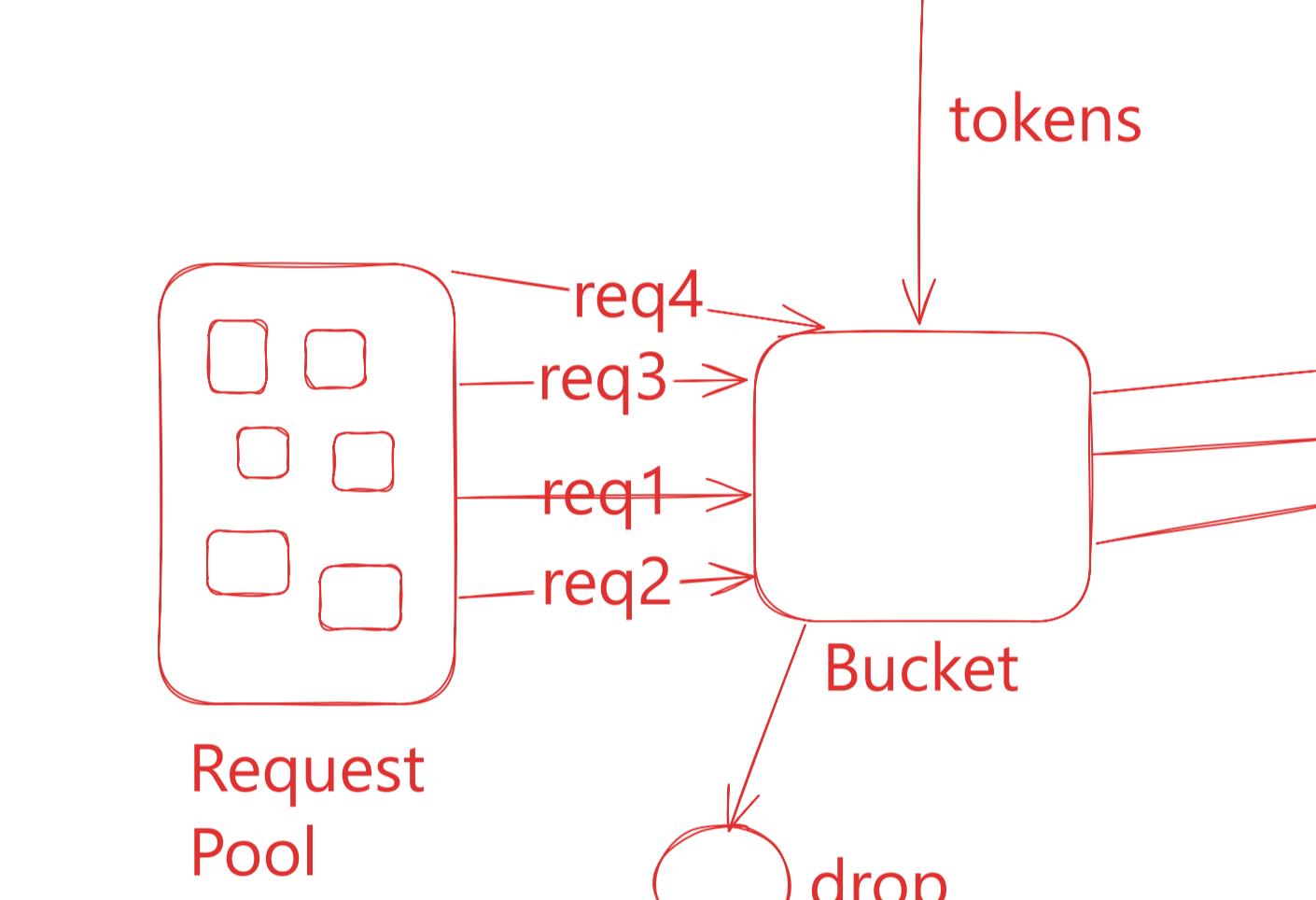
1. Client side : worst
2. Server side
3. Middleware



Application use case:

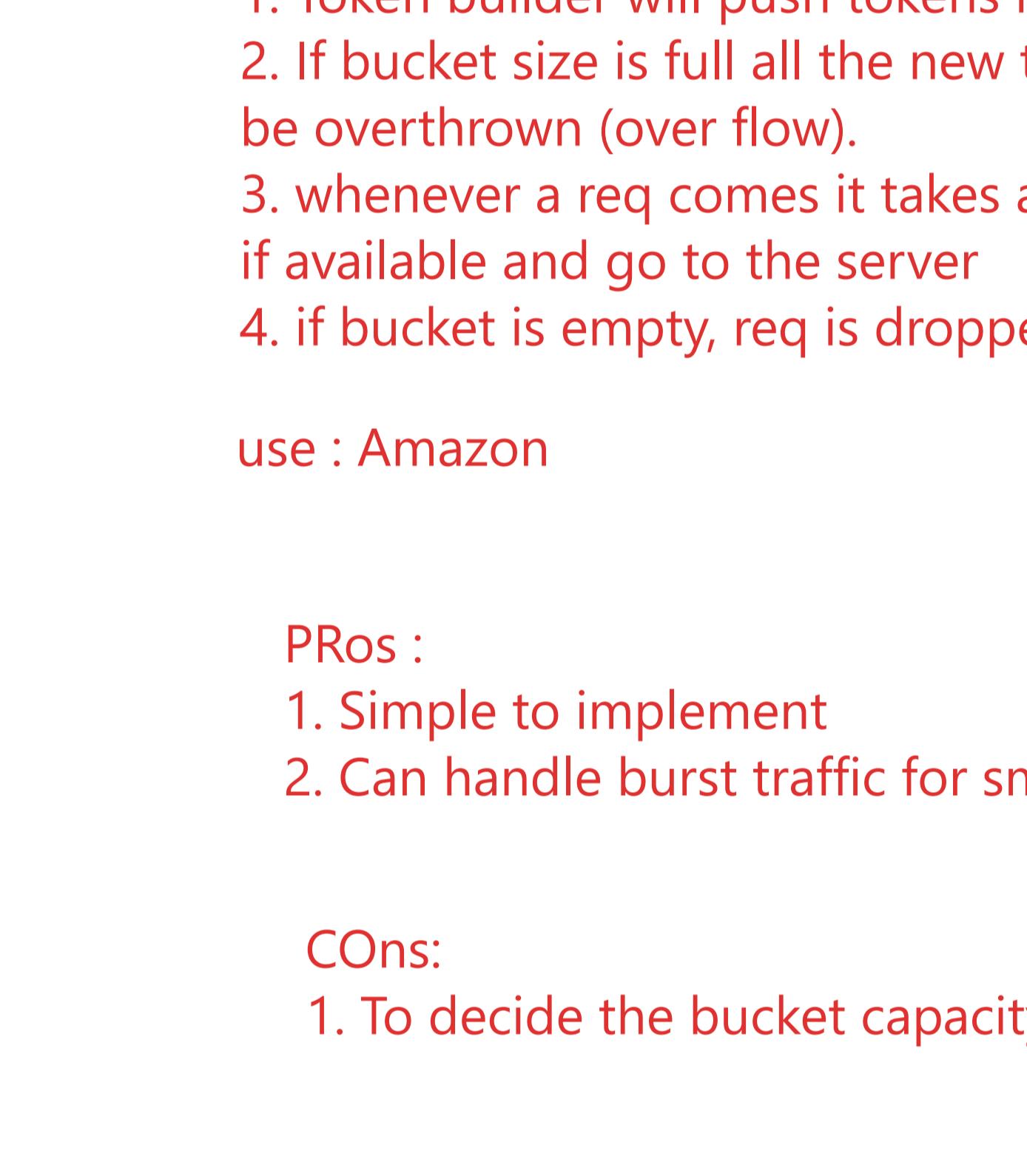
1. Server side language(Java, C++, javascript, Python) should be fast

