Overload Control for µs-scale RPCs with Breakwater

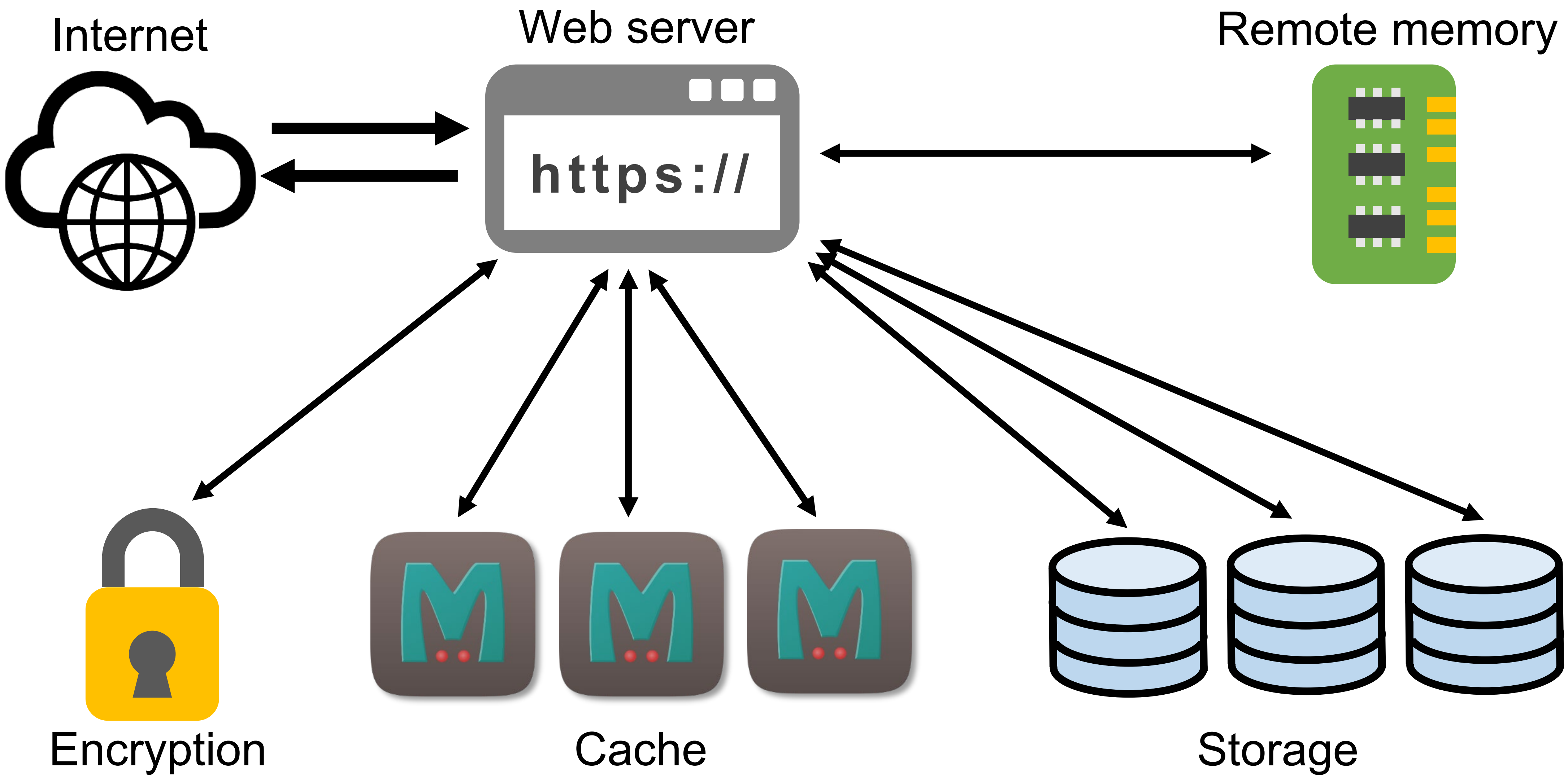
Inho Cho, Ahmed Saeed, Joshua Fried,
Seo Jin Park, Mohammad Alizadeh, Adam Belay

CSAIL

Trend: µs-scale RPCs

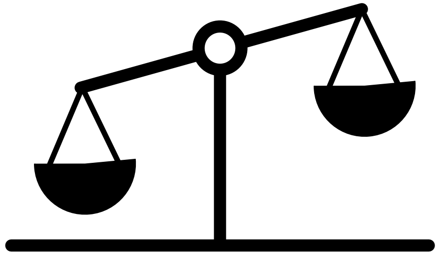
1. Fast Network: Network latency (~ 5 us)
2. Fast Storage: M.2 NVME SSD (~ 20 us)
3. In-memory operations: Memcached, Reddis, Ignite

Trend: High Fan-out

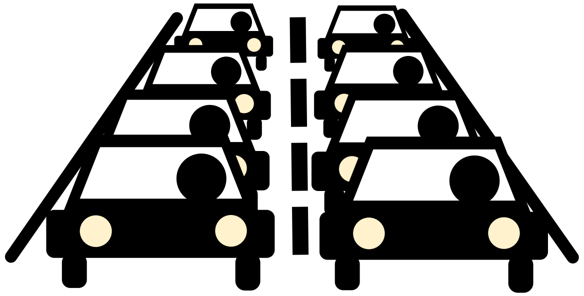


Causes of Server Overload

Load Imbalance

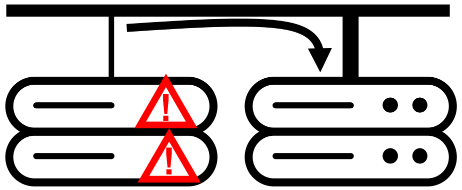


Unexpected user traffic

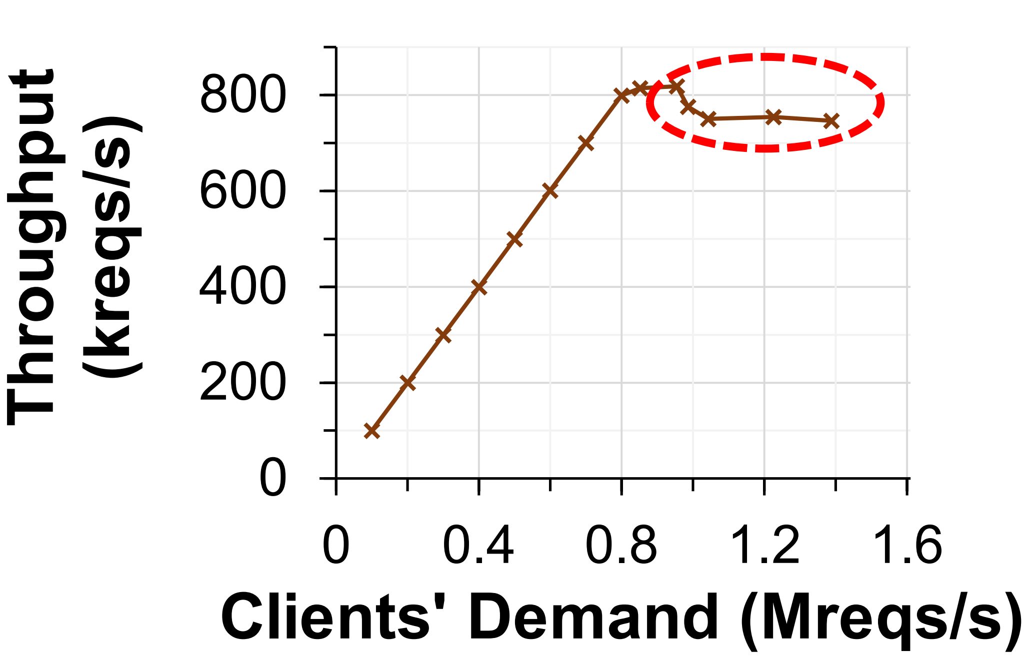


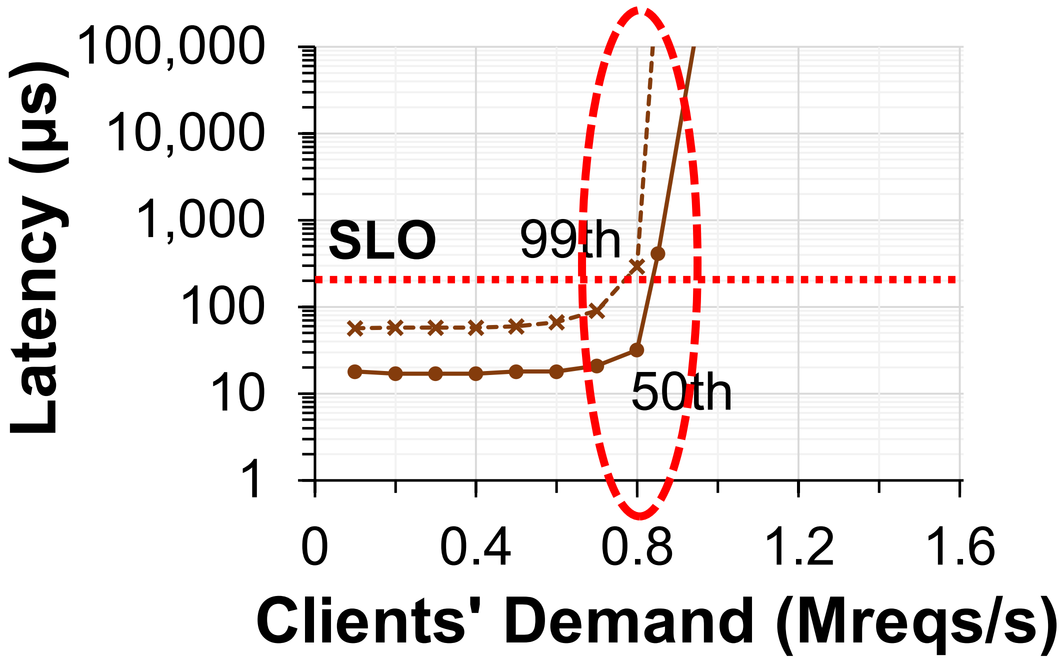
Packet bursts

Redirected traffic due to failure



Performance Without Overload Control



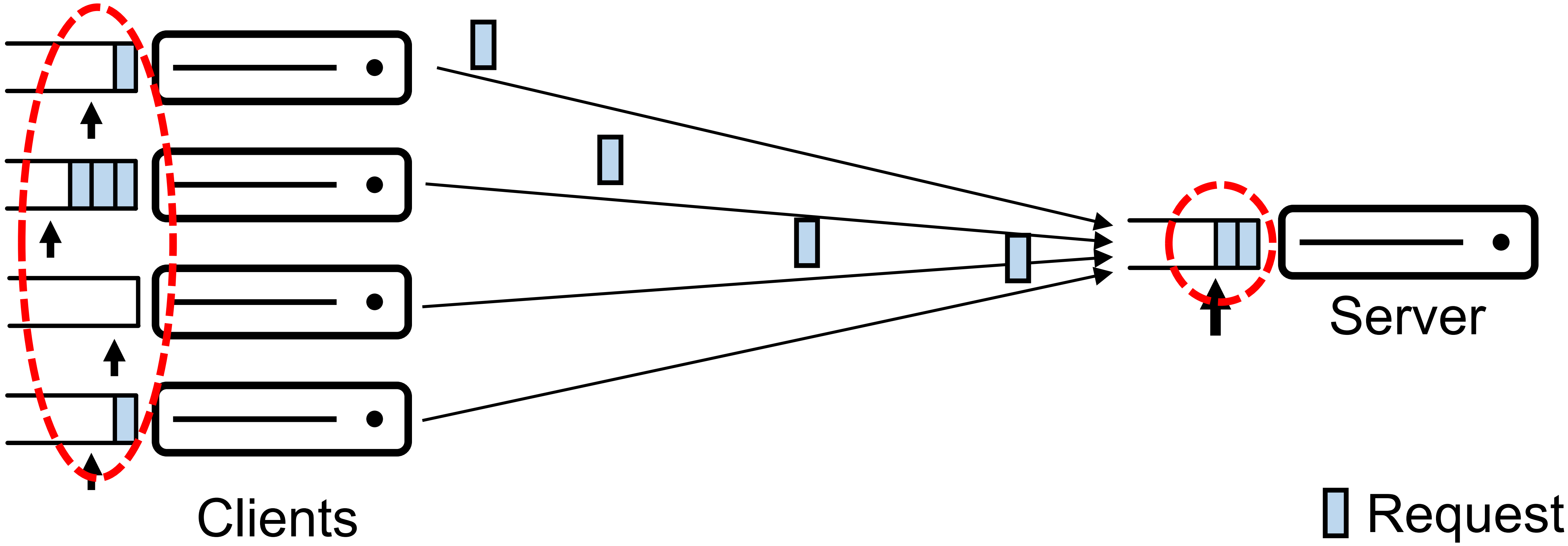


Without overload control, server overload makes almost all requests violate its SLO.

