Overload Control for µs-scale RPCs with Breakwater

Inho Cho, Ahmed Saeed, Joshua Fried,
Seo Jin Park, Mohammad Alizadeh, Adam Belay

CSAIL

Trend: µs-scale RPCs

2010

Storage: SATA SSD (~ 90 us)
Network: ~ 100 us

→

2020

Storage: M.2 NVMe SSD (~ 20 us)
Network: ~ 5 us

Trend: µs-scale RPCs

2010

Storage: SATA SSD (~ 90 us)
Network: ~ 100 us

× 1/4 →

2020

Storage: M.2 NVMe SSD (~ 20 us)
Network: ~ 5 us

Trend: µs-scale RPCs

2010

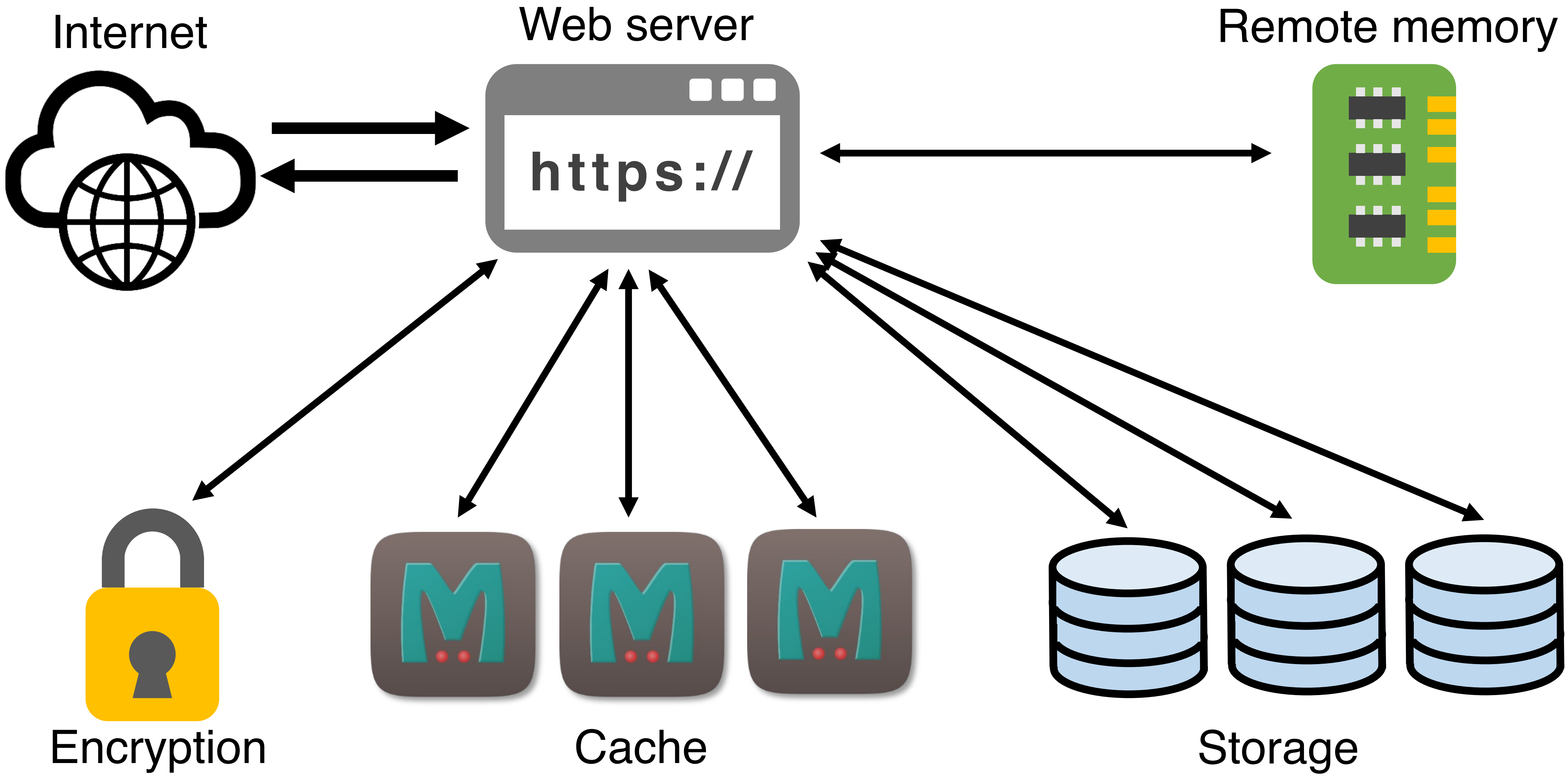
Storage: SATA SSD (~ 90 us)
Network: ~ 100 us

$\xrightarrow[\times\ 1/20]{\times\ 1/4}$

2020

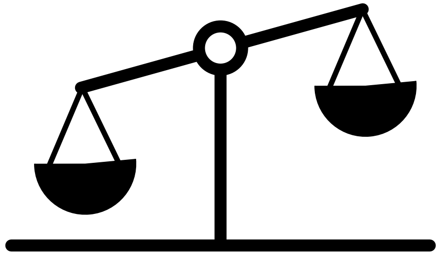
Storage: M.2 NVMe SSD (~ 20 us)
Network: ~ 5 us

Trend: µs-scale SLOs

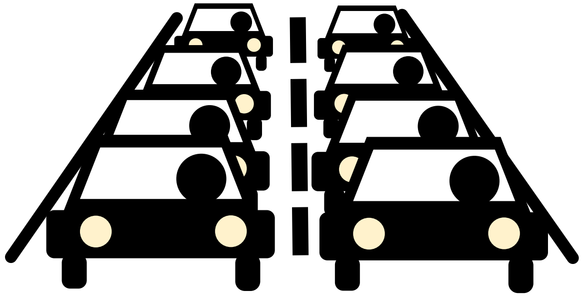


Server Overload

Load Imbalance

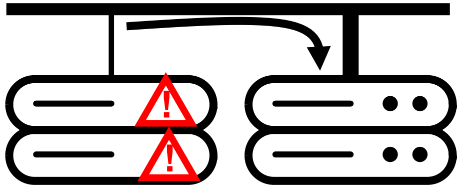


Unexpected user traffic

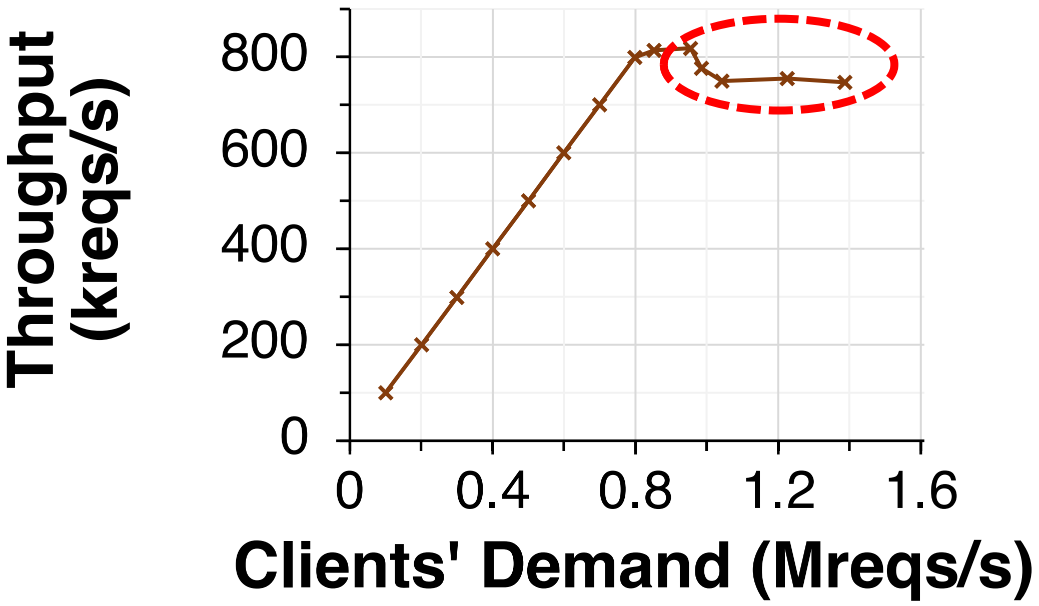


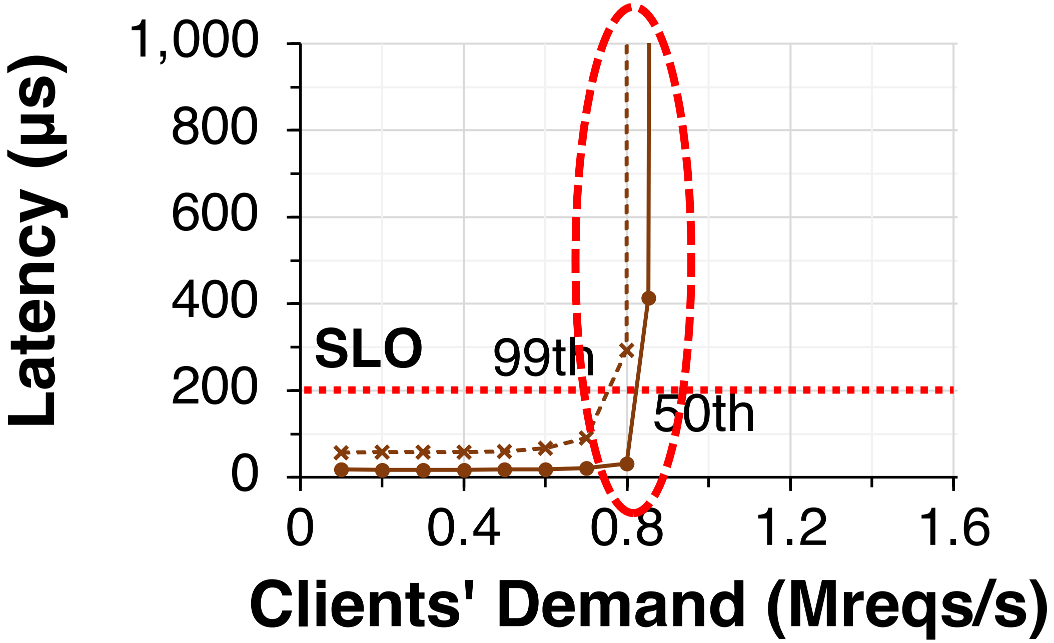
Packet bursts

Redirected traffic due to failure



Server Overload

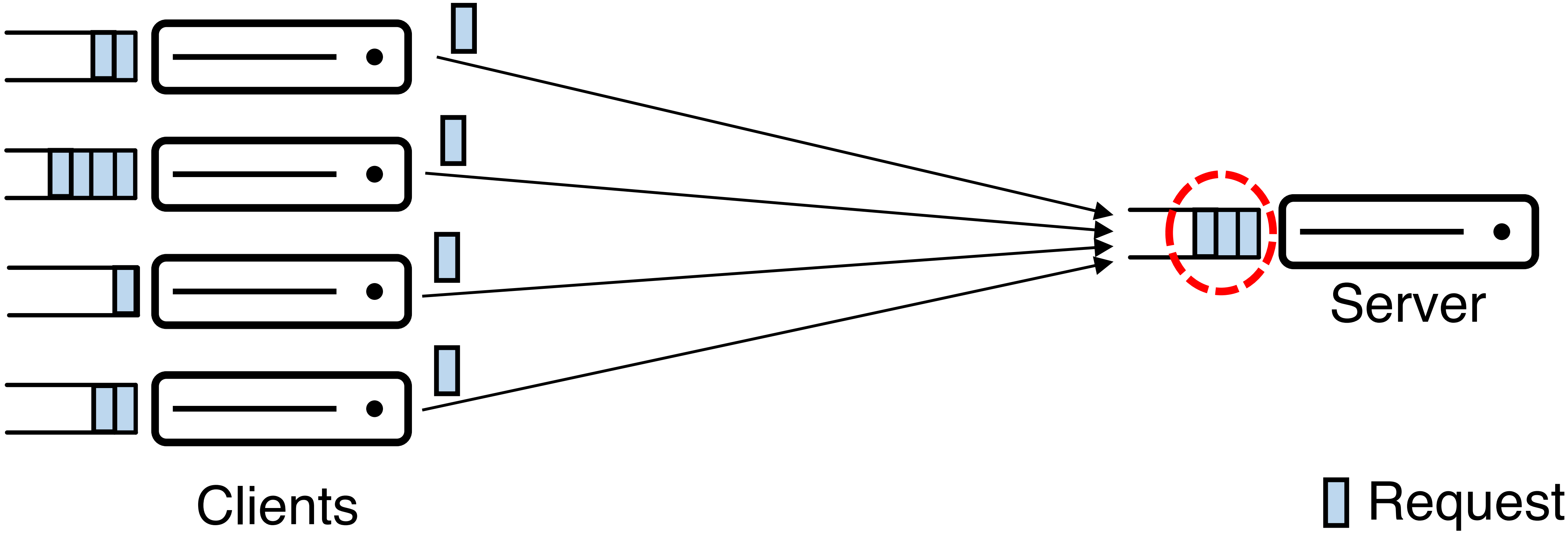




Without overload control, server overload makes almost all requests violate its SLO.

