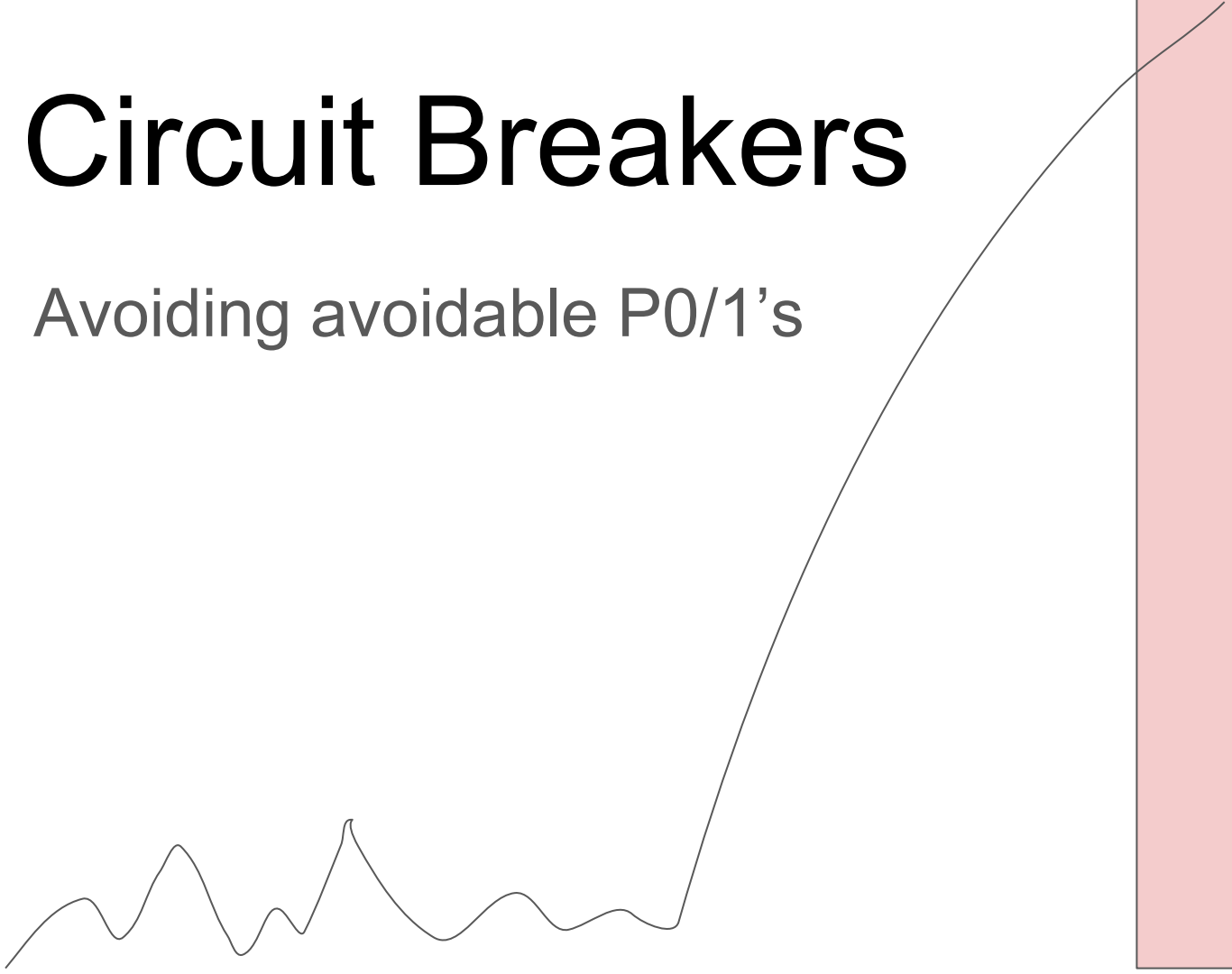
Circuit Breakers

Avoiding avoidable P0/1’s



Garrow Bedrossian
@grokfail

Follow

"if you switch one of the microservices off and anything else breaks, you don't really have a microservice architecture, you just have a distributed monolith!"
Domain Modelling made functional, by Scott Wlaschin

9:09 PM - 19 May 2018

291 Retweets **651** Likes

Synchronous Requests are The Devil™

Sync requests should be avoided in the web thread

Offload things via Sidekiq/AMQP whenever you can

Return a 201 Accepted, and find a way to communicate with your APIs clients

If client is backend system, use webhooks (hit an endpoint)

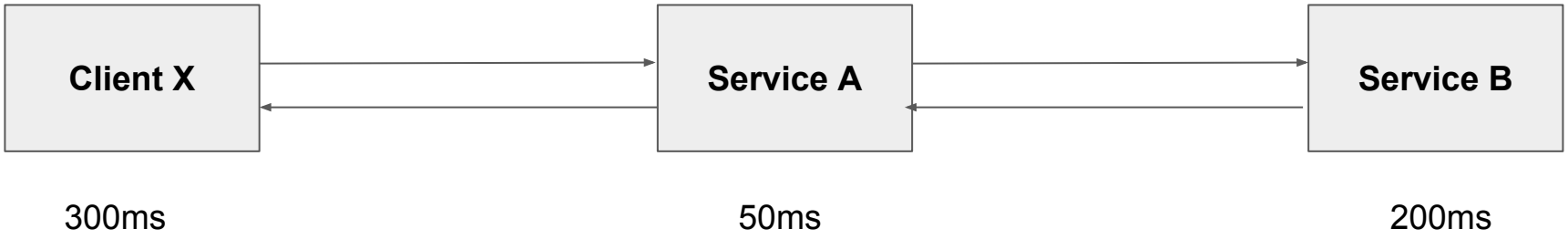
If client is mobile/web/etc, return a websocket URL to subscribe for updates

A Common Sync Call

Service A aims to respond in 300ms

Service B usually responds in 200ms

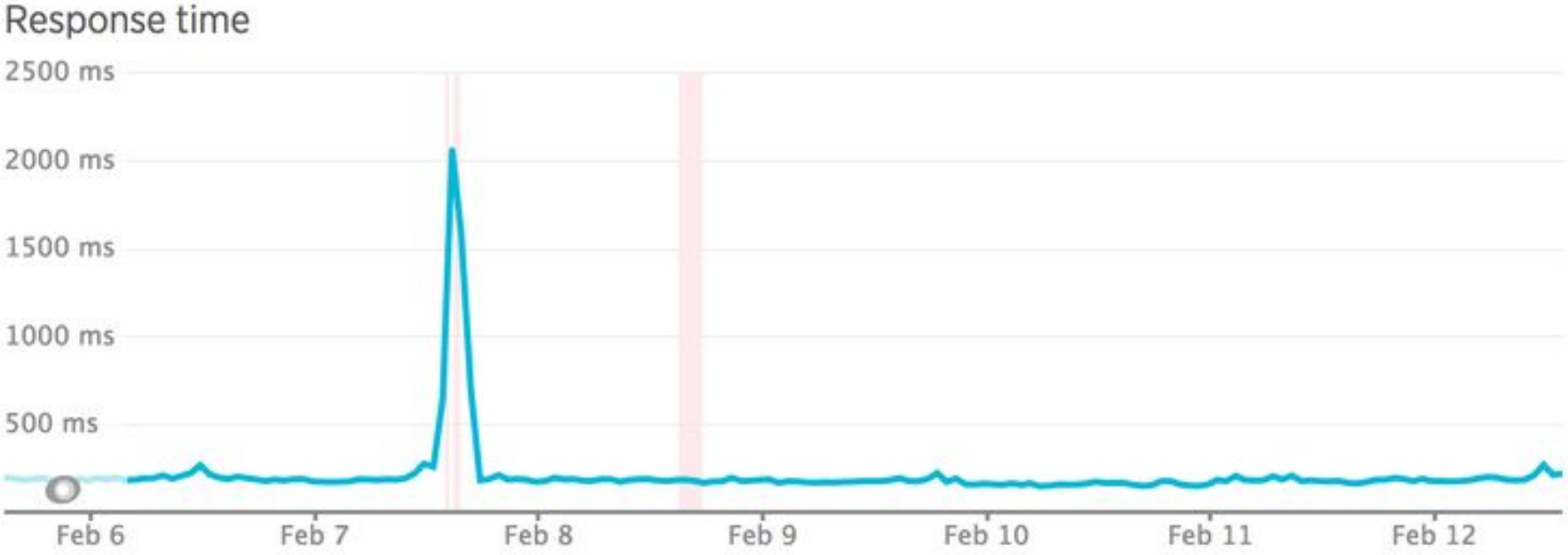
Ignoring network latency and content download times…