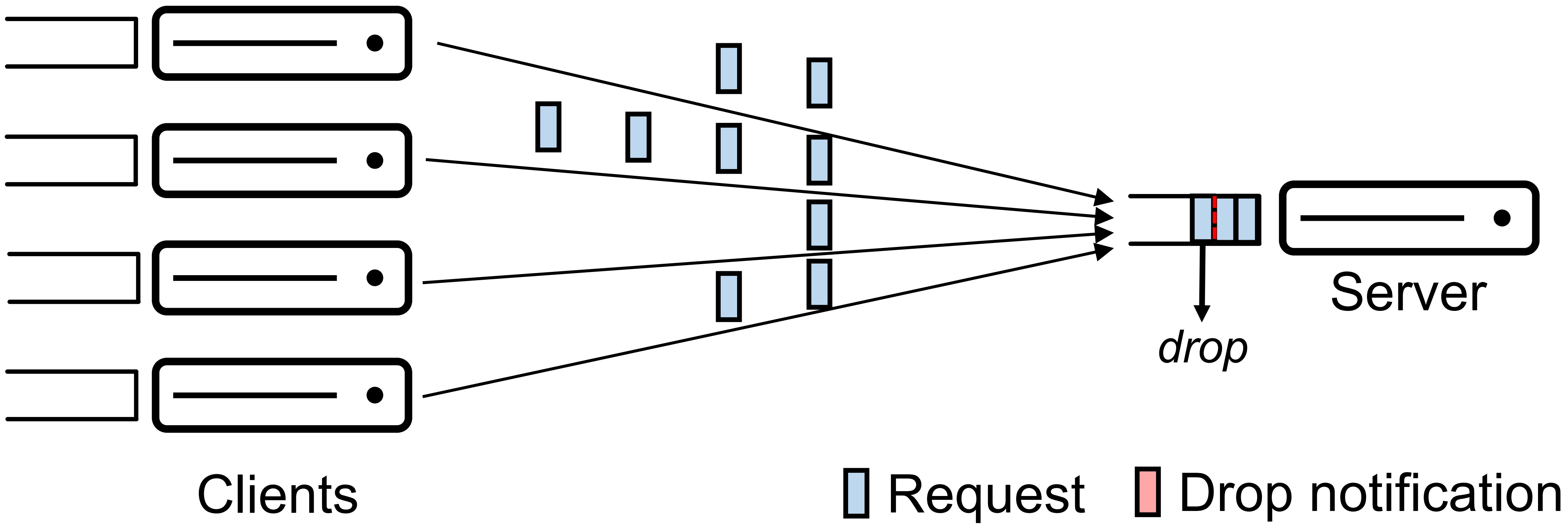
Ideal Overload Control

should keep request queue **short**, but **not empty**

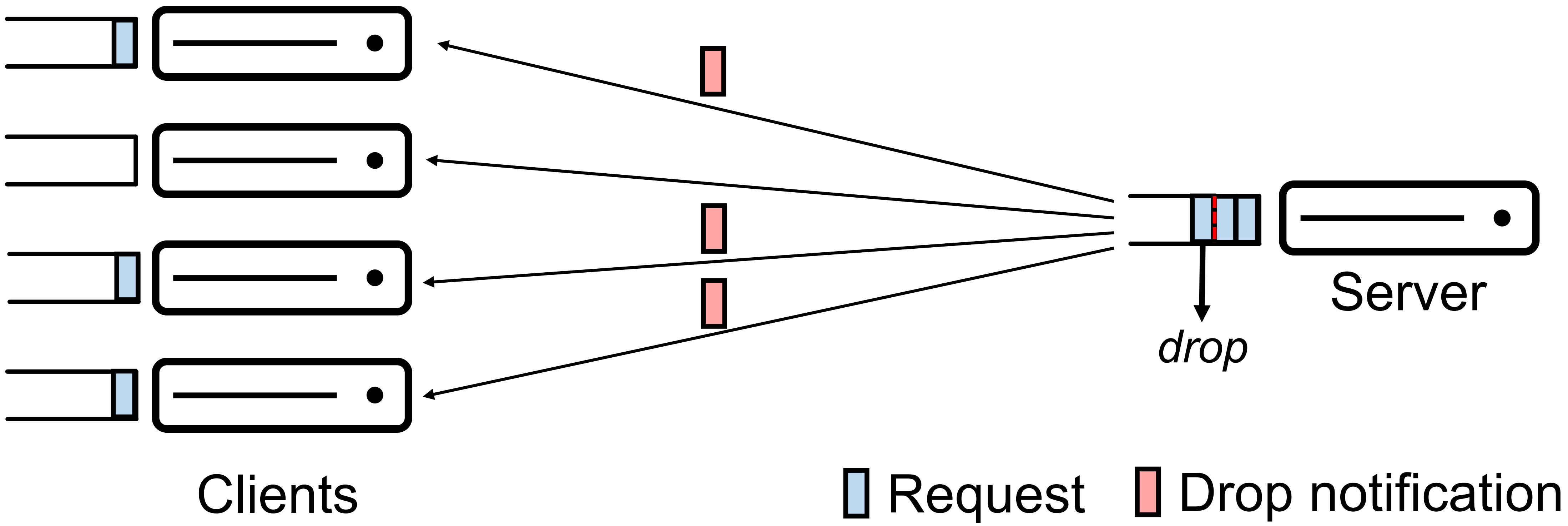
should inform clients about overload quickly



Strawman #1: Server-side AQM

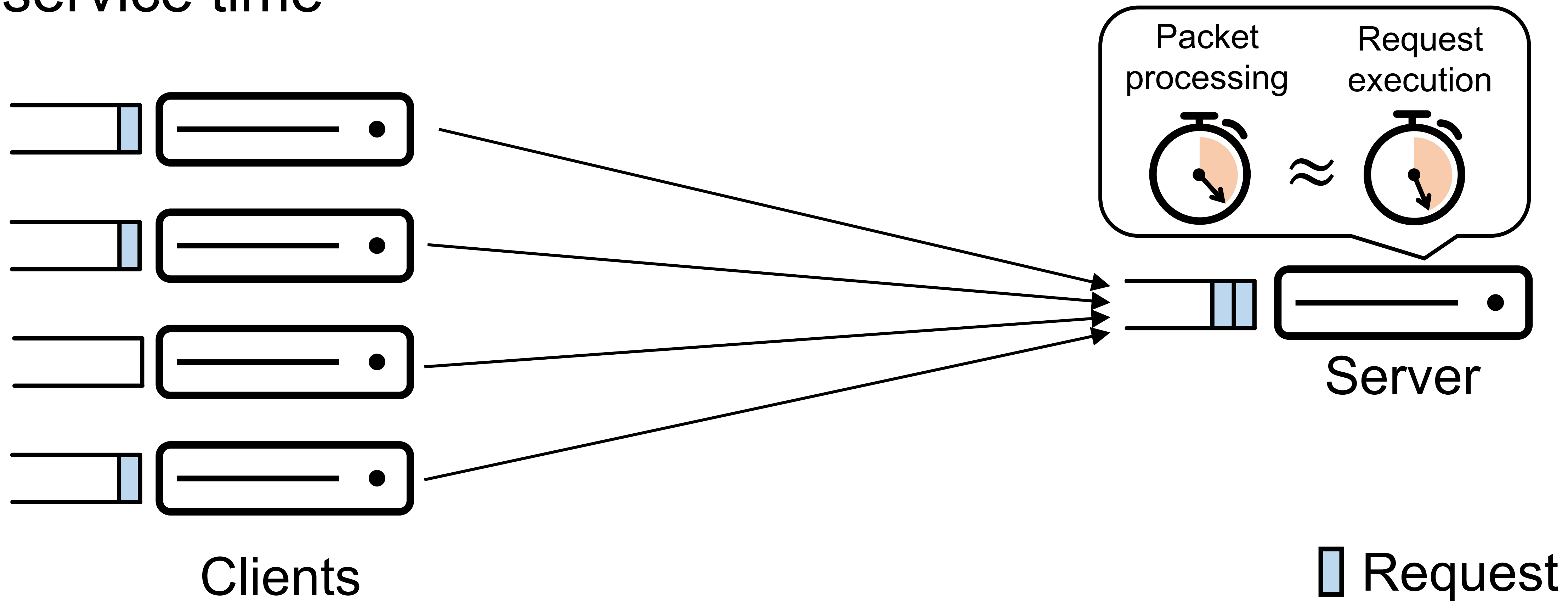


Strawman #1: Server-side AQM

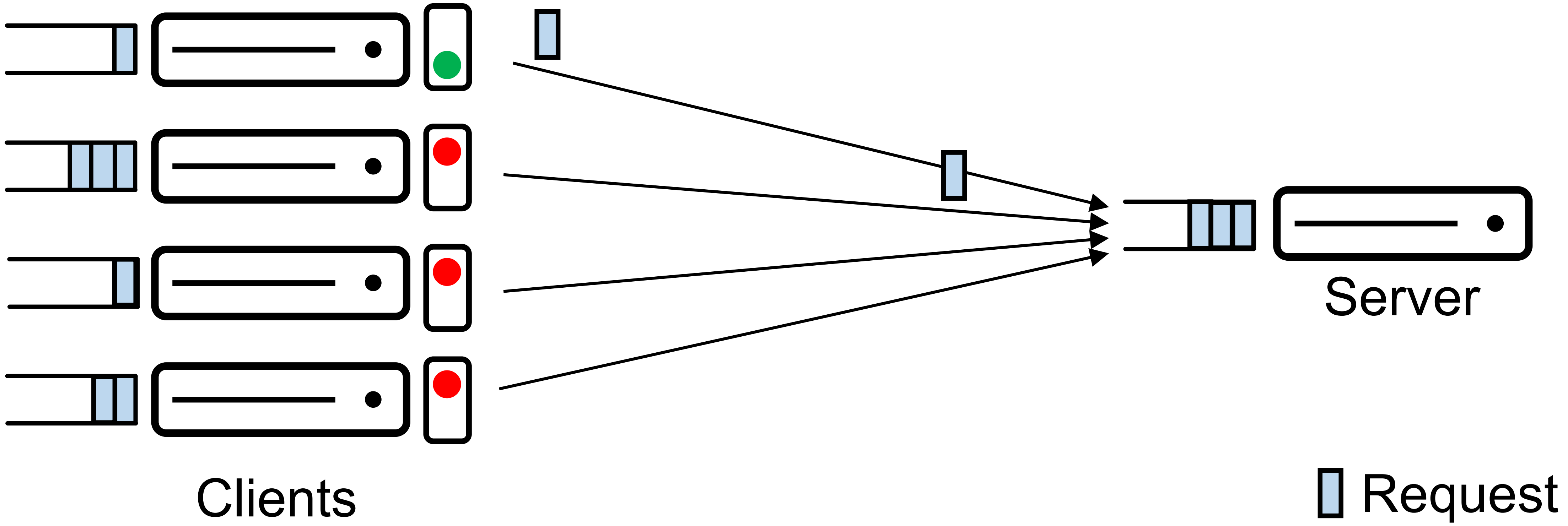


Strawman #1: Server-side AQM

The cost of packet processing is comparable to the service time

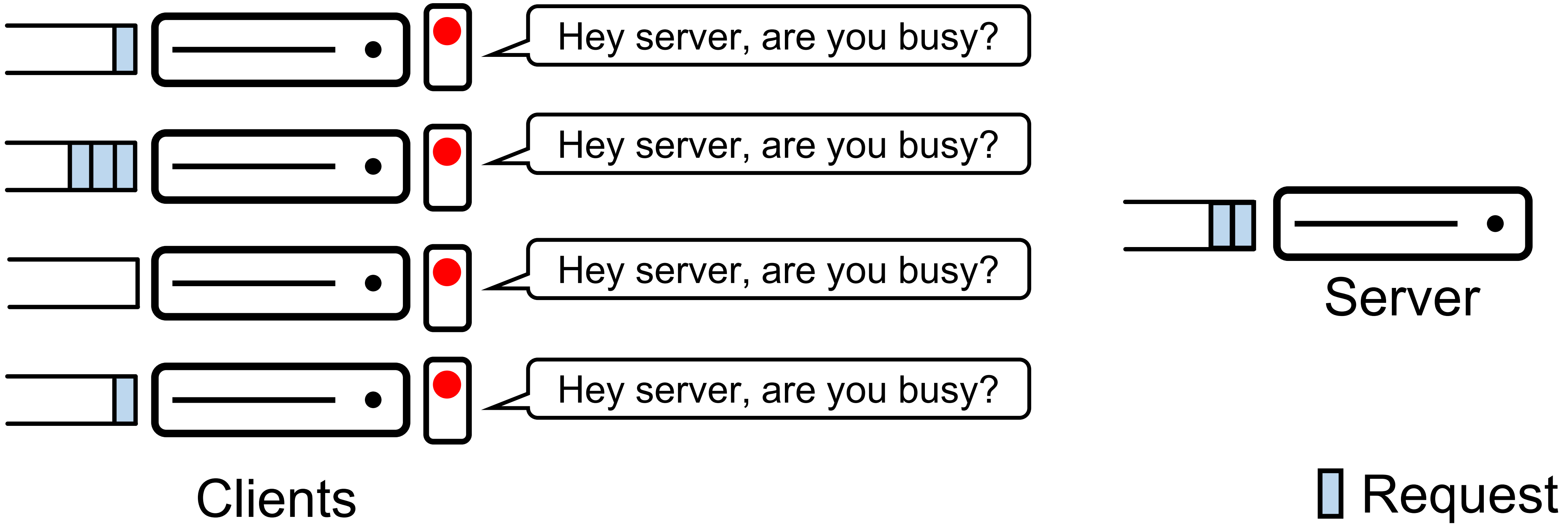


Strawman #2: Client Rate limiting



Strawman #2: Client Rate limiting

Probing server status incur high message overhead



Breakwater

Overload control scheme for µs-scale RPCs

Components	Benefits
1. Credit-based admission control	Coordinates requests with minimum delay
2. Demand speculation	Minimizes message overhead
3. Delay-based AQM	Ensures low tail latency