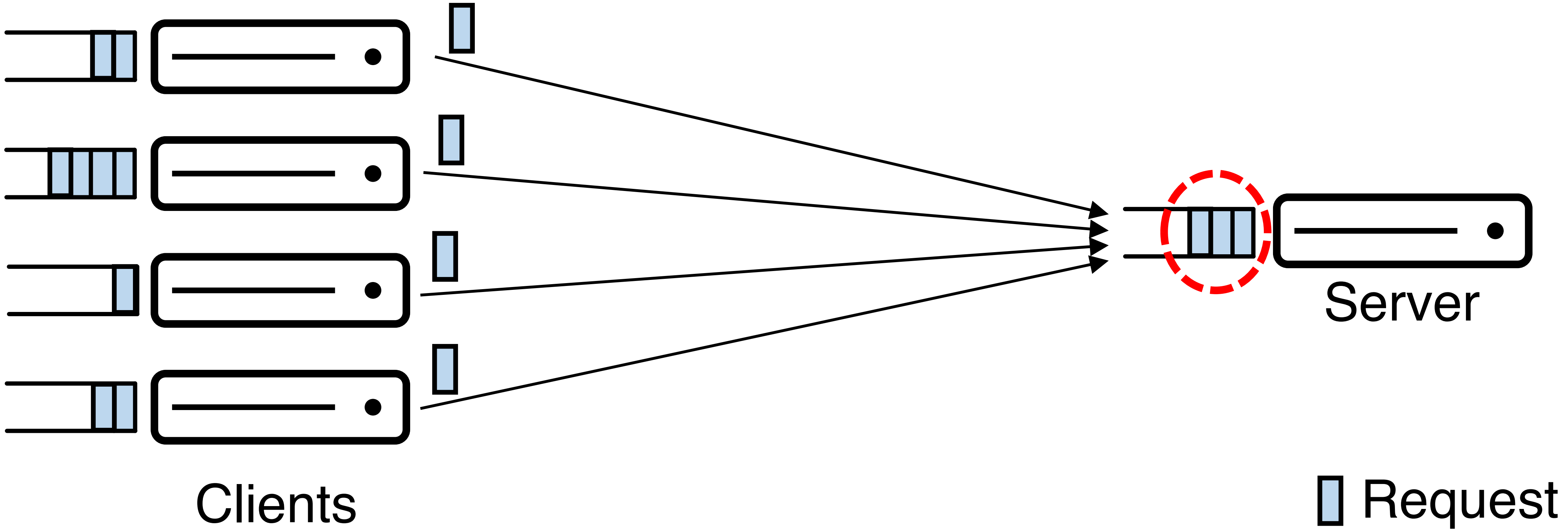
Ideal Overload Control

Should keep request **short**, but **not empty**



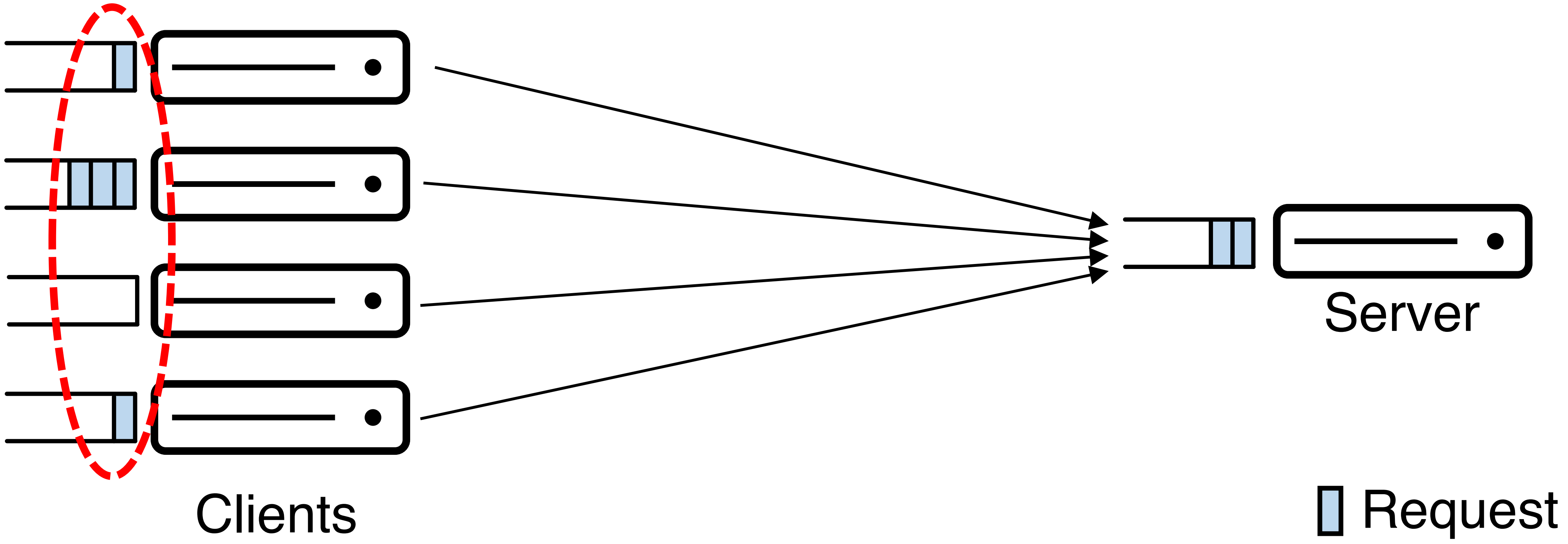
Ideal Overload Control

Should keep request **short**, but **not empty**

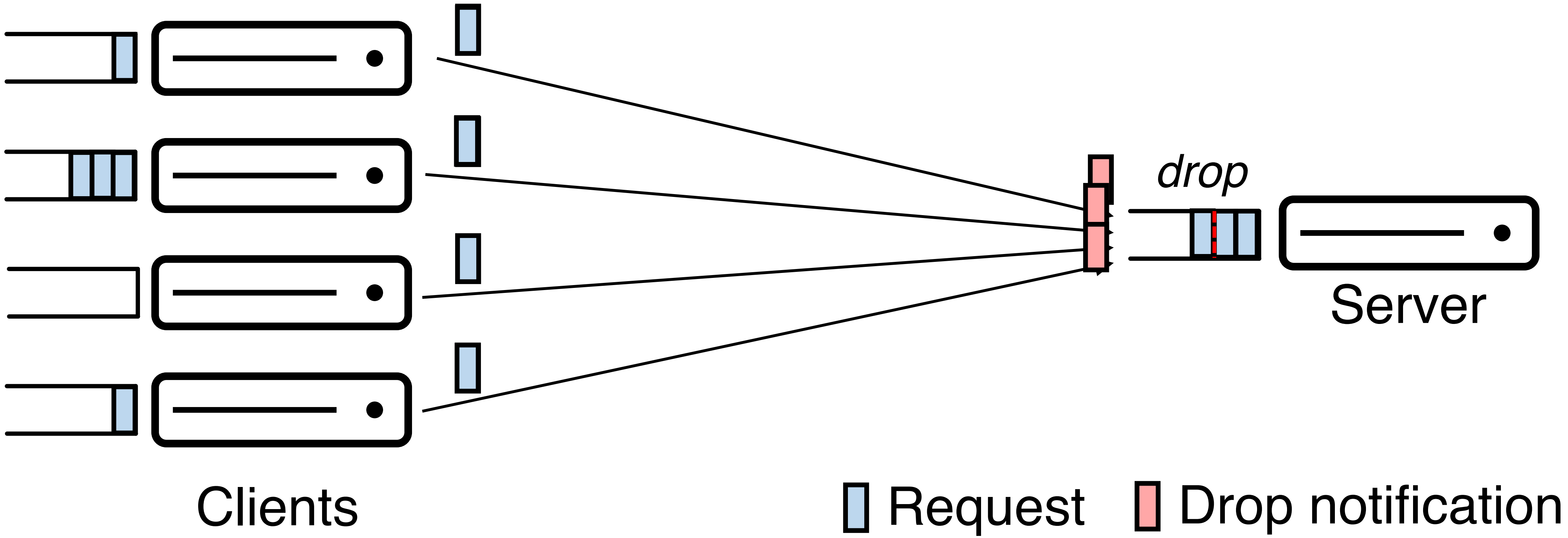


Ideal Overload Control

Should keep request **short**, but **not empty**

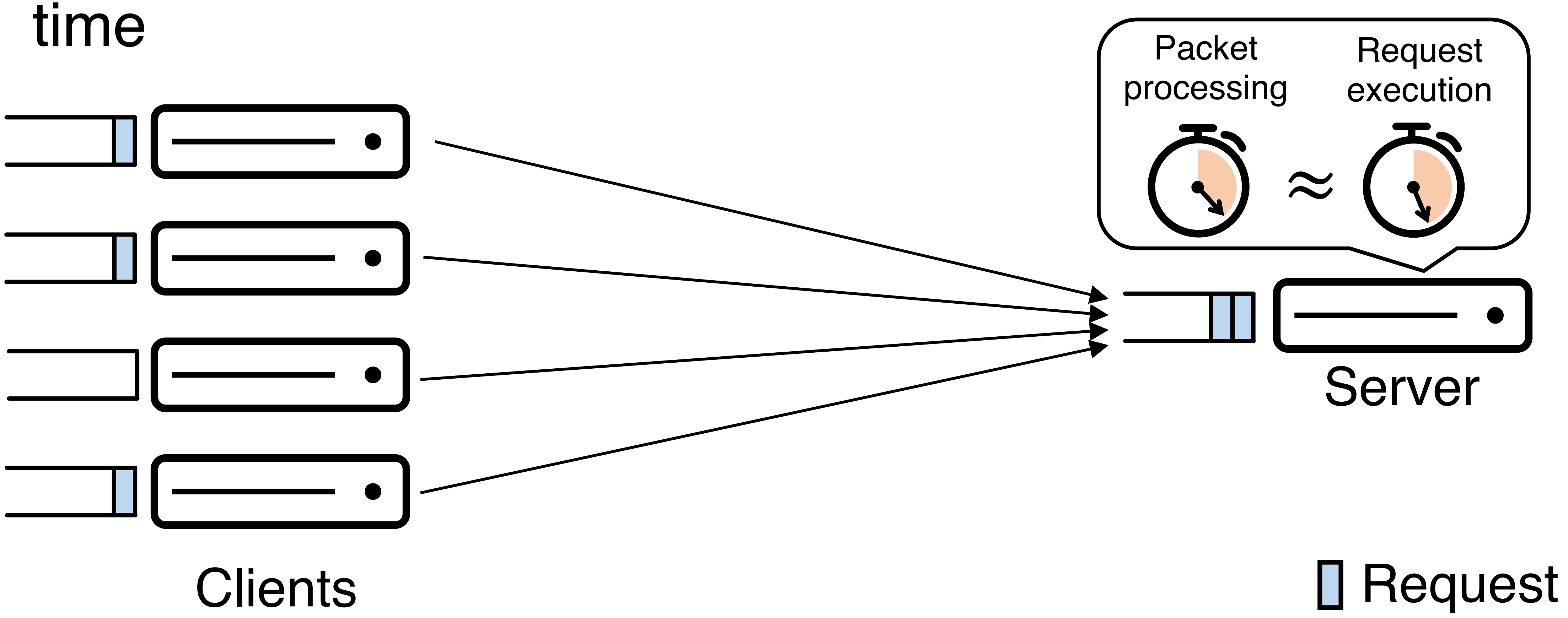


Strawman #1: Server-side AQM

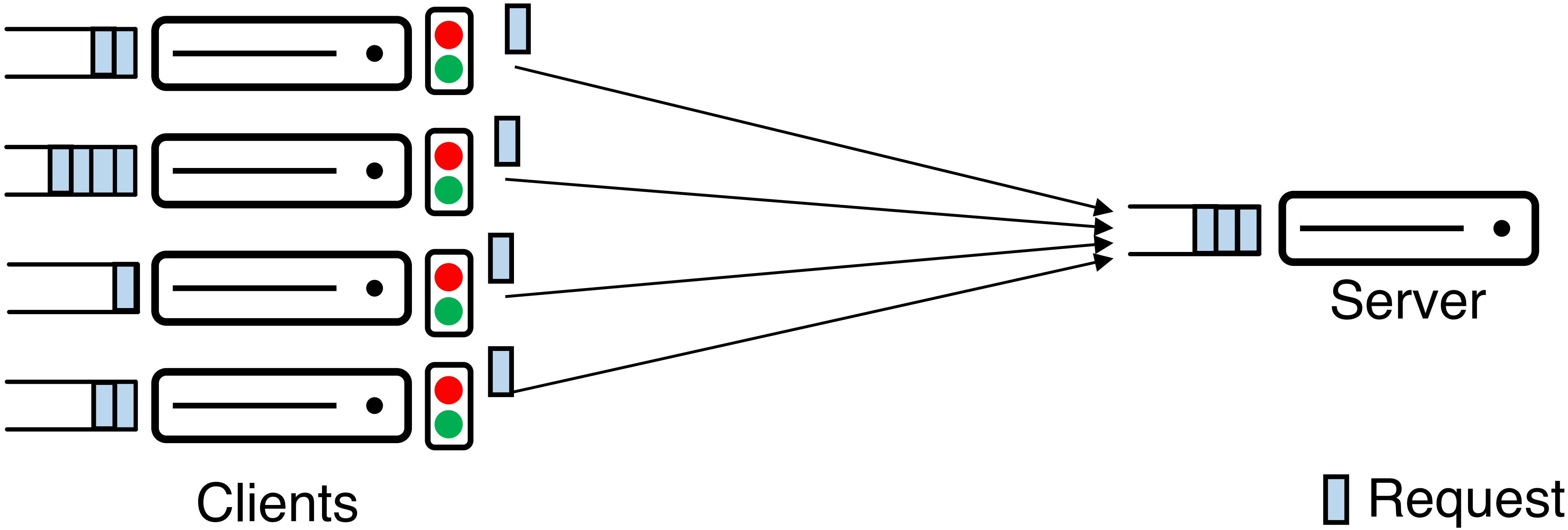


Strawman #1: Server-side AQM

Cost of packet processing is comparable to the service time

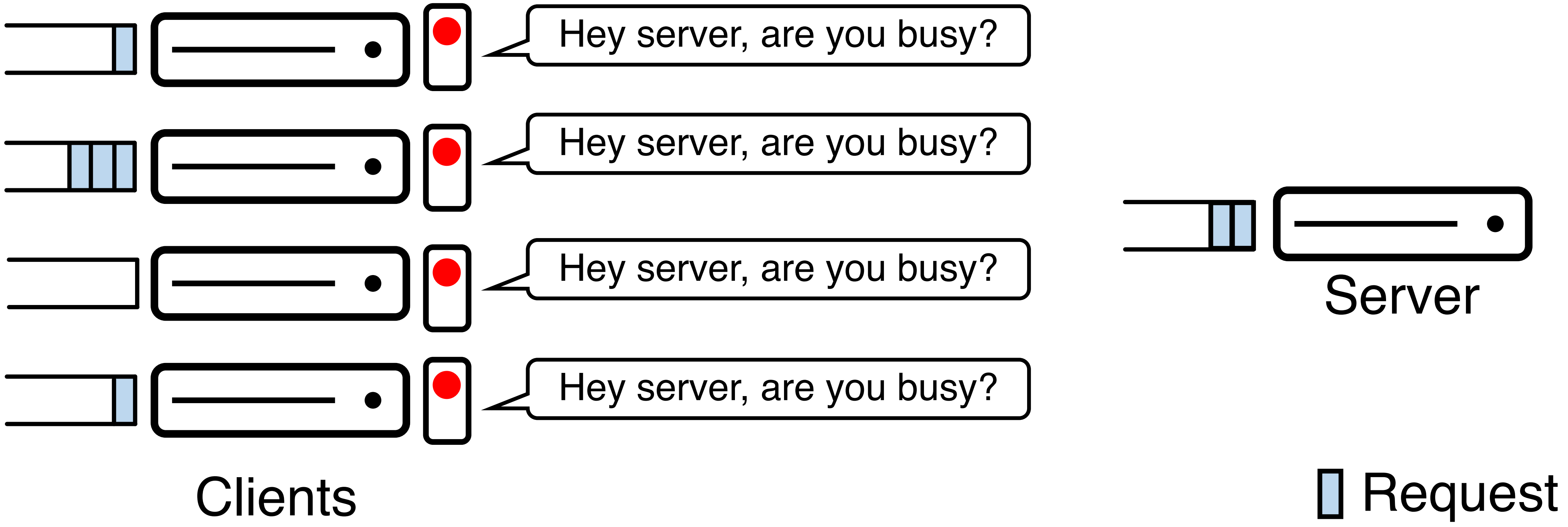


Strawman #2: Client Rate limiting



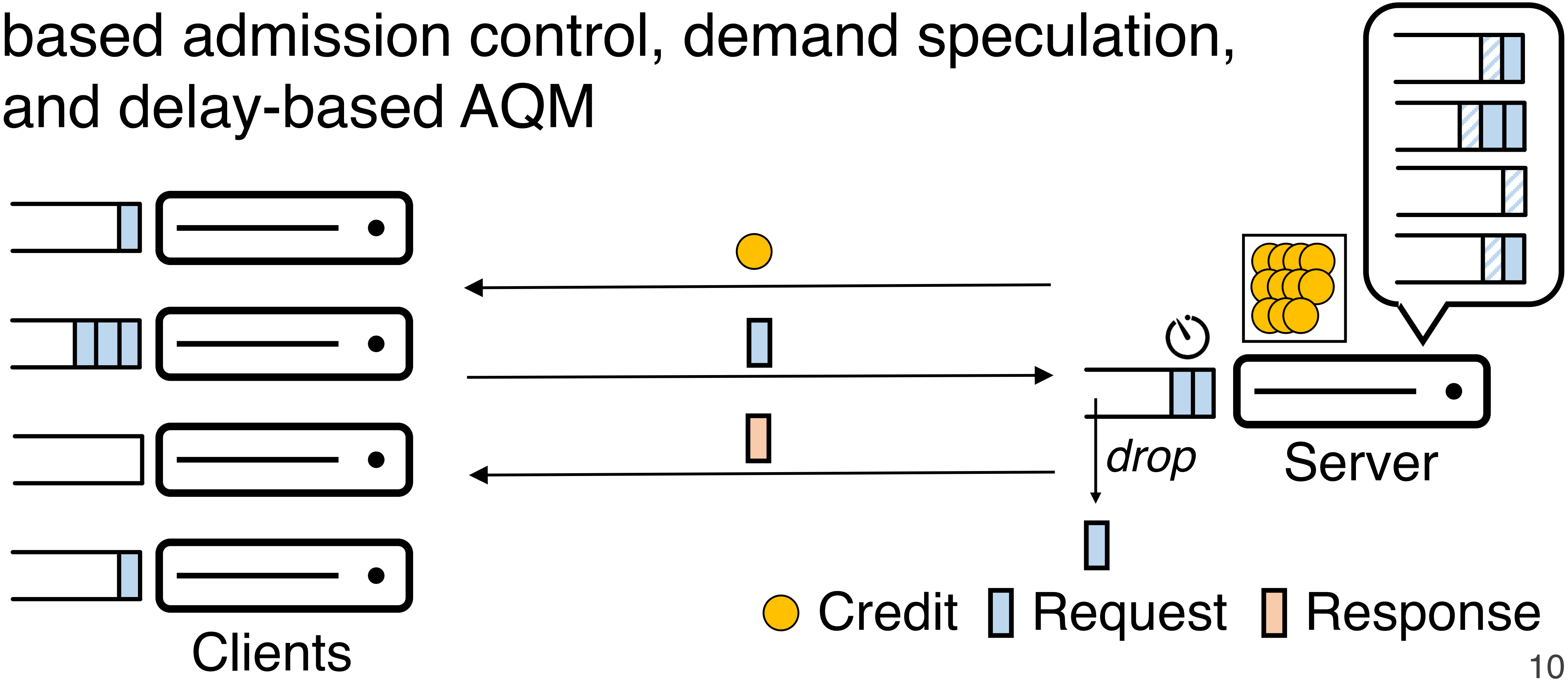
Strawman #2: Client Rate limiting

Cost of packet processing is comparable to the service time


Hey server, are you busy?
Hey server, are you busy?
Hey server, are you busy?
Hey server, are you busy?
Server
Clients
▯ Request

Breakwater

Overload control for µs-scale RPCs with credit-based admission control, demand speculation, and delay-based AQM



