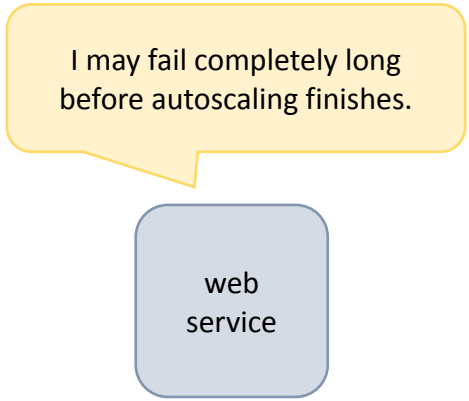
Load shedding

autoscaling takes time

- monitoring system needs time to collect and aggregate metric data
- provisioning new machines takes time
- deploying services and bootstrapping on new machines takes time

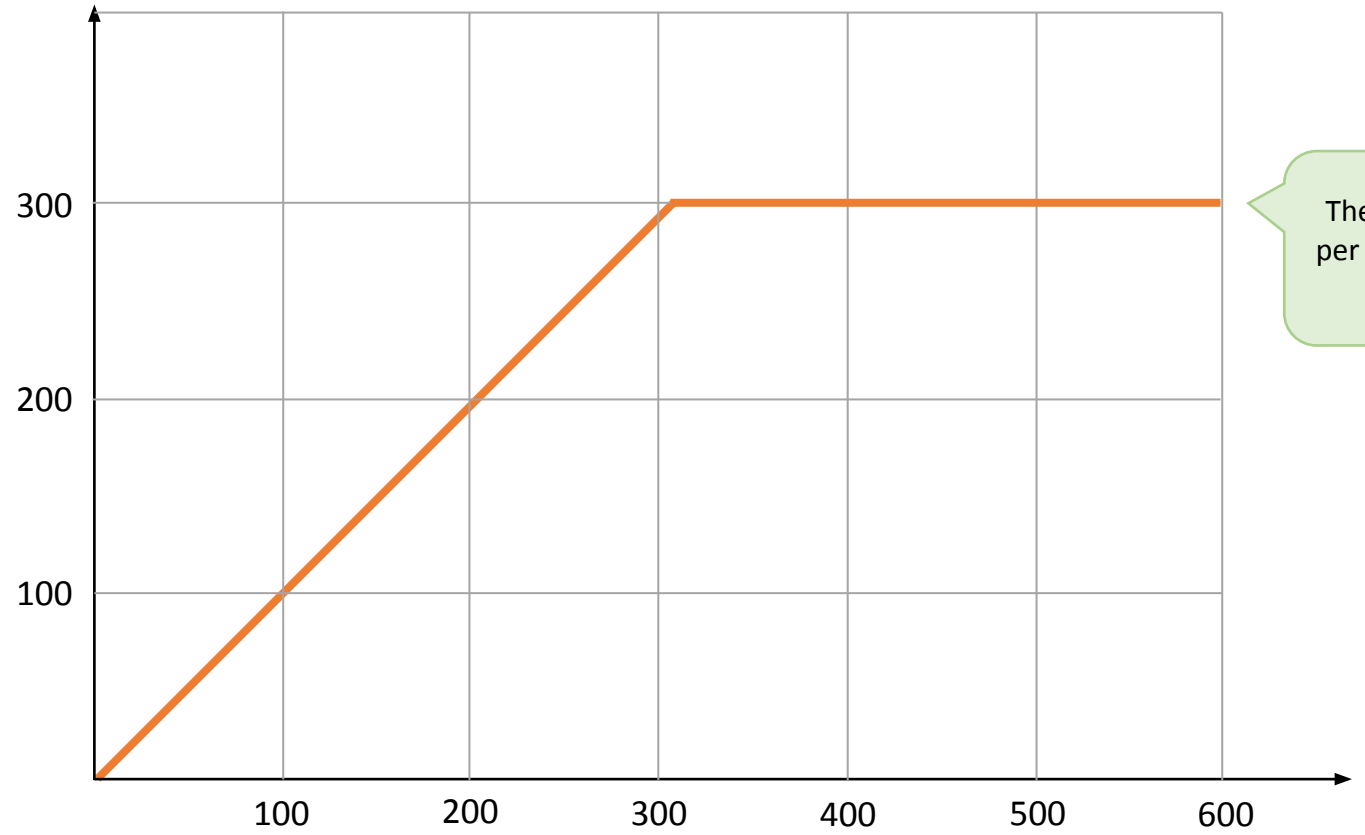


load shedding and rate limiting to the rescue

Load shedding

when the server approaches overload,
it starts to drop incoming requests

number of requests
successfully processed on the server
(per second)



number of requests
sent to the server
(per second)

To determine this value we run a load test and evaluate metrics (CPU, memory, latency)

The server accepts 300 requests per second and drops all requests over this limit.