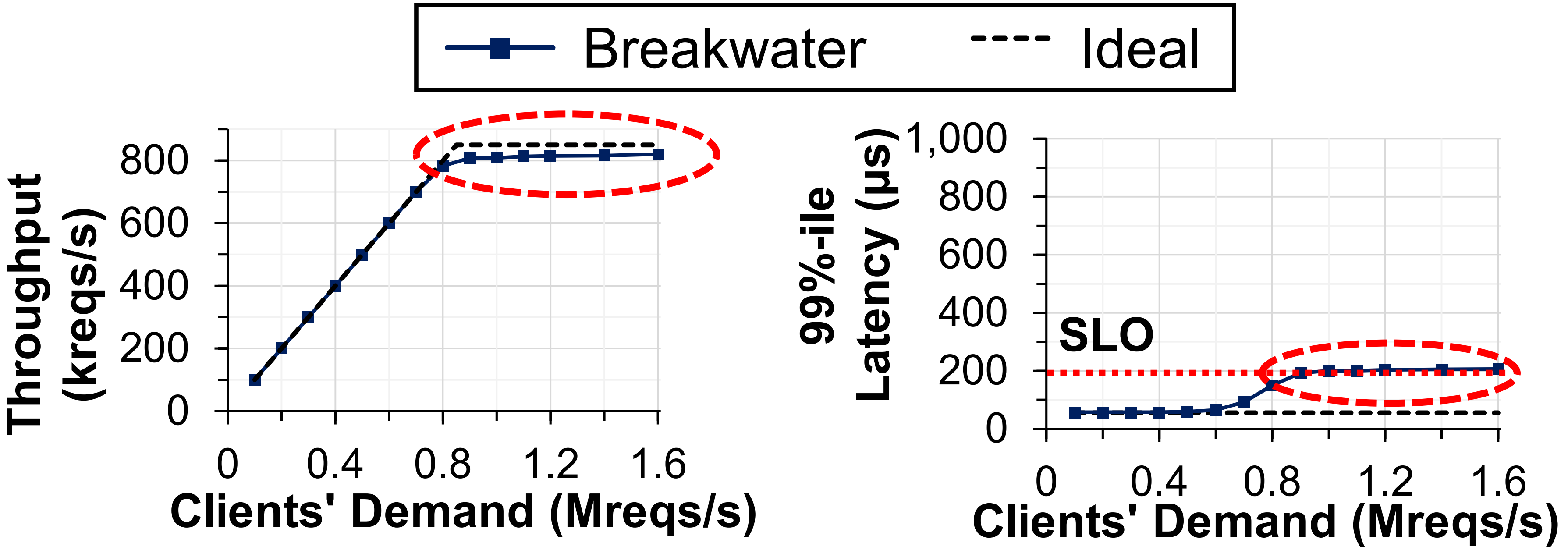
Breakwater's benefits

Handles server overload with **μs-scale RPCs** with

(1) High throughput

(2) Low and bounded tail latency

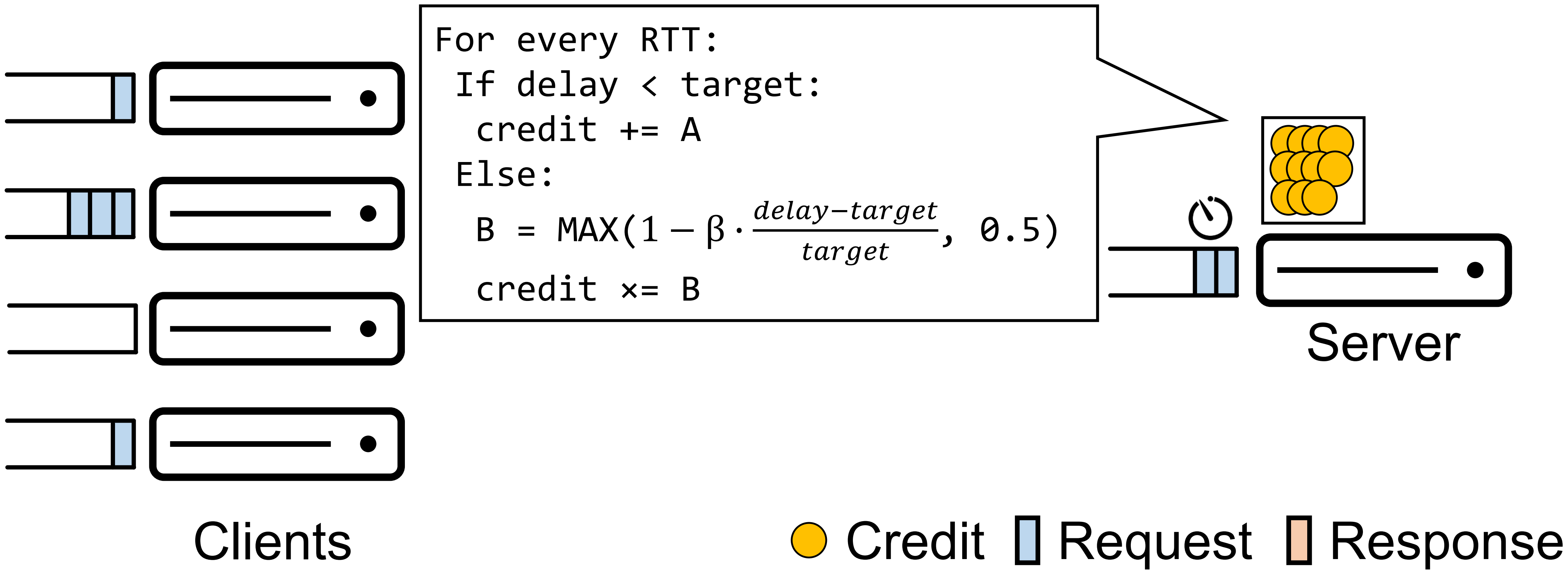
(3) Scalability to a large number of clients



Queueing delay as congestion signal

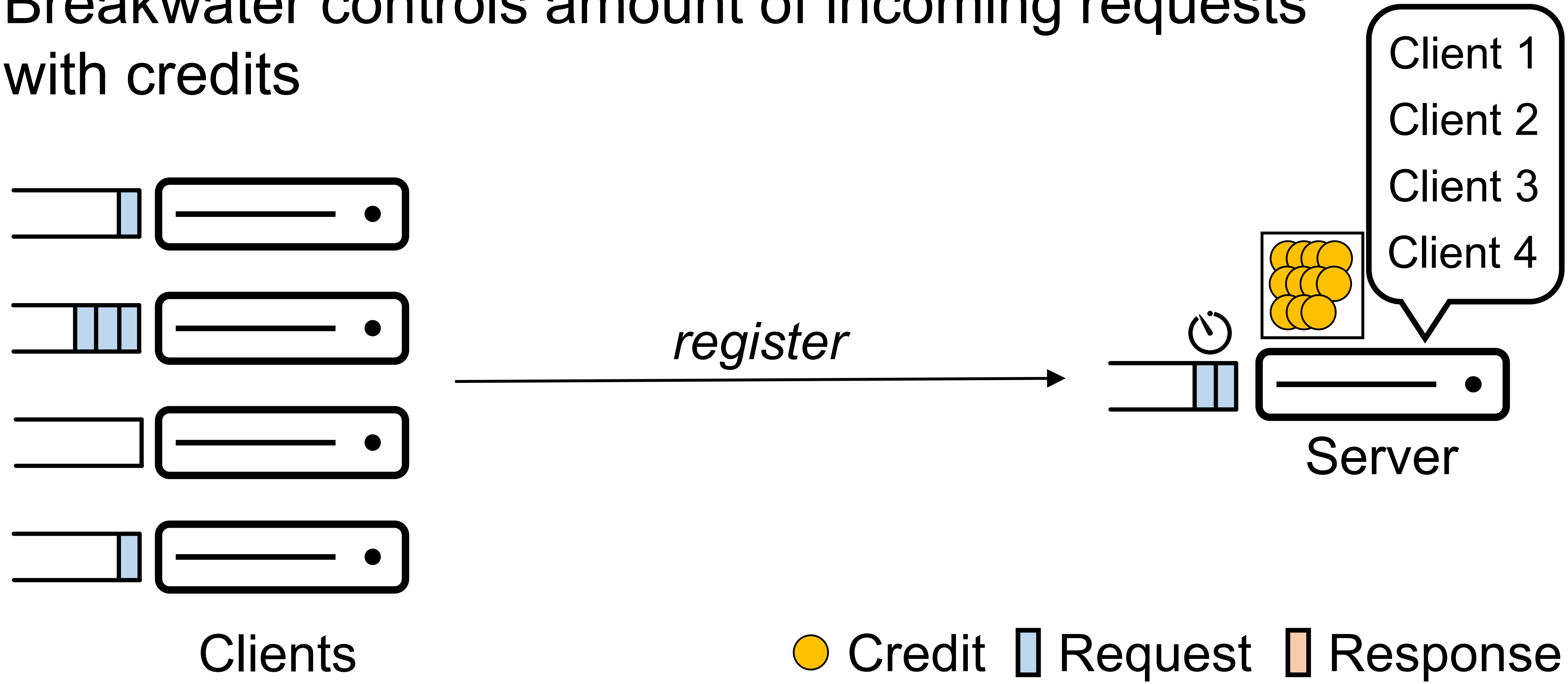
Comp. #1: Credit-based admission control

Breakwater controls amount of incoming requests with credits



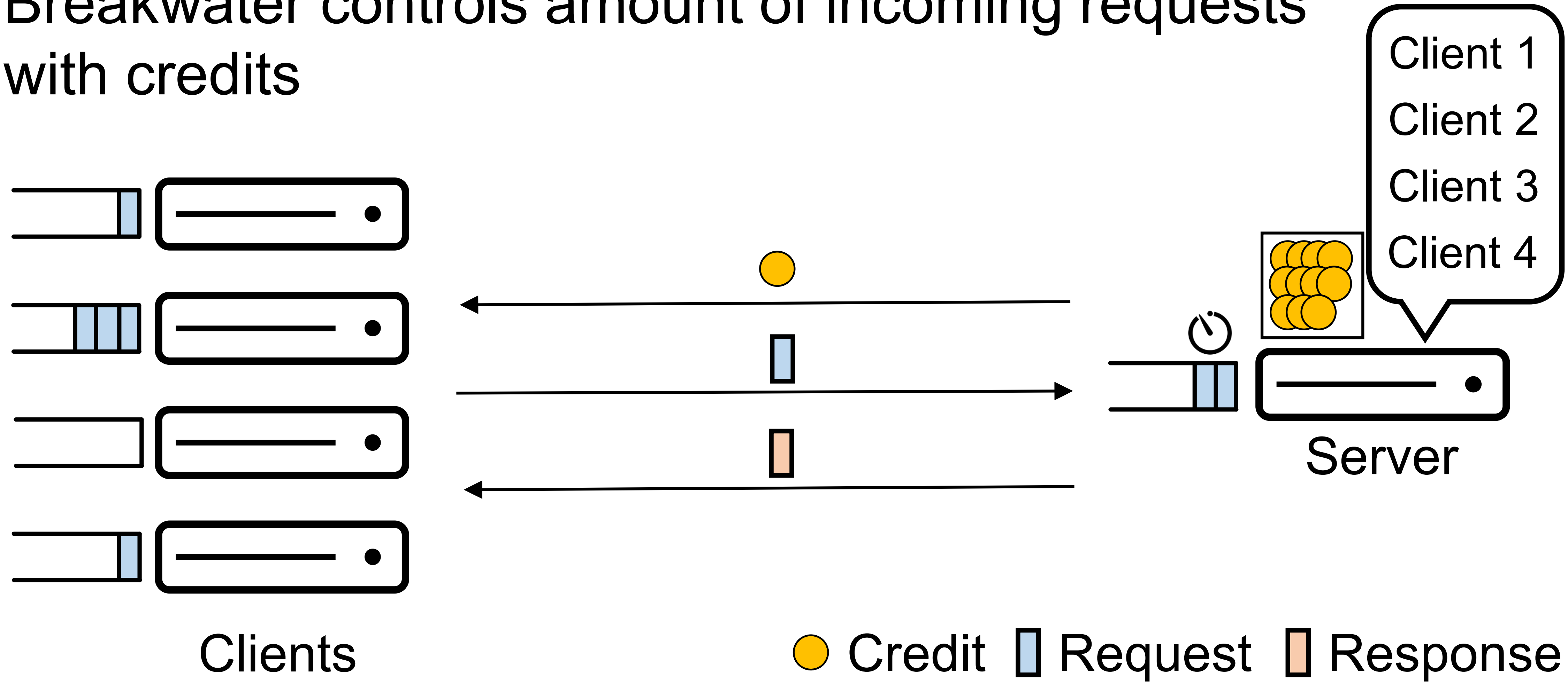
Comp. #1: Credit-based admission control

