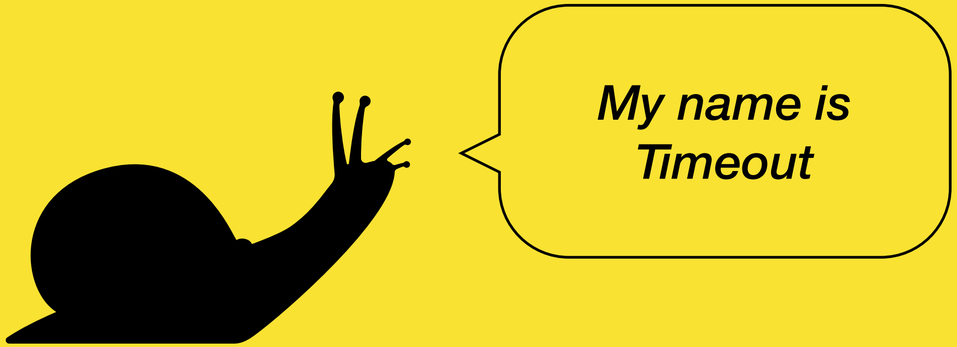
Using fallback to build reliable services

Anthony Regeda

Using fallback to build reliable services

Anthony Regeda




https://www.facebook.com/GoJuno/

Routing service

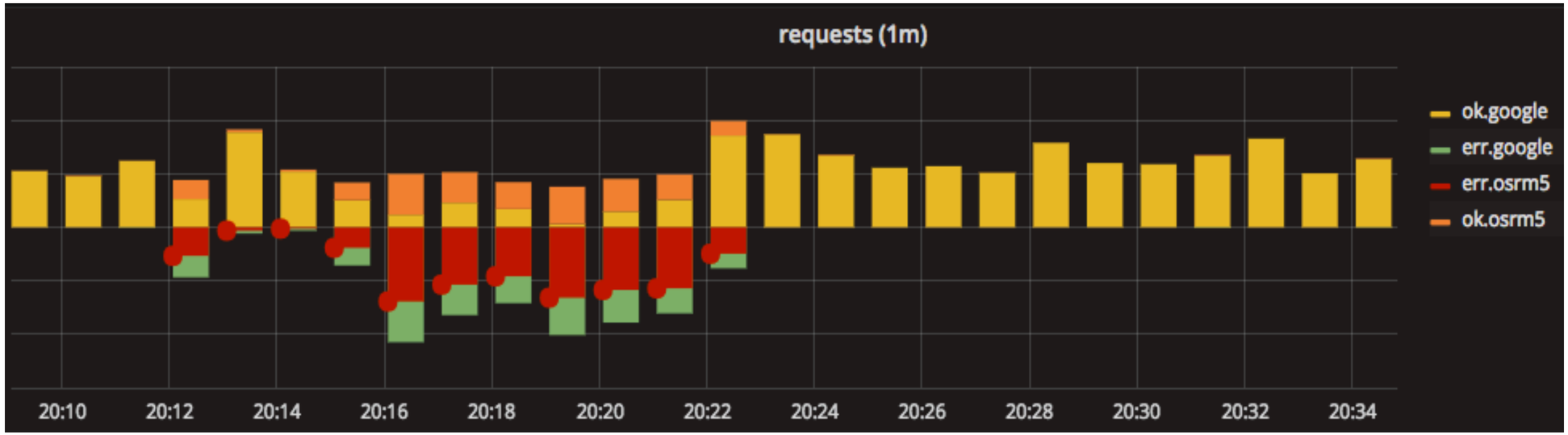
- Fare estimation (price per minute/km)
- Provider estimation (time of arrival)
- Navigation (figure out where is a car)

Service numbers

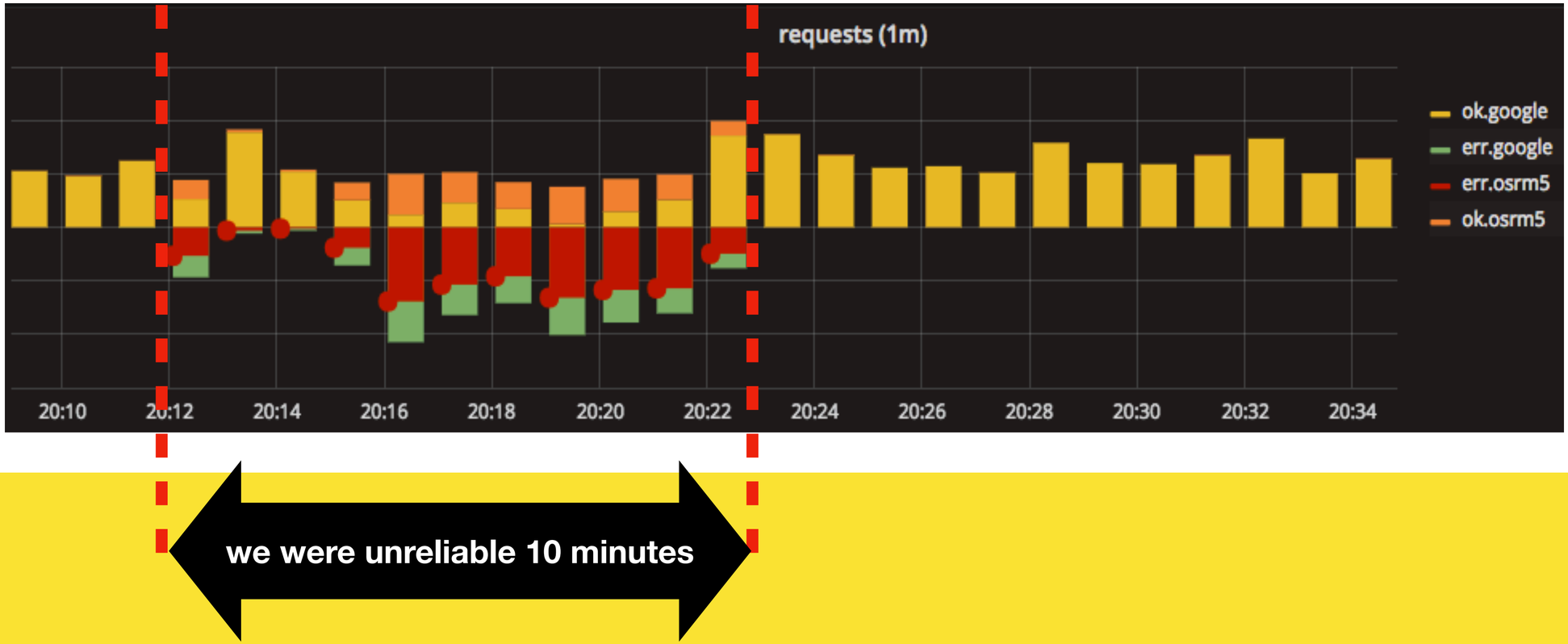
- 1kk requests per day
- response time up to 2 seconds in 0.99 quantile


OSRM
OPEN SOURCE ROUTING MACHINE
Dirty
Accurate
Quick
Slow

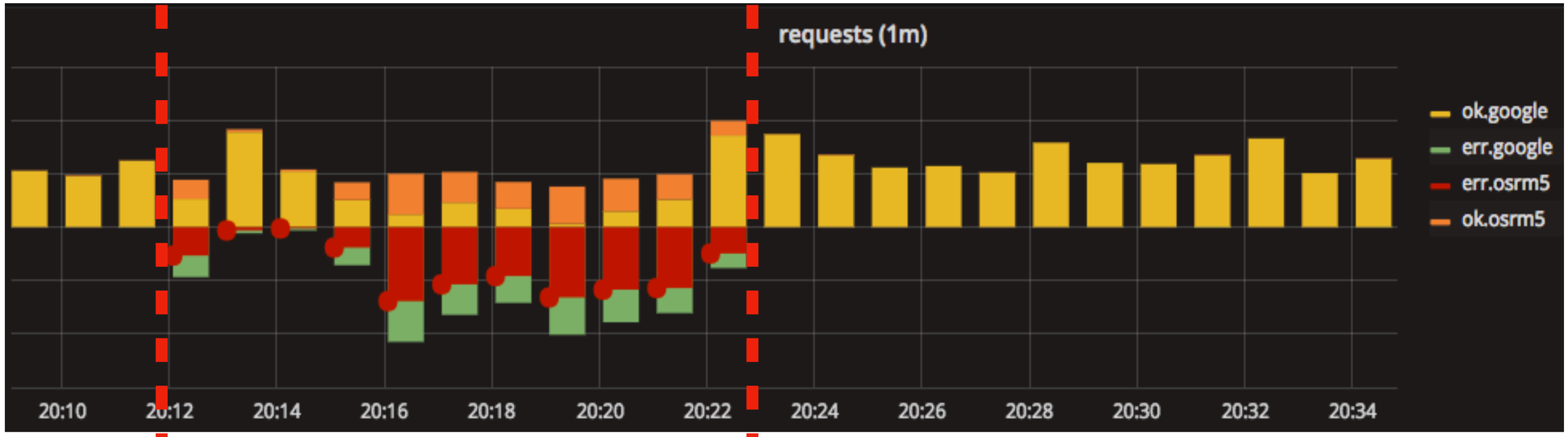
Once day…


requests (1m)
ok.google
err.google
err.osrm5
ok.osrm5
20:10
20:12
20:14
20:16
20:18
20:20
20:22
20:24
20:26
20:28
20:30
20:32
20:34

Once day…


requests (1m)
ok.google
err.google
err.osrm5
ok.osrm5
20:10
20:12
20:14
20:16
20:18
20:20
20:22
20:24
20:26
20:28
20:30
20:32
20:34
we were unreliable 10 minutes

Once day…



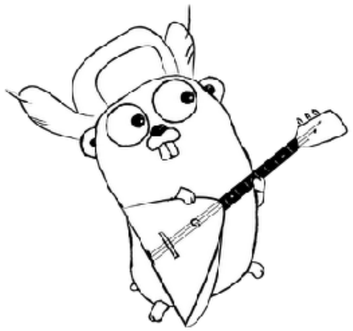
we were unreliable 10 minutes

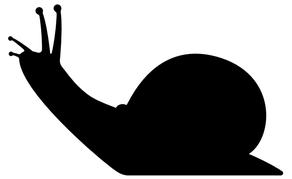
Can’t estimate a ride!
Can’t find a car!
:lol:

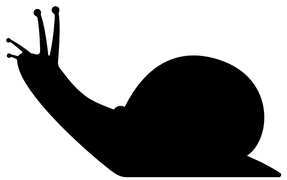
A lot of errors like…

```
handler returned an error: ERR_INTERNAL http(500) [failover: failed to
route: google - context deadline exceeded, osrm - request canceled]
```

failover, failback, fallback

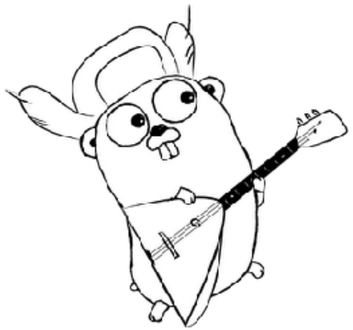


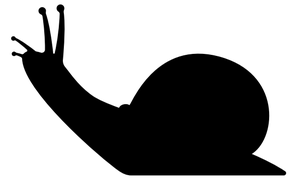


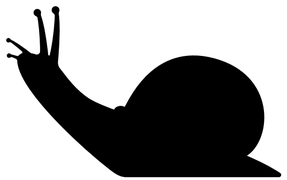


fallback

failover, failback, fallback

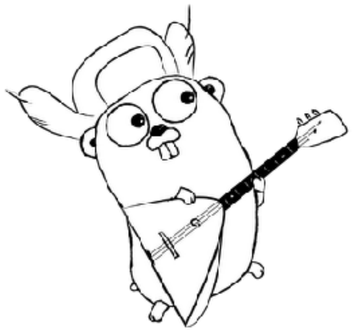


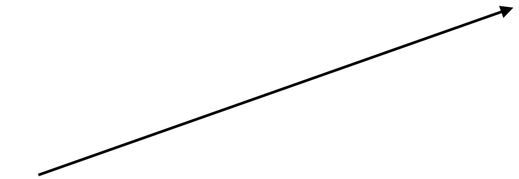


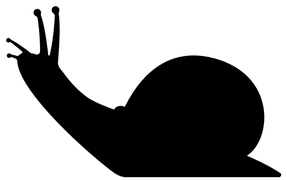


fallback

failover, failback, fallback

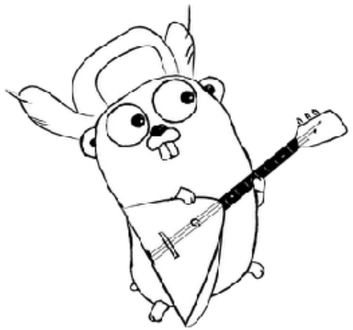




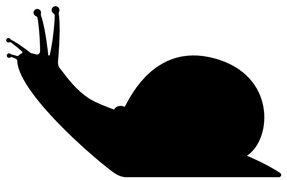


fallback

failover, failback, fallback

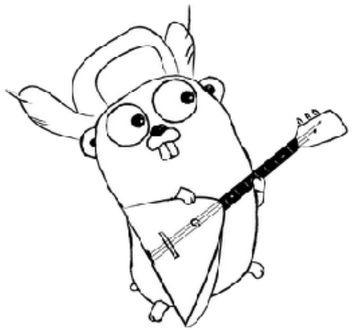


failover