A Common Sync Call Problem

227 ms
RESP. TIME

28 cpm
THROUGHPUT

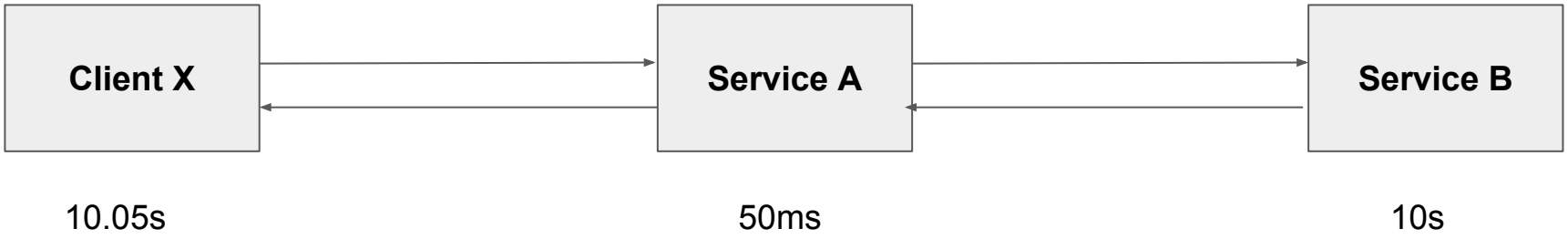


A Common Sync Call Problem

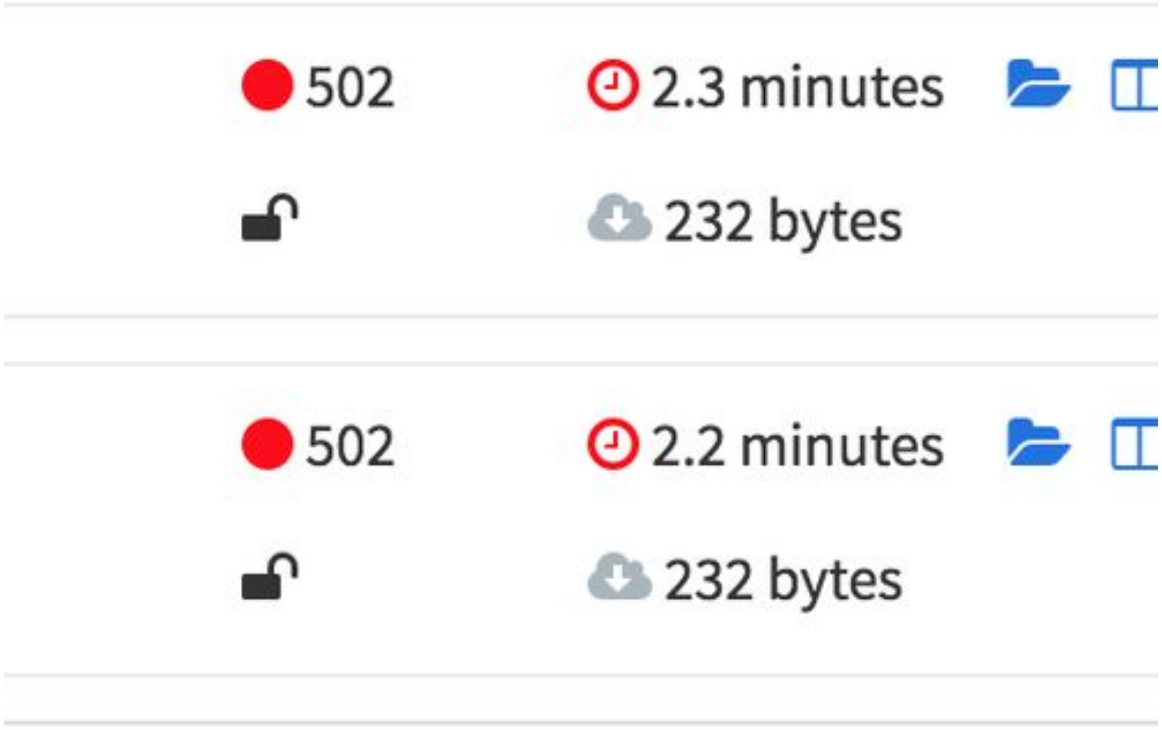
Service B is having a bad day

Service A gives it too much of a leash

Client X is stuck waiting for 10+ seconds

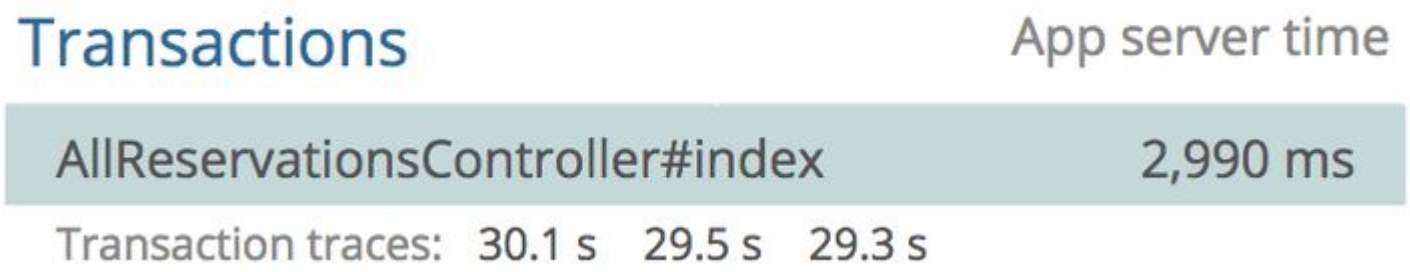


The Keycard Troubles of 2016



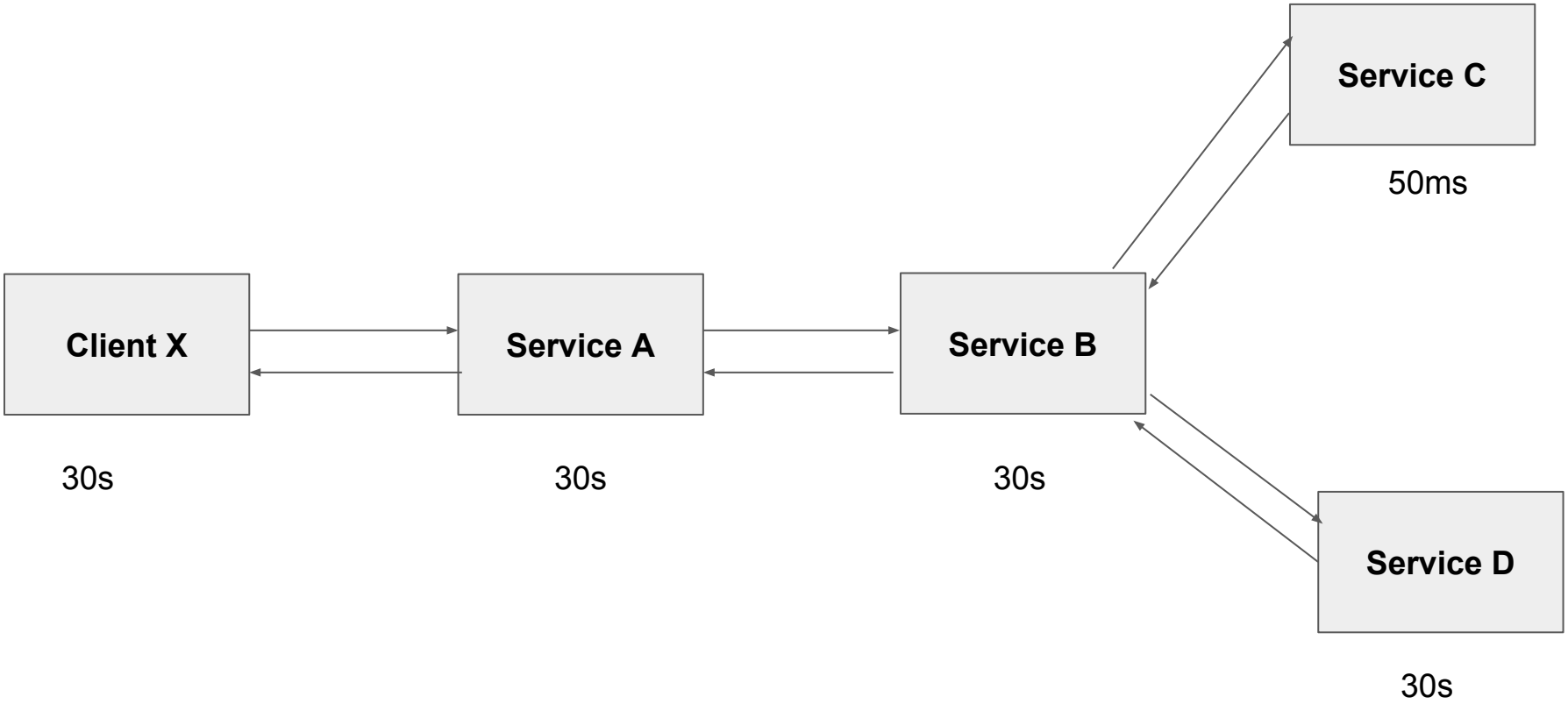
Heroku Chop of 30s

For the systems still on Heroku, web threads terminate after **thirty seconds**