Breakwater controls amount of incoming requests with credits



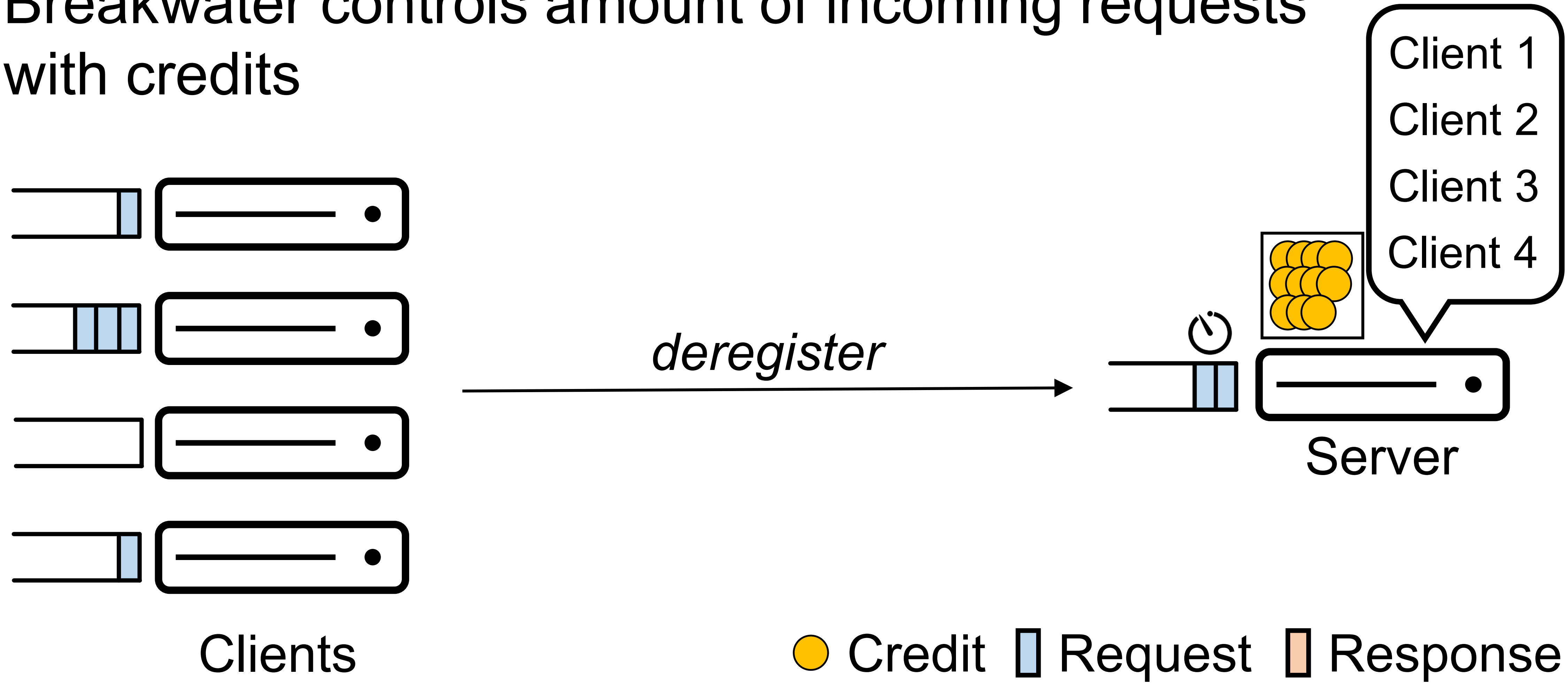
Comp. #1: Credit-based admission control

Breakwater controls amount of incoming requests with credits



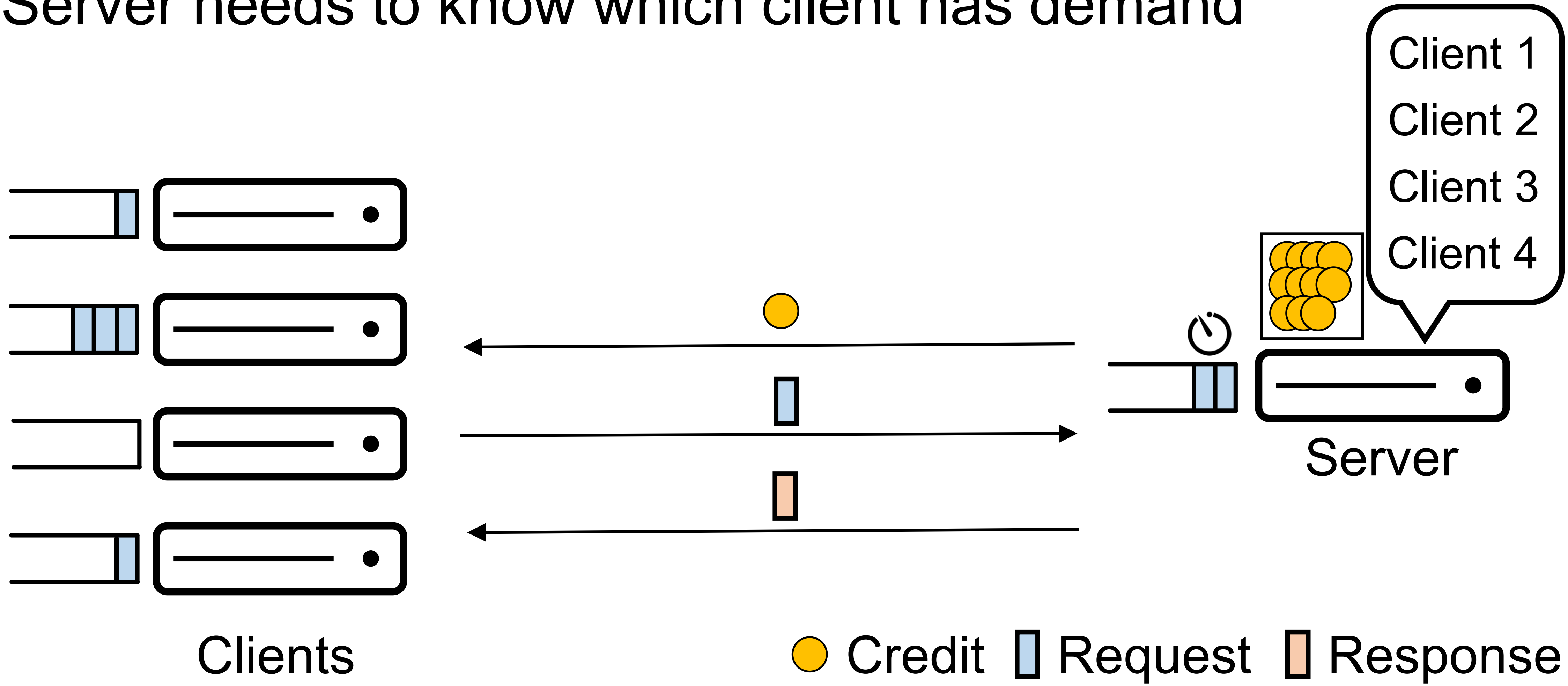
Comp. #1: Credit-based admission control