Load shedding

implementation options

thread per connection

limit the number of
connections

limit the number of
request processing threads
(same effect as limiting connections)

monitor system performance metrics and
drop requests when the threshold is reached

thread per request

with non-blocking I/O

limit the number of
connections

limit the number of
request processing threads

monitor system performance metrics and
drop requests when the threshold is reached

event loop

limit the number of
connections

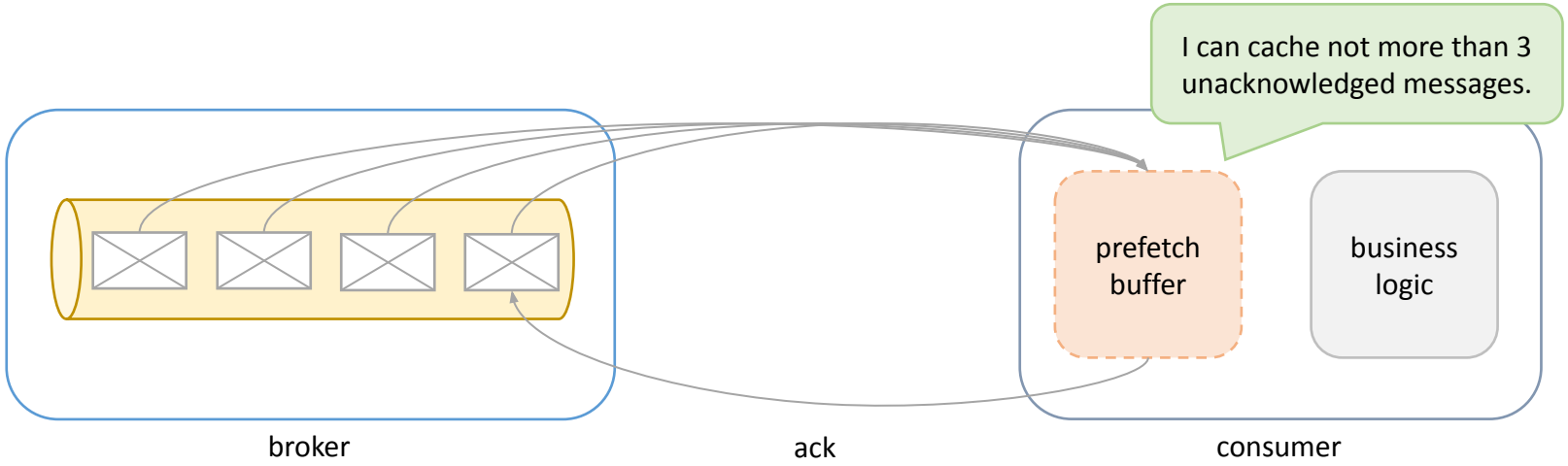
limit the size of
the task queue

monitor system performance metrics and
drop requests when the threshold is reached

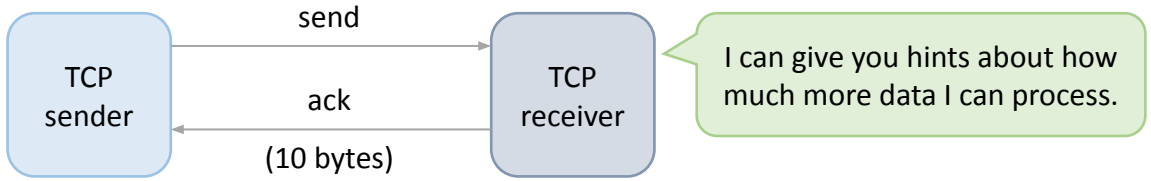
Load shedding

example

consumer prefetch (RabbitMQ)
helps avoid overloading the consumer with too many messages



TCP flow control



Load shedding

important considerations

- request priority

  health checks, people over robots

- request cost

  drop expensive requests first

- request duration

  LIFO over FIFO, timeout hints

- autoscaling

  autoscaling threshold < load shedding threshold