That’s better than ∞ seconds but… 😬

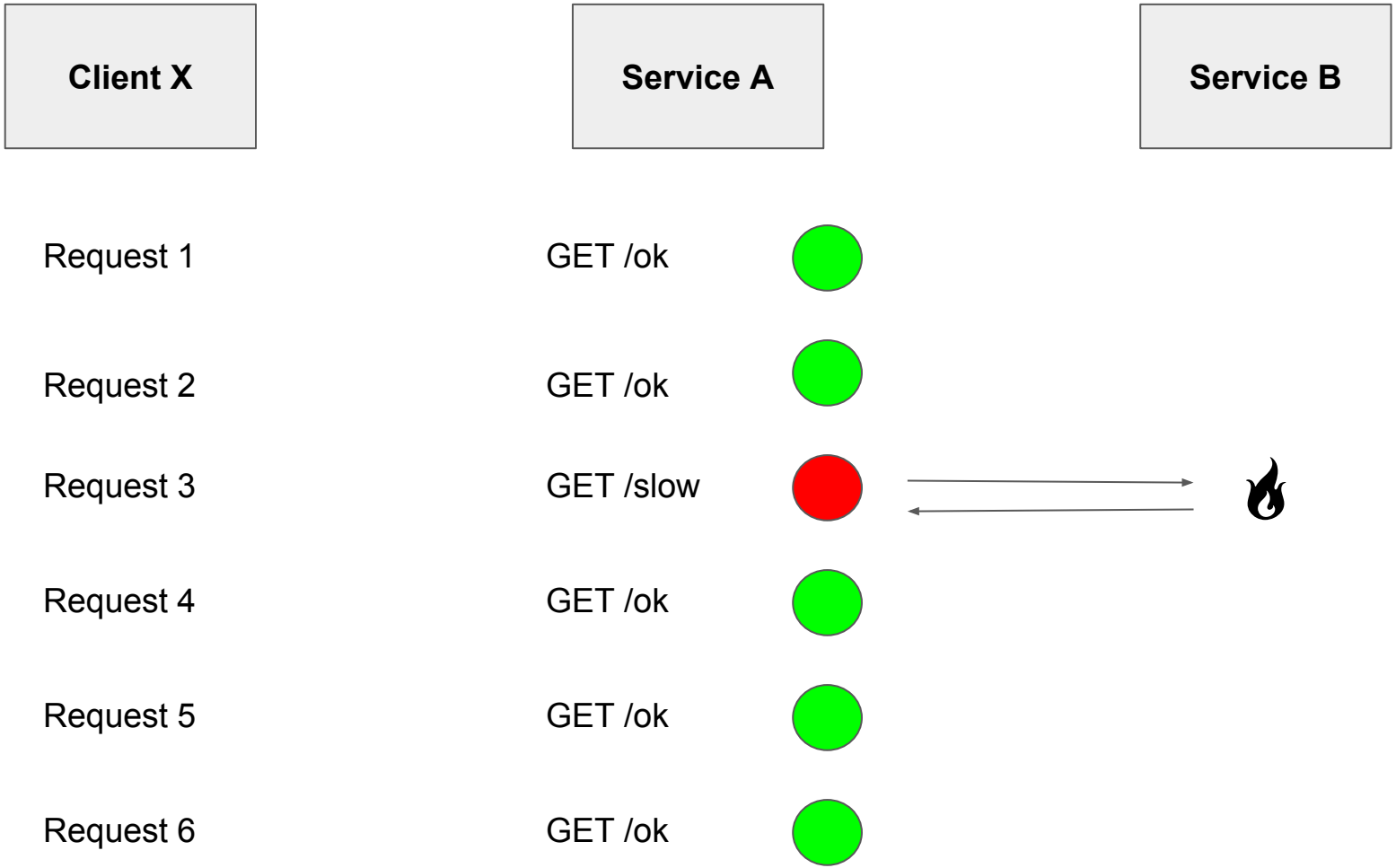
A Common Sync Call Problem

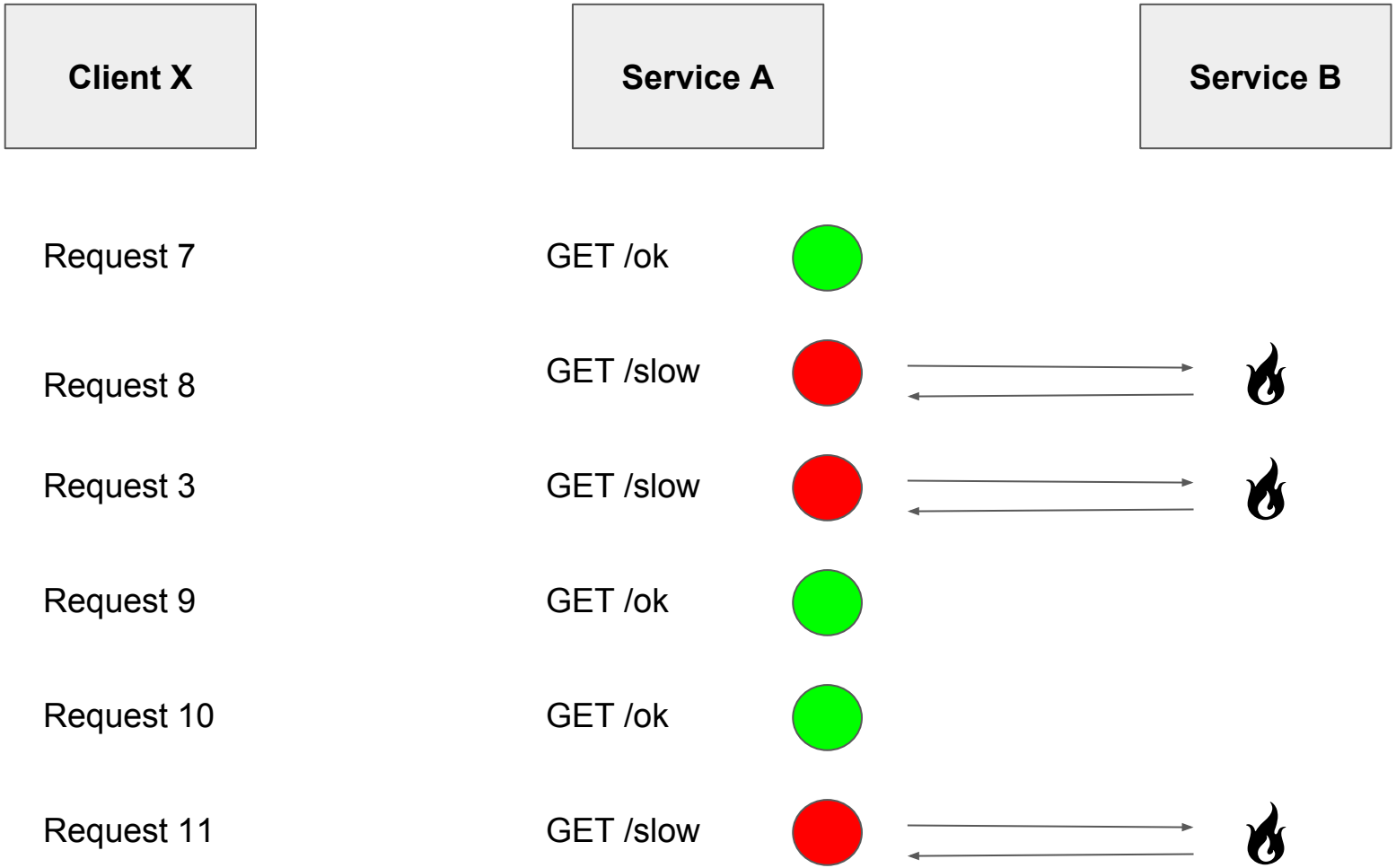


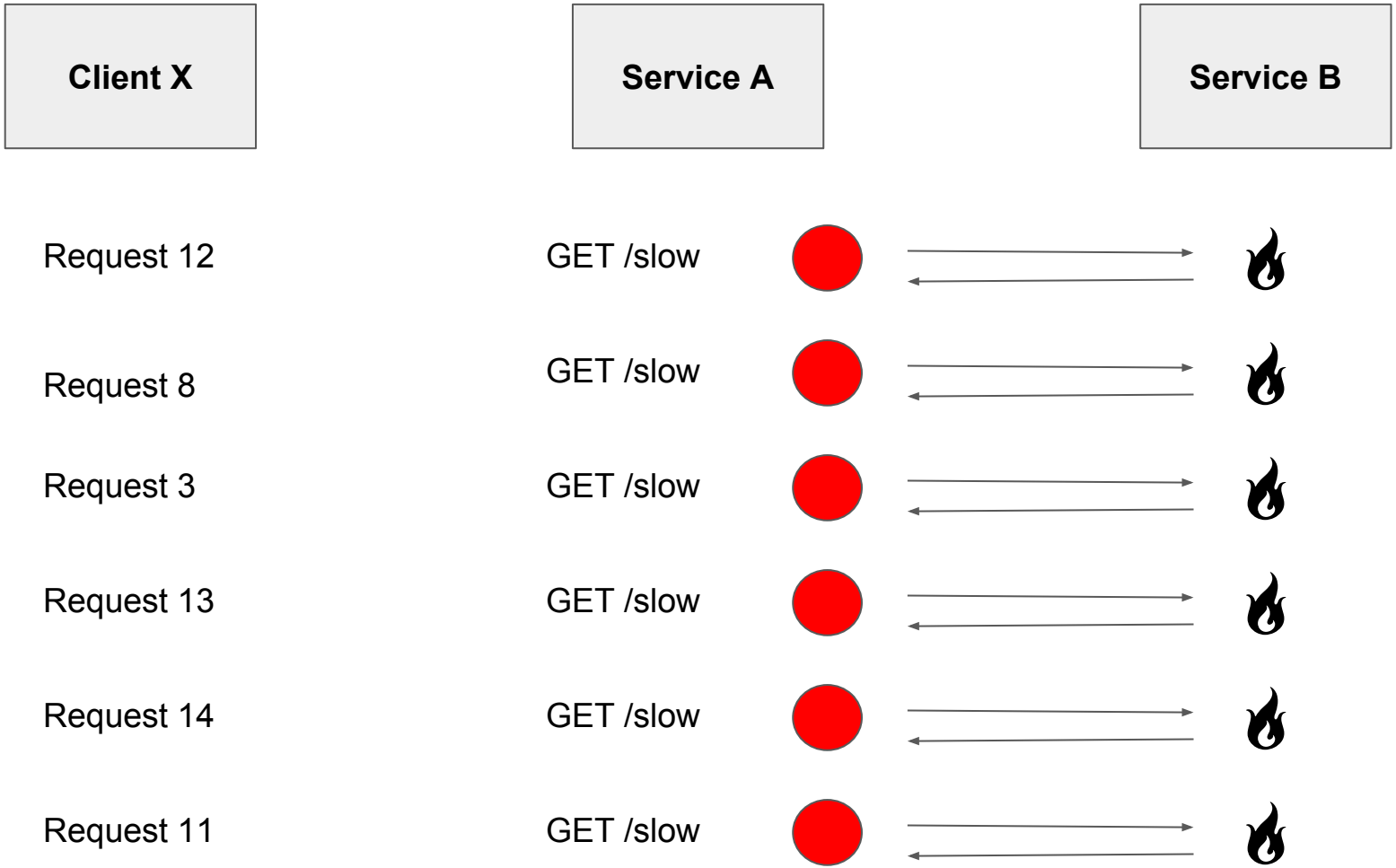
Sucks for this specific user

Waiting 30 seconds to be told something failed?

Waiting *another* 30 seconds to be told it **still** failed?

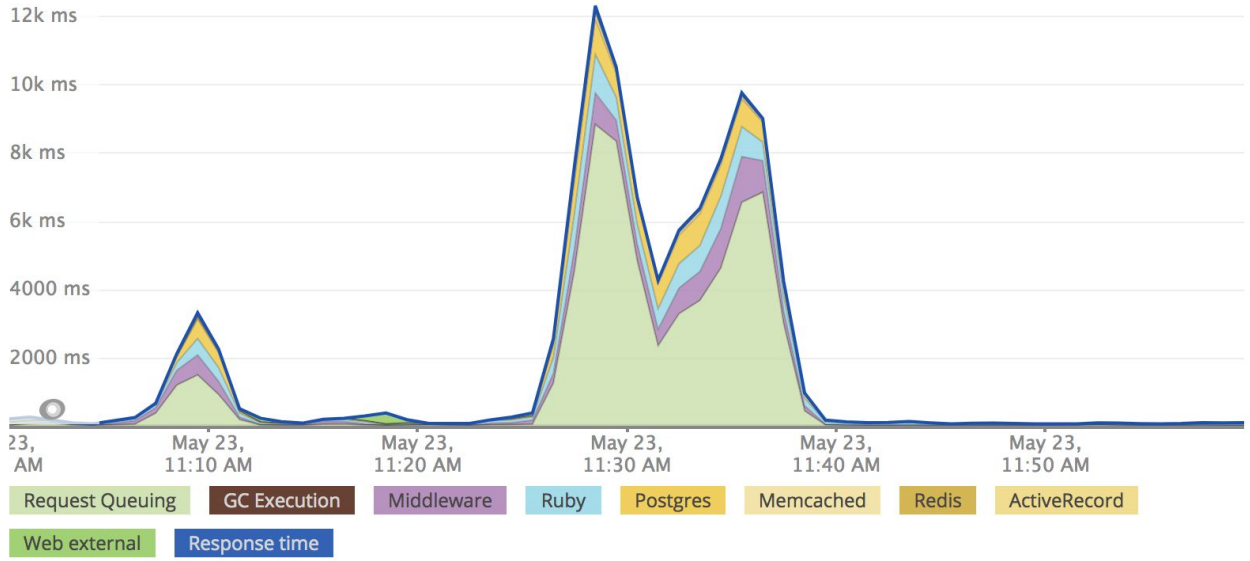

Client X
Service A
Service B
Request 1
GET /ok
Request 2
GET /ok
Request 3
GET /slow
Request 4
GET /ok
Request 5
GET /ok
Request 6
GET /ok