Breakwater controls amount of incoming requests with credits



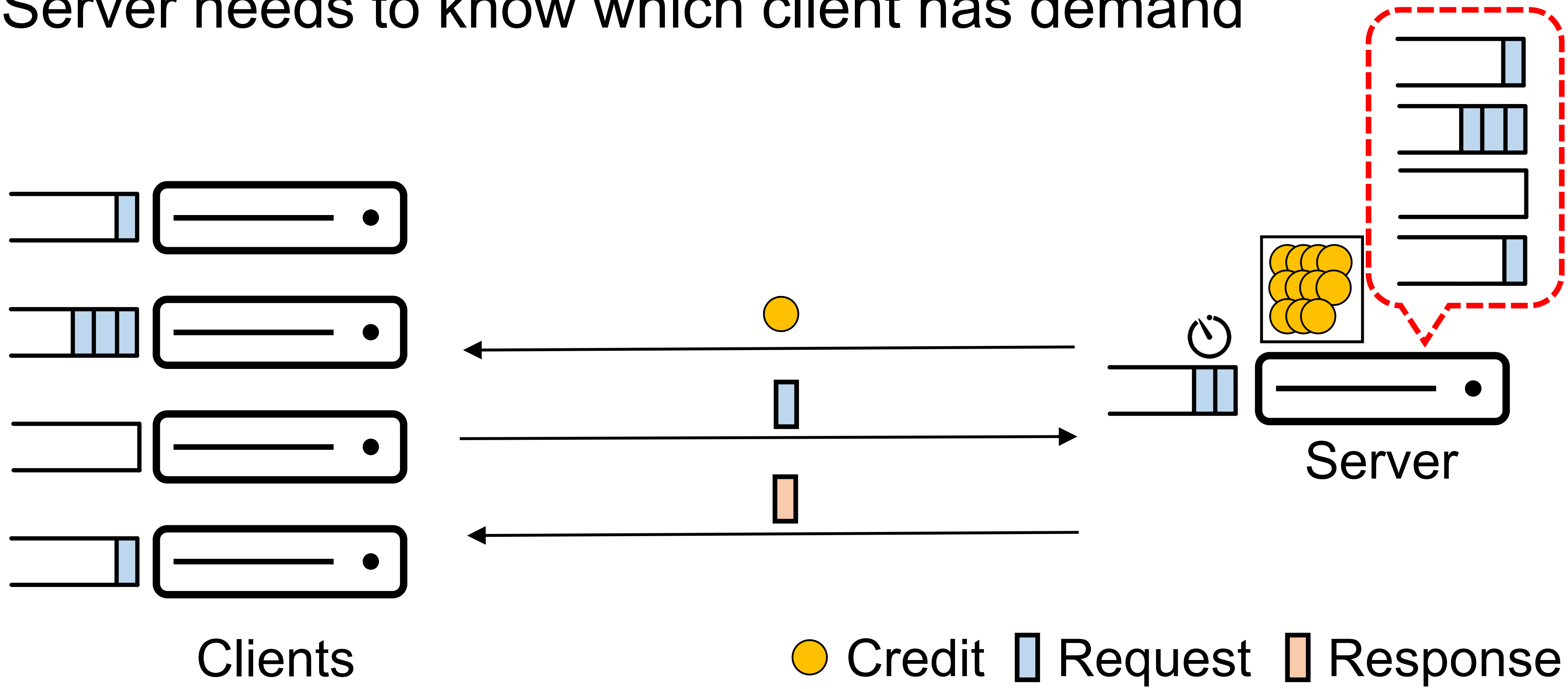
Demand Message Overhead

Server needs to know which client has demand



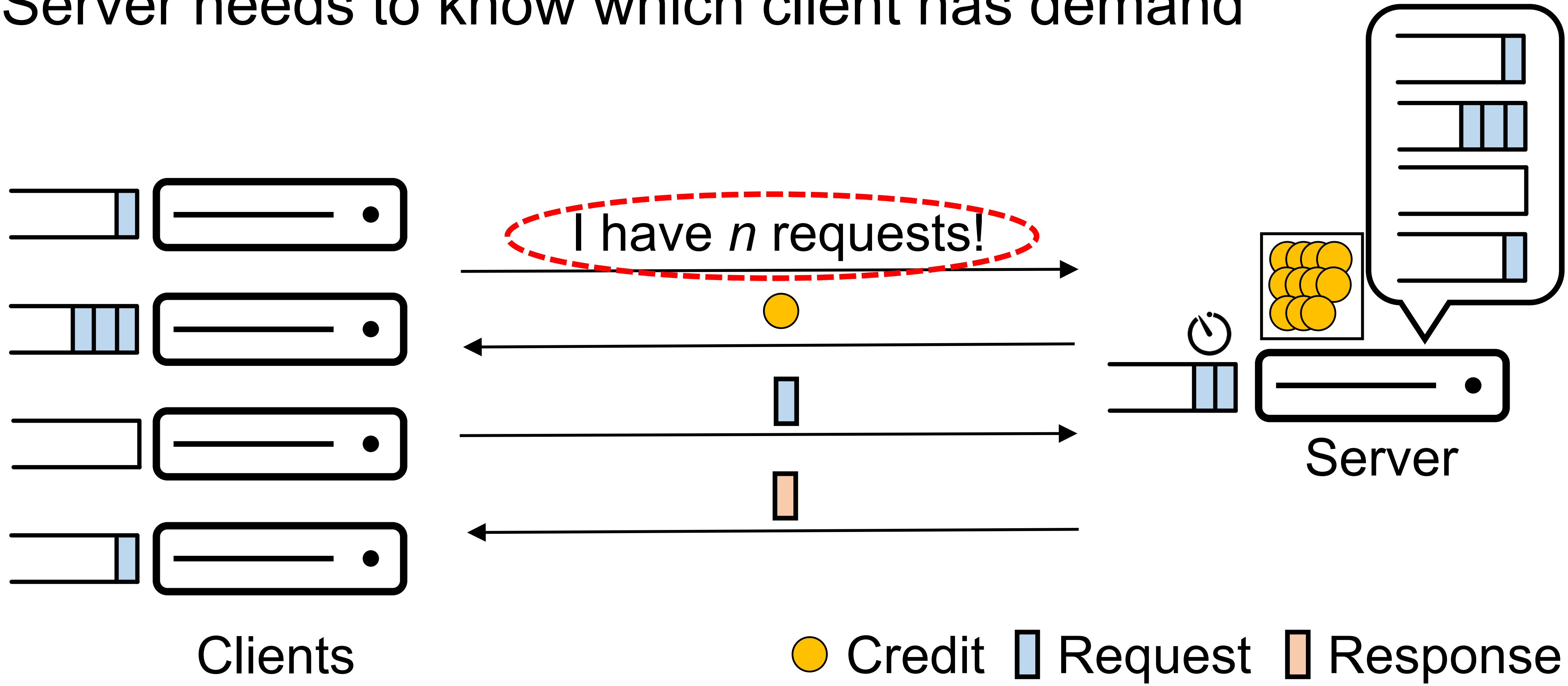
Demand Message Overhead

Server needs to know which client has demand



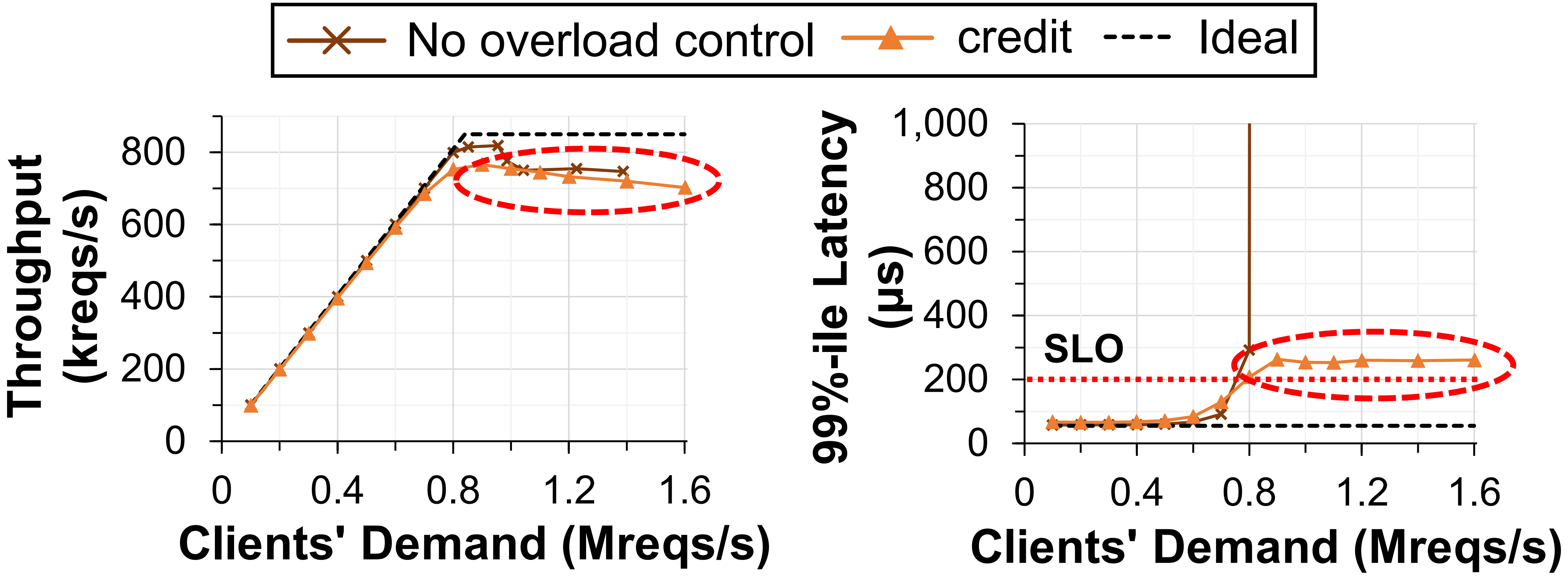
Demand Message Overhead

Server needs to know which client has demand



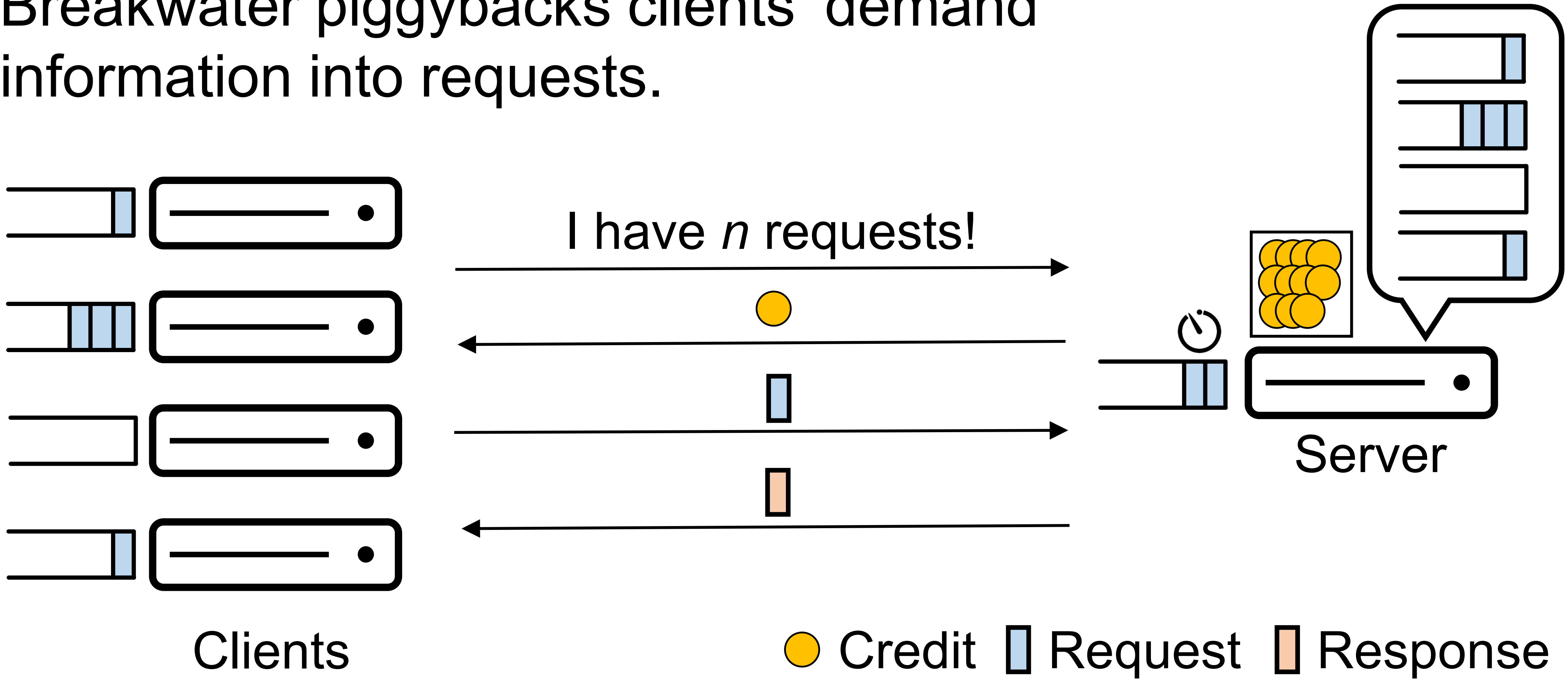
Impact of Credit-based Admission Control

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



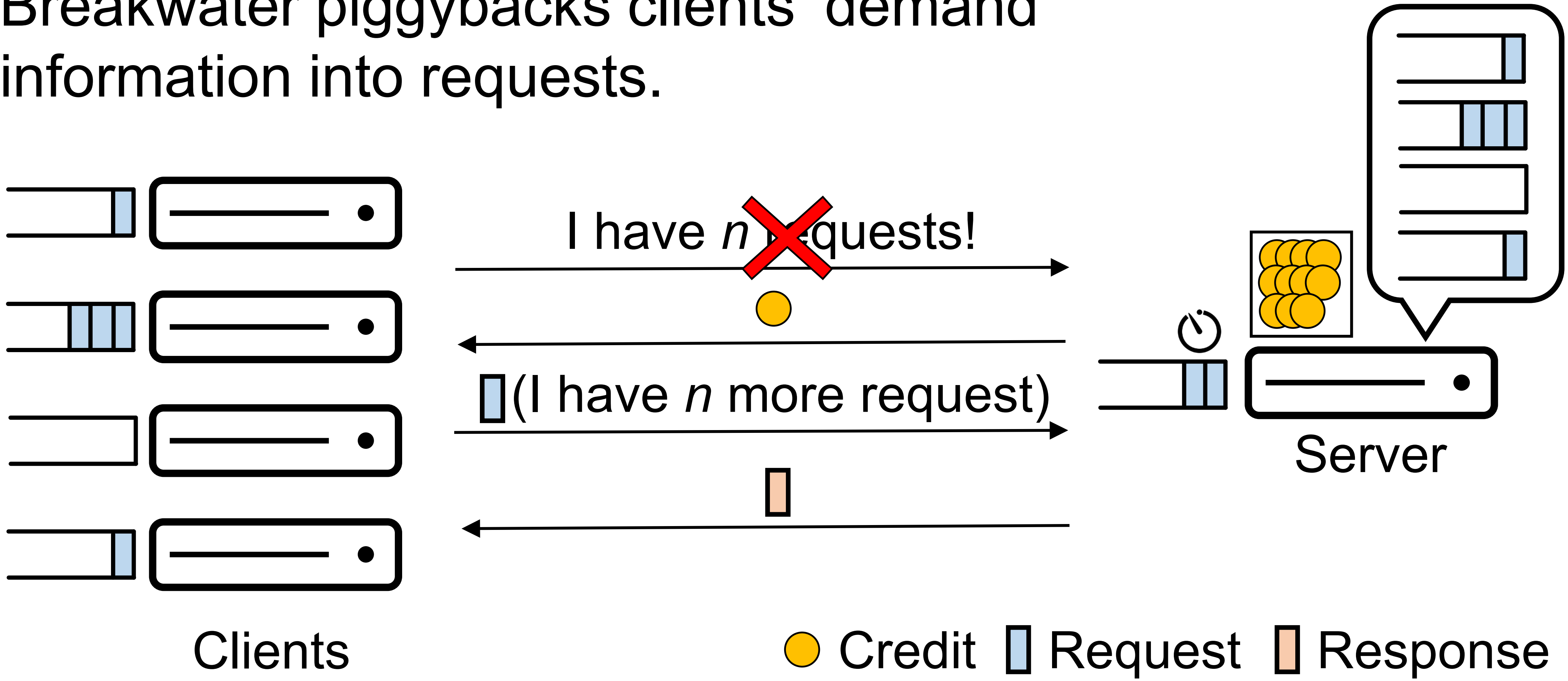
Piggybacking Demand Information

Breakwater piggybacks clients' demand information into requests.



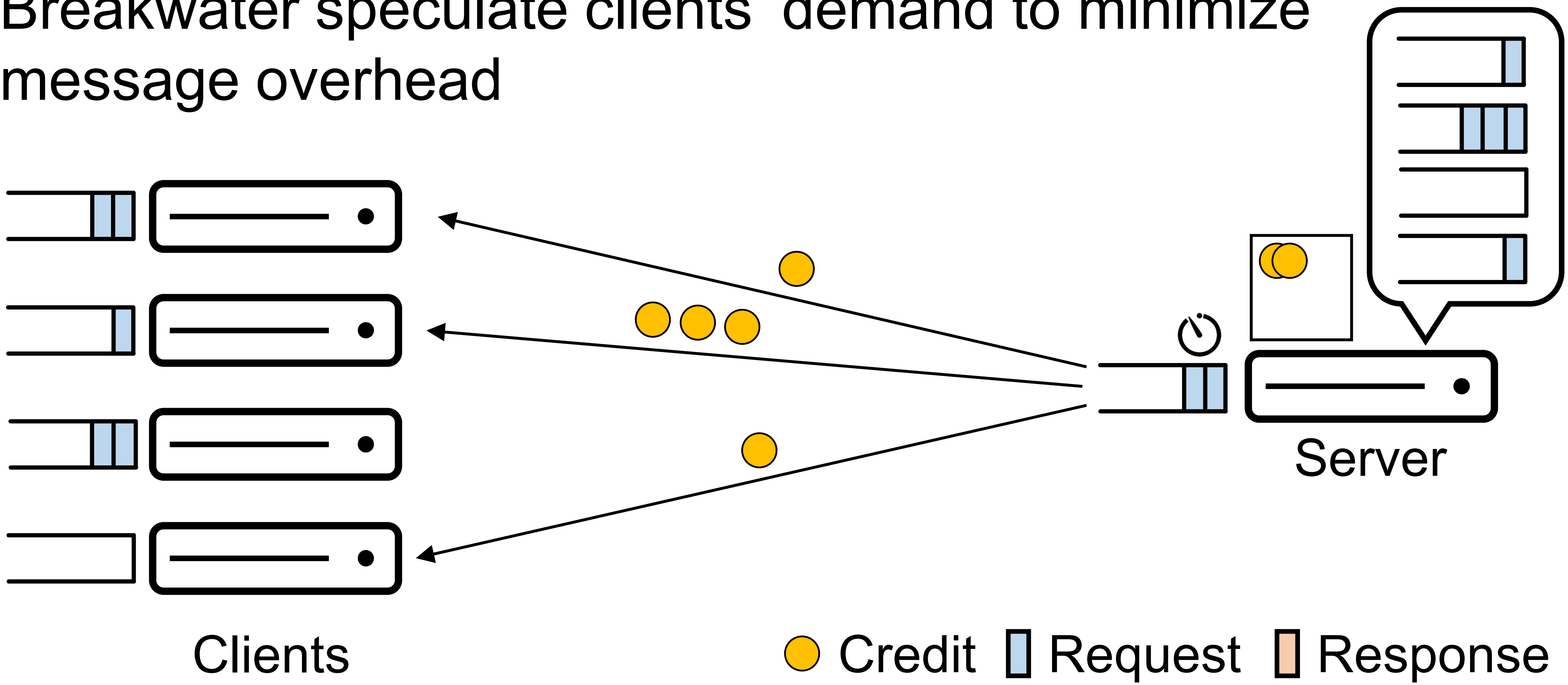
Piggybacking Demand Information

Breakwater piggybacks clients' demand information into requests.