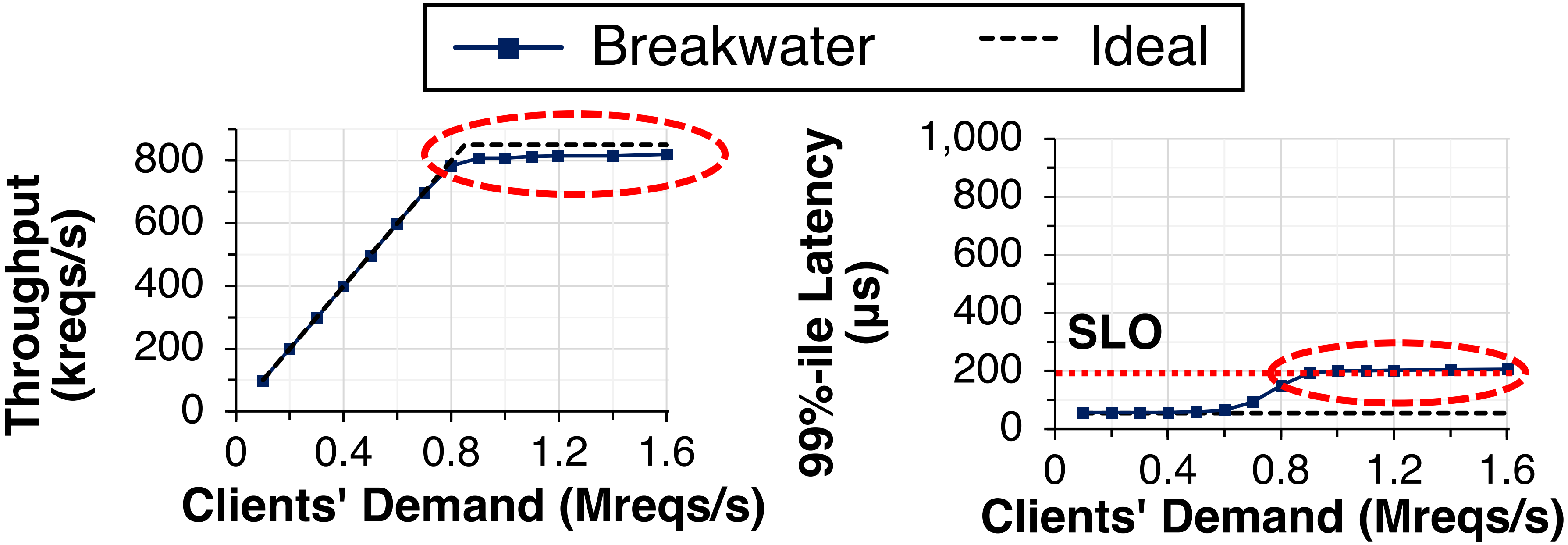
Breakwater

(1) High throughput

(2) Low and bounded tail latency

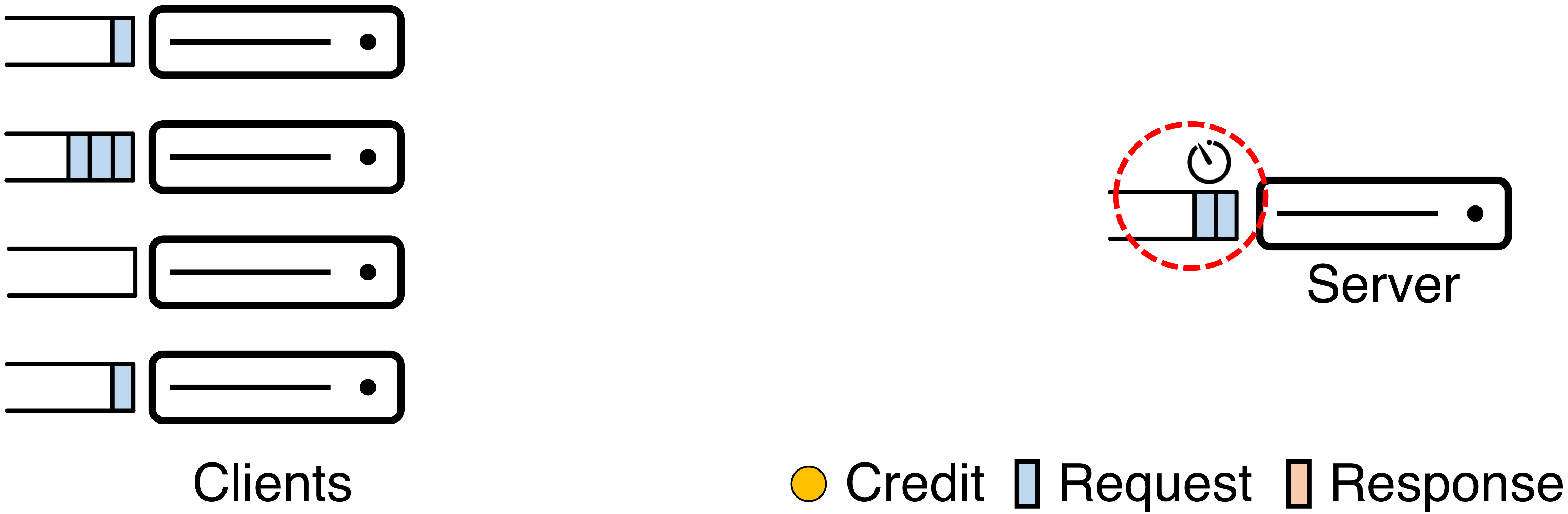
(3) Fast feedback for the rejected requests

(4) Scalability to a large number of clients



Queueing delay as congestion signal

Breakwater uses request queueing delay as a congestion signal



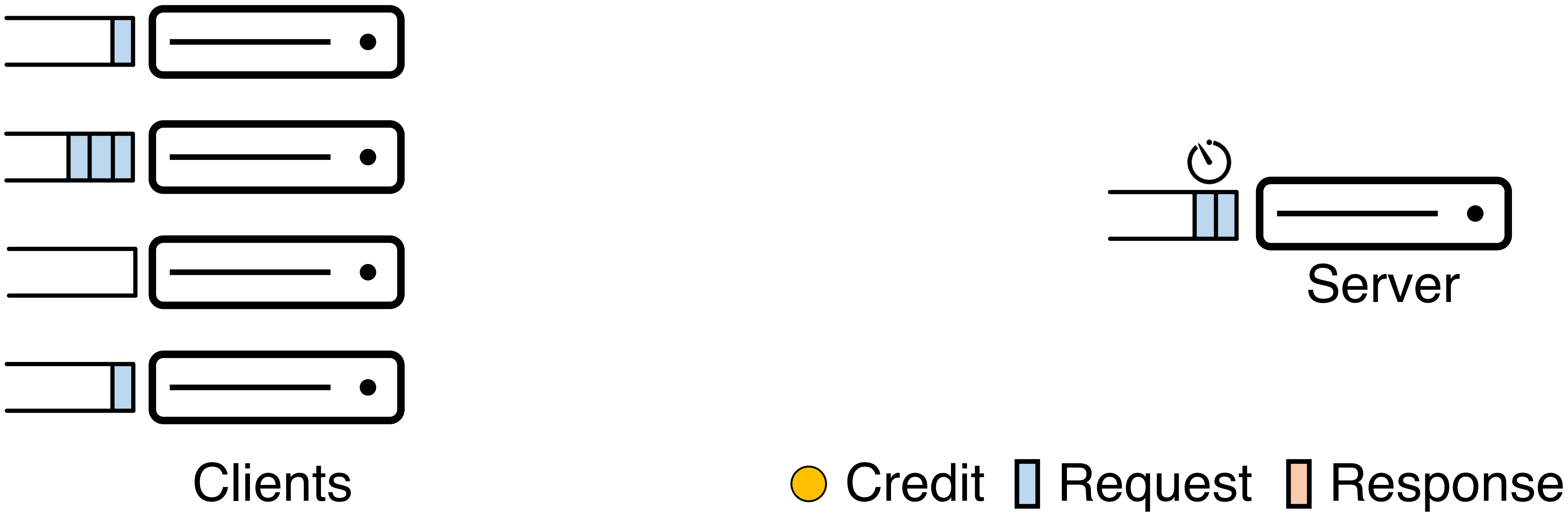
Server

Clients

● Credit ▯ Request ▯ Response

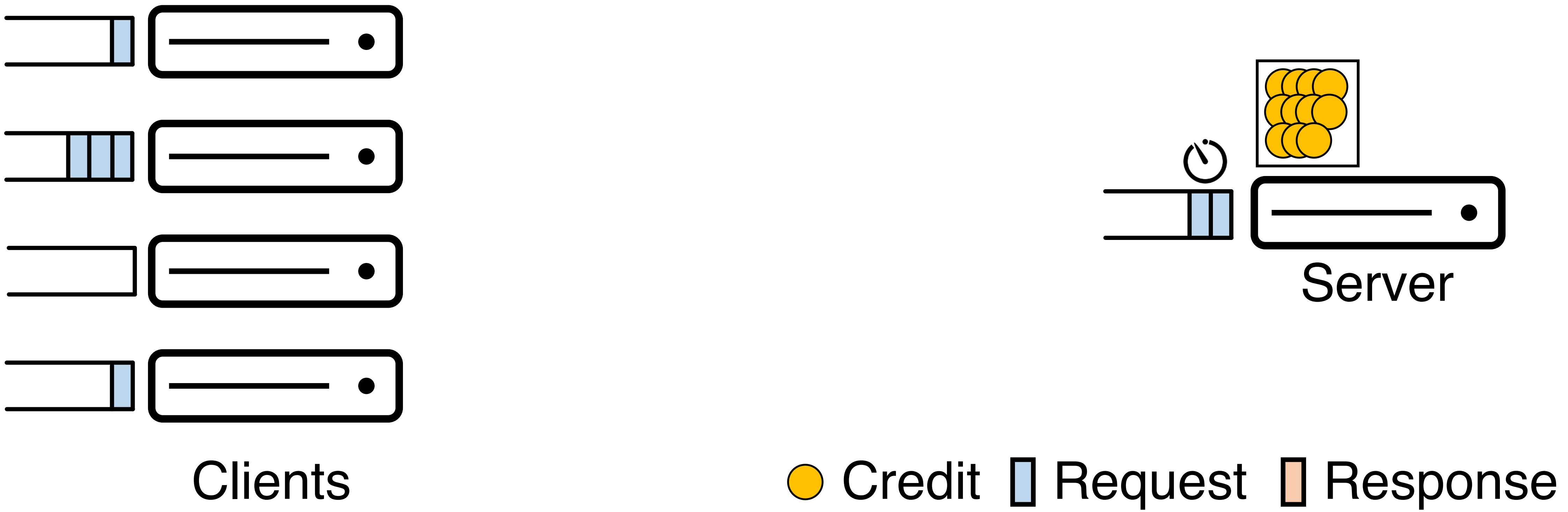
Credit-based admission control

Breakwater controls amount of incoming requests with credits



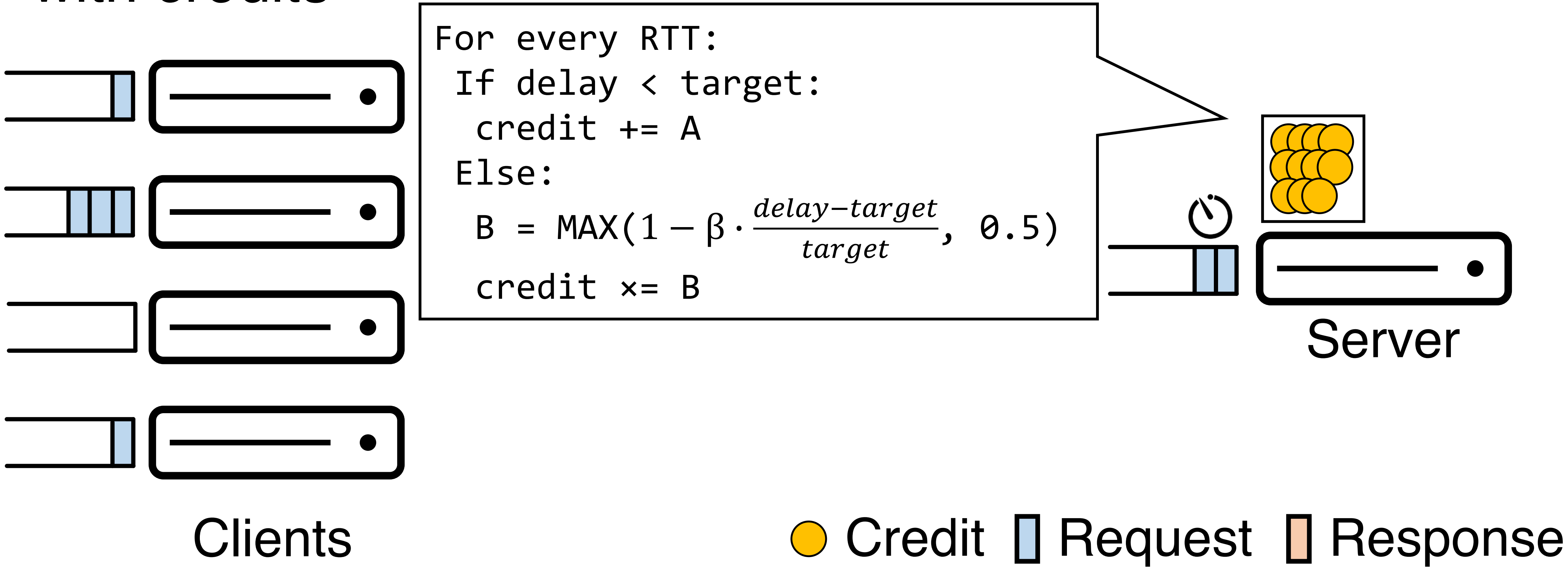
Credit-based admission control

Breakwater controls amount of incoming requests with credits



Credit-based admission control

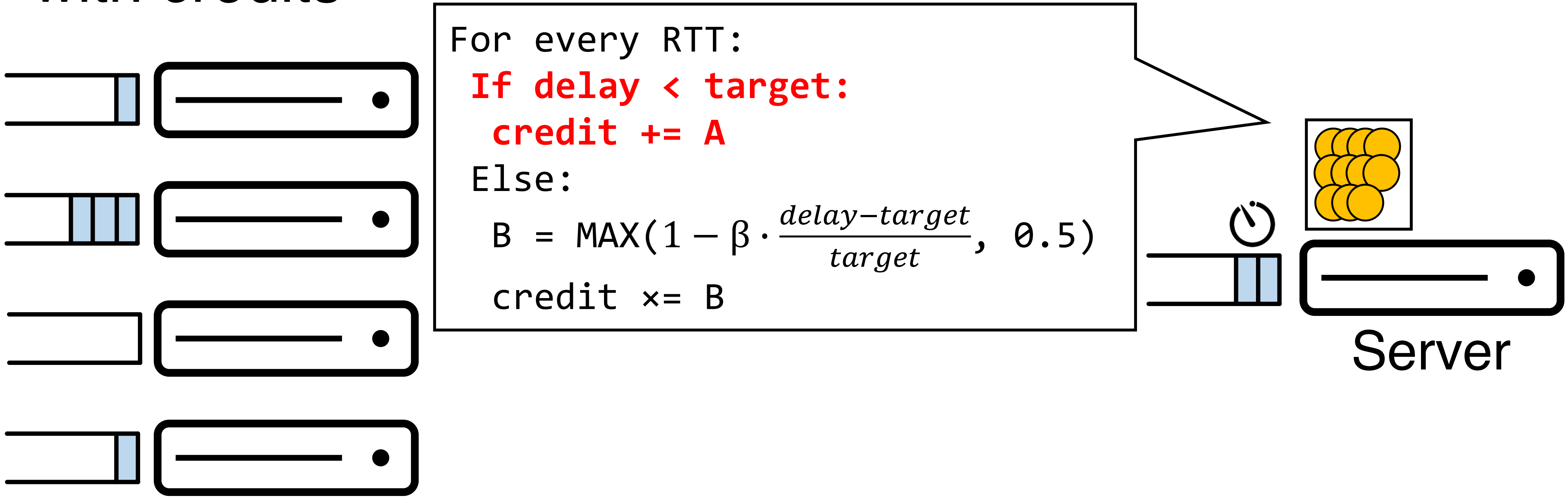
Breakwater controls amount of incoming requests with credits



```
For every RTT:
 If delay < target:
  credit += A
 Else:
  B = MAX($1 - \beta \cdot \frac{delay - target}{target}$, 0.5)
  credit ×= B
```

Credit-based admission control

Breakwater controls amount of incoming requests with credits



```
For every RTT:
 If delay < target:
  credit += A
 Else:
  B = MAX($1 - \beta \cdot \frac{delay-target}{target}$, 0.5)
  credit ×= B
```