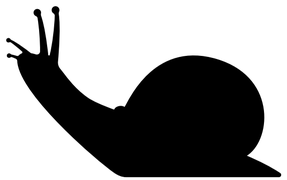


fallback

failover, failback, fallback

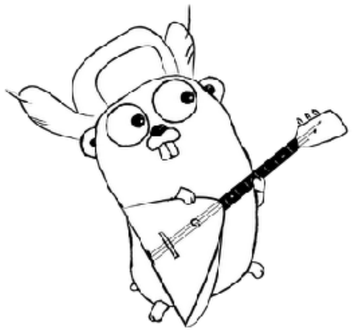


failover

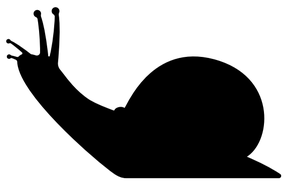


fallback

failover, failback, fallback

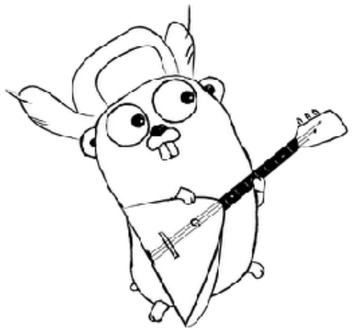


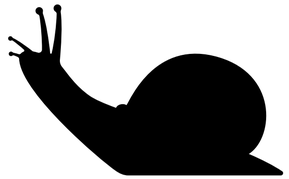
failover



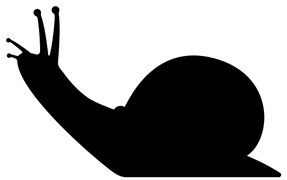
fallback

failover, failback, fallback



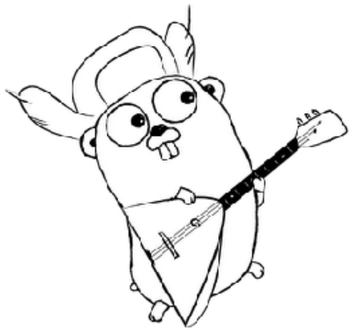


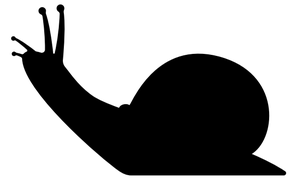
failover



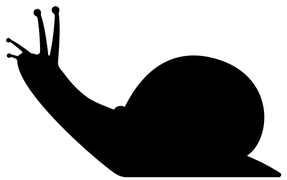
fallback

failover, failback, fallback



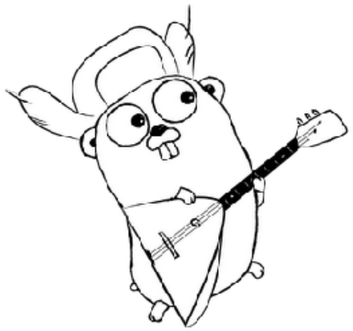


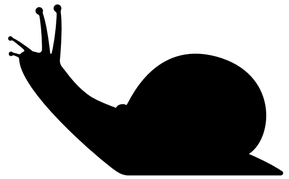
failback



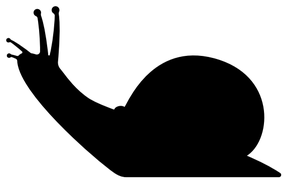
fallback

failover, failback, fallback