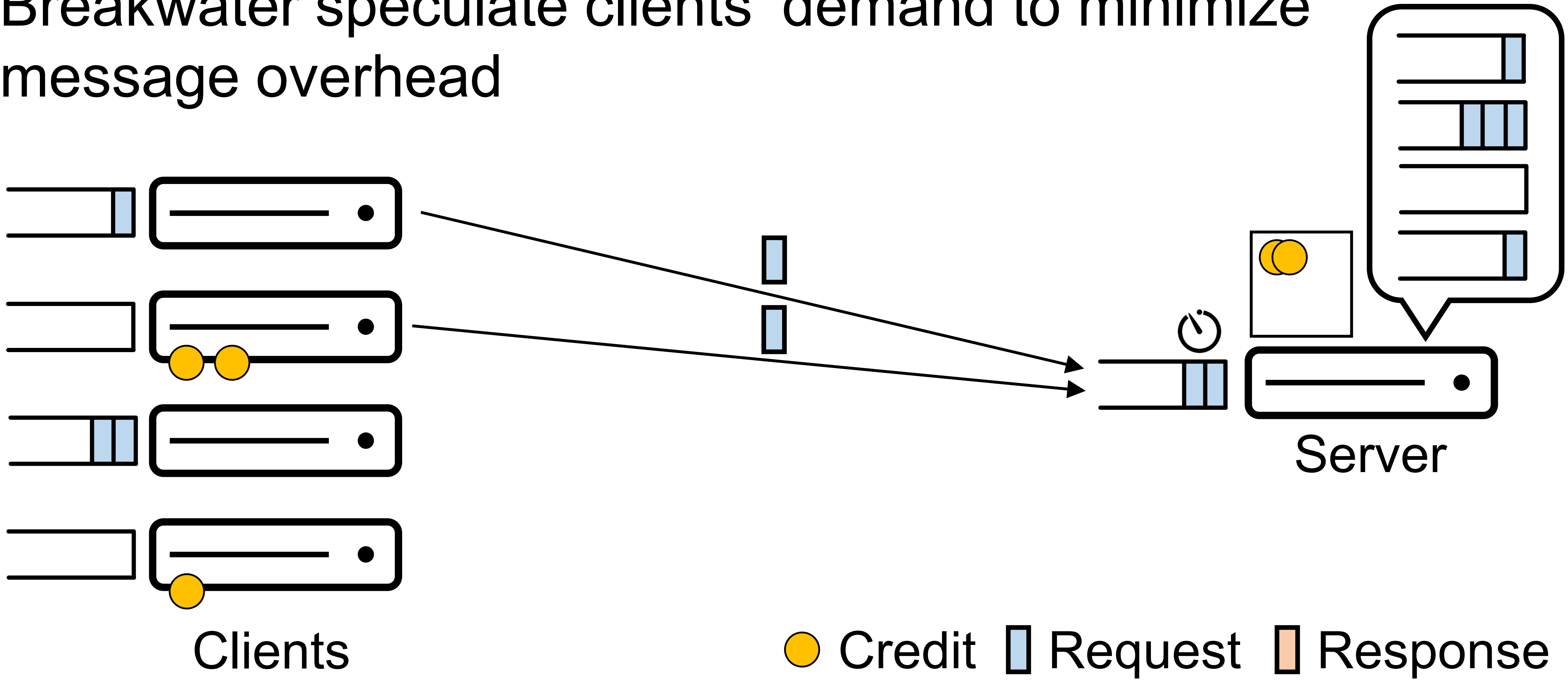
Comp. #2: Demand Speculation

Breakwater speculate clients' demand to minimize message overhead



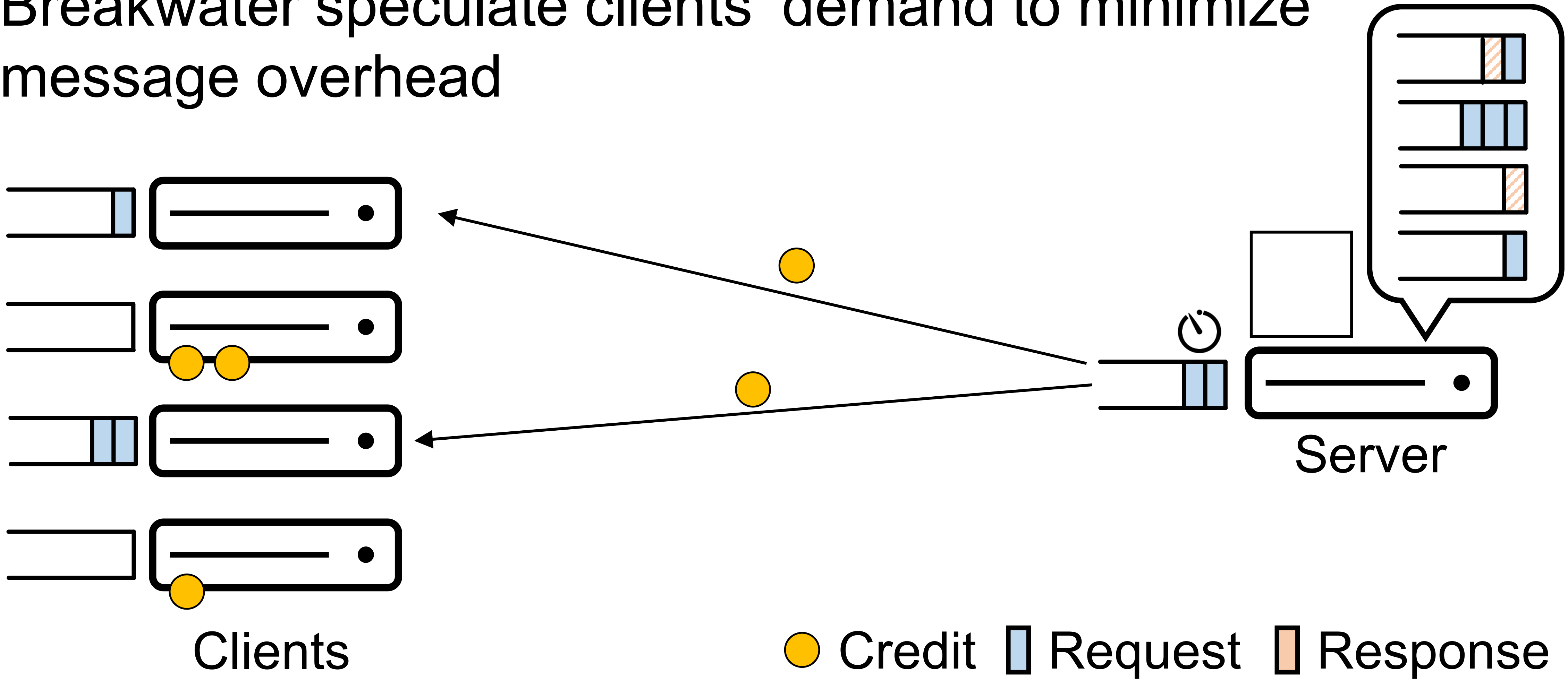
Comp. #2: Demand Speculation

Breakwater speculate clients' demand to minimize message overhead



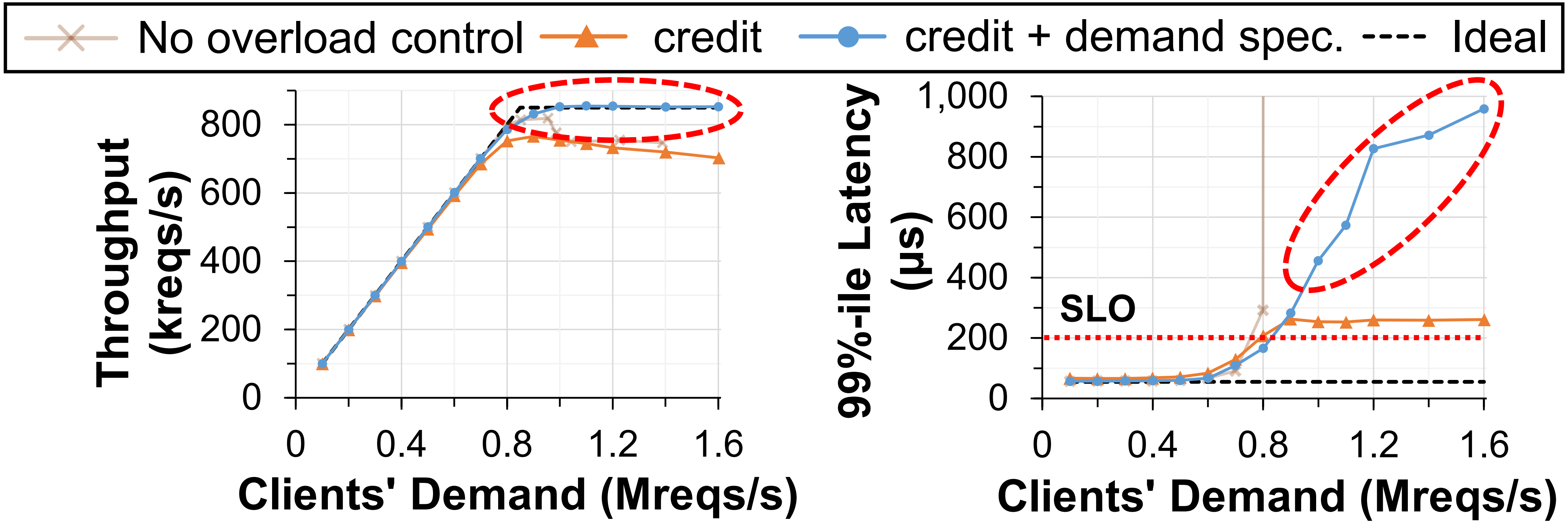
Comp. #2: Demand Speculation

Breakwater speculate clients' demand to minimize message overhead



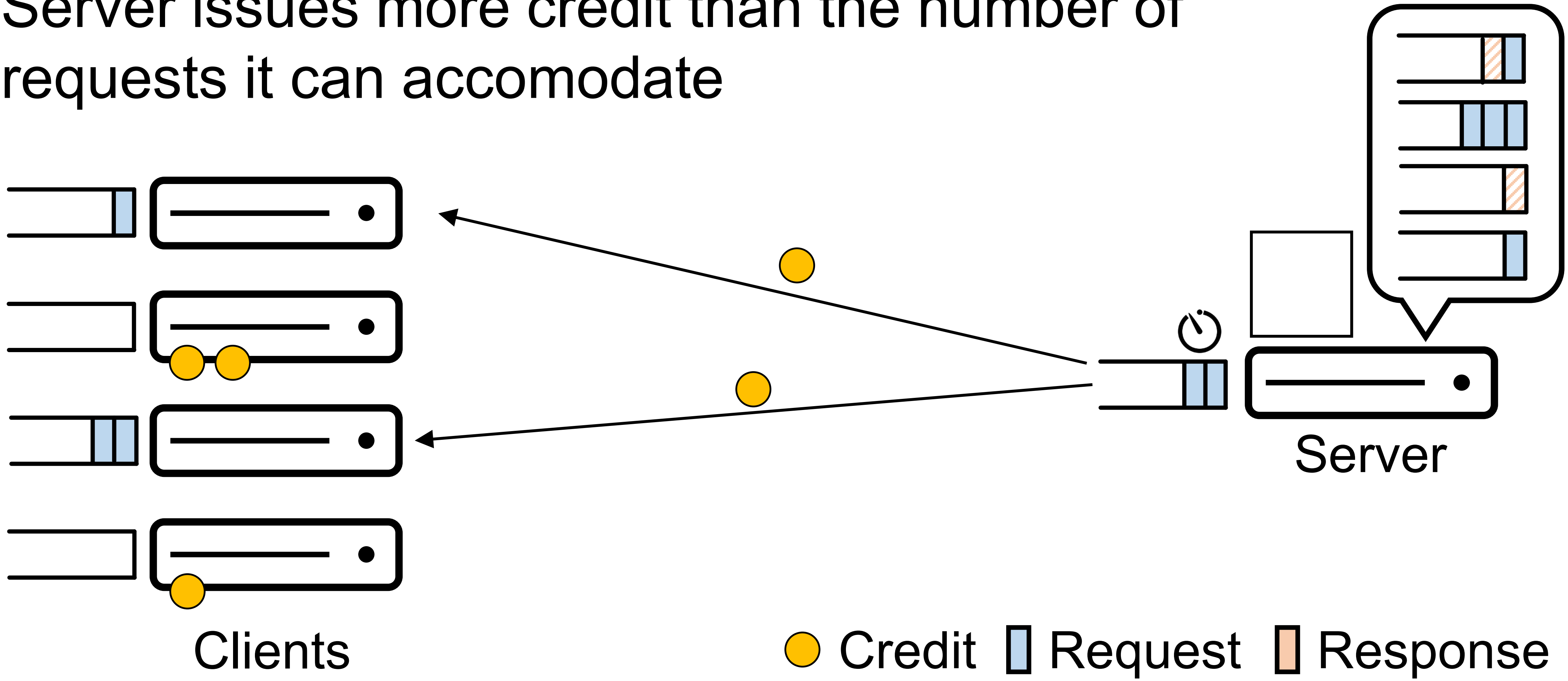
Impact of Adding Demand Speculation

Demand speculation improves throughput with higher tail latency