Server side

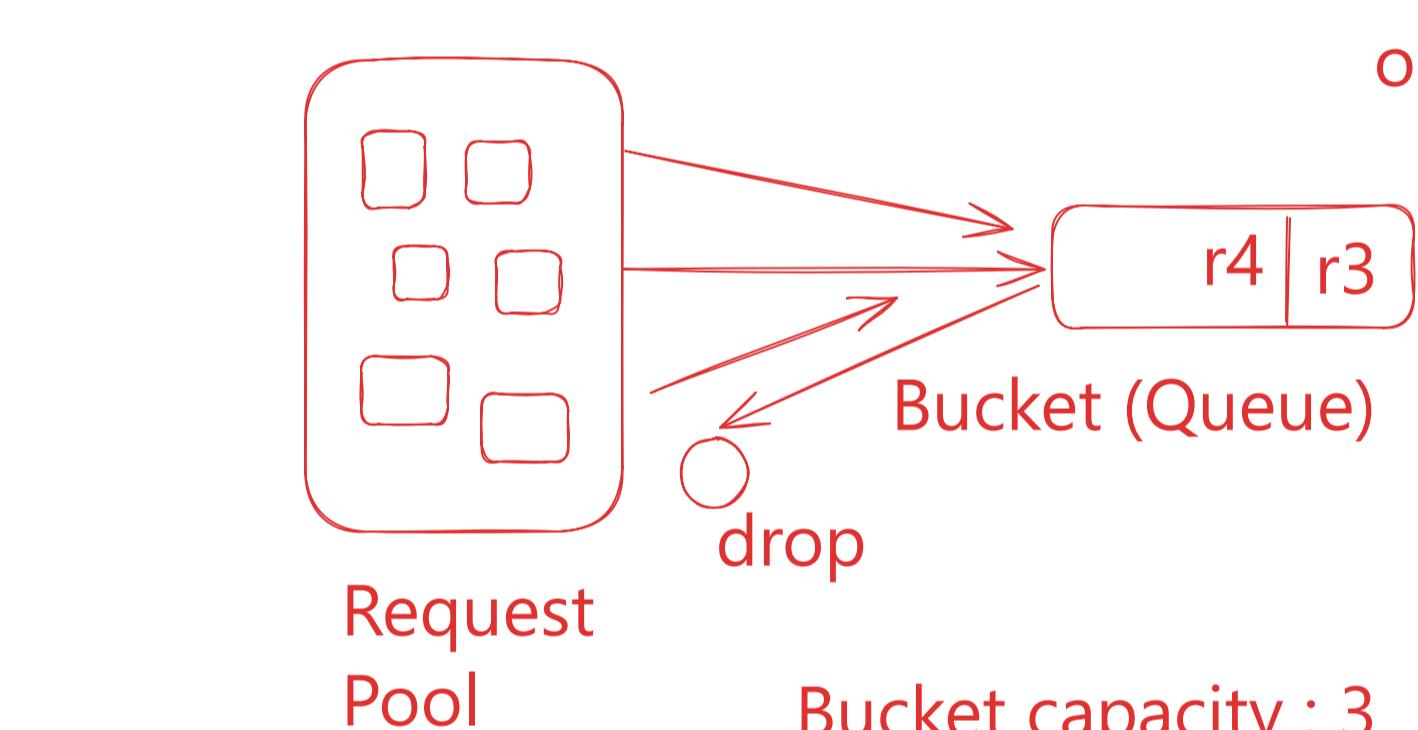


2. Generally 3rd party Rate limiter providers.