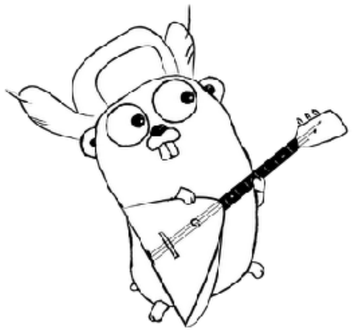


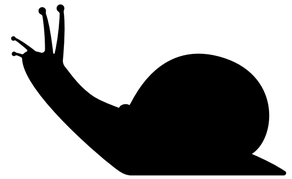
failback

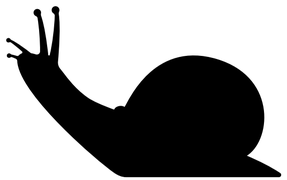


fallback

failover, failback, fallback

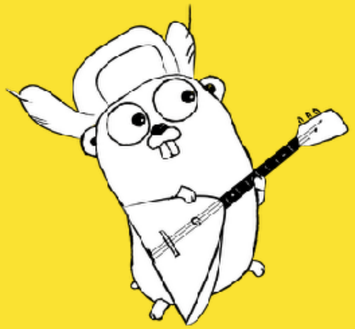






fallback

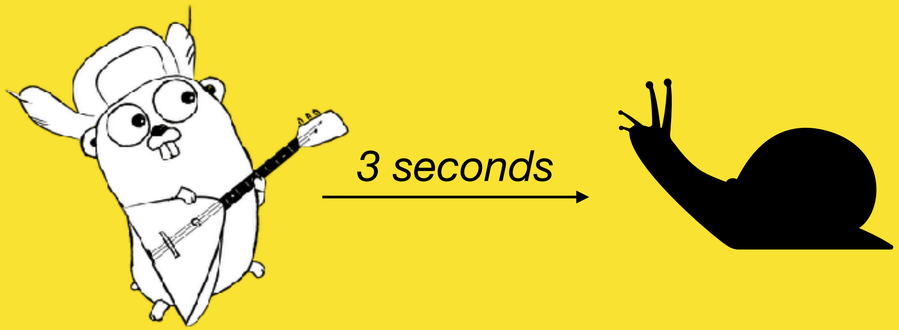
Naive approach – fallback on error



Google

OSRM

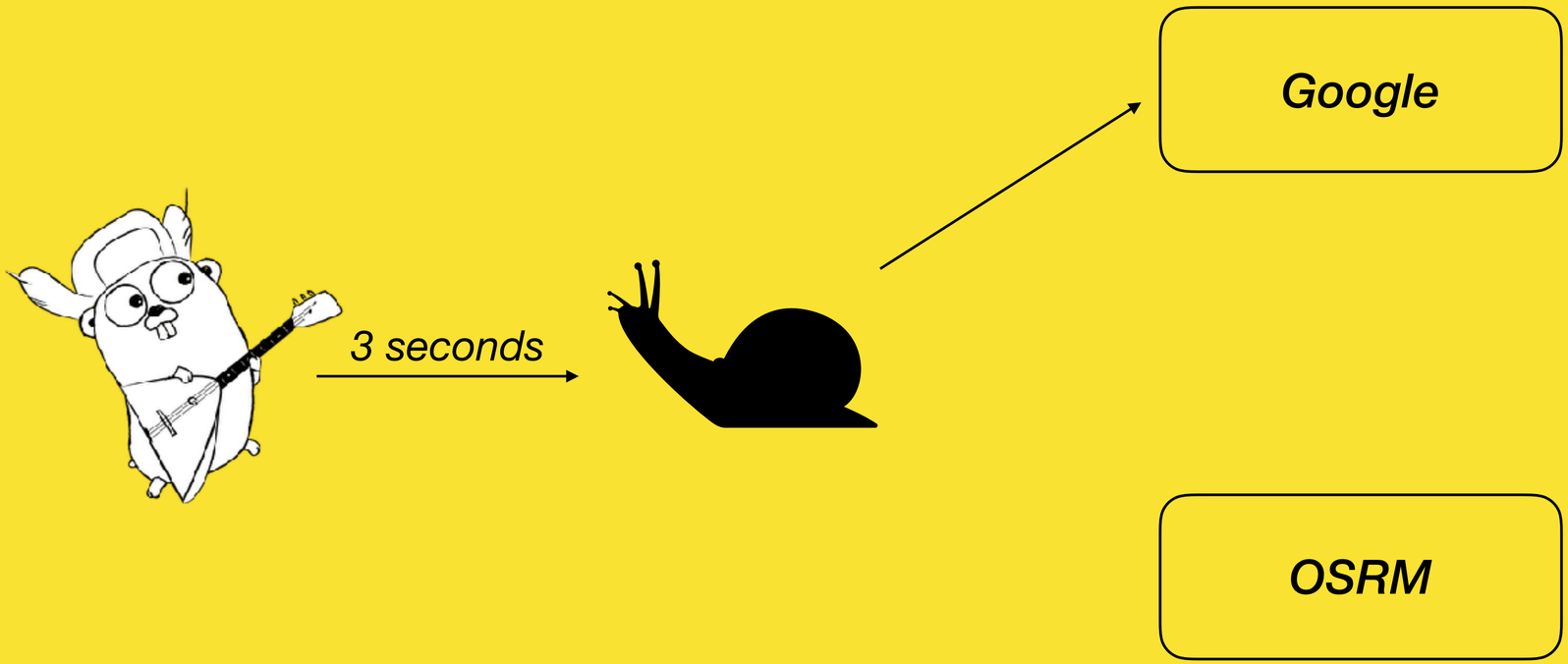
Naive approach – fallback on error



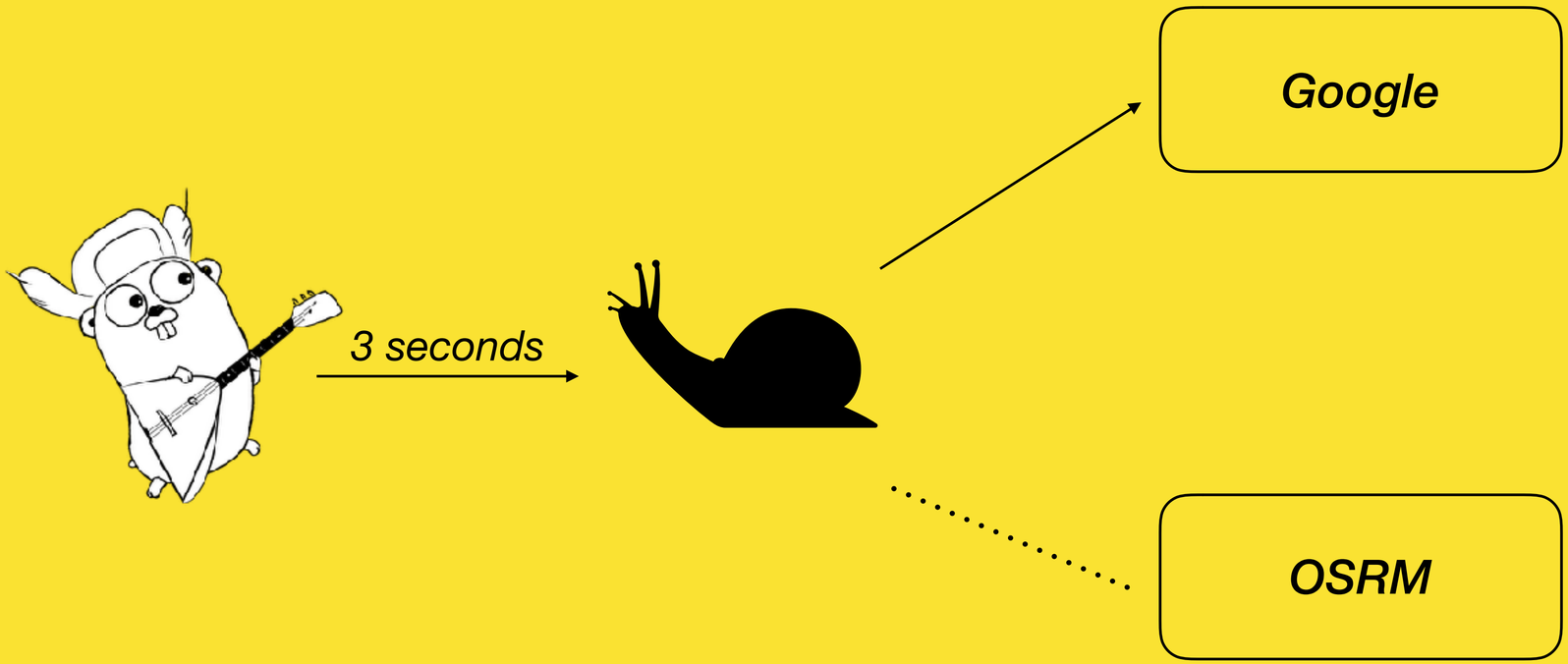
Google

OSRM

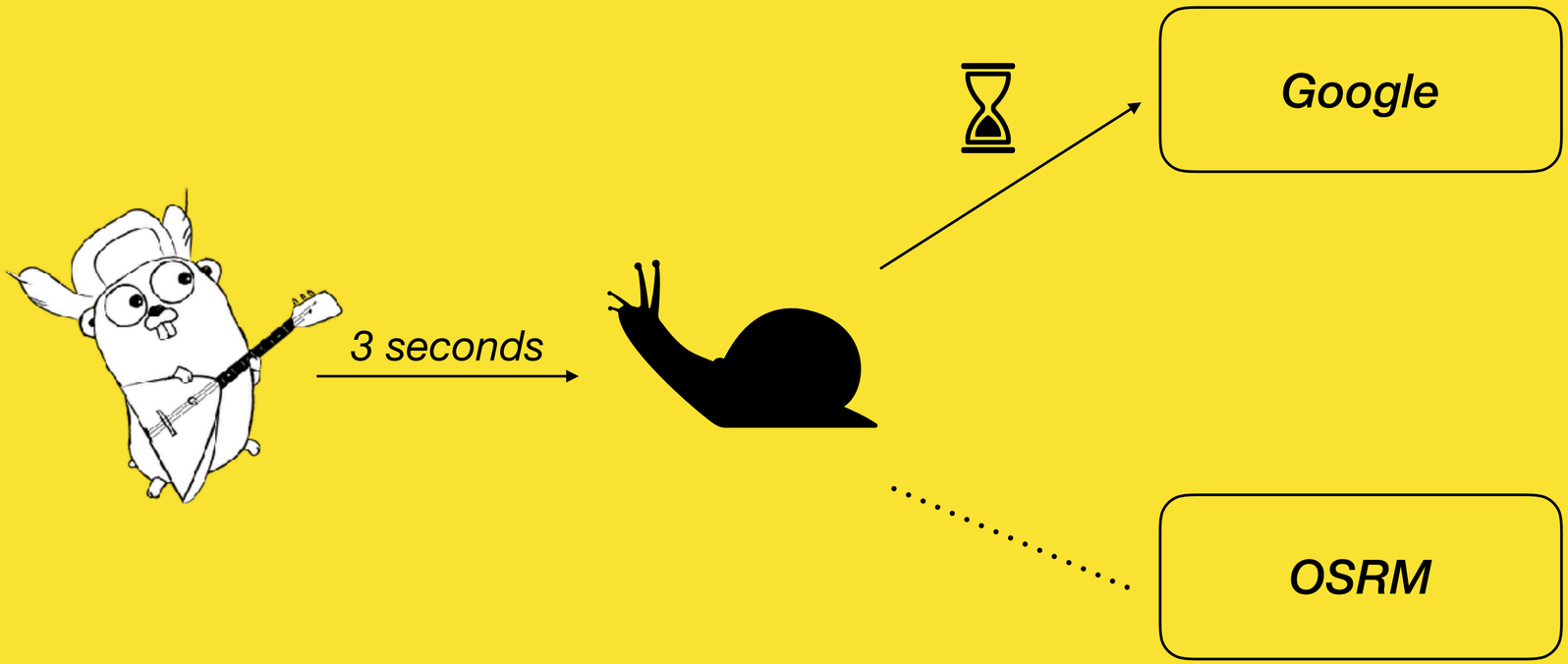
Naive approach – fallback on error



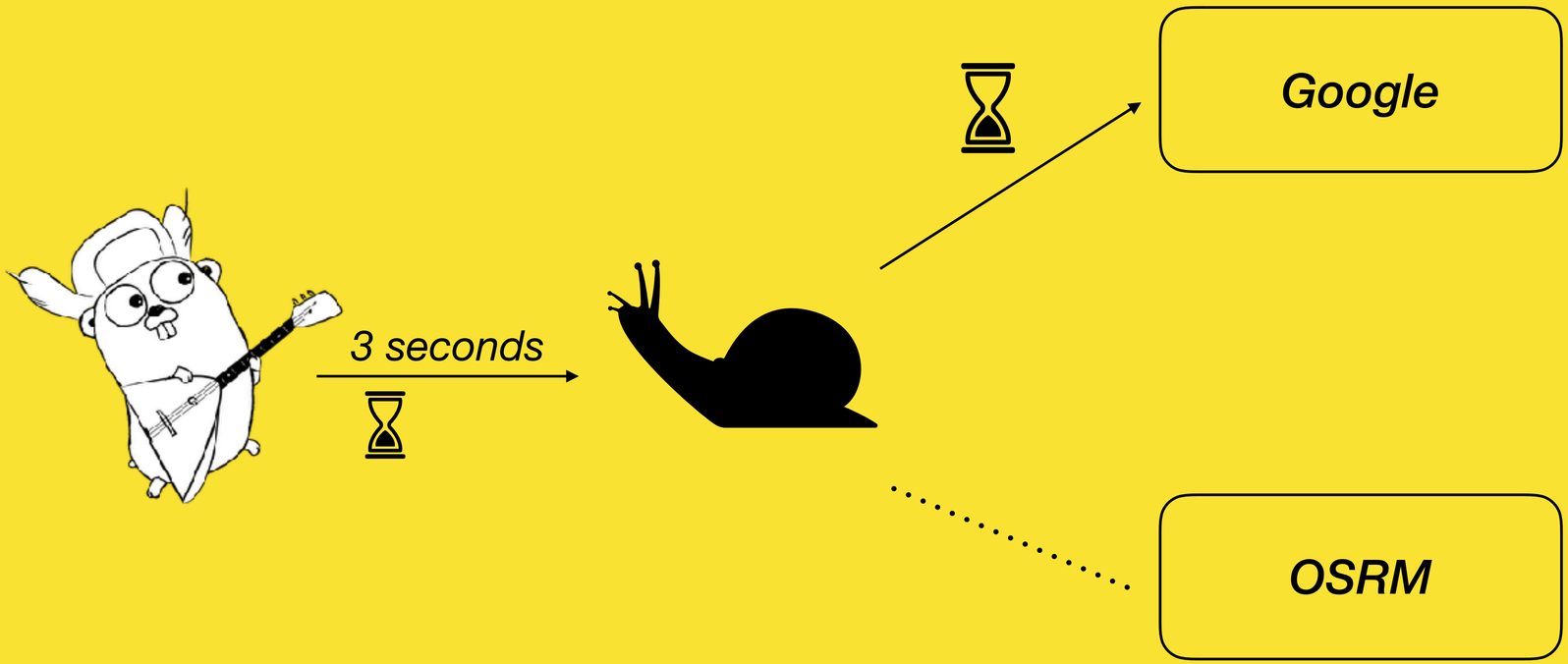
Naive approach – fallback on error



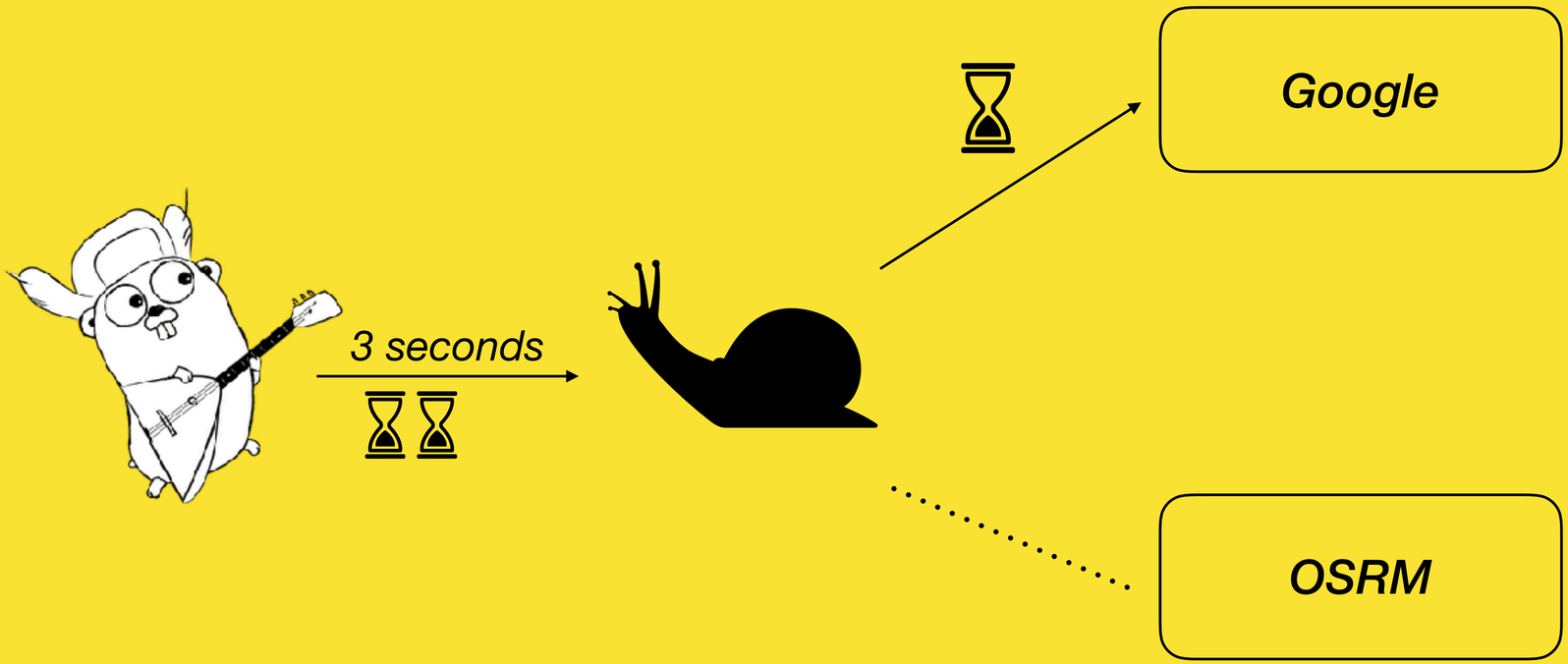
Naive approach – fallback on error



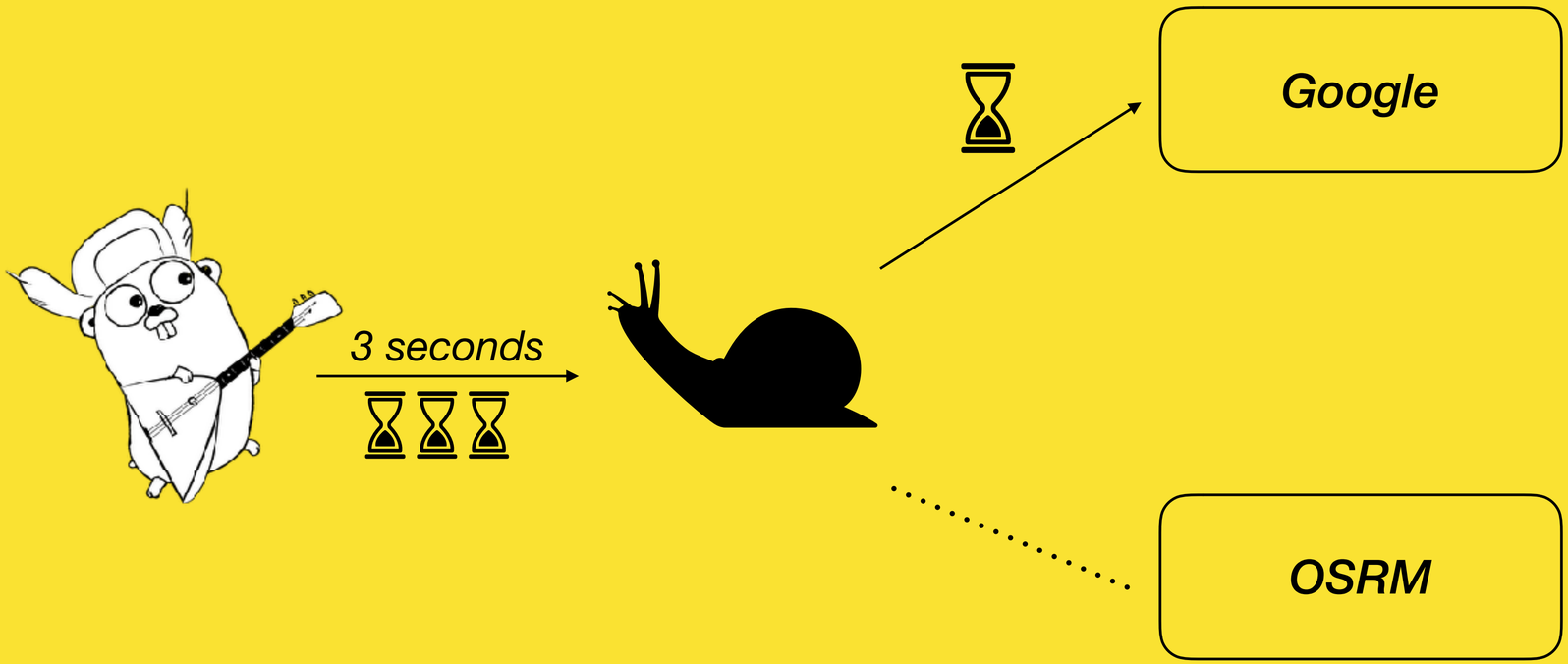
Naive approach – fallback on error