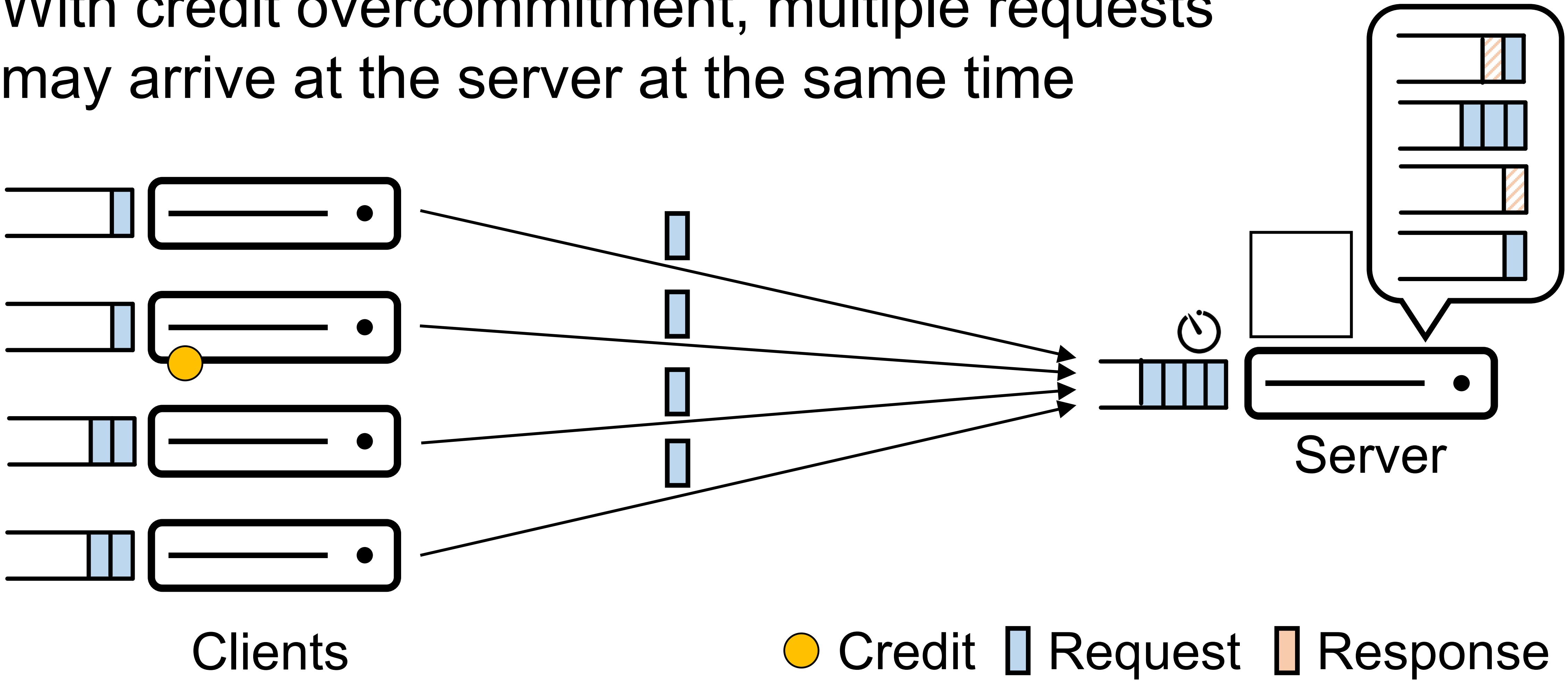
Credit Overcommitment

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



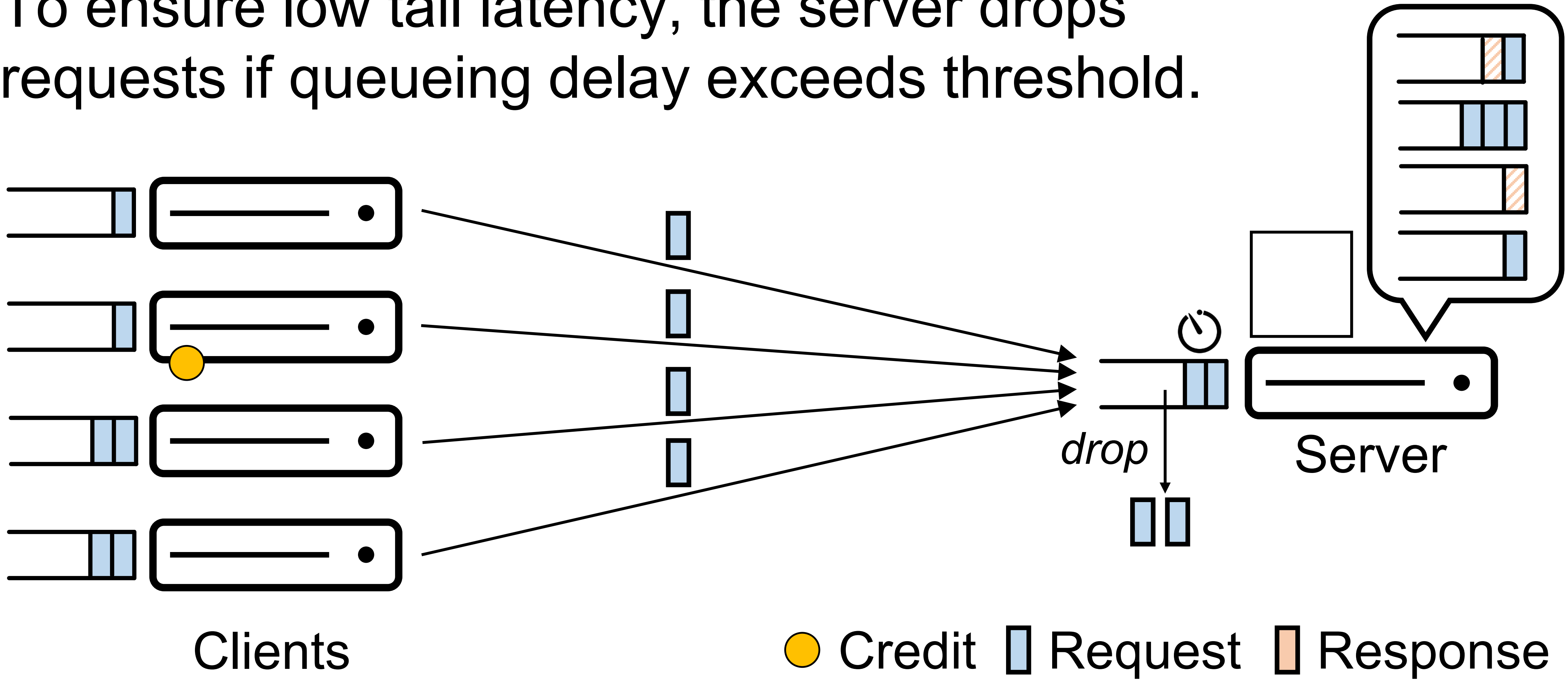
Incast Causing Long Queue

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



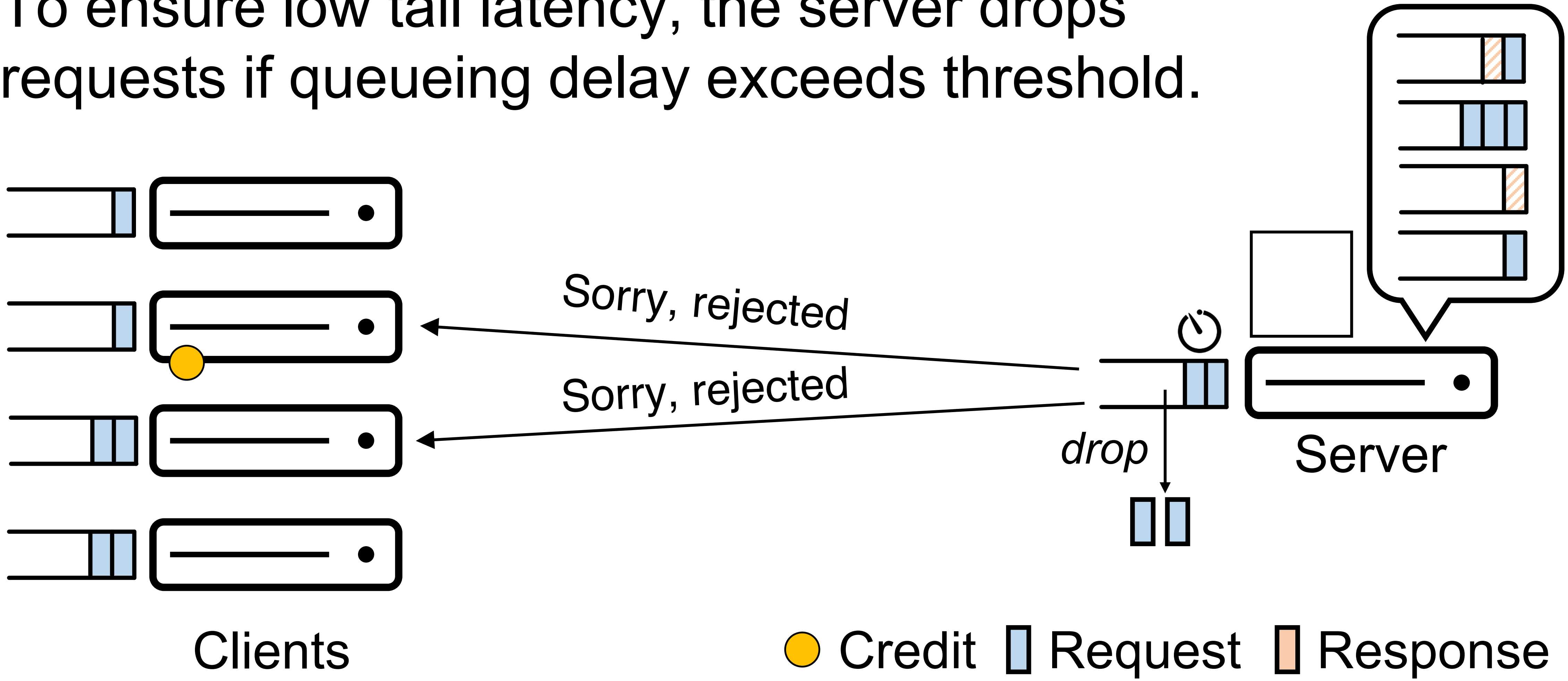
Comp. #3: Delay-based AQM

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



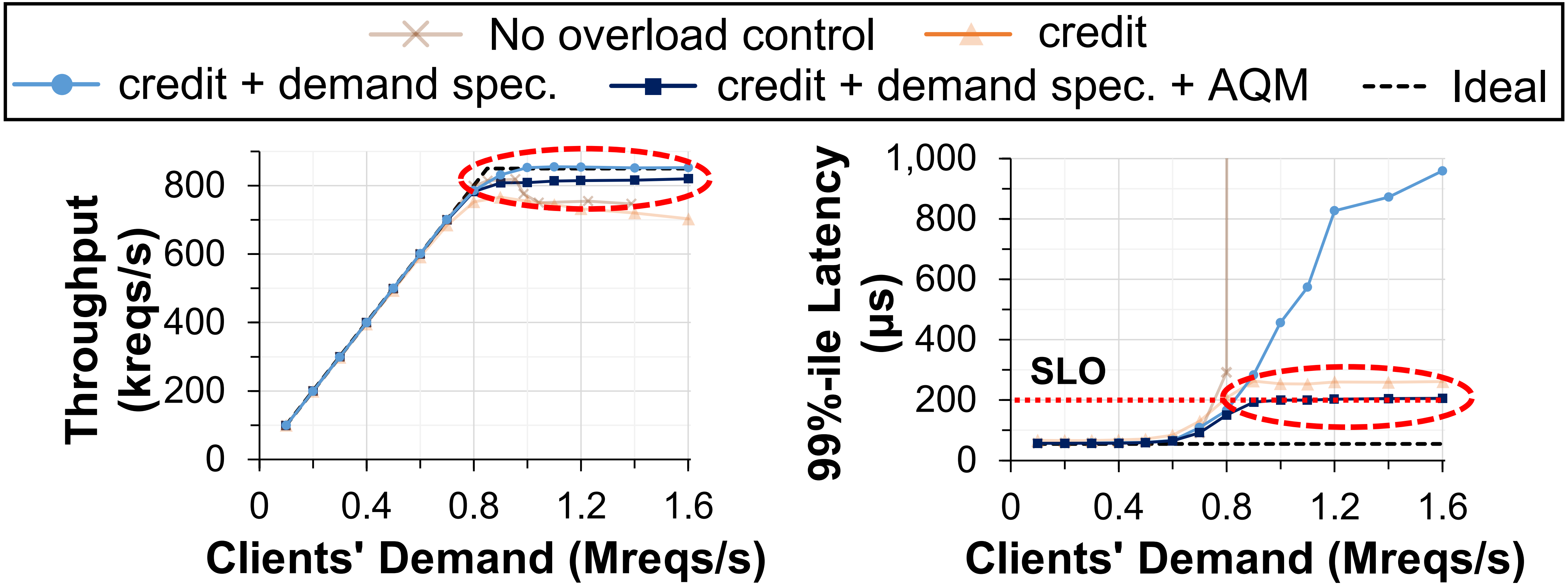
Comp. #3: Delay-based AQM

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 notification for the rejected requests?
(3) Is Breakwater scalable to many clients?