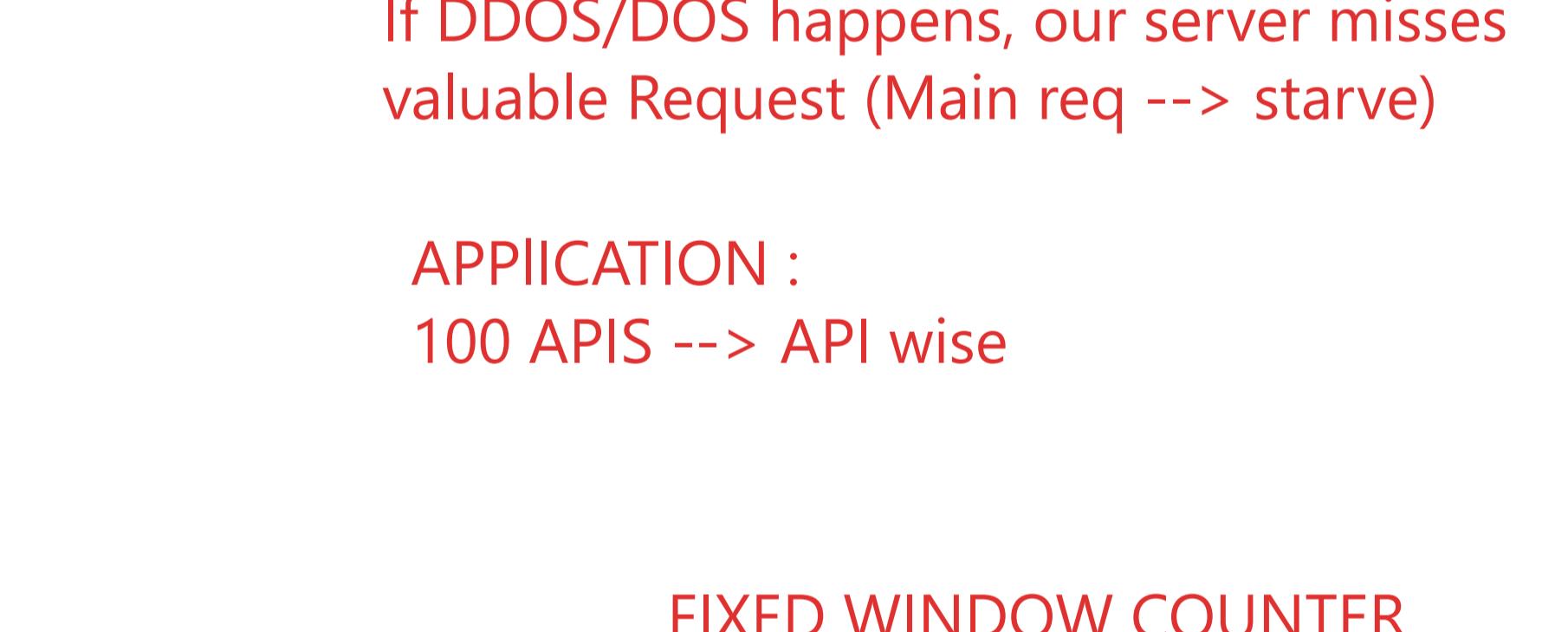
Amazon aws. ( API Gateway)



Rate limiting on What ??



1. IP Address basis
2. Userid basis
3. other way (primary key, candidate key)



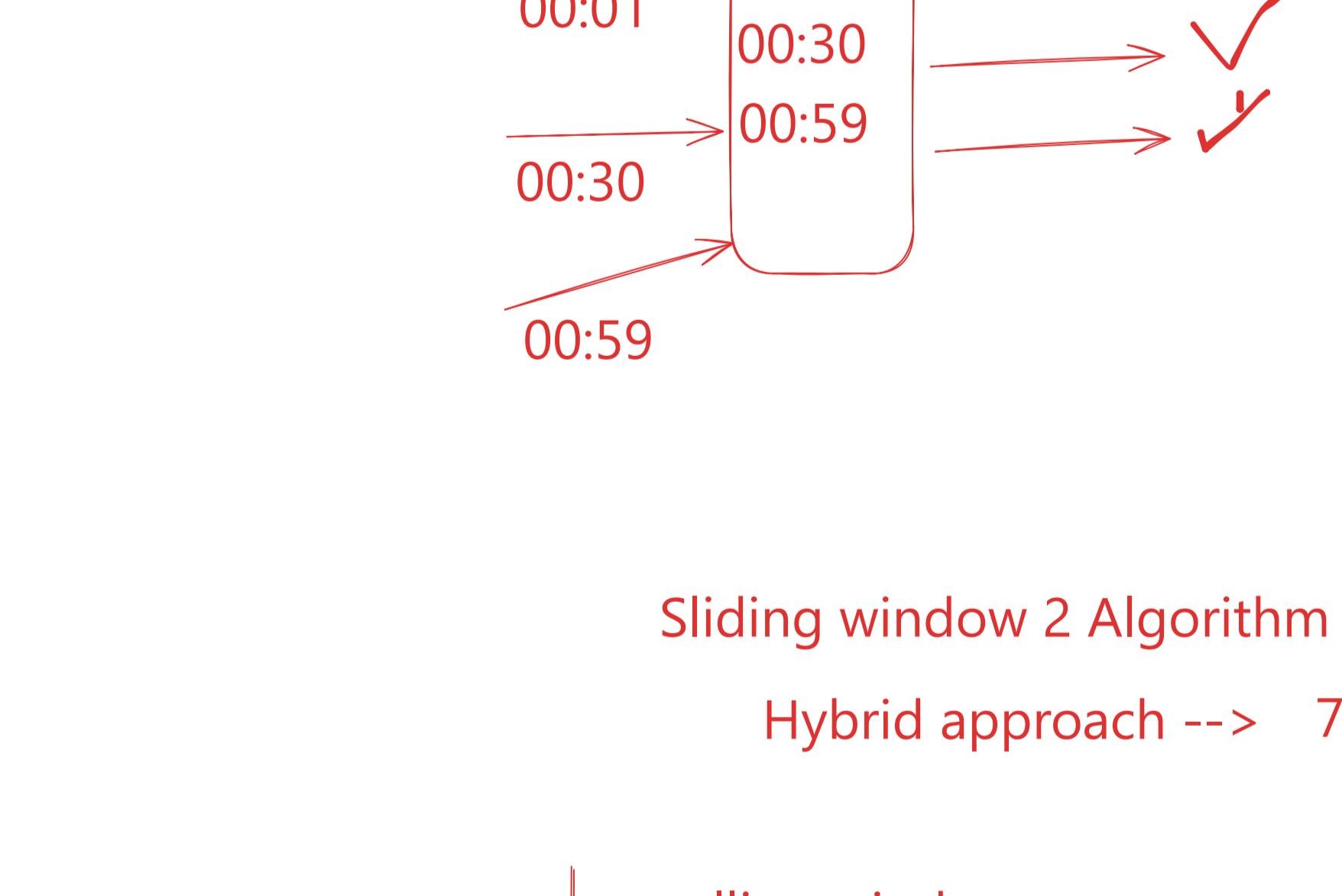
Response codes:  
200 : Ok  
429 : Too many Requests

Headers:  
X-api-limit : 2  
x-duration : seconds

Algorithms of Rate Limiting

1. Token Bucket
2. Leaky Bucket
3. Fixed window counter
4. Sliding window log
5. sliding window counter

Token Bucket Algorithm



1. Bucket size : 20
2. Inflow : 10 tokens/sec

1. Token builder will push tokens in fixed intervals
2. If bucket size is full all the new tokens will be overwritten (over flow).
3. whenever a req comes it takes a token from bucket if available and go to the server
4. if bucket is empty, req is dropped.

use : Amazon

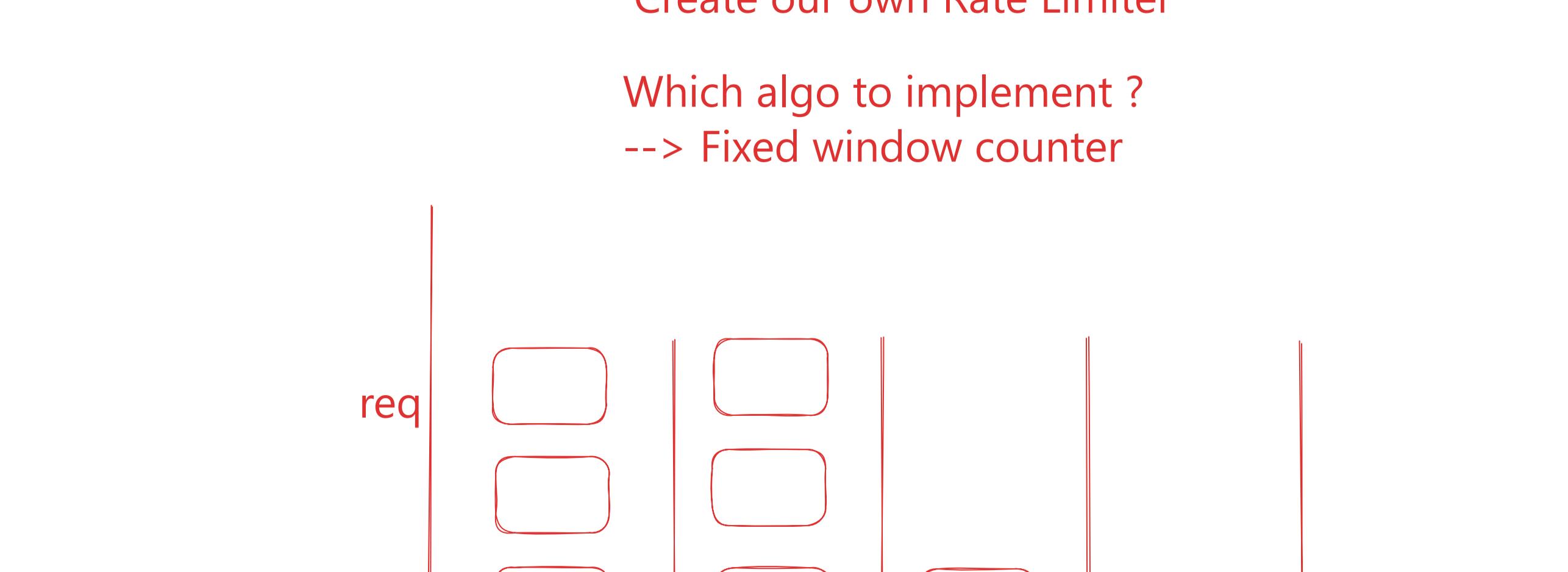
PROs :

1. Simple to implement
2. Can handle burst traffic for small duration

COs:

1. To decide the bucket capacity and inflow

Leaky Bucket (Global Rate limiting)



App : 1000 users

IP level : every unique limit

Pros :

Simple to implement

Even though burst traffic is there, our server will not crash

Cons:

If DDOS/DOS happens, our server misses valuable Request (Main req --> starve)

APPLICATION :

100 APIs --> API wise

FIXED WINDOW COUNTER

1. In a fixed interval (Say a sec, or a minute) Only a specific no. of req can come.



1. Bucket size : 20
2. Inflow : 10 tokens/sec

1. If bucket size is full all the new tokens will be overwritten (over flow).
2. whenever a req comes it takes a token from bucket if available and go to the server
3. if bucket is empty, req is dropped.

use : Amazon

Leaky Bucket (Global Rate limiting)



App : 1000 users

IP level : every unique limit

Pros :

Simple to implement

Even though burst traffic is there, our server will not crash

Cons:

If DDOS/DOS happens, our server misses valuable Request (Main req --> starve)

APPLICATION :

100 APIs --> API wise

Sliding window Log Algorithm

Best Algorithm for Rate Limiting

Strict

Slow & memory consuming

Algorithm:

1. It will store each req in a log file.
2. Whenever a req comes, it will first remove all the outdated req from the file.
3. Then it will log the new req and check, if the counter limit reached, drop the req else send it to server.

3 req/sec



Sliding window 2 Algorithm

Hybrid approach --> 7 req/ minute



Formula : No. of req in curr interval

$$+ \frac{\text{No. of req in prev interval}}{70\%}$$

\*

70%

$$= 4 + 5 * 0.7 = 7.5$$

Create our own Rate Limiter

Which algo to implement ?

--> Fixed window counter



Excepted Counter = 3

current counter = 3

Db ? SQL / NoSQL

Db : It is slow.

Cache : In-Memory Cache : Redis