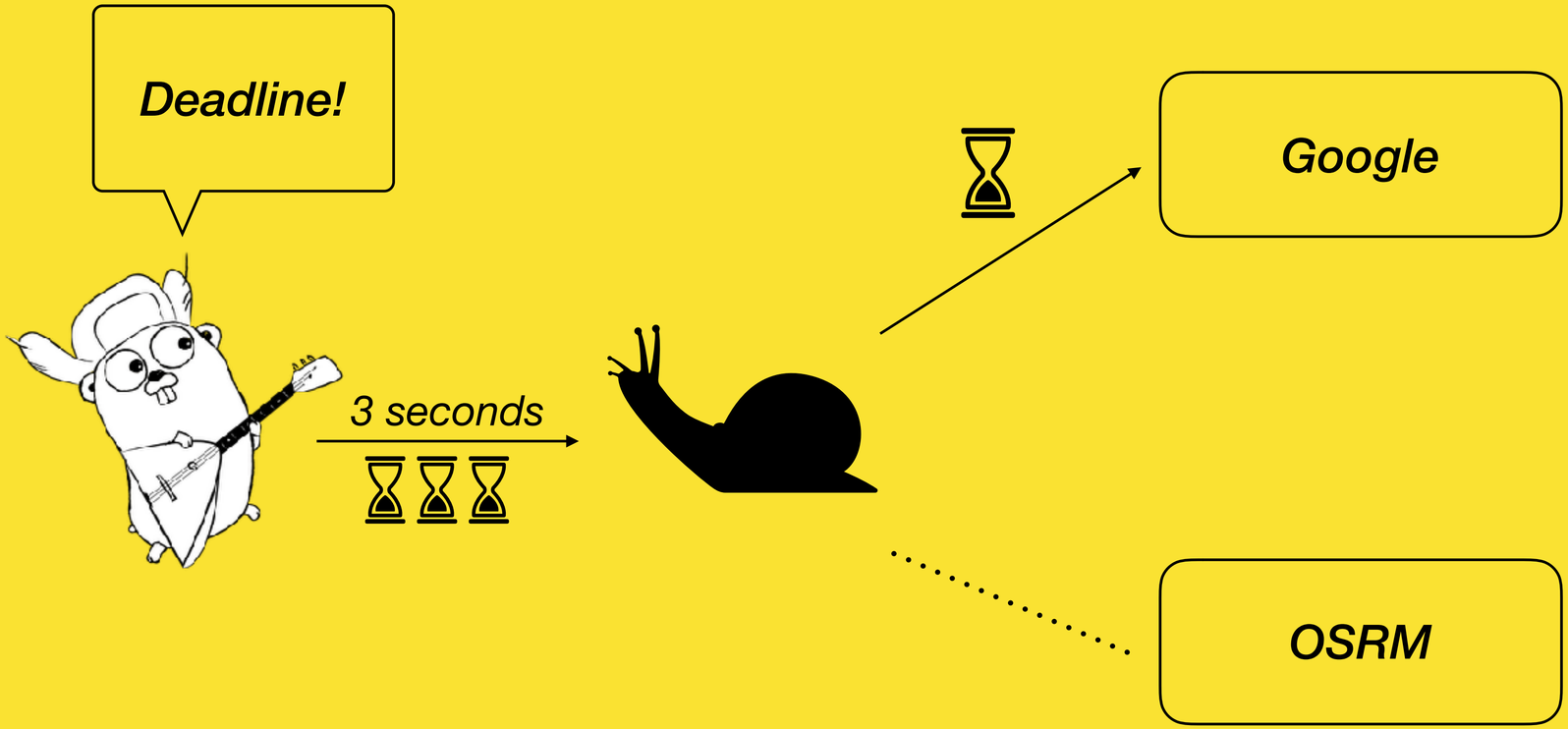
Naive approach – fallback on error



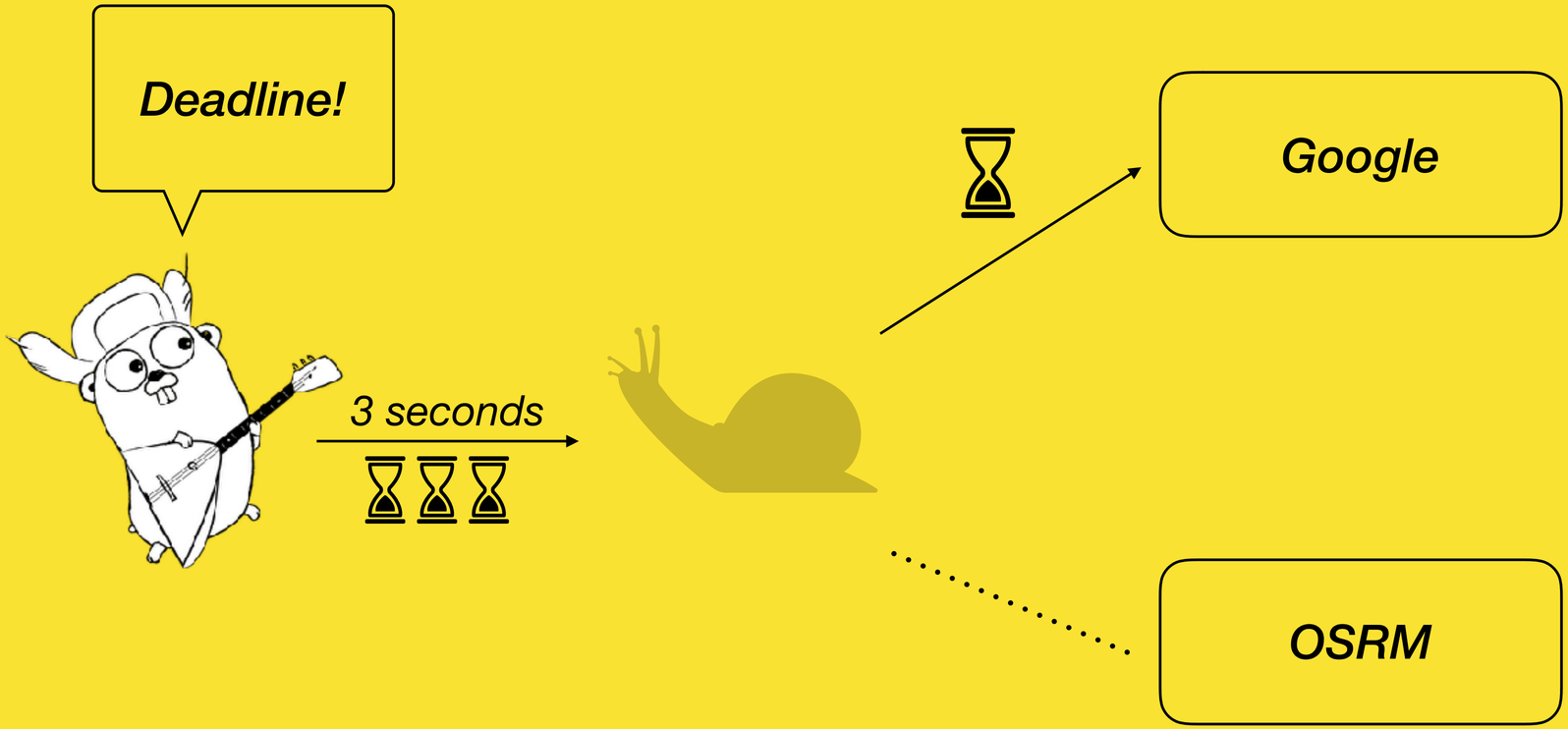
Naive approach – fallback on error



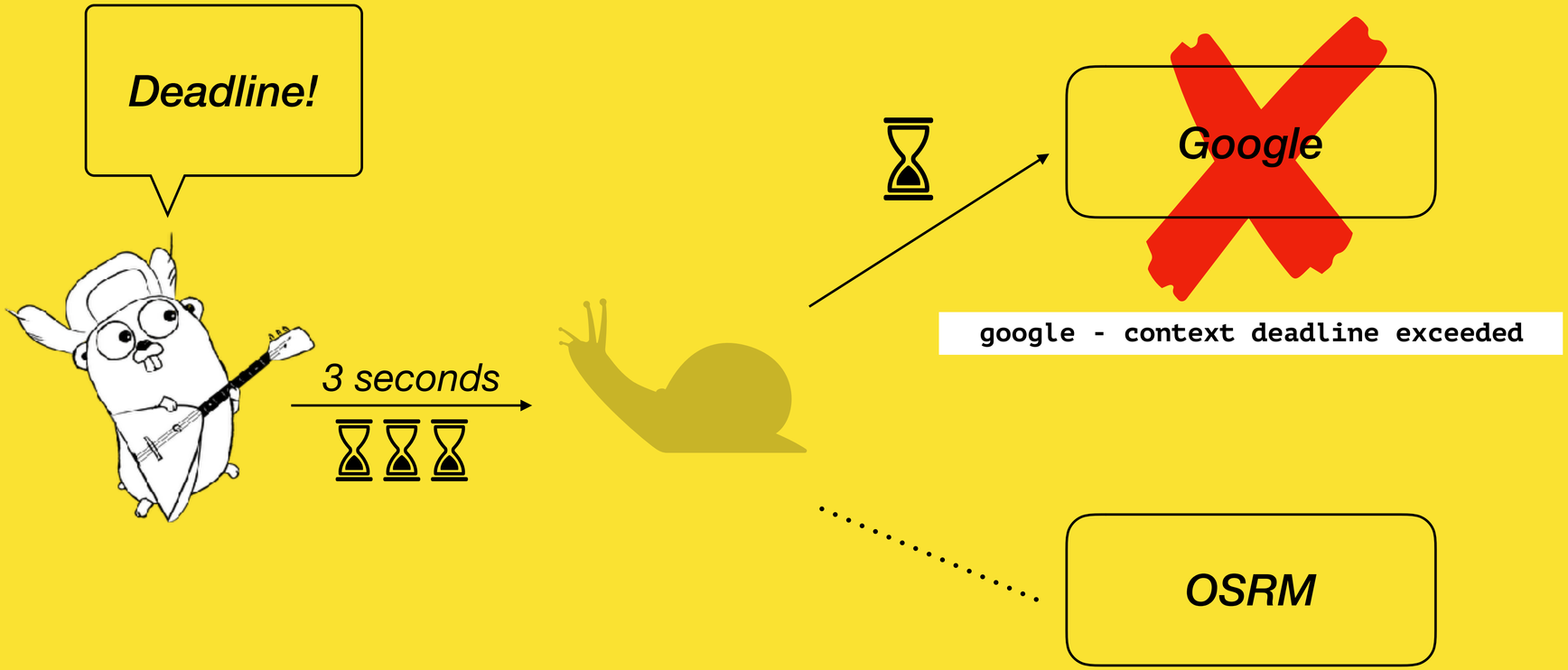
Naive approach – fallback on error



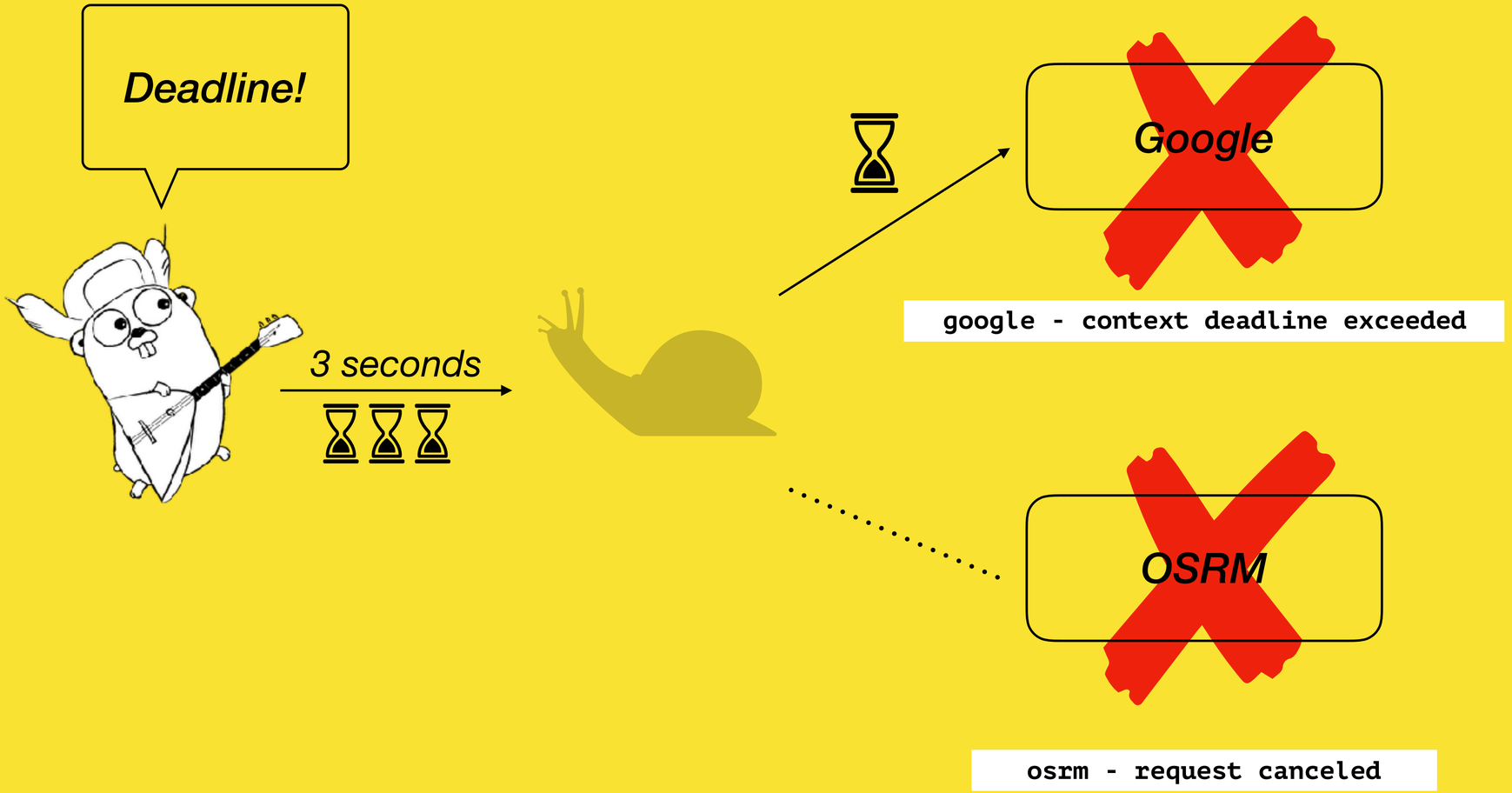
Naive approach – fallback on error



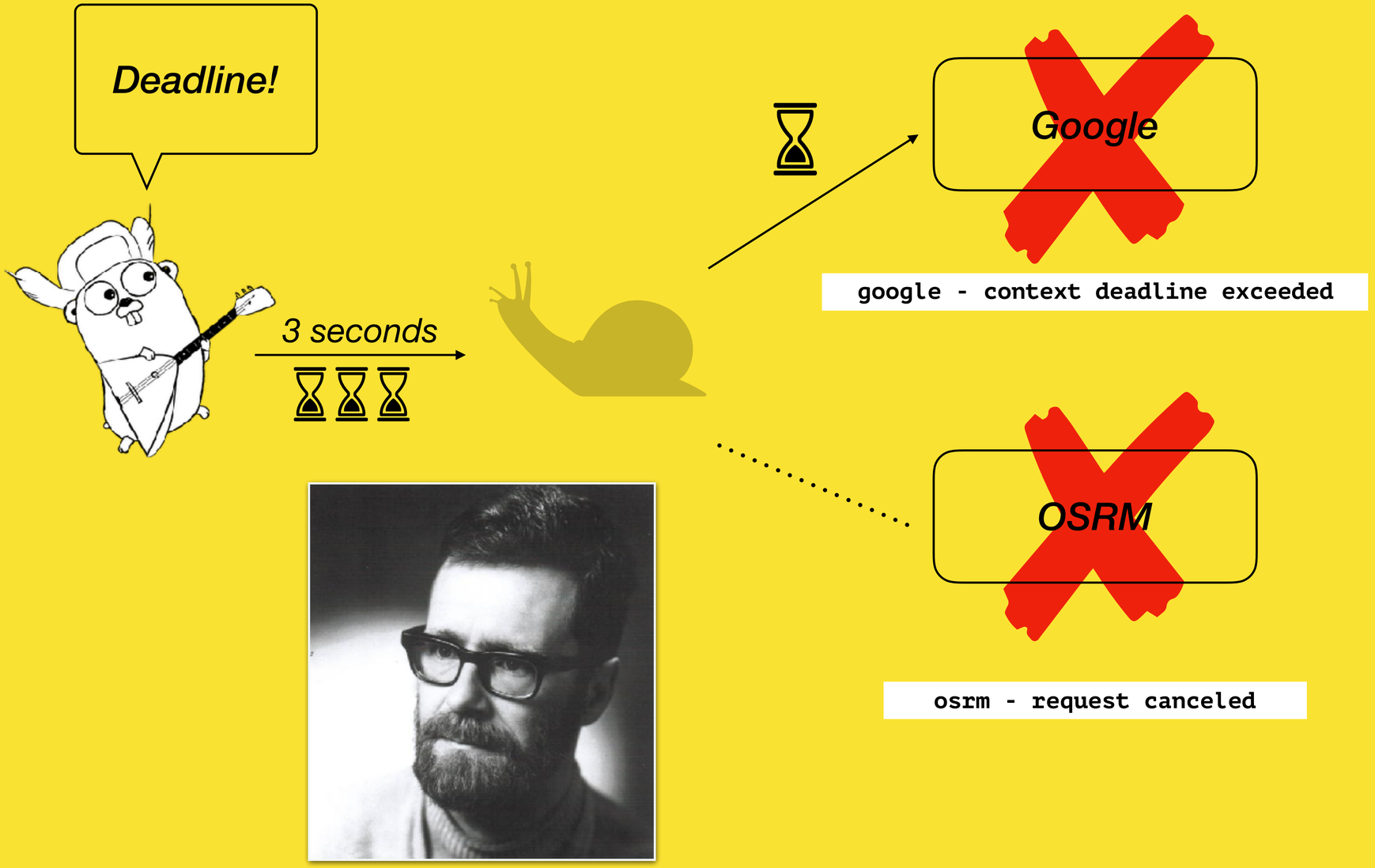
Naive approach – fallback on error



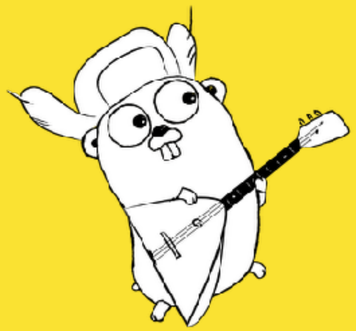
Naive approach – fallback on error

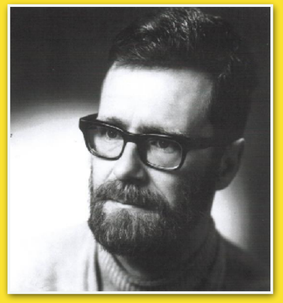


Naive approach – fallback on error



Brainy approach – shifted fallback

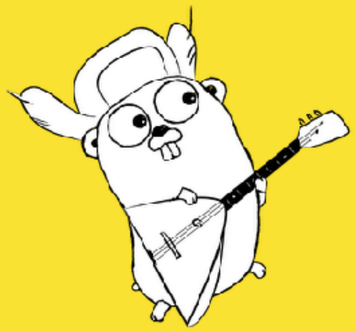




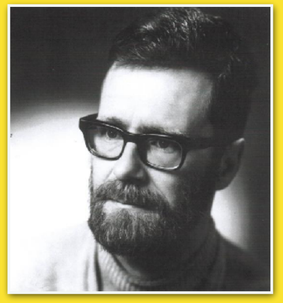
Google

OSRM

Brainy approach – shifted fallback



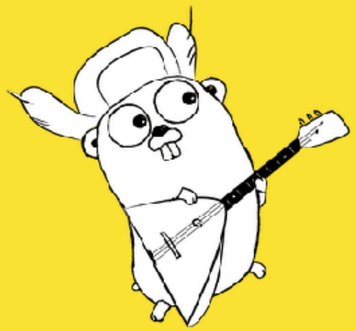