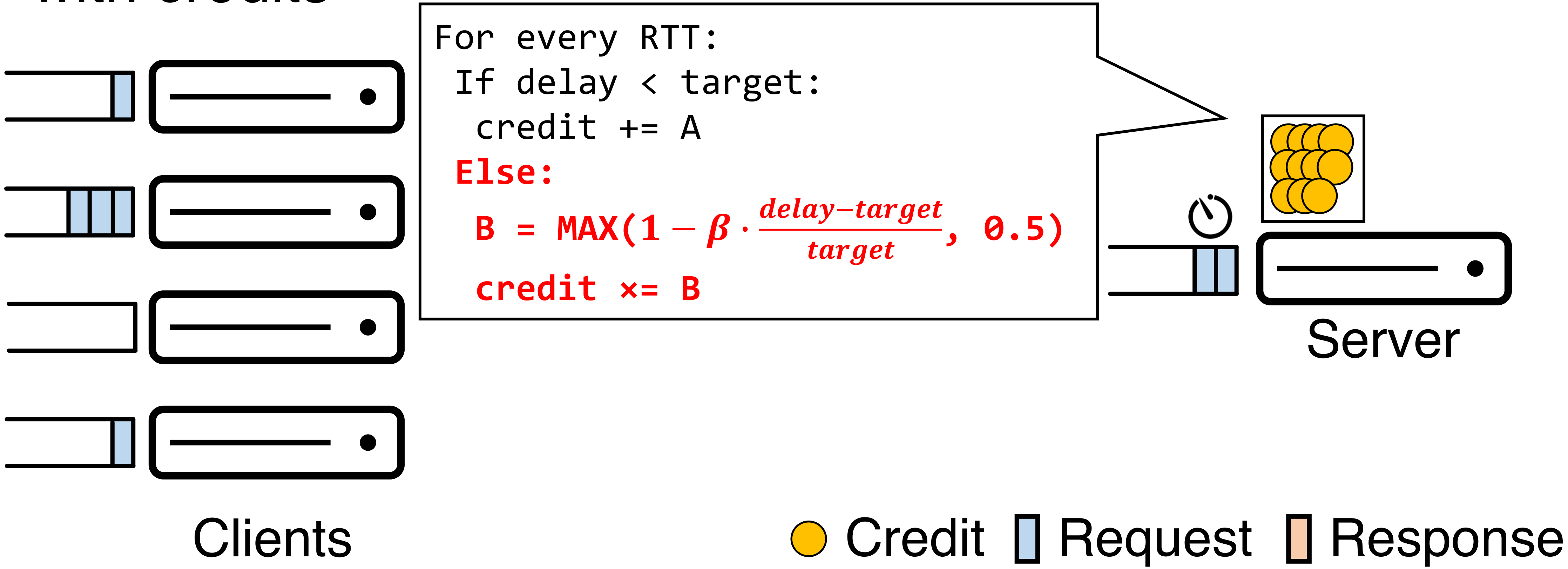
Server

Clients

● Credit ▯ Request ▯ Response

Credit-based admission control

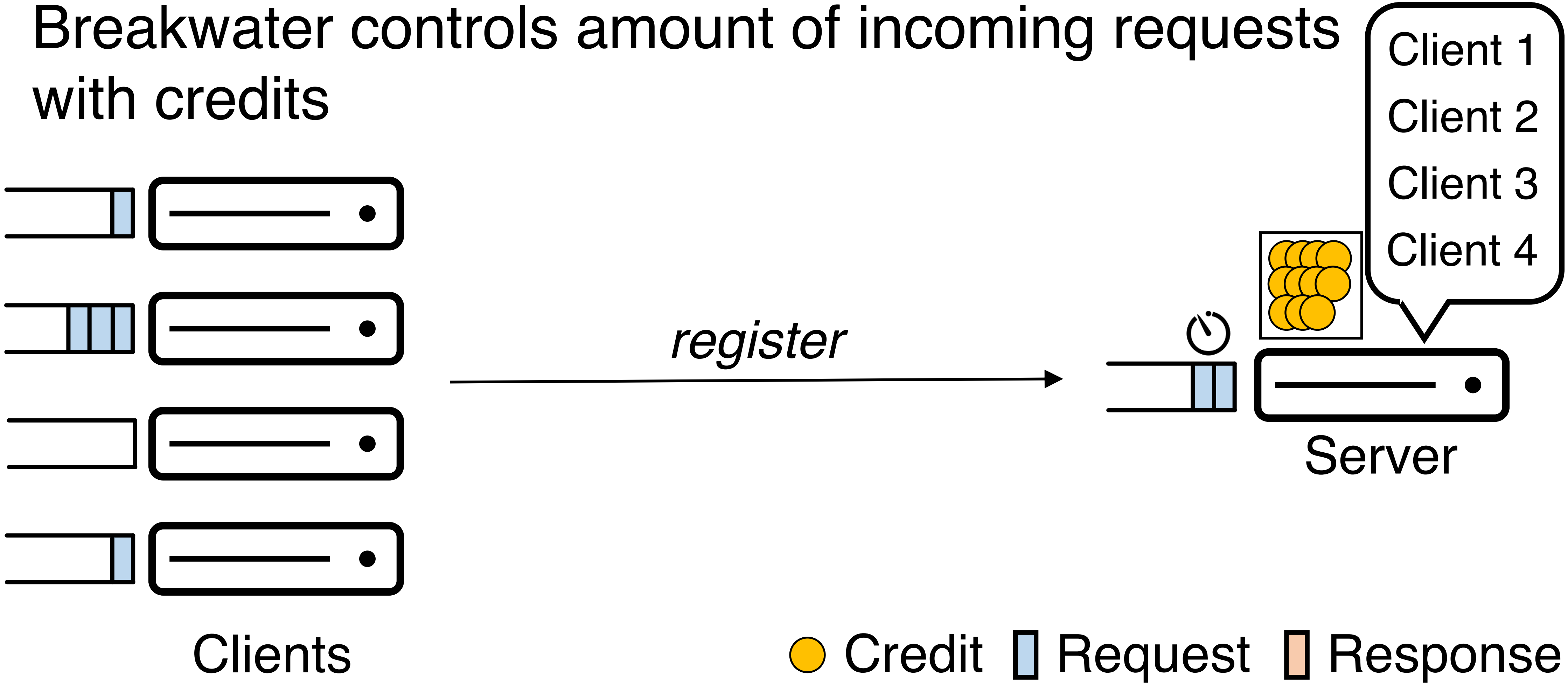
Breakwater controls amount of incoming requests with credits



```
For every RTT:
 If delay < target:
  credit += A
 Else:
  B = MAX($1 - \beta \cdot \frac{delay - target}{target}$, 0.5)
  credit ×= B
```