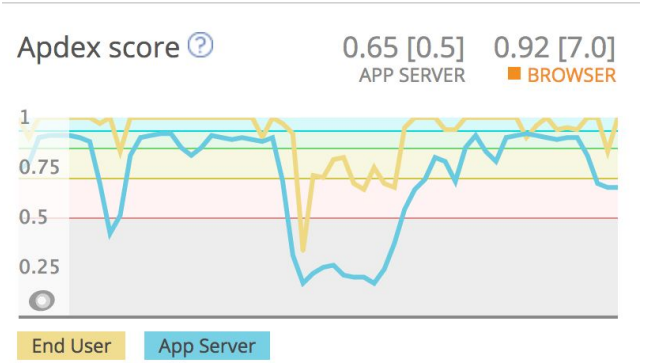

Client X
Service A
Service B
Request 7
GET /ok
Request 8
GET /slow
Request 3
GET /slow
Request 9
GET /ok
Request 10
GET /ok
Request 11
GET /slow

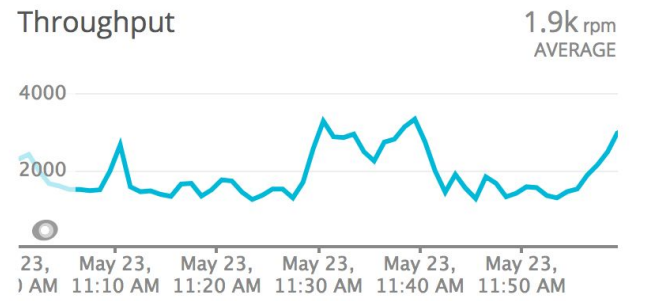

Client X
Service A
Service B
Request 12
GET /slow
Request 8
GET /slow
Request 3
GET /slow
Request 13
GET /slow
Request 14
GET /slow
Request 11
GET /slow

Sucks for all other users of any clients talking to that

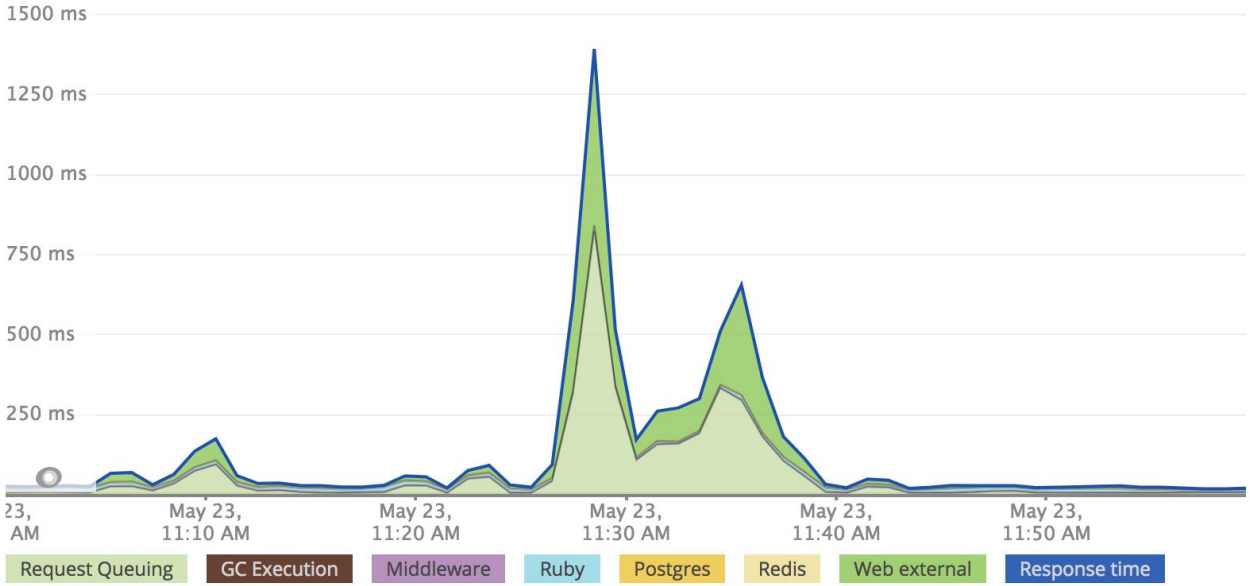
Web threads sit there spinning on slow requests