3 seconds



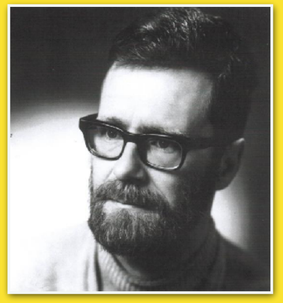
Google

OSRM

Brainy approach – shifted fallback



3 seconds



Google

OSRM

Brainy approach – shifted fallback

Brainy approach – shifted fallback

Brainy approach – shifted fallback

Brainy approach – shifted fallback

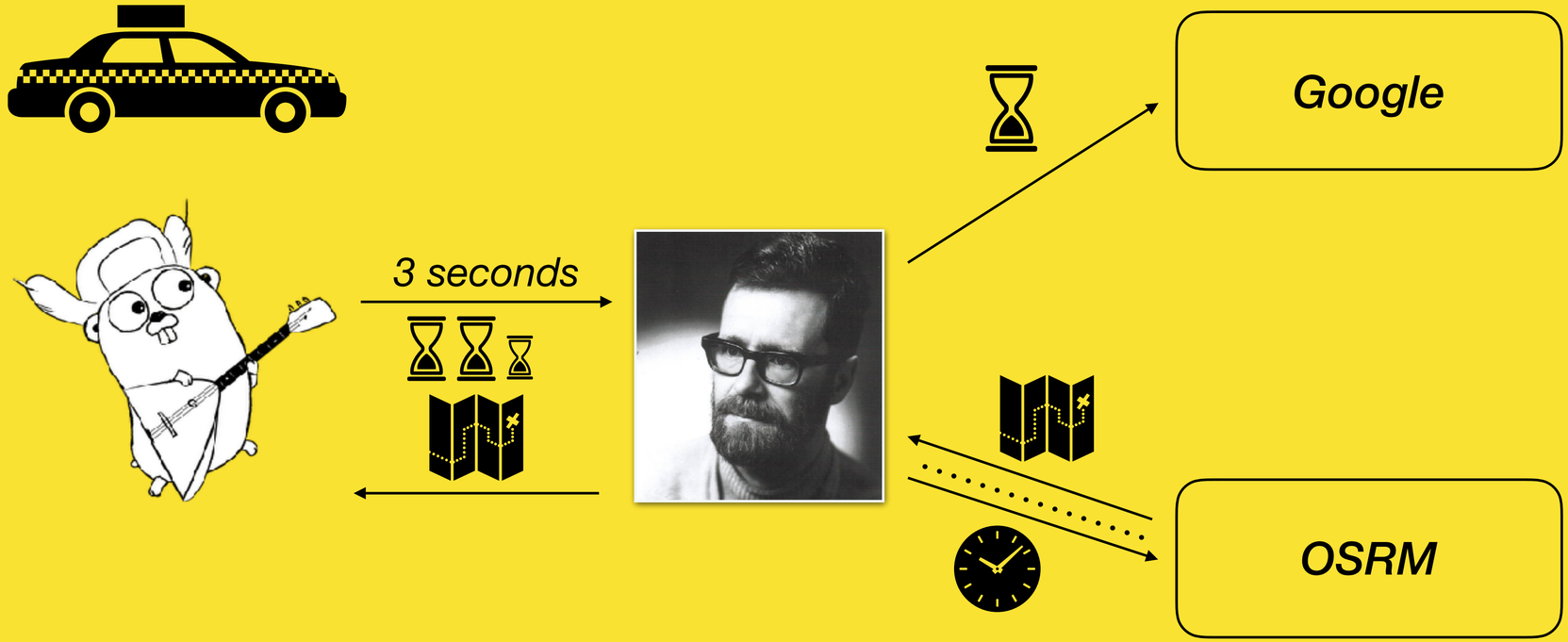
Brainy approach – shifted fallback

Brainy approach – shifted fallback

Brainy approach – shifted fallback

Brainy approach – shifted fallback

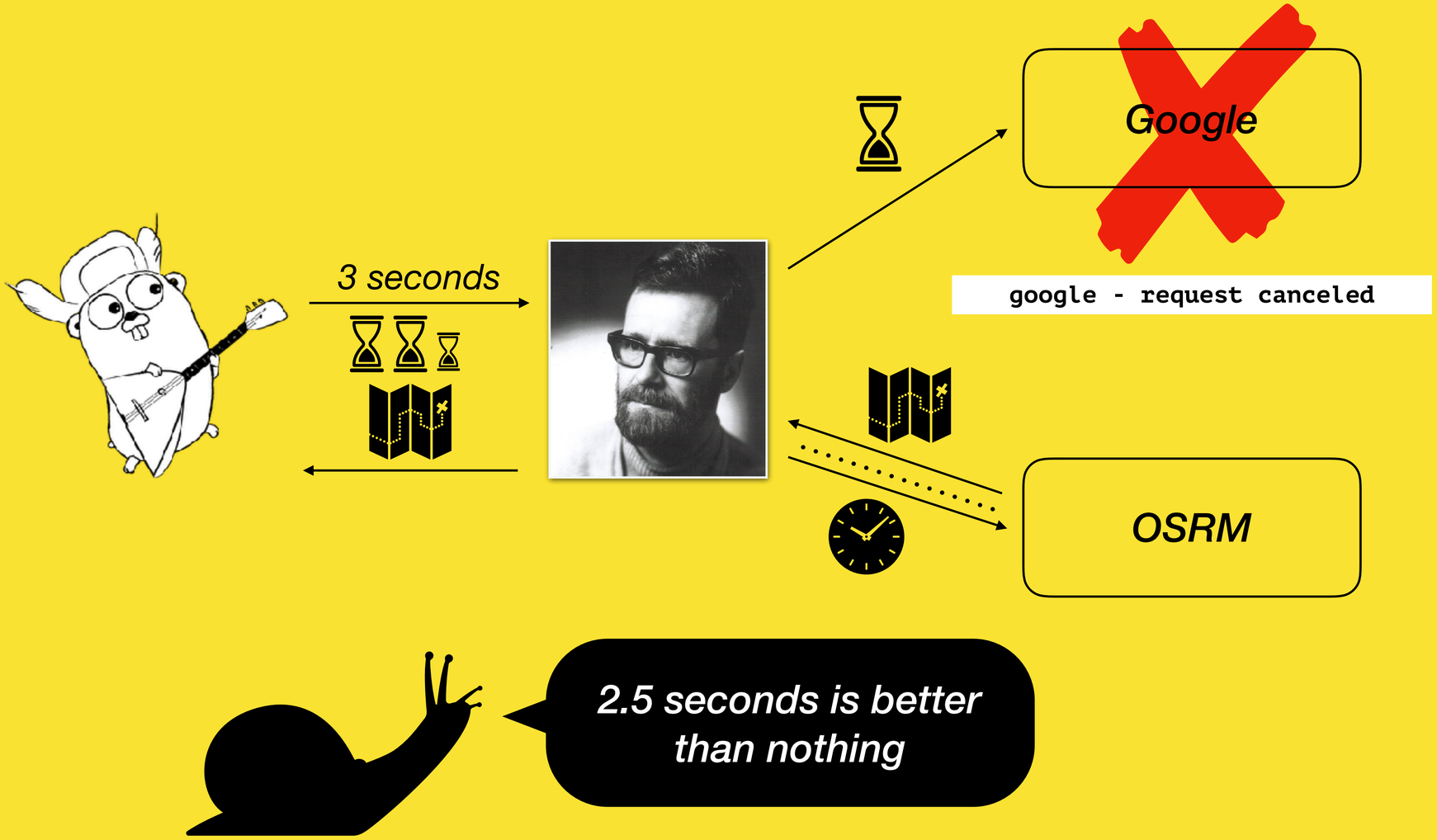
Brainy approach – shifted fallback

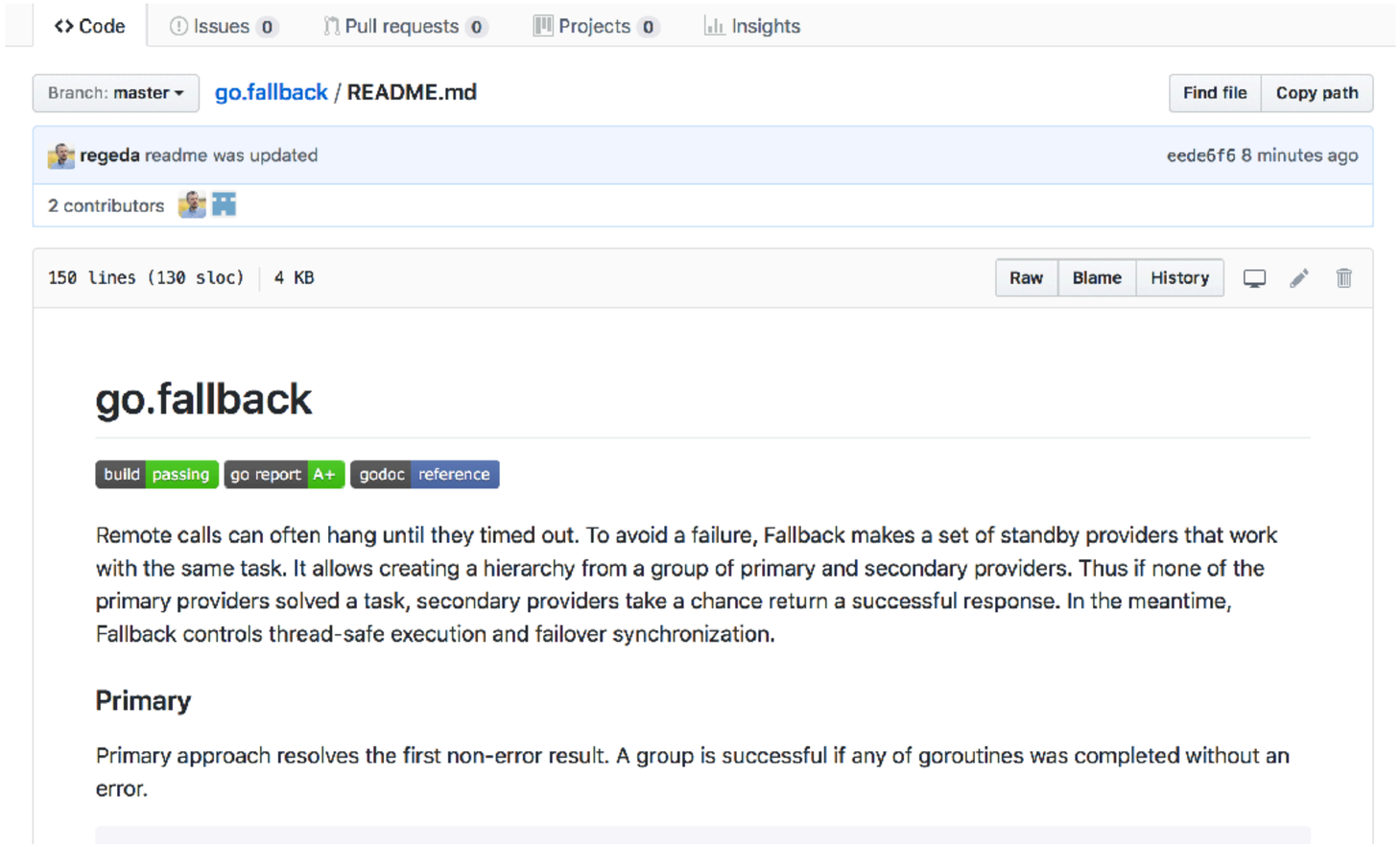


Brainy approach – shifted fallback

Brainy approach – shifted fallback

Brainy approach – shifted fallback




<> Code
Issues 0
Pull requests 0
Projects 0
Insights
Branch: master
go.fallback / README.md
Find file
Copy path
regeda readme was updated
eede6f6 8 minutes ago
2 contributors
150 lines (130 sloc)
4 KB
Raw
Blame
History
go.fallback
build passing
go report A+
godoc reference
Remote calls can often hang until they timed out. To avoid a failure, Fallback makes a set of standby providers that work with the same task. It allows creating a hierarchy from a group of primary and secondary providers. Thus if none of the primary providers solved a task, secondary providers take a chance return a successful response. In the meantime, Fallback controls thread-safe execution and failover synchronization.
Primary
Primary approach resolves the first non-error result. A group is successful if any of goroutines was completed without an error.

go.fallback

Ingredients

Ingredients

- sync.WaitGroup

Ingredients

- sync.WaitGroup
- sync.Cond

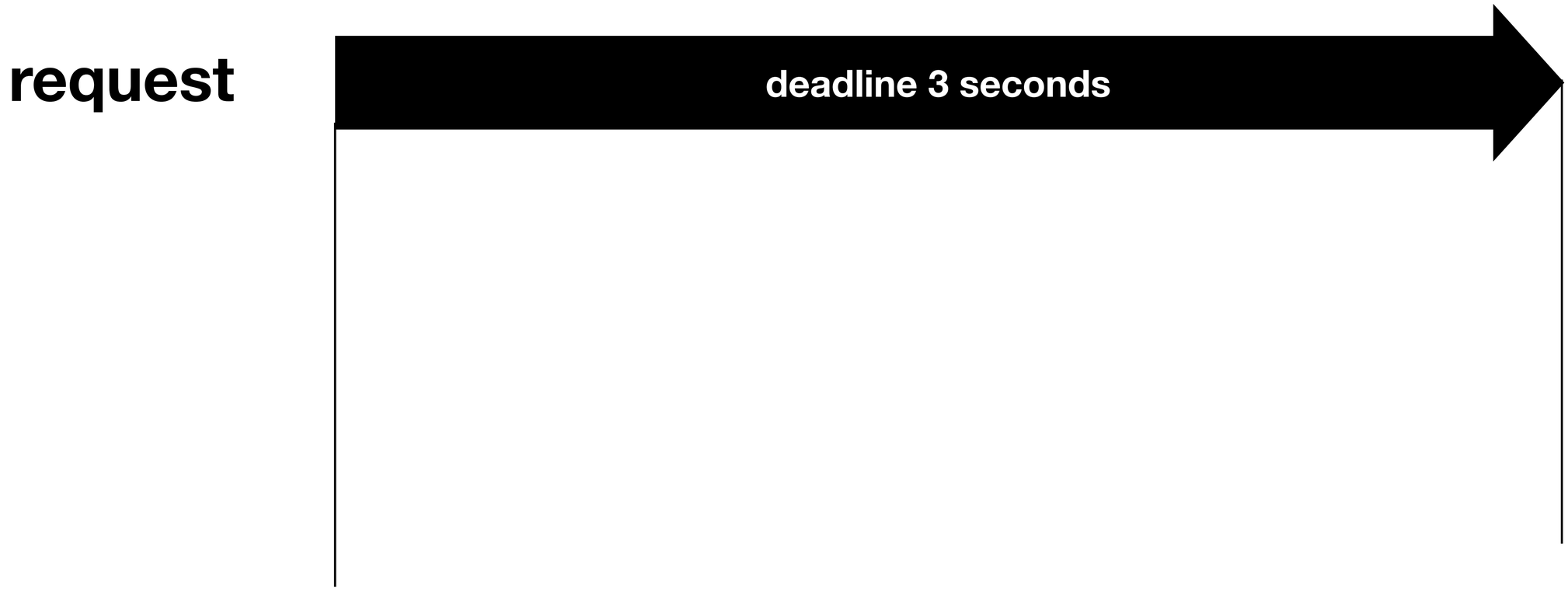
Ingredients

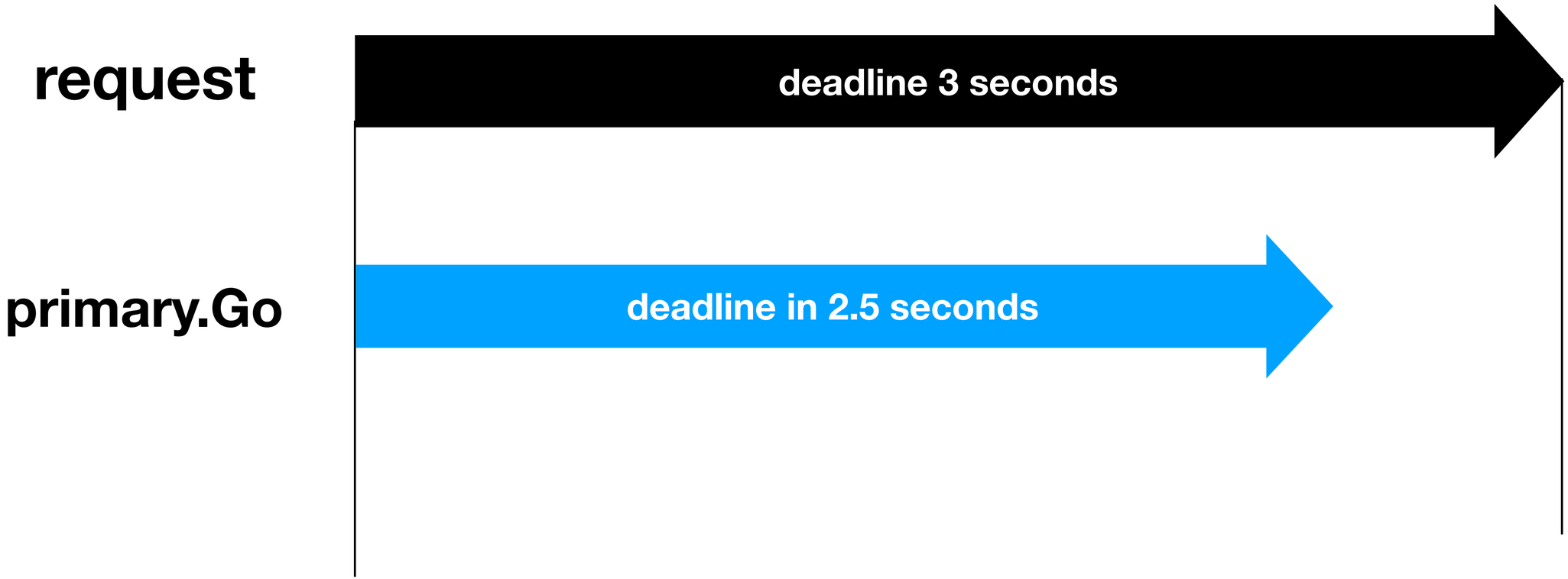
- sync.WaitGroup
- sync.Cond
- context.Context

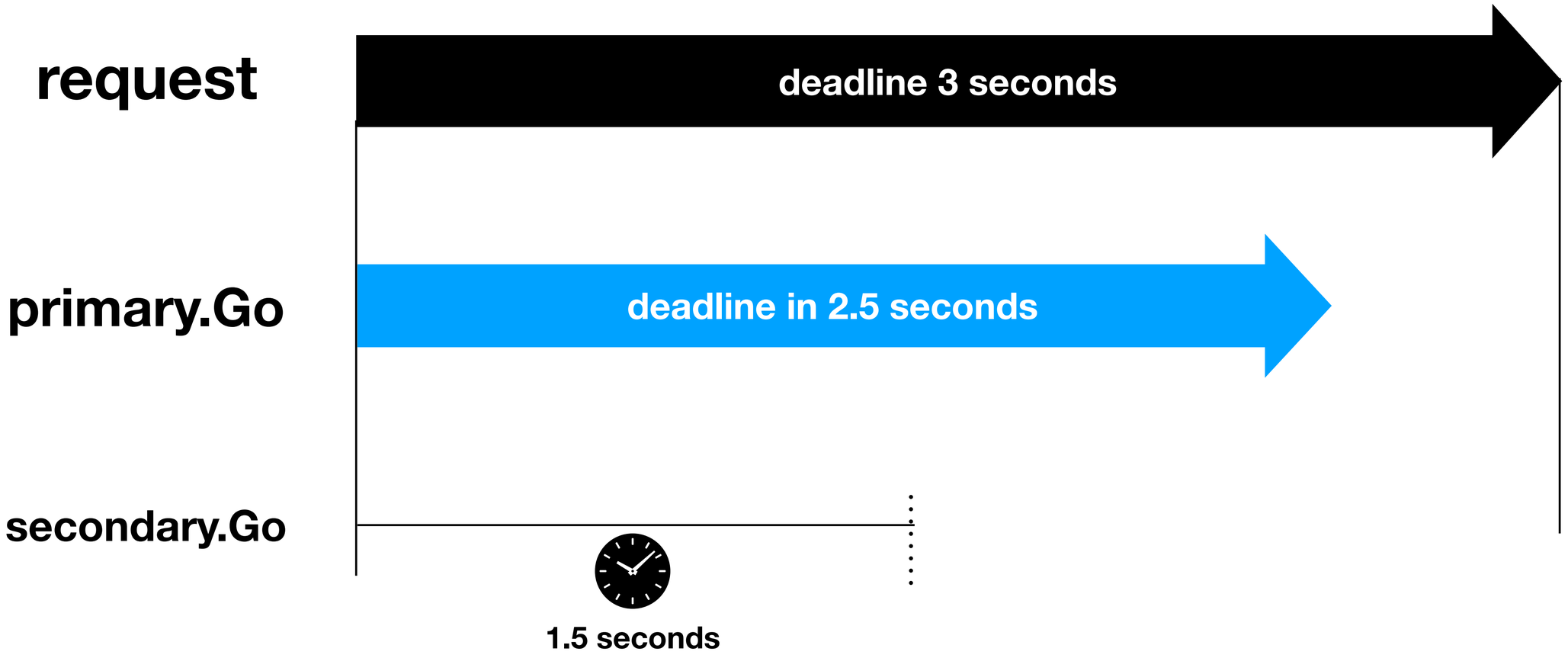
```
type Func func() (func(), error)

type Group interface {
    Go(Func)
    Wait() bool
}
```

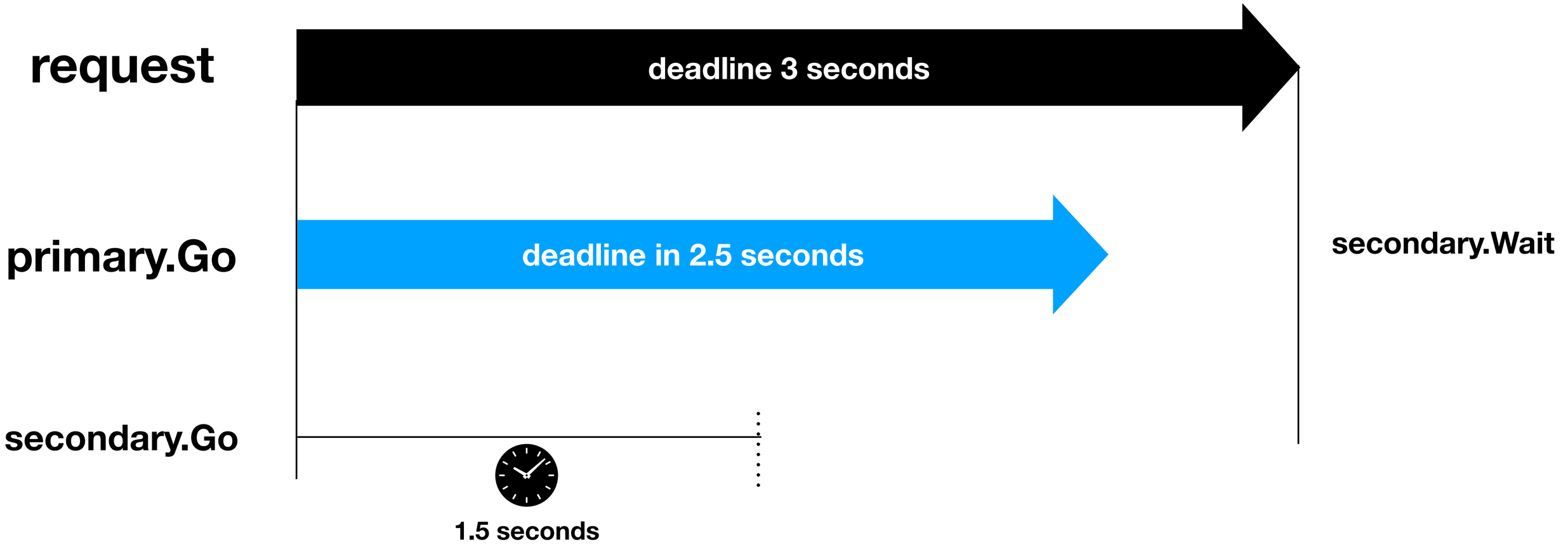
```
func NewPrimary() *Primary
func NewSecondary(Group) *Secondary
```