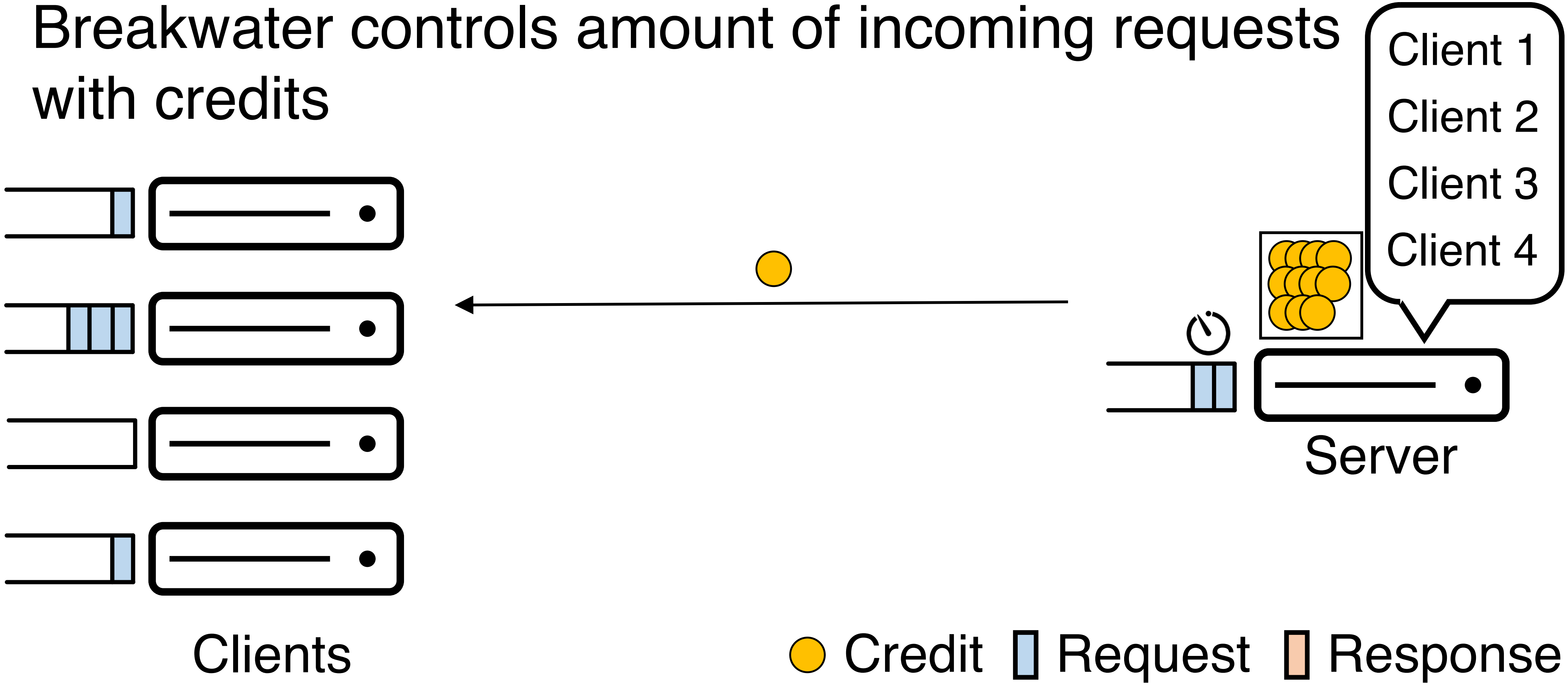
Credit-based admission control

Breakwater controls amount of incoming requests with credits



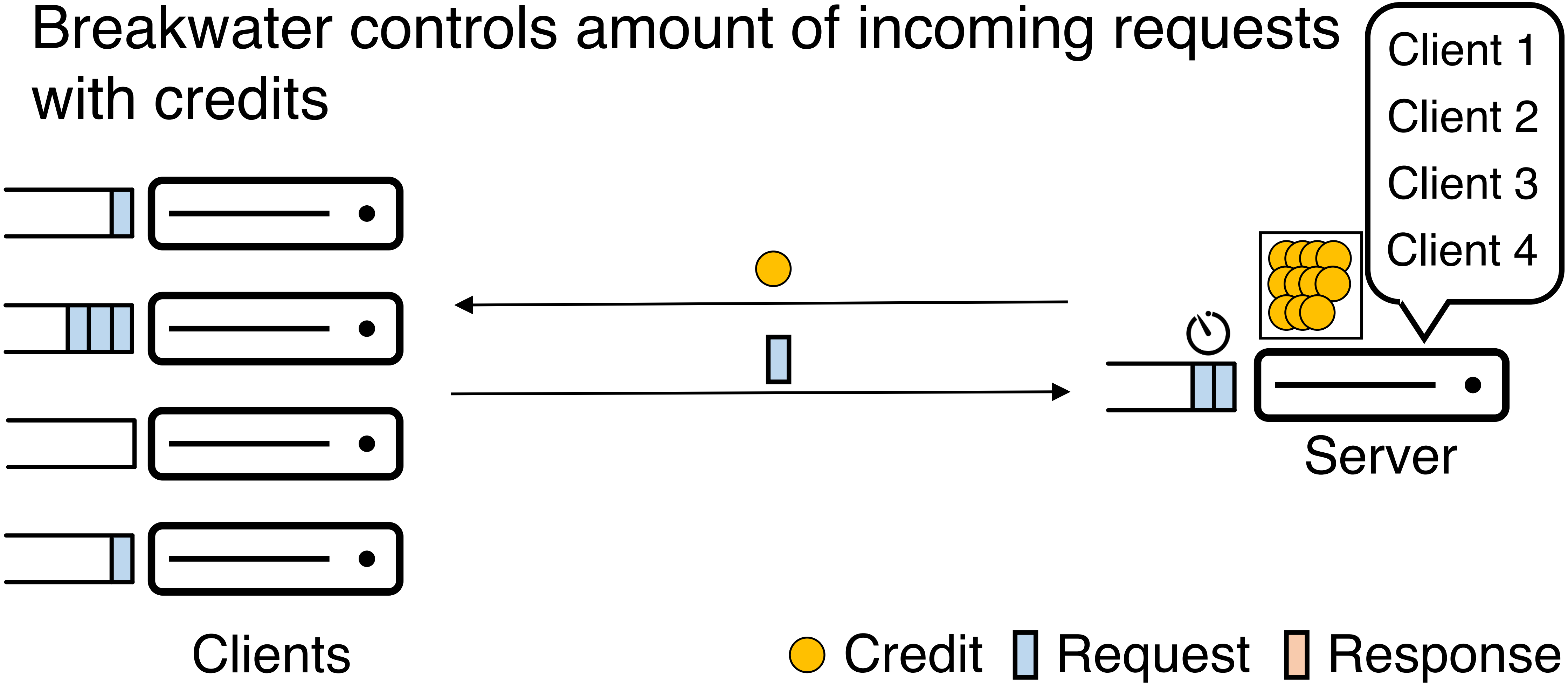
Credit-based admission control

Breakwater controls amount of incoming requests with credits



Credit-based admission control

Breakwater controls amount of incoming requests with credits



Client 1

Client 2

Client 3

Client 4

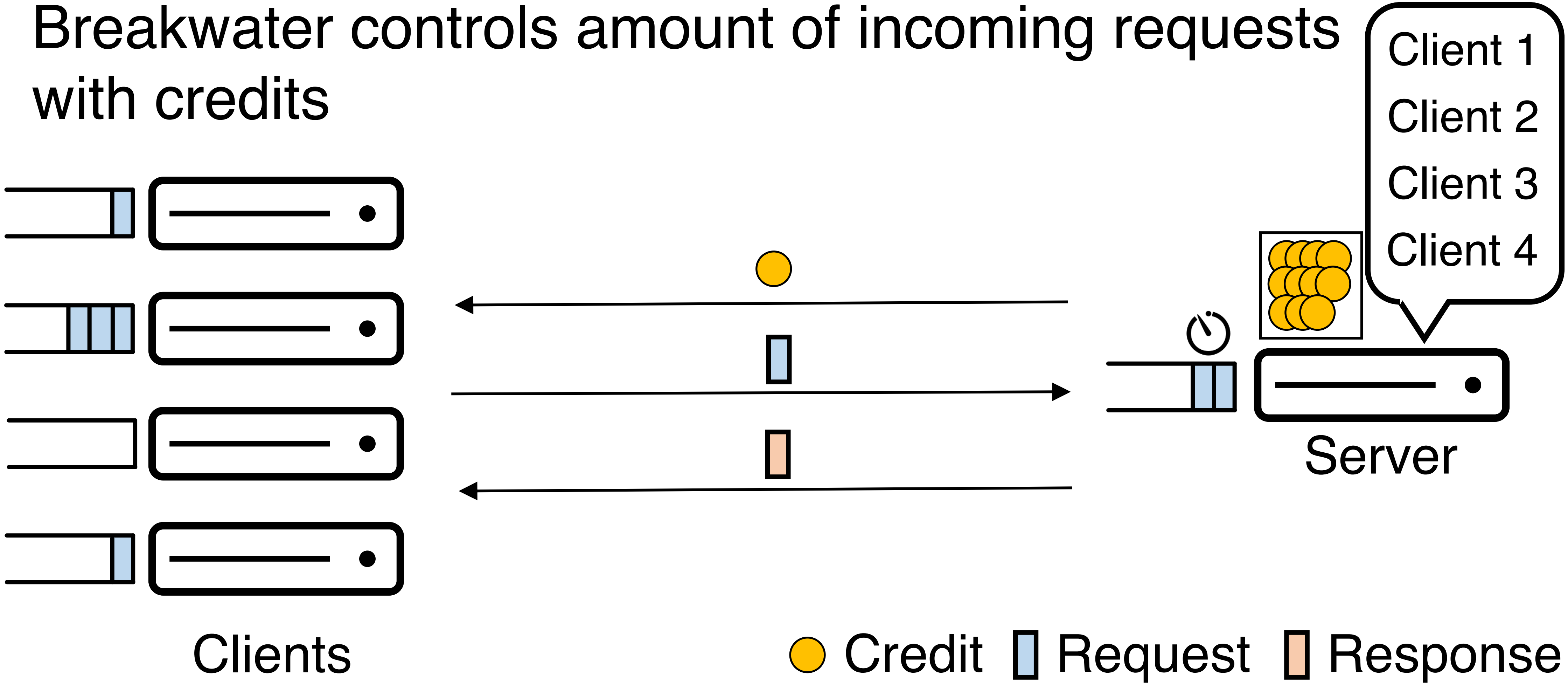
Server

Clients

● Credit ▯ Request ▯ Response

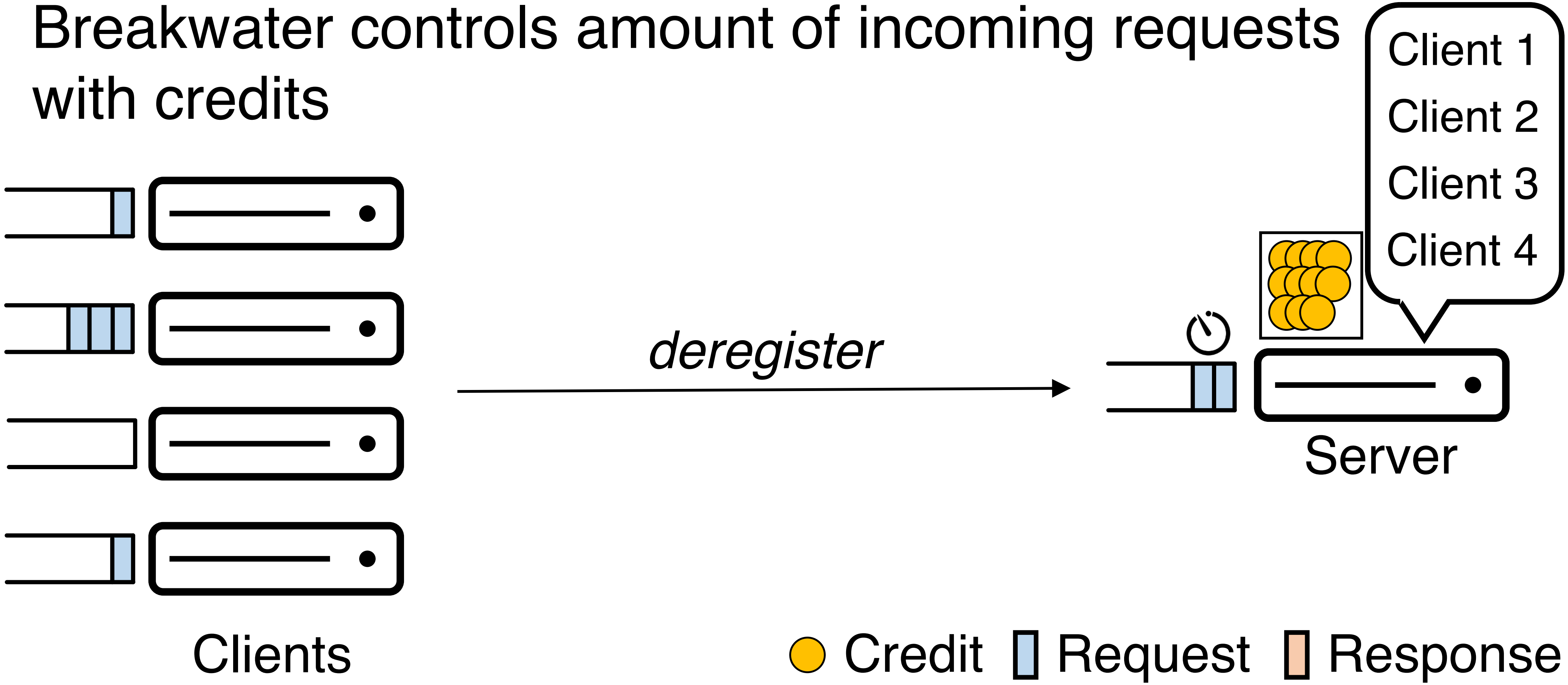
Credit-based admission control

Breakwater controls amount of incoming requests with credits



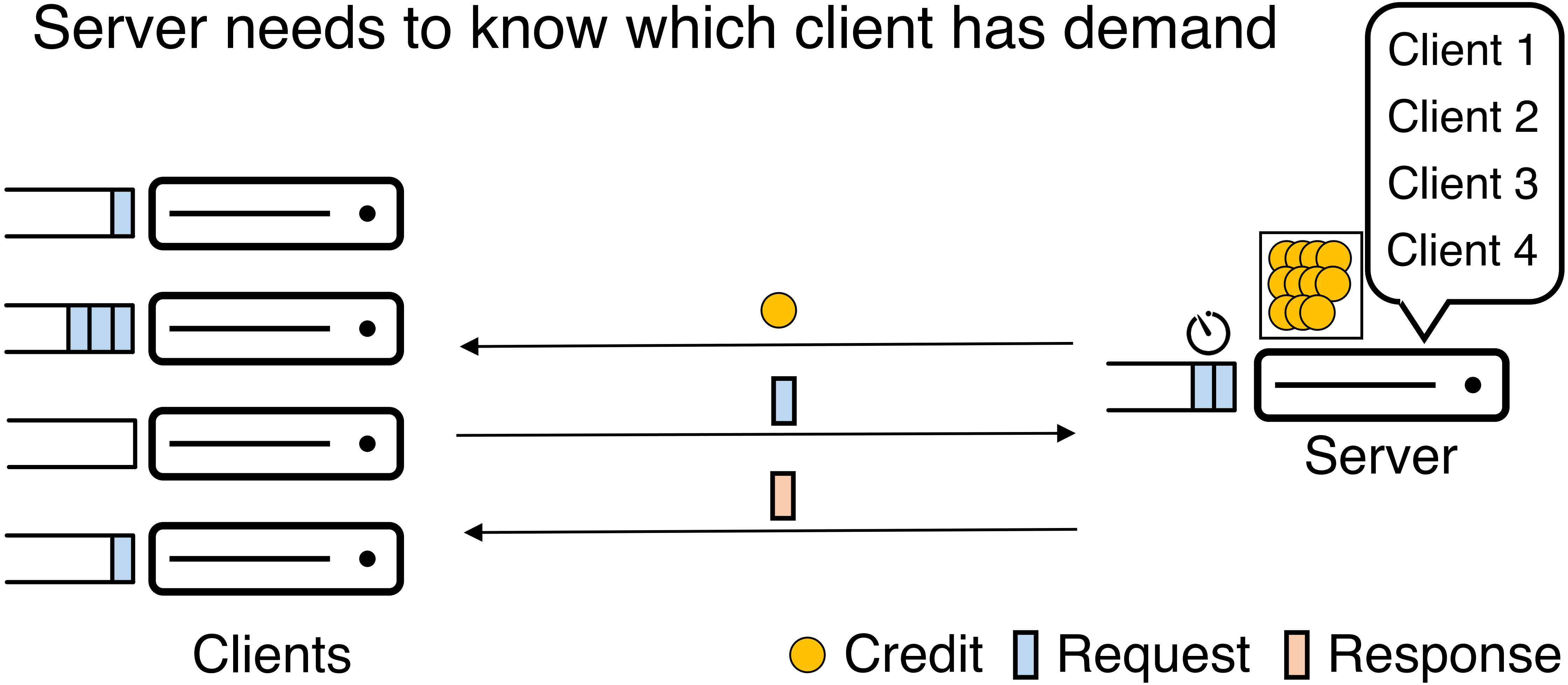
Credit-based admission control

Breakwater controls amount of incoming requests with credits



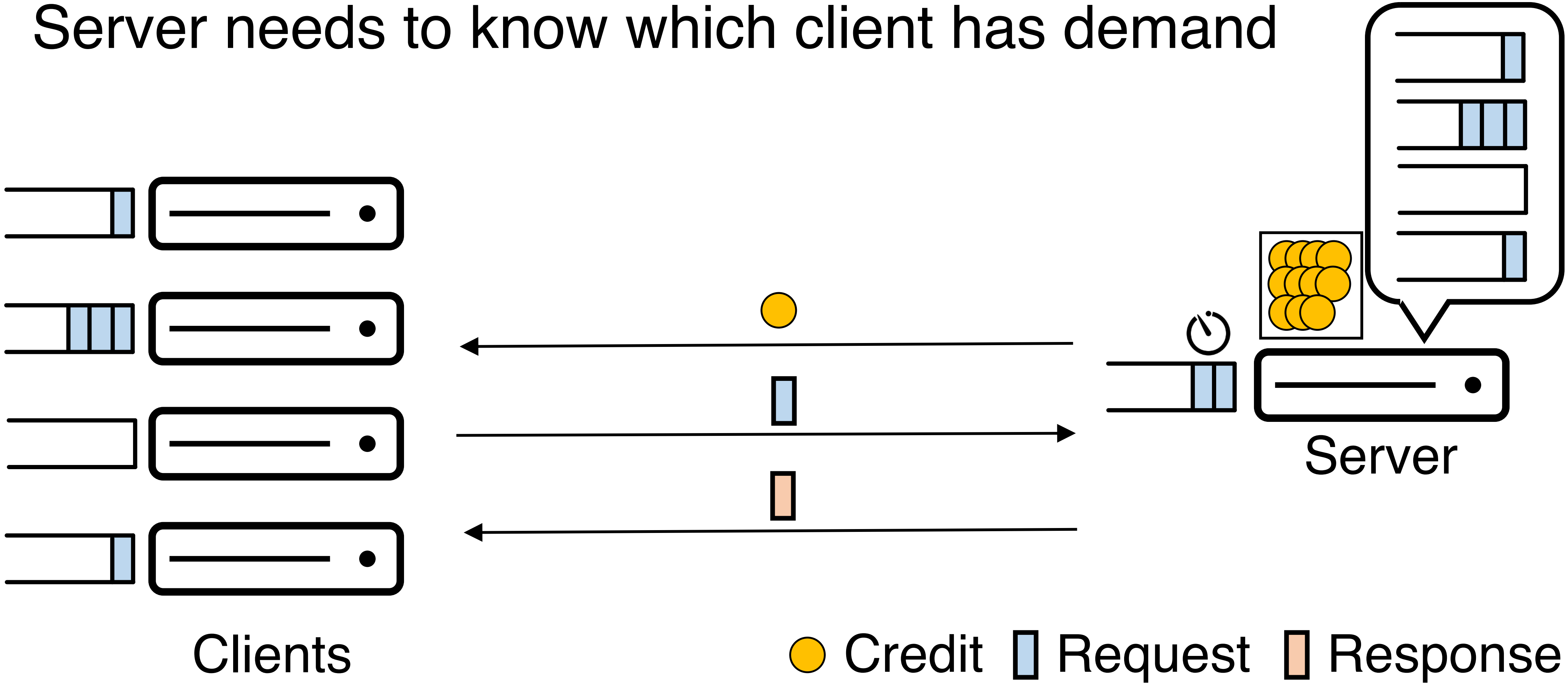
Message Overhead

Server needs to know which client has demand



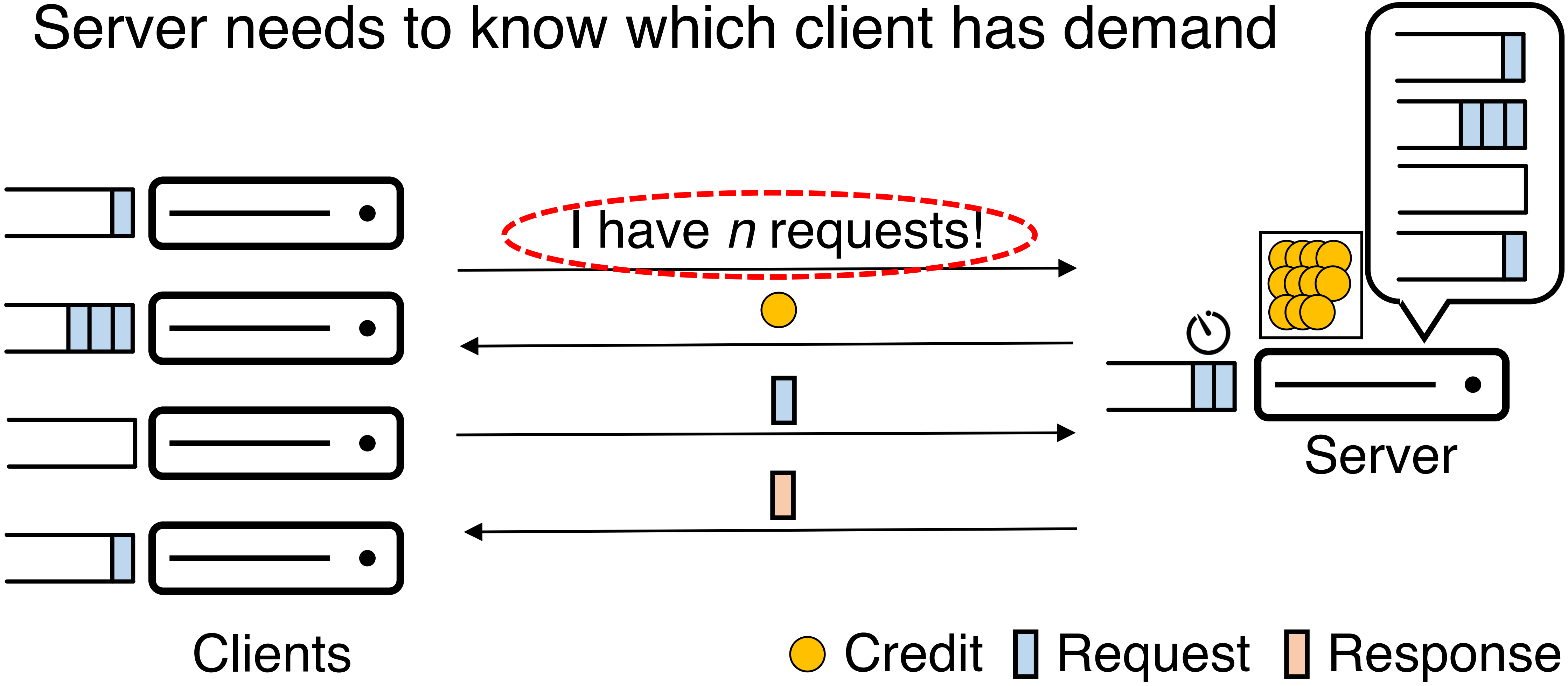
Message Overhead

Server needs to know which client has demand



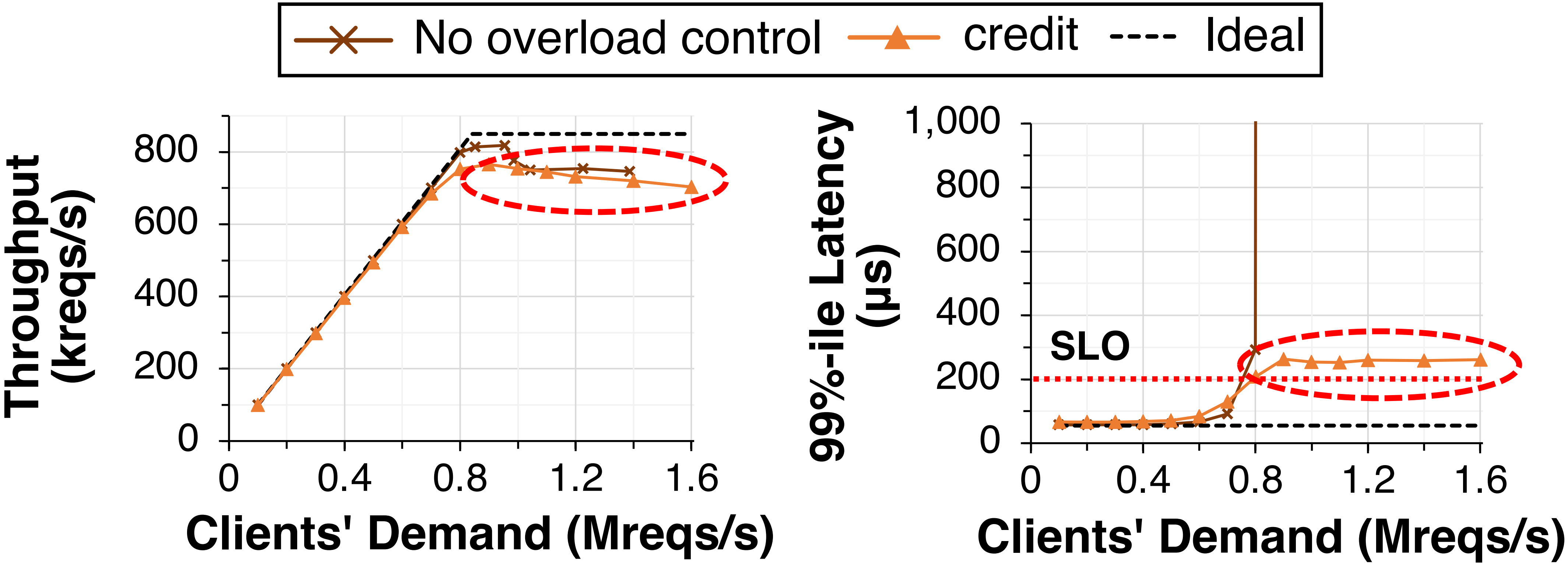
Message Overhead

Server needs to know which client has demand



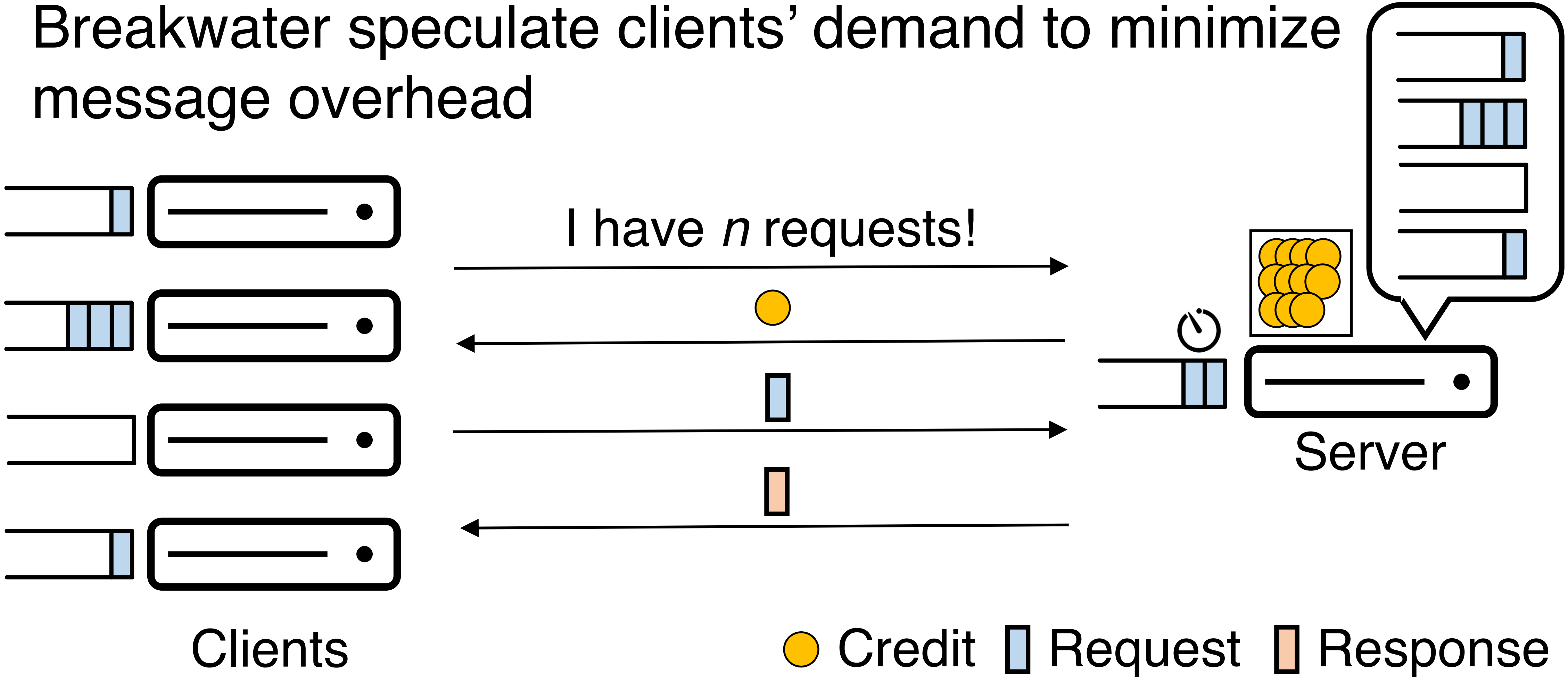
Impact of Credit-based admission control

Credit-based admission control has lower and bounded tail latency but lower throughput.