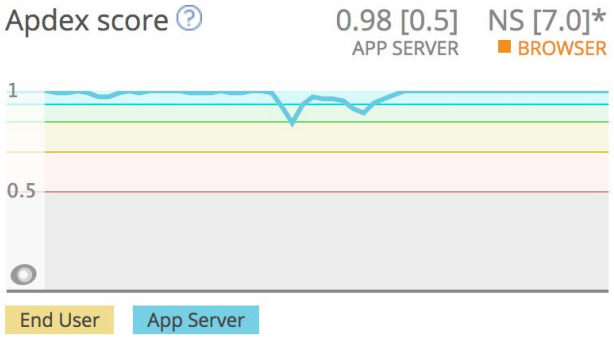


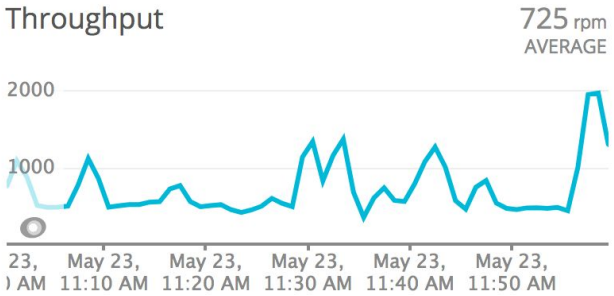




And the users of any services talking to THAT







For each thread that gets stuck, given that thread is stuck for 30s, if most requests go through in 100ms, **that's 3000 potential requests not being handled**.

Setting a timeout of 10s gives that thread 20s of extra life, meaning **2000 more successful requests.**

Step 1: Set Timeouts

Ruby

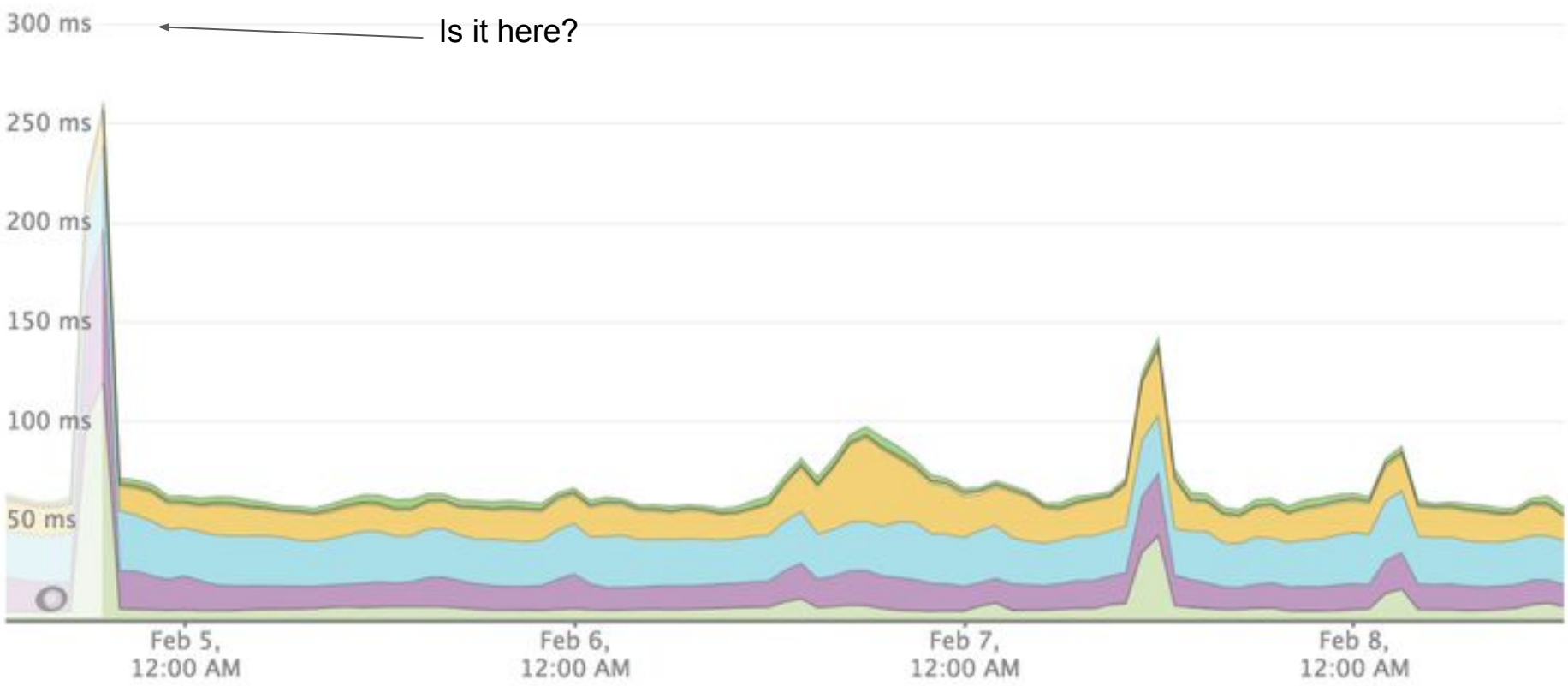
```
conn = Faraday.new('http://foo');

conn.get do |req|
  req.url '/bar'
  req.options.timeout = 0.5           # open/read timeout in seconds
  req.options.open_timeout = 0.2      # connection open timeout in seconds
End
```

Step 1: Set Timeouts

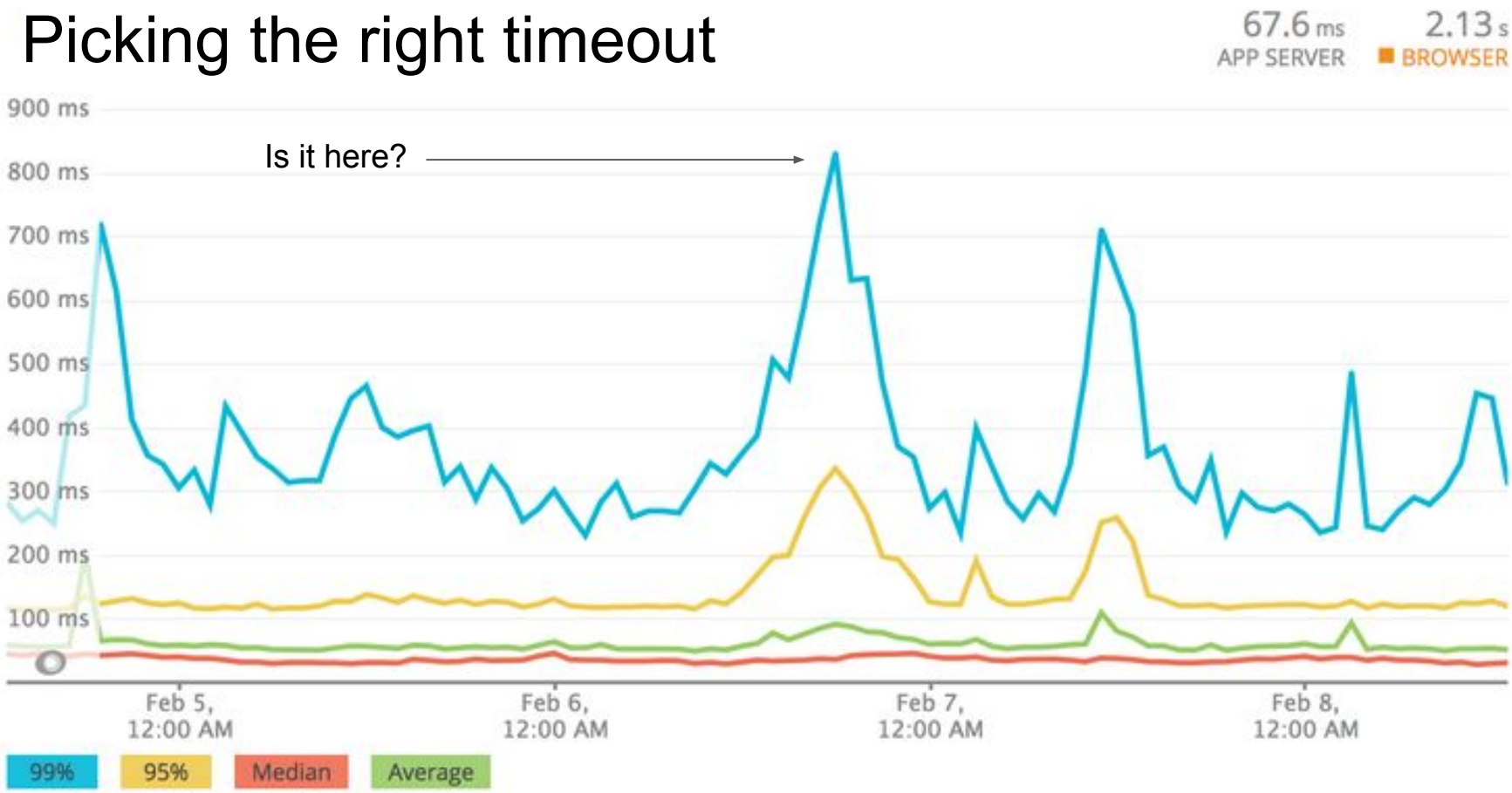
PHP

```
$client->request('GET', '/delay/5', ['timeout' => 2]);
```