Baselines:

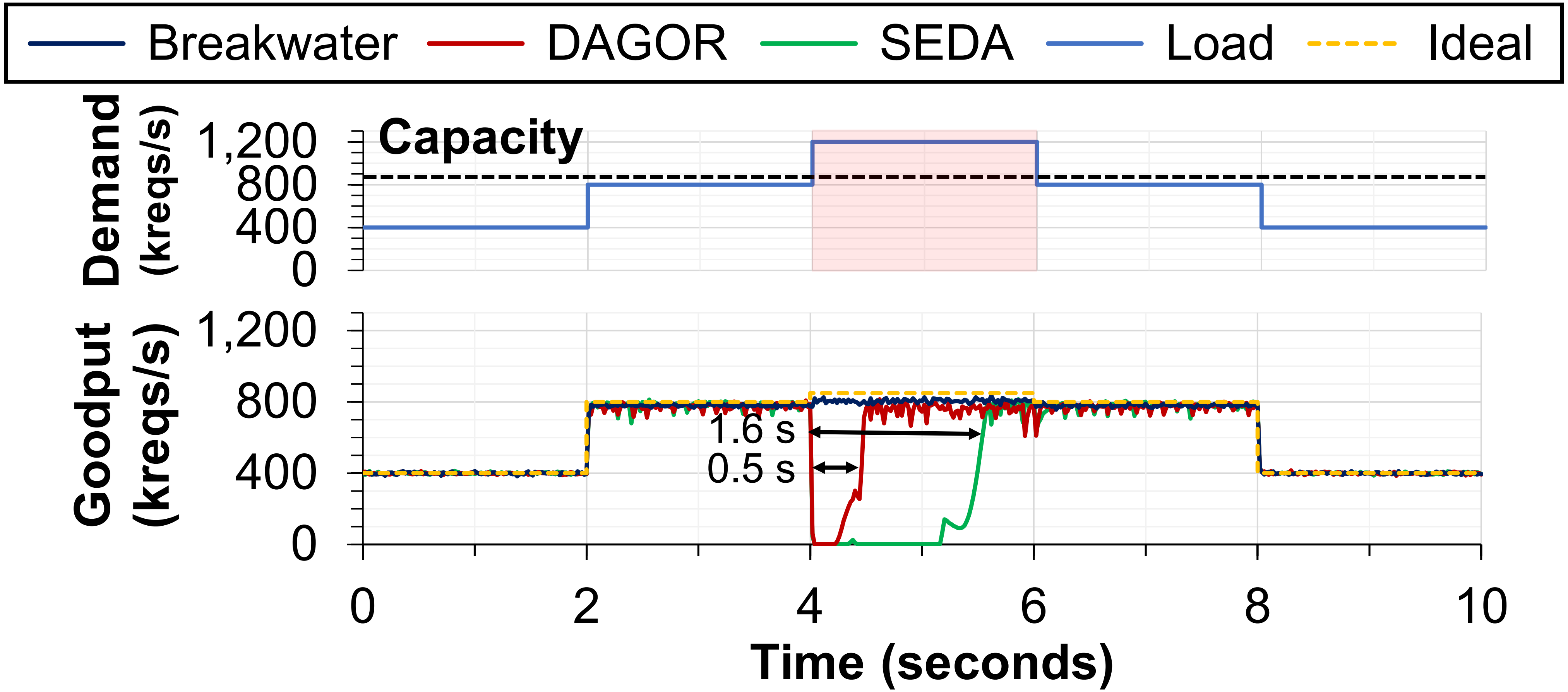
DAGOR

priority-based overload control used in WeChat

SEDA

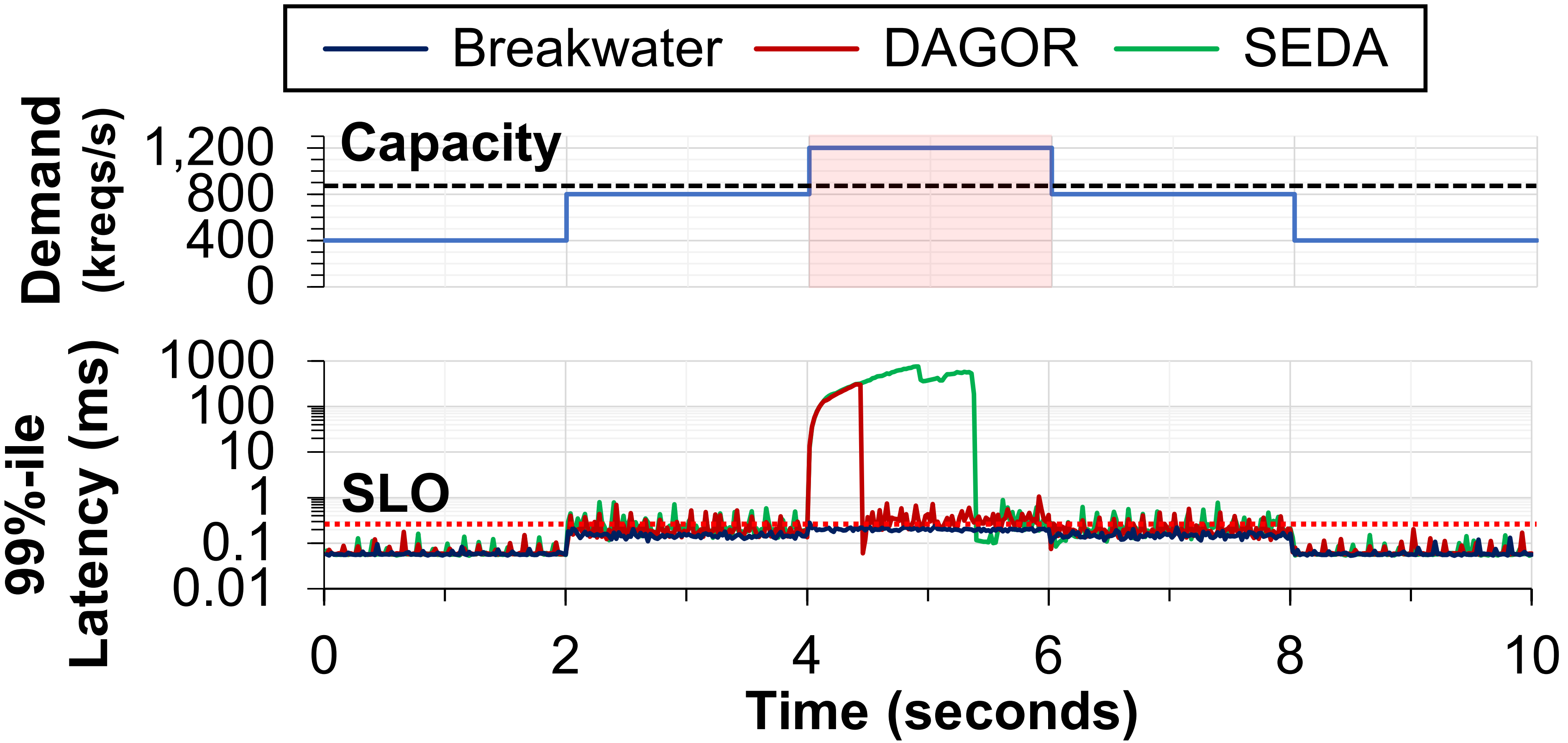
adaptive overload control for staged event-driven architecture

High Goodput with Fast Convergence



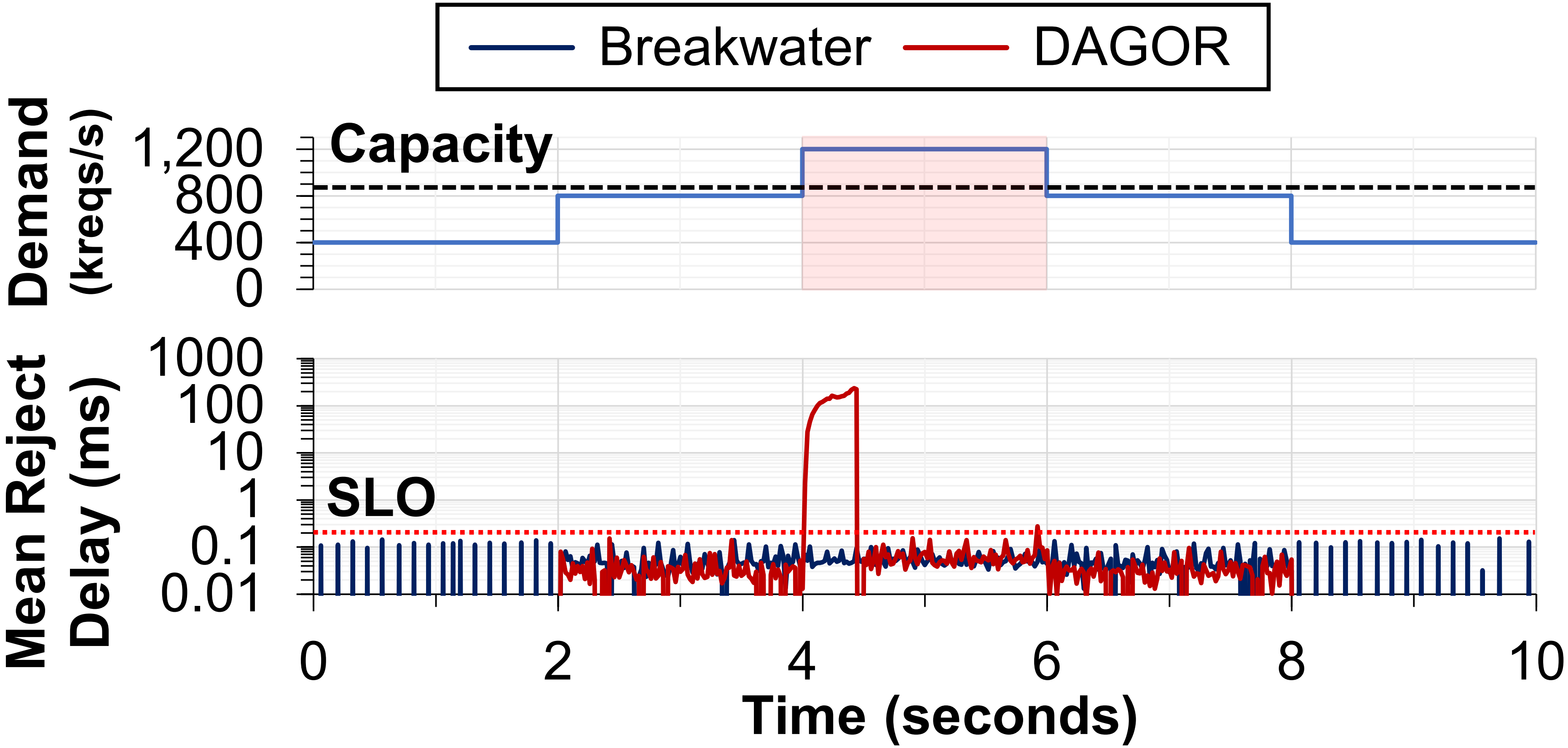
Breakwater converges to higher goodput 25x faster than DAGOR and 79x faster than SEDA.

Low and Bounded Tail Latency



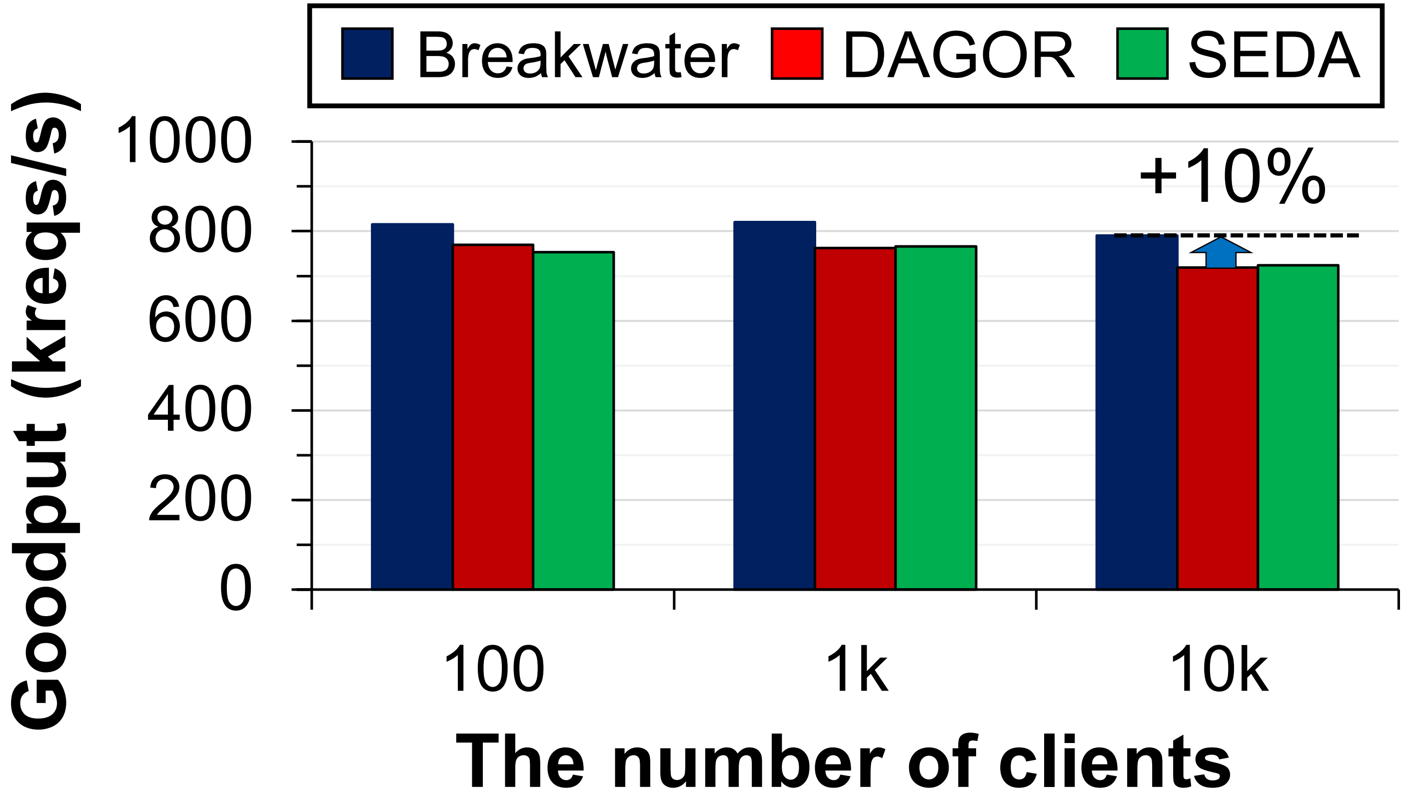
Breakwater maintains low tail latency even with load spikes.

Fast Notification of Reject



Breakwater notifies rejected request to clients before violating its SLO.

Scalability to Many Clients



Breakwater easily scales to 10,000 clients.

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 notification** 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>