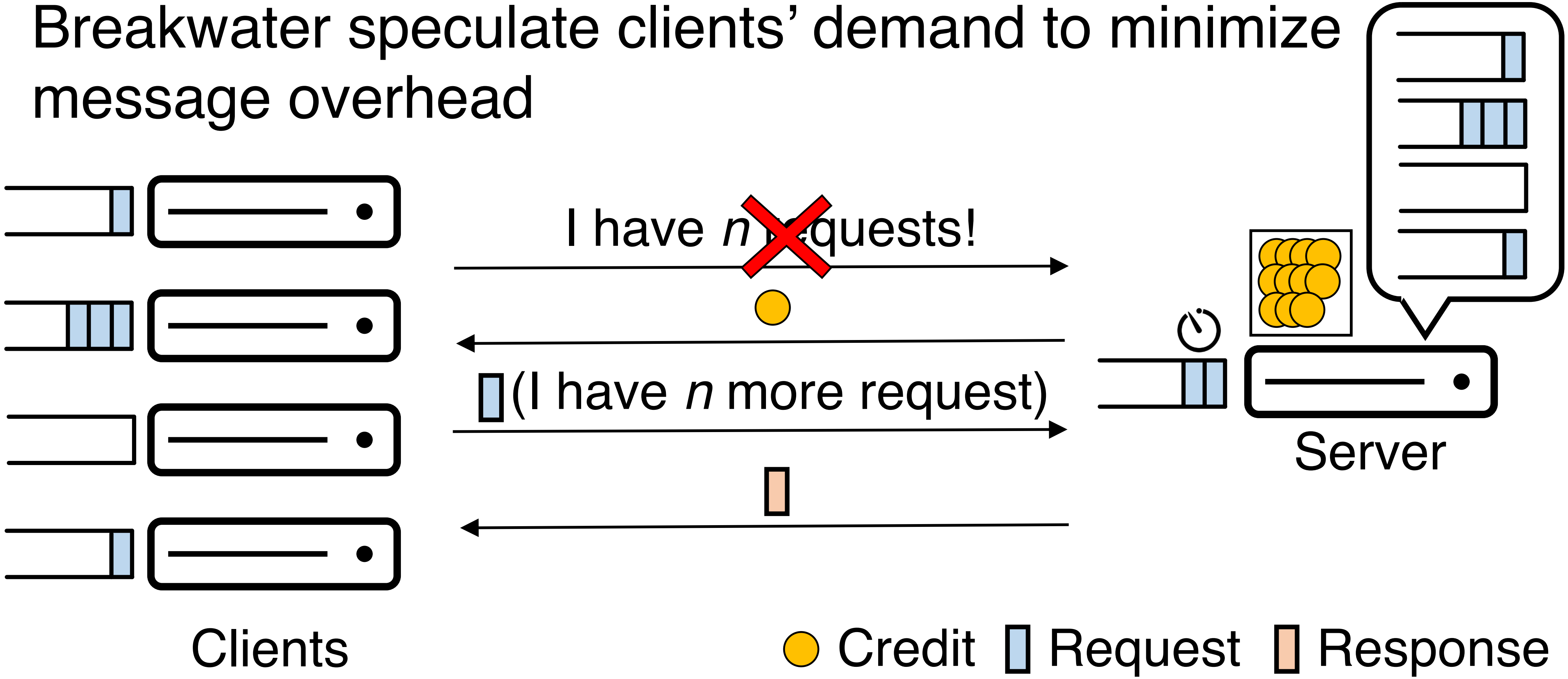
Demand Speculation

Breakwater speculate clients' demand to minimize message overhead



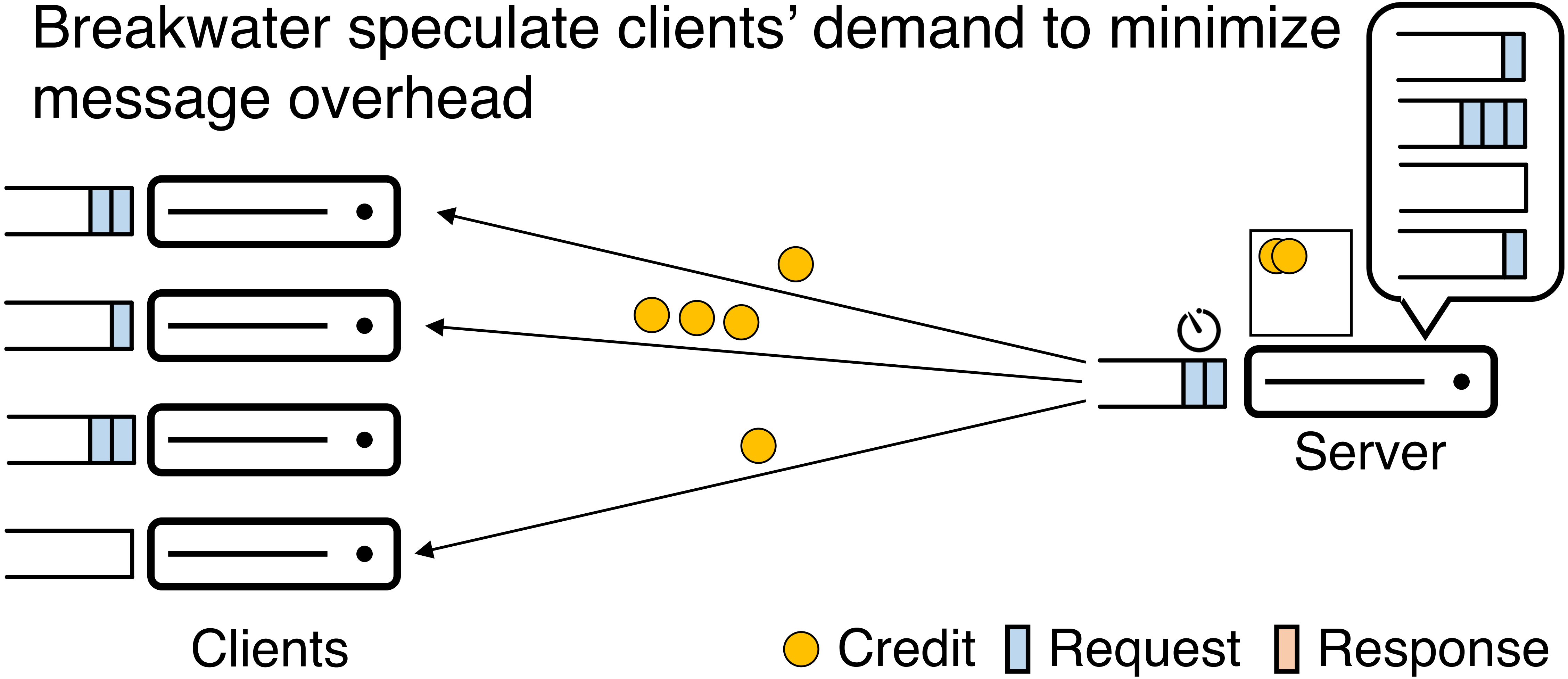
Demand Speculation

Breakwater speculate clients' demand to minimize message overhead



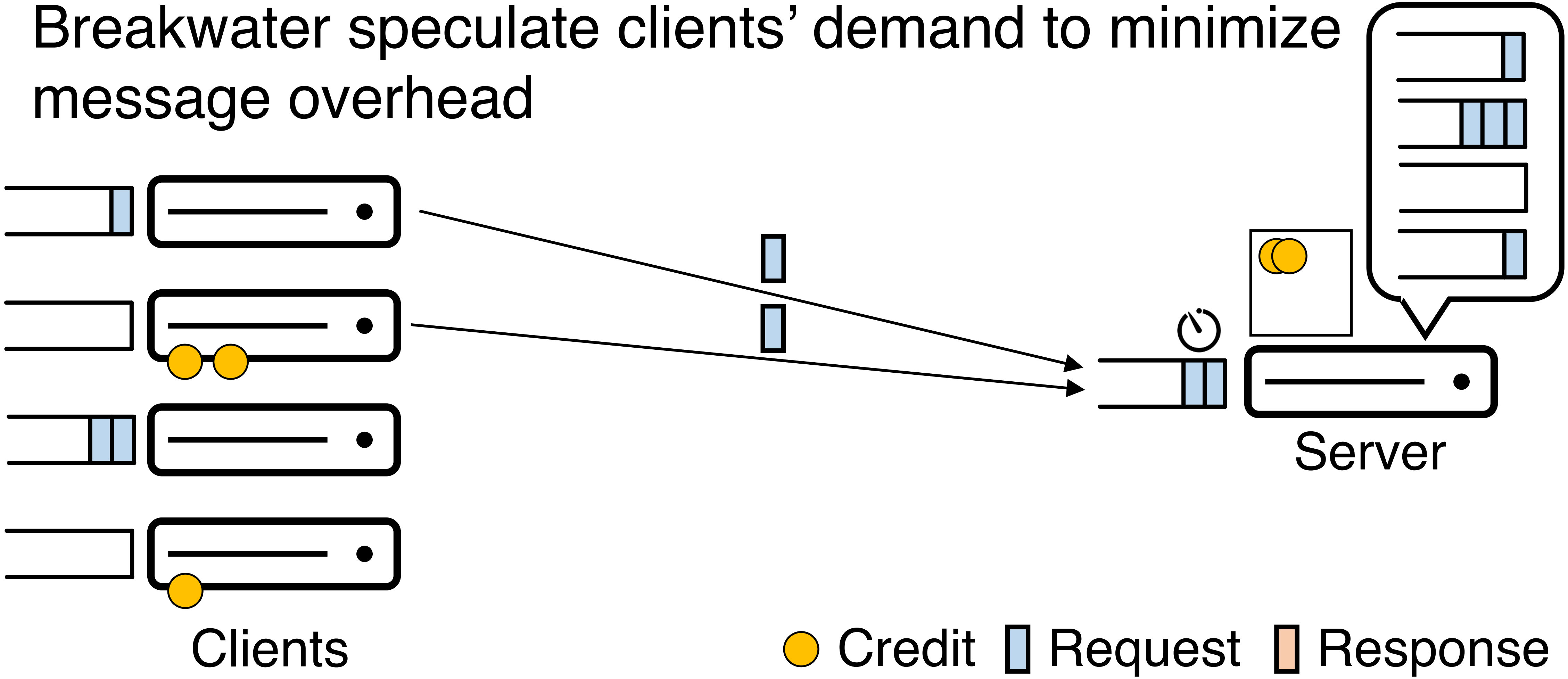
Demand Speculation

Breakwater speculate clients' demand to minimize message overhead



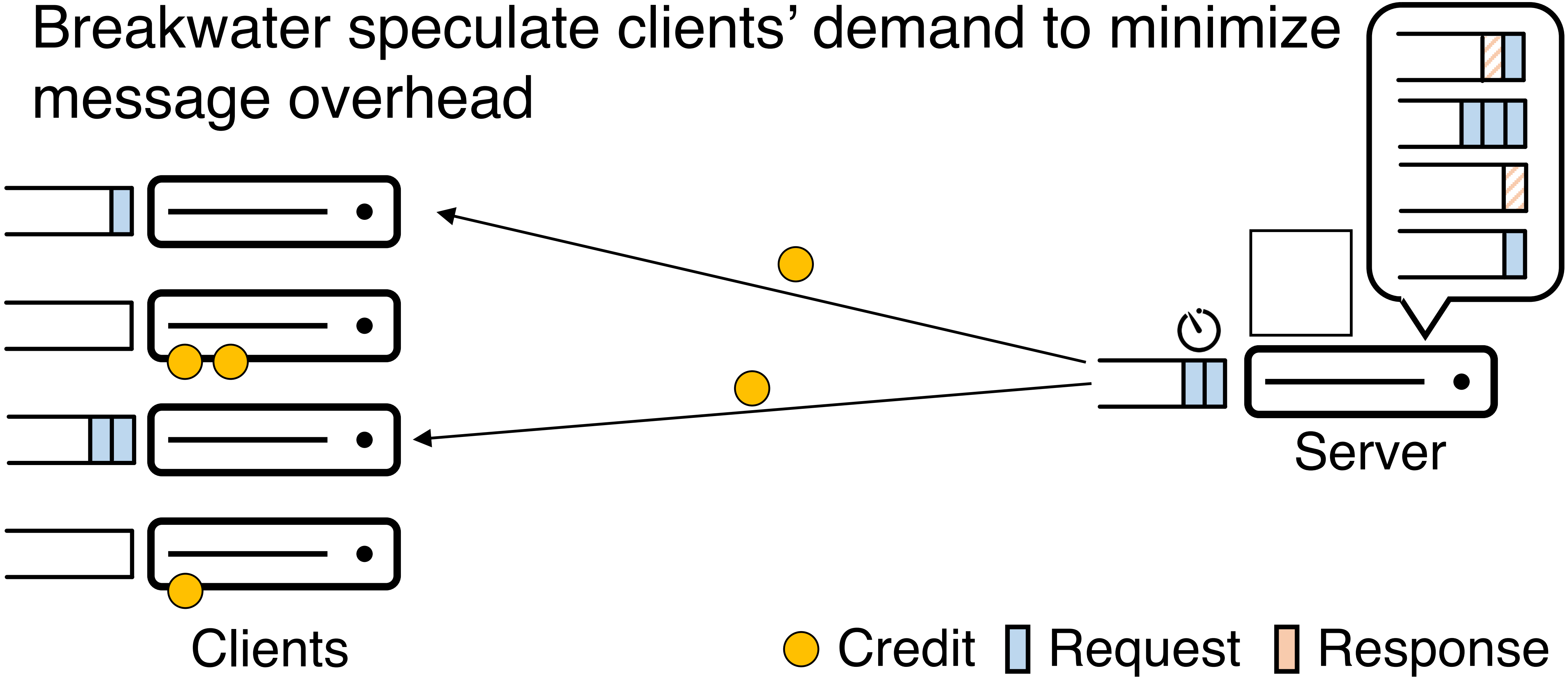
Demand Speculation

Breakwater speculate clients’ demand to minimize message overhead



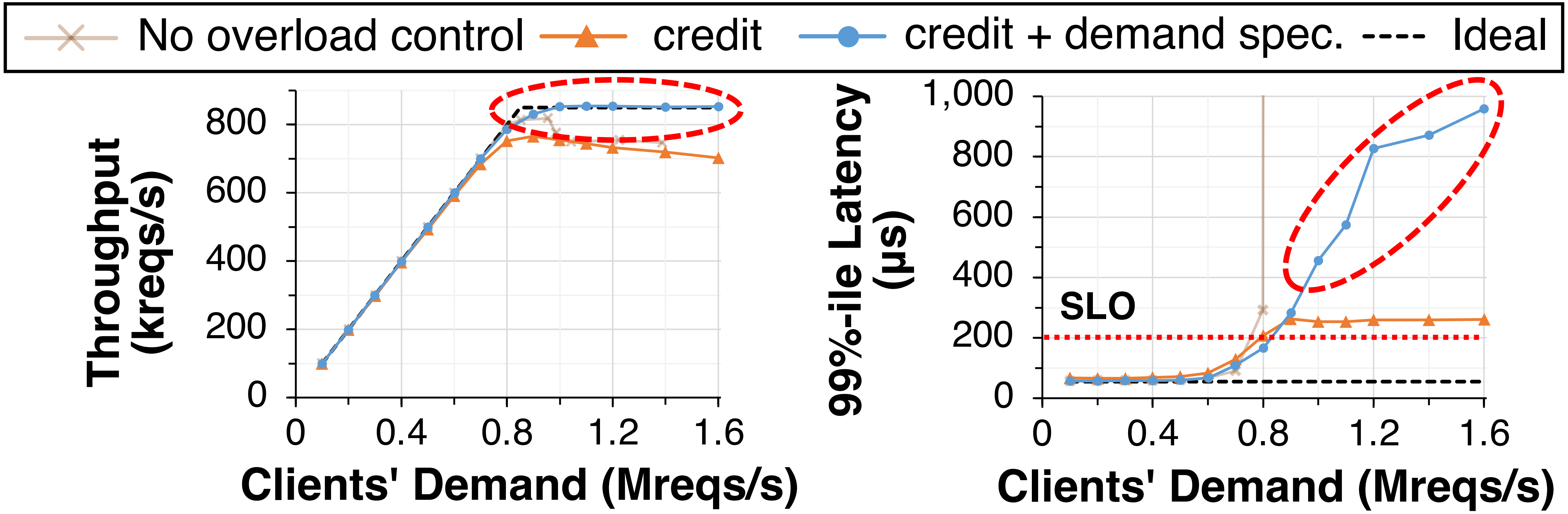
Demand Speculation

Breakwater speculate clients' demand to minimize message overhead



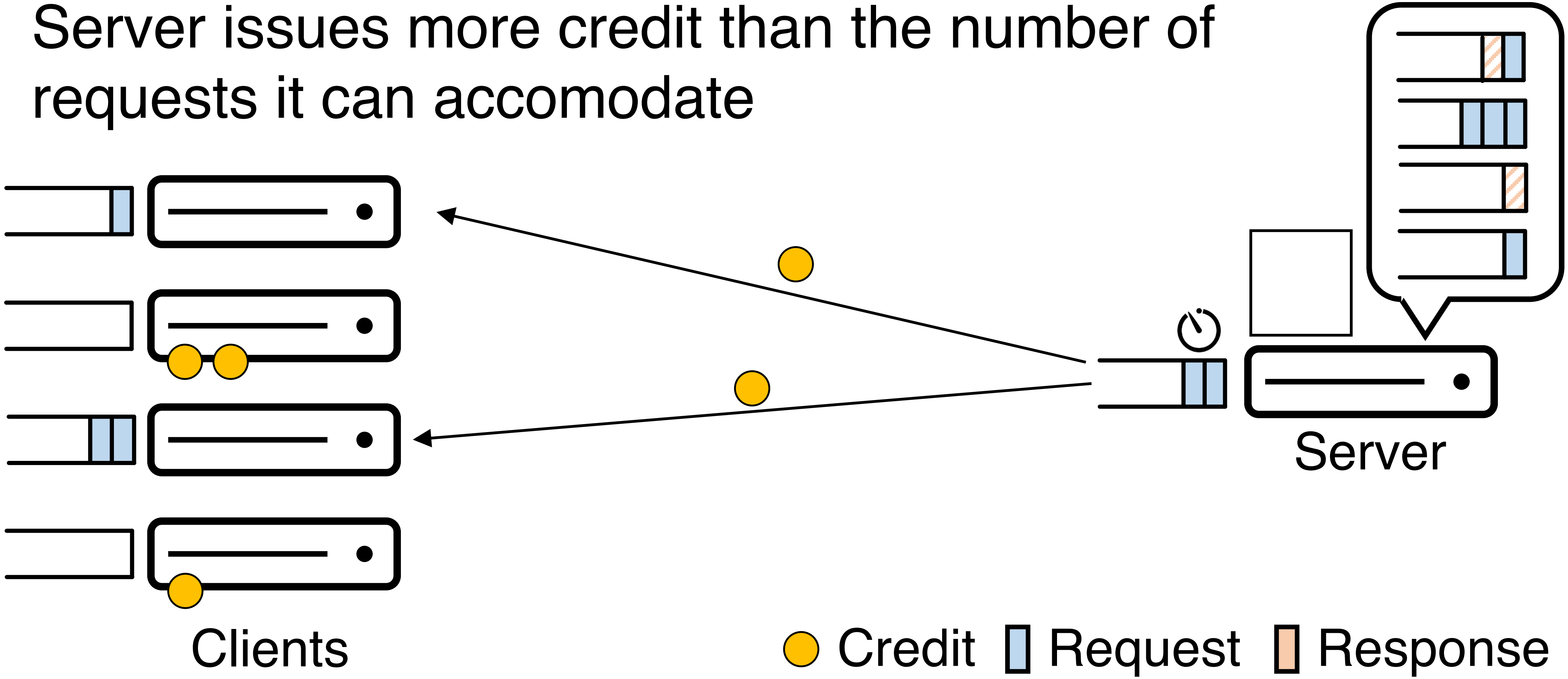