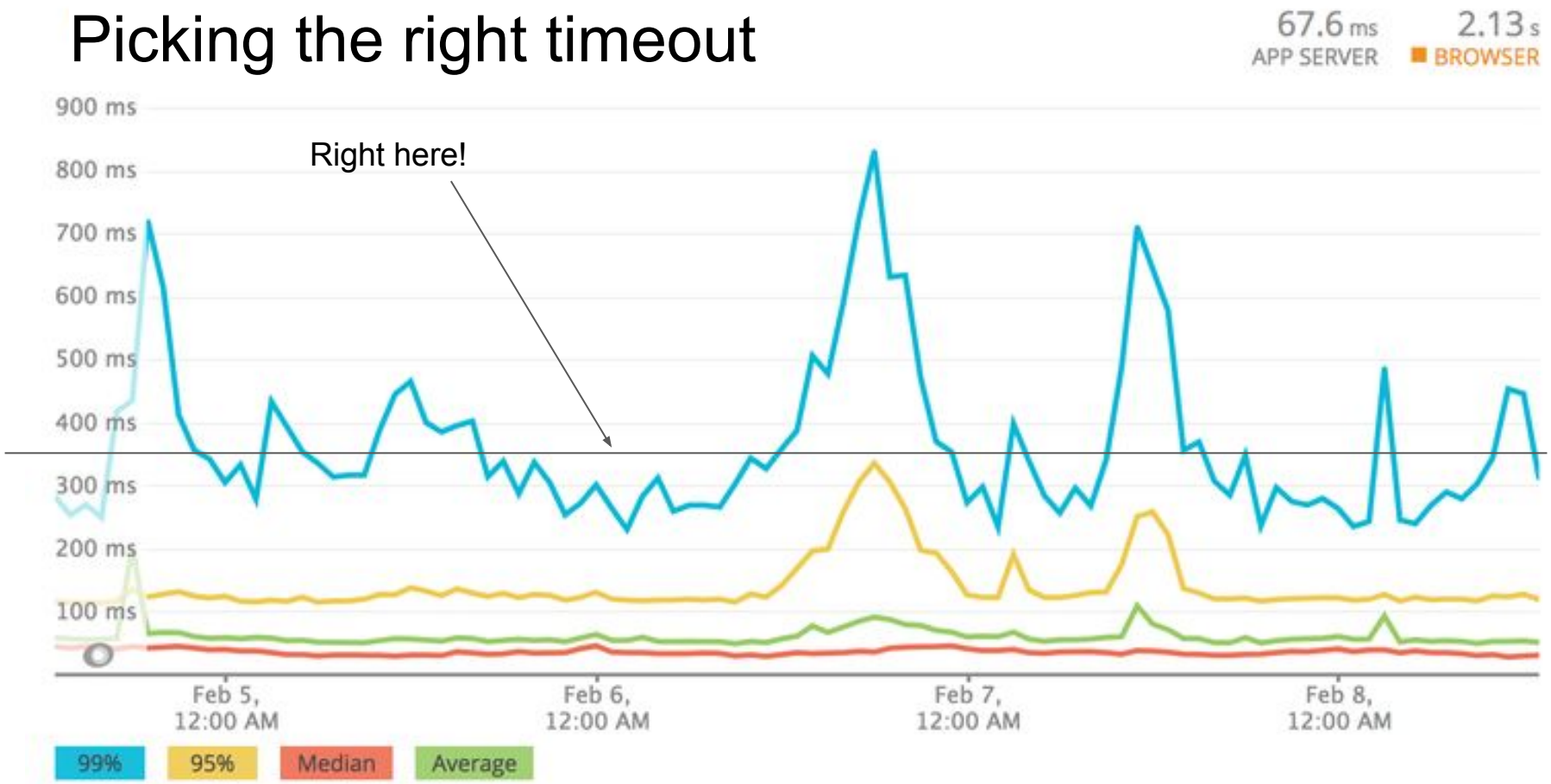
Picking the right timeout



Picking the right timeout



Picking the right timeout



Step 2: Retries

Try a second time

Maybe you'll get in the 99% of quicker responses!

Ruby

```
Faraday.new do |conn|
    conn.request :retry, max: 1, interval: 0.05
end
```

Add up all timeouts and retries

Service A calls B, C, D & E

4 services, with timeout = 2s, retries = 2 (3 total attempts)

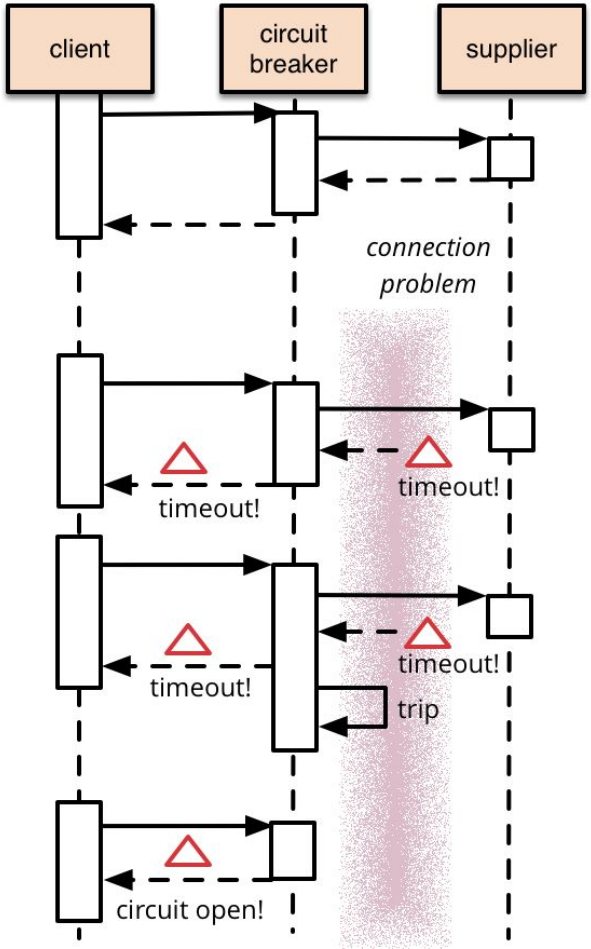
4 * 2 * 3 = 24 seconds!!

That’s a lot of time to be waiting for stuff

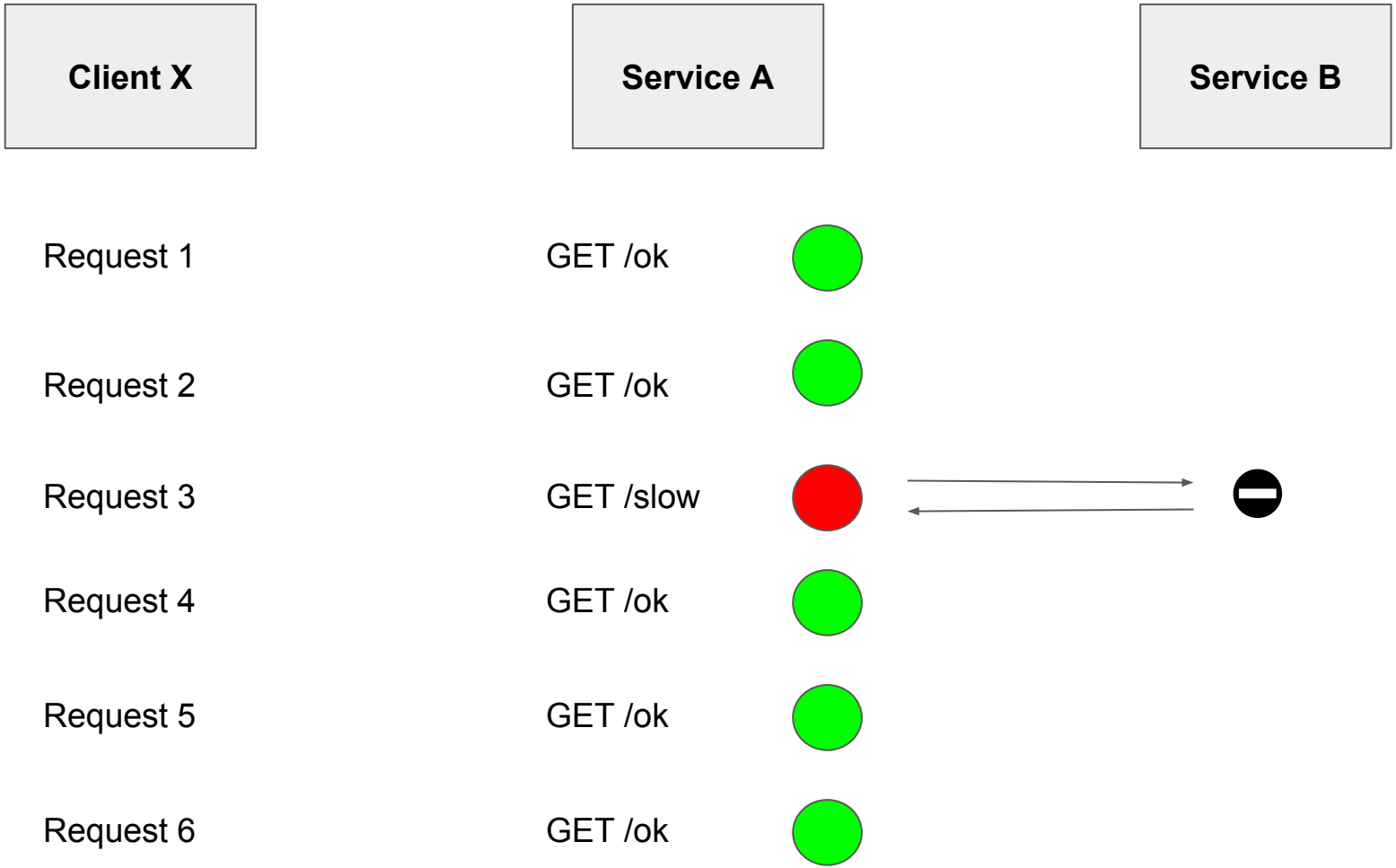
Step 3: Circuit Breakers

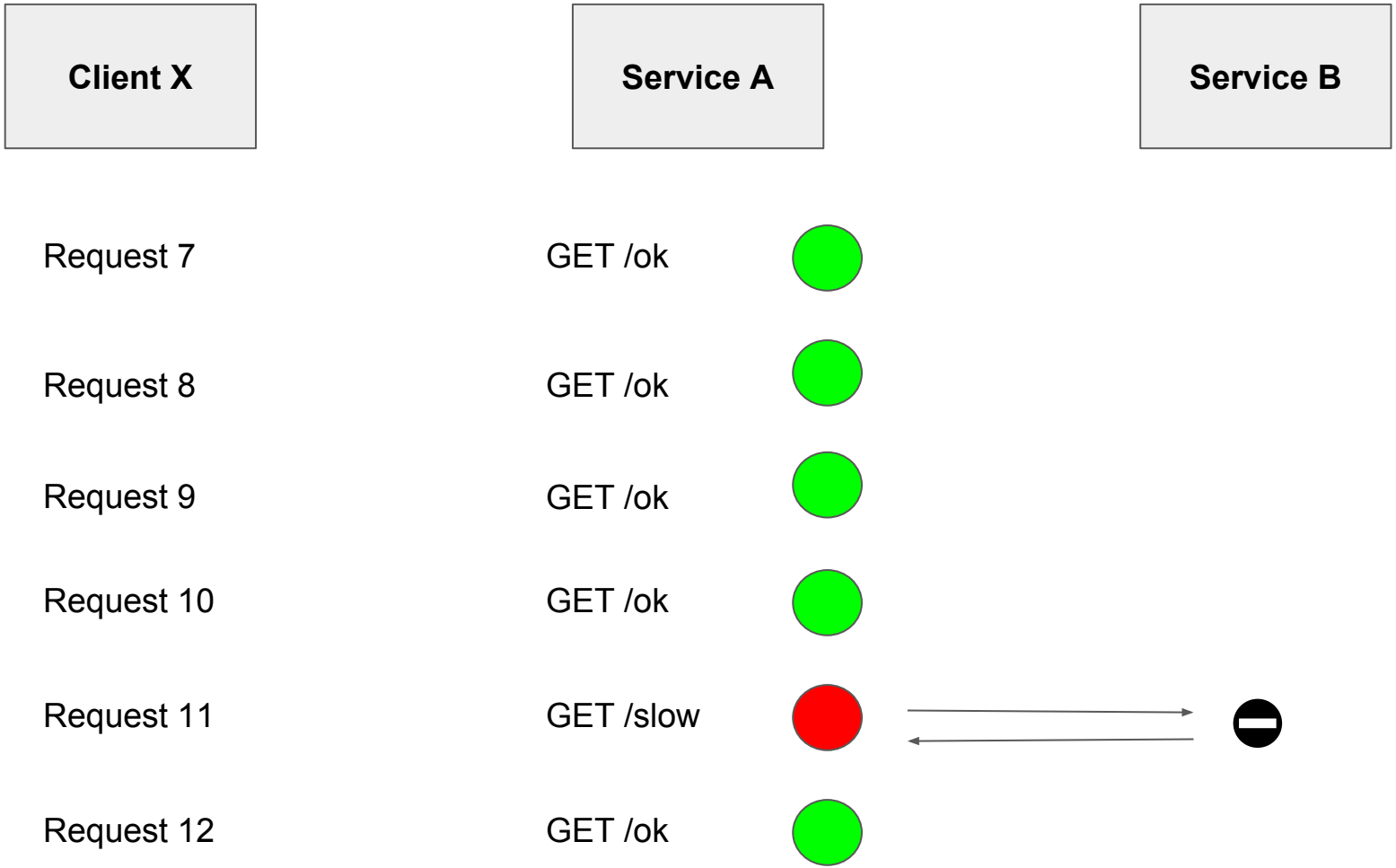
Don’t wait for stuff that **probably won’t work**

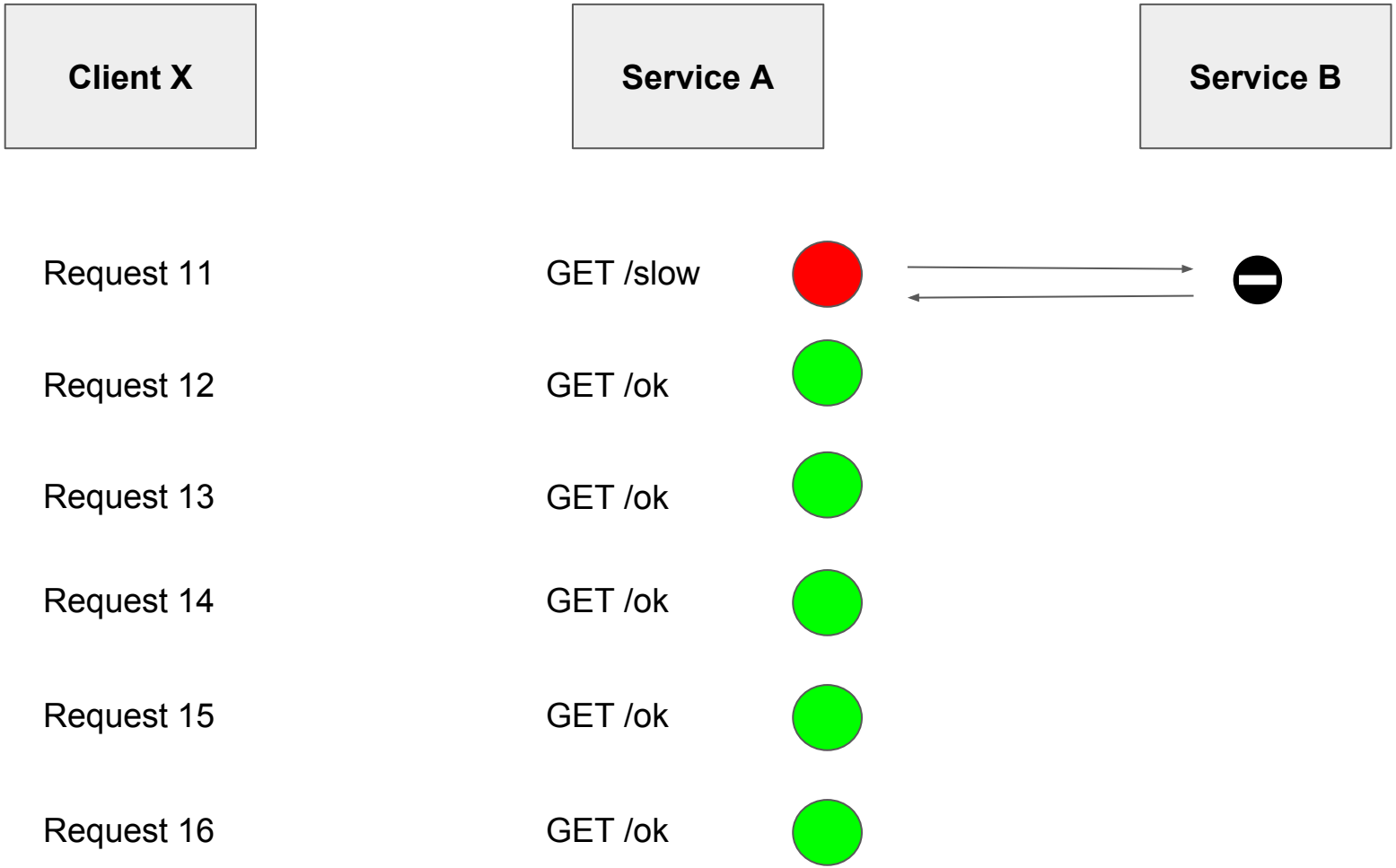
Free up web workers to answer requests that **probably will work!**



https://martinfowler.com/bliki/CircuitBreaker.html


Client X
Service A
Service B
Request 1
GET /ok
Request 2
GET /ok
Request 3
GET /slow
Request 4
GET /ok
Request 5
GET /ok
Request 6
GET /ok