Impact of Adding Demand Speculation

Demand speculation improves throughput with higher tail latency



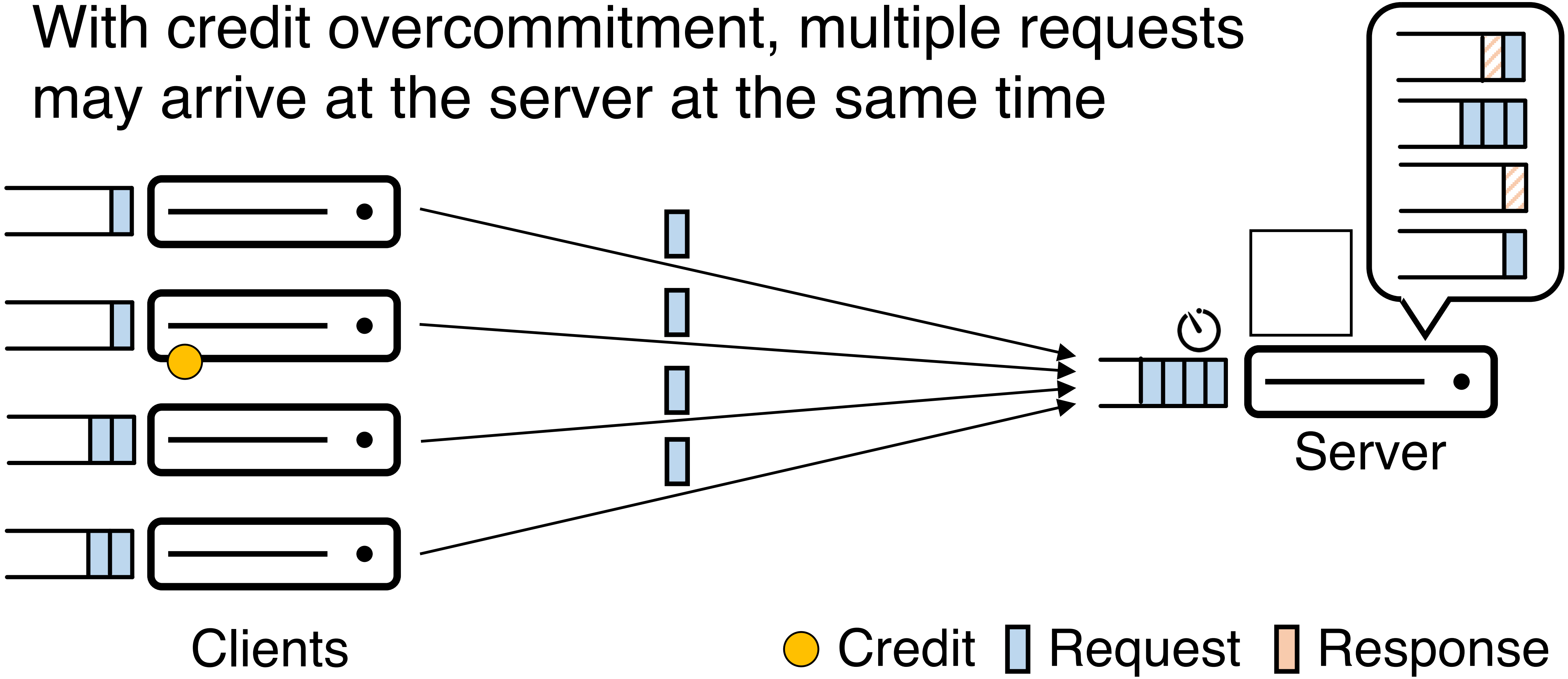
Credit Overcommitment

Server issues more credit than the number of requests it can accomodate



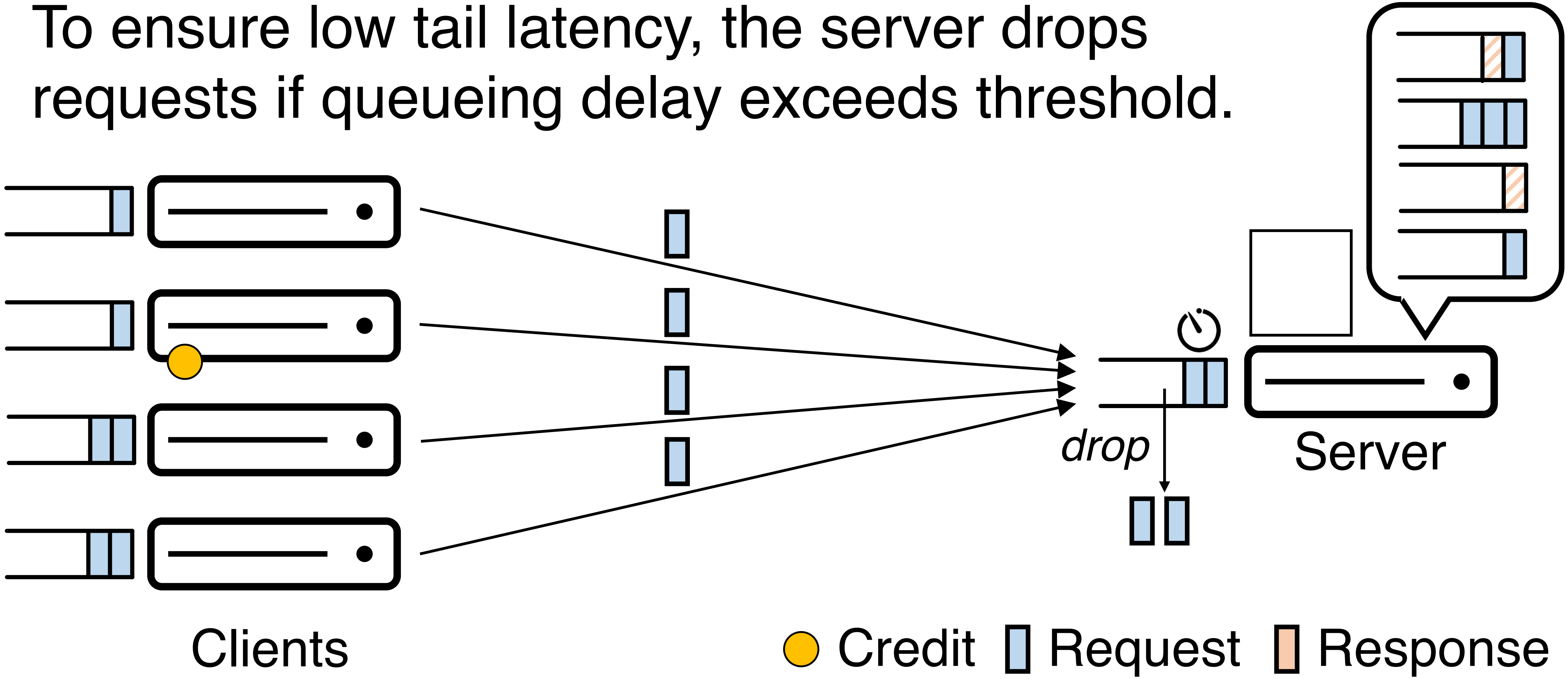
Incast

With credit overcommitment, multiple requests may arrive at the server at the same time



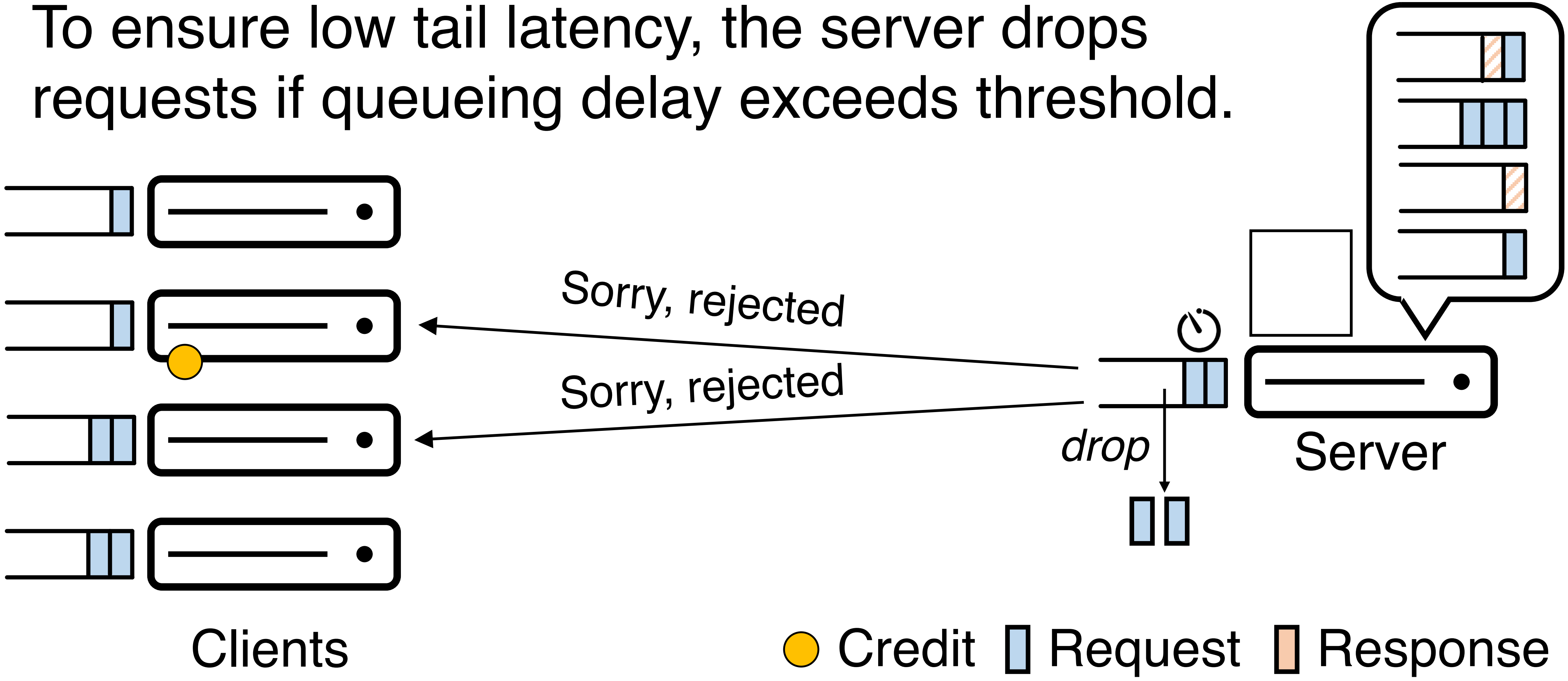
Delay-based AQM

To ensure low tail latency, the server drops requests if queueing delay exceeds threshold.



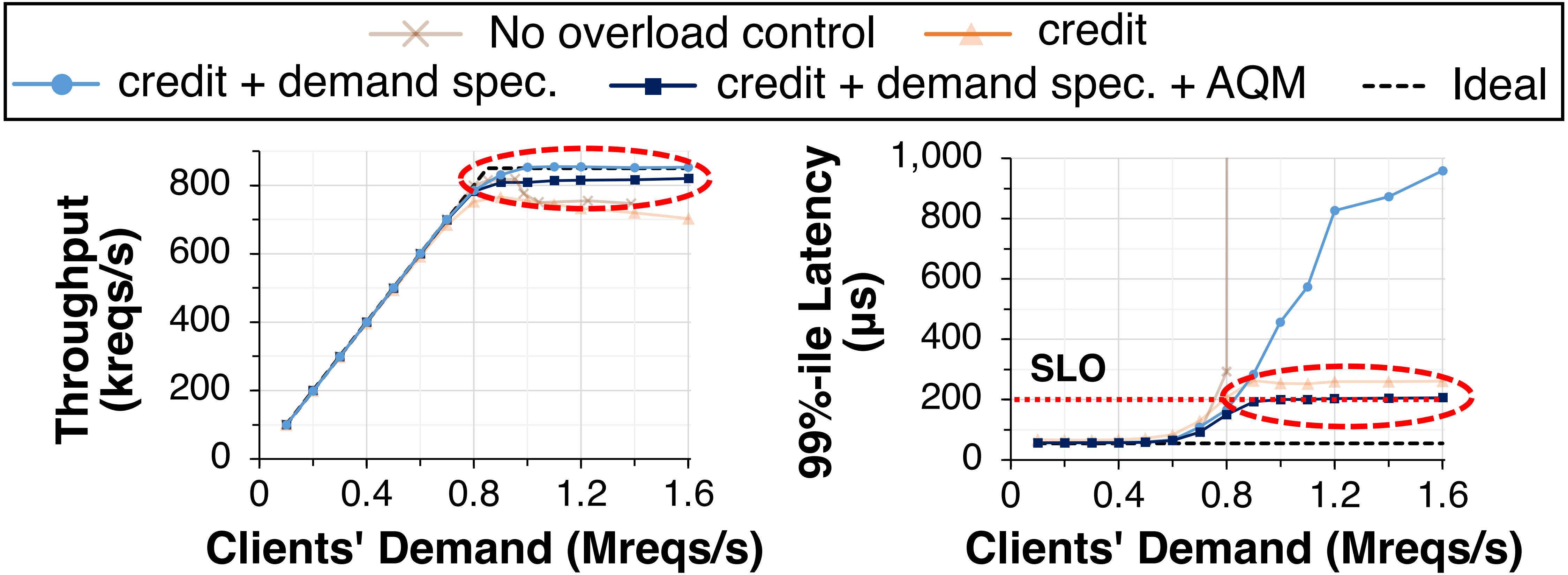
Delay-based AQM

To ensure low tail latency, the server drops requests if queueing delay exceeds threshold.