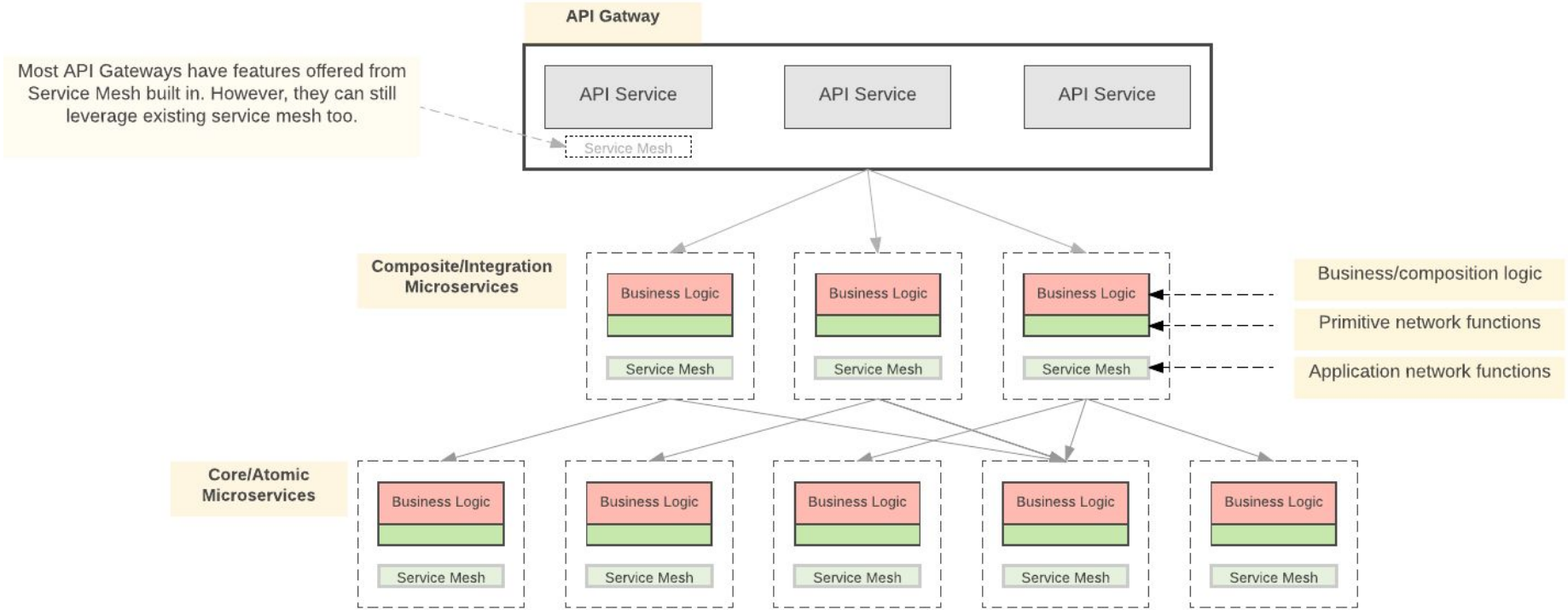

Client X
Service A
Service B
Request 7
GET /ok
Request 8
GET /ok
Request 9
GET /ok
Request 10
GET /ok
Request 11
GET /slow
Request 12
GET /ok


Client X
Service A
Service B
Request 11
GET /slow
Request 12
GET /ok
Request 13
GET /ok
Request 14
GET /ok
Request 15
GET /ok
Request 16
GET /ok

One Option: Code Breakers

```
require 'faraday'
require 'circuitbox/faraday_middleware'

conn = Faraday.new(:url => "http://example.com") do |c|
  c.use Circuitbox::FaradayMiddleware
end

response = conn.get("/api")
if response.success?
  # success
else
  # failure or open circuit
end
```

Another Option: Service Mesh!



Service Mesh is a network communication infrastructure which allows your to decouple and offload most of the application network functions from your service code.

Envoy - Service Mesh from Lyft

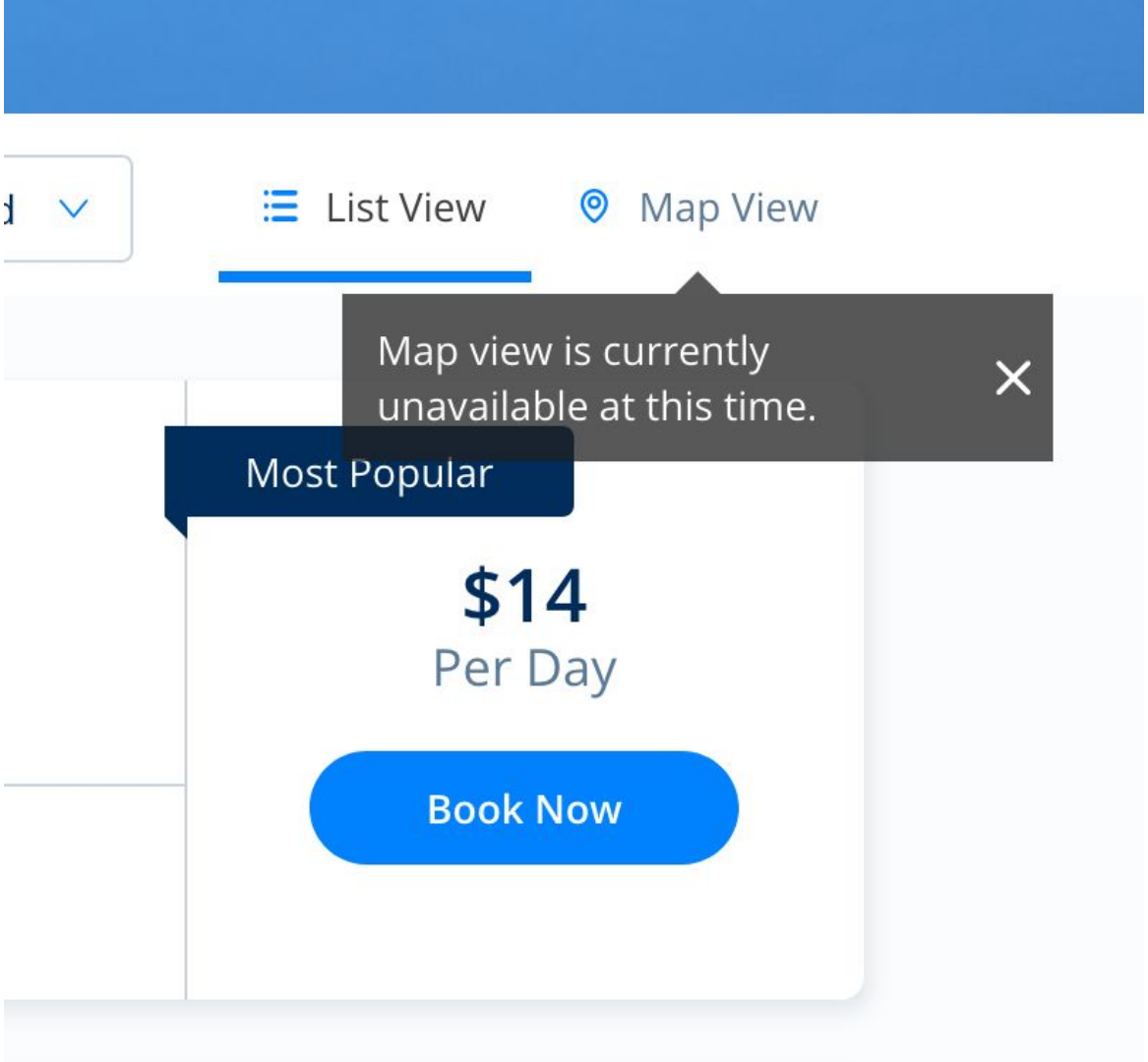
https://www.envoyproxy.io/

Demo/Guide on creating Circuit Breakers

http://blog.christianposta.com/microservices/01-microservices-patterns-with-envoy-proxy-part-i-circuit-breaking/

Comparing Envoy and Istio Circuit Breaking

http://blog.christianposta.com/microservices/comparing-envoy-and-istio-circuit-breaking-with-netflix-hystrix/


List View
Map View
Map view is currently unavailable at this time.
Most Popular
$14
Per Day
Book Now

me

from other drivers and park confidently

Recommended

Lowest Price

$11.95

Per Day

Book Now

tle To JFK

Min

All this and more

apisyouwonthate.com

Selling this book on pre-order!

Coupon code: APIDAYS2018