Impact of Adding Delay-based AQM

Breakwater achieves high throughput and low and bounded tail latency at the same time



Evaluation

Testbed Setup

- xl170 in Cloudlab
- 11 machines are connected to a single switch
- 10 client machines / 1 server machine
- Implementation on Shenango as a RPC layer

Synthetic Workload

- Clients generate request with open-loop Poisson process
- Requests spin-loops specified amount of time at server
- Exponential service time distribution with 10µs average

Evaluation

(1) Does Breakwater achieves high throughput and low tail latency even with demand spikes?

(2) Does Breakwater provides fast feedback for the rejected requests?

(3) Is Breakwater scalable to the number of clients?

Baselines:

DAGOR

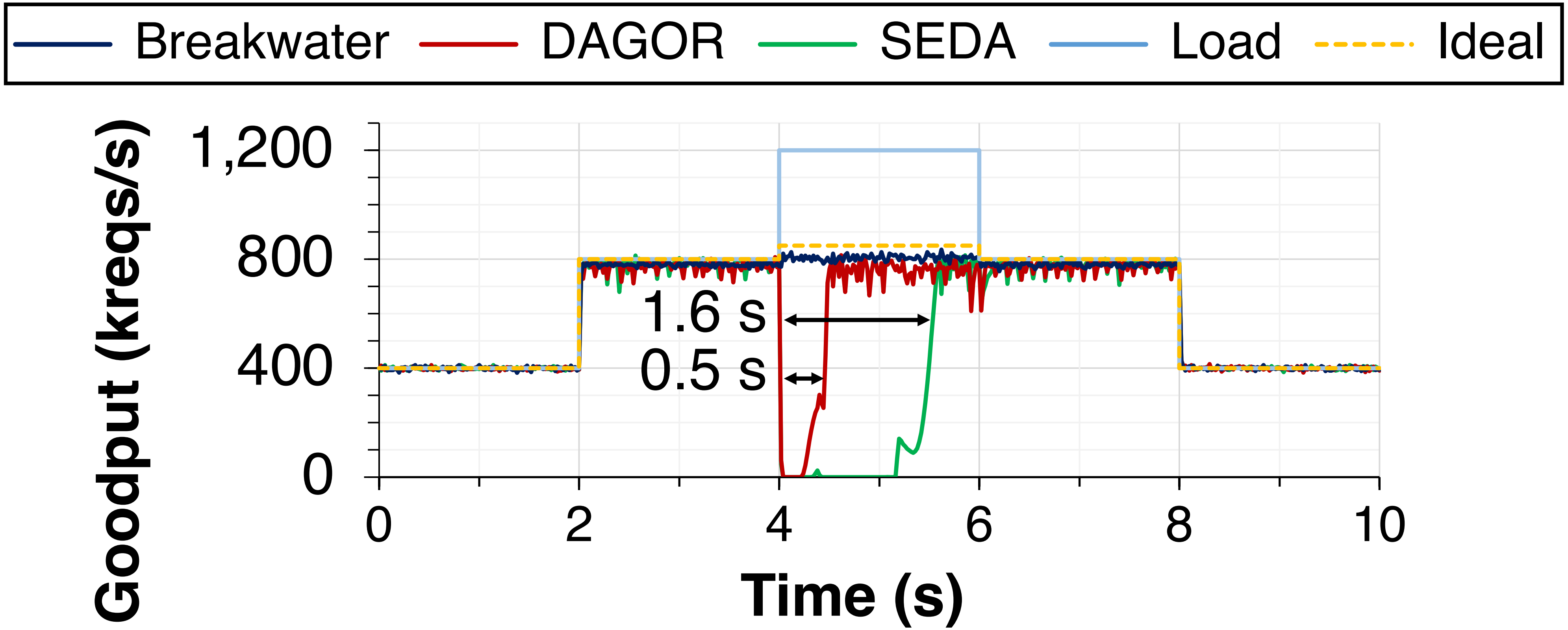
priority-based overload control used in WeChat

SEDA

adaptive overload control for staged event-driven architecture

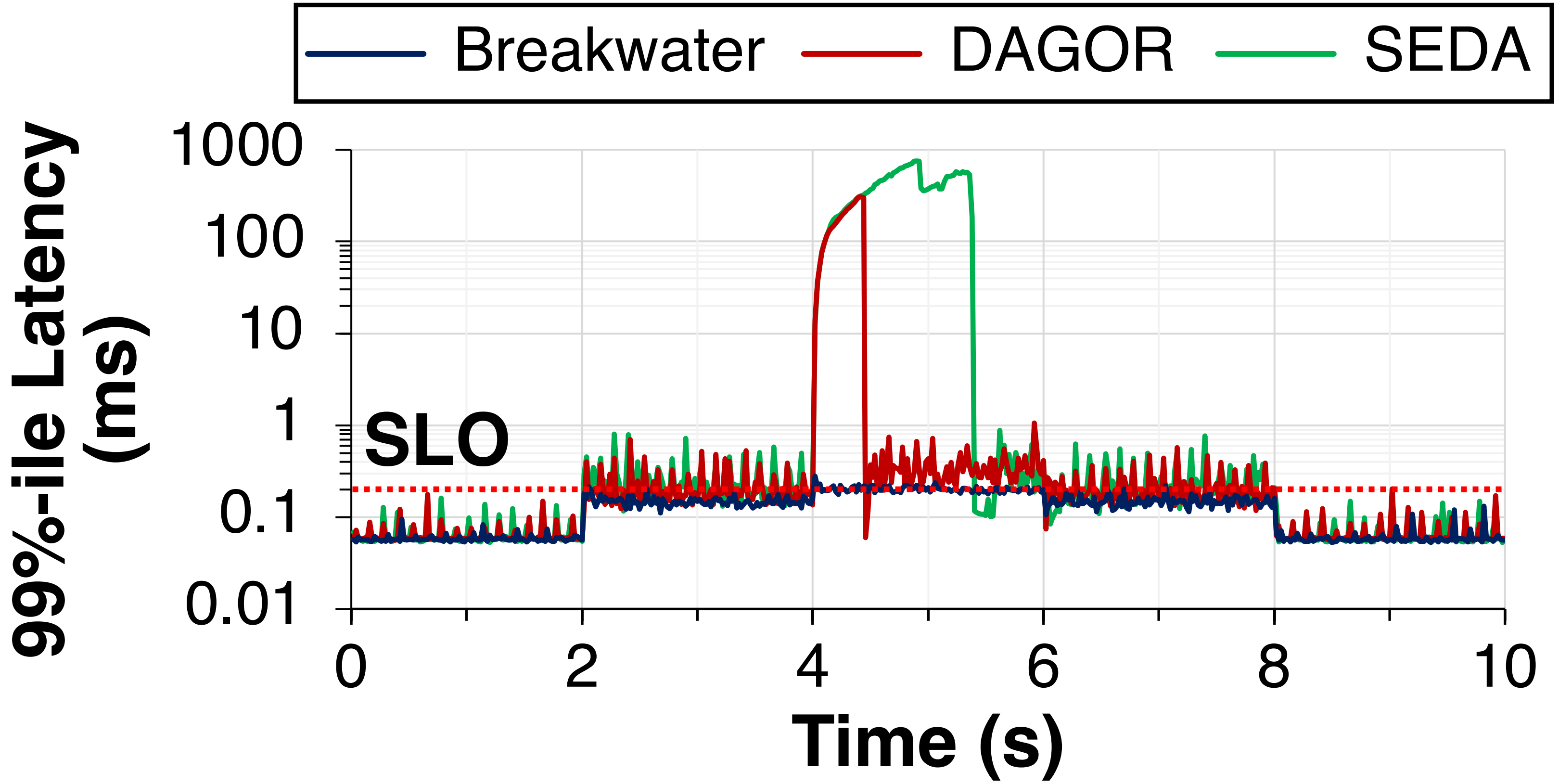
High Goodput with fast convergence

Breakwater achieves high goodput with fast convergence with sudden load shift.



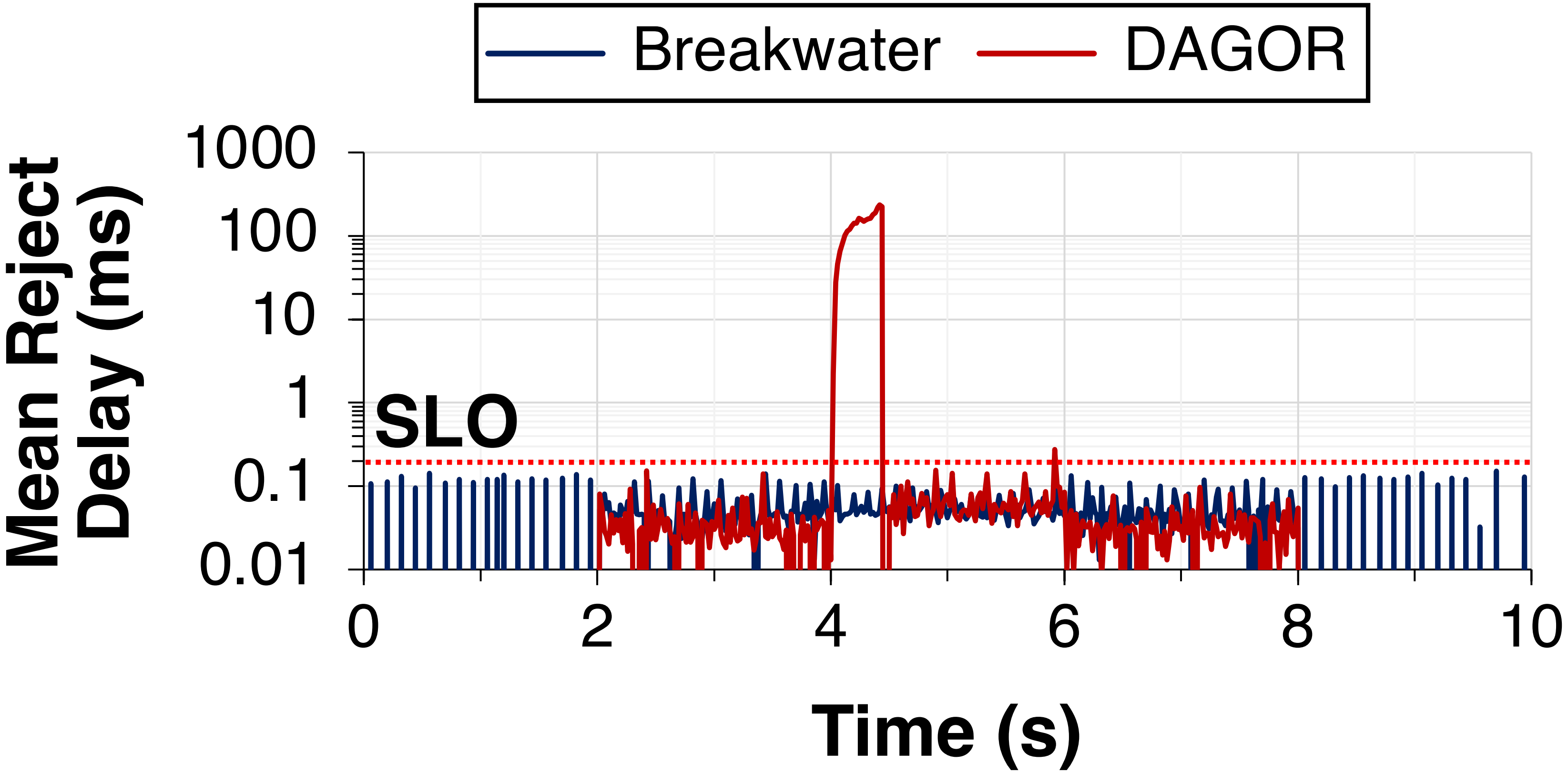
Low and Bounded Tail Latency

Breakwater has low and bounded tail latency even with sudden load shift



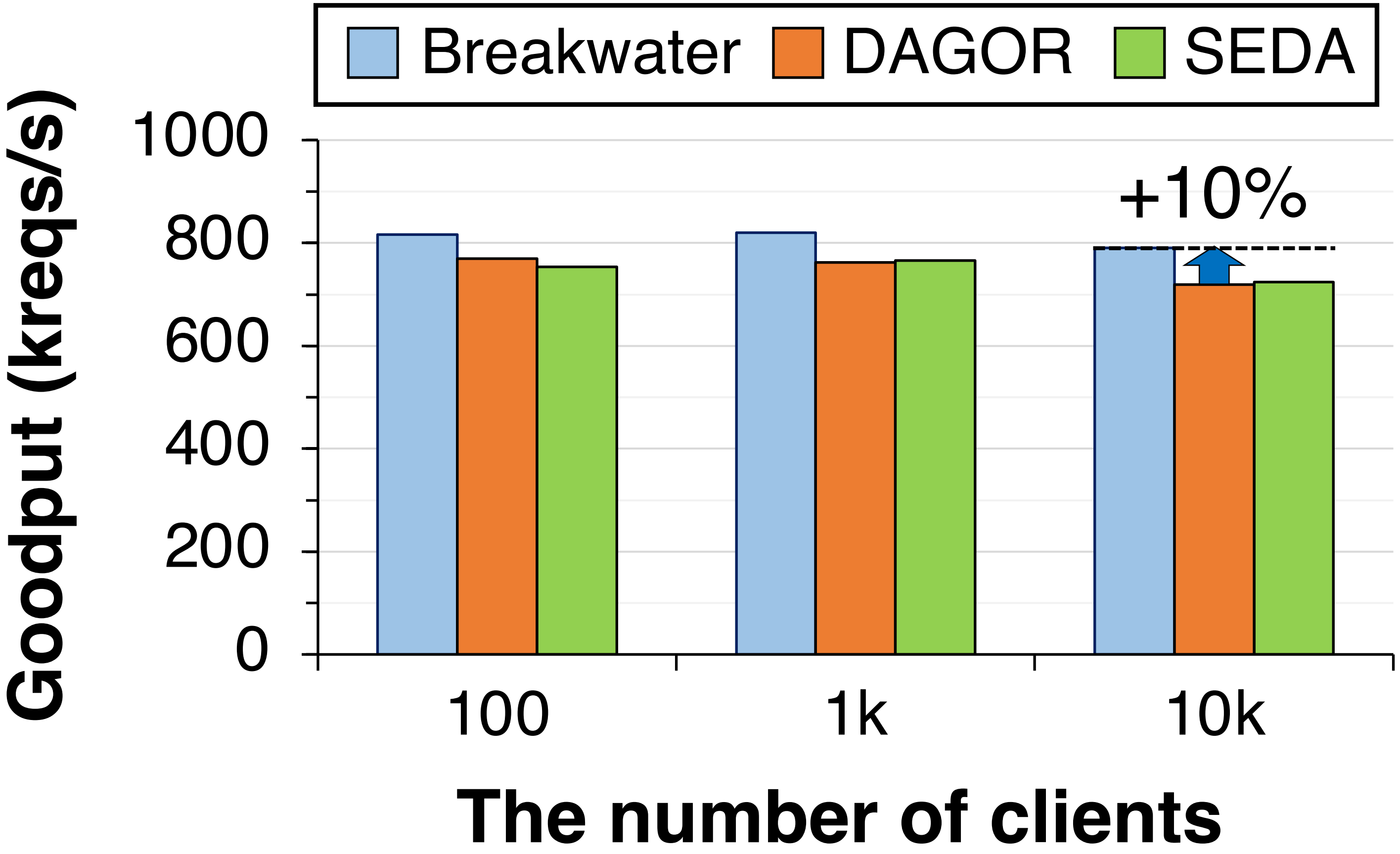
Fast Feedback

Breakwater notifies clients of rejected request in timely manner



Scalability

Breakwater is more scalable than existing overload controls



Conclusion

- Breakwater is a **server-driven credit-based** overload control system for µs-scale RPCs
- Breakwater's key components include
  (1) Credit-based admission control
  (2) Demand speculation
  (3) Delay-based AQM
- Our evaluation shows that Breakwater achieves
  (1) **Low & bounded tail latency** with **high throughput**
  (2) **Fast feedback** for a rejected request
  (3) **Scalability** to many clients

Thank you!

Breakwater is available at

`inhocho89.github.io/breakwater/`

Questions?

Inho Cho <inhocho@csail.mit.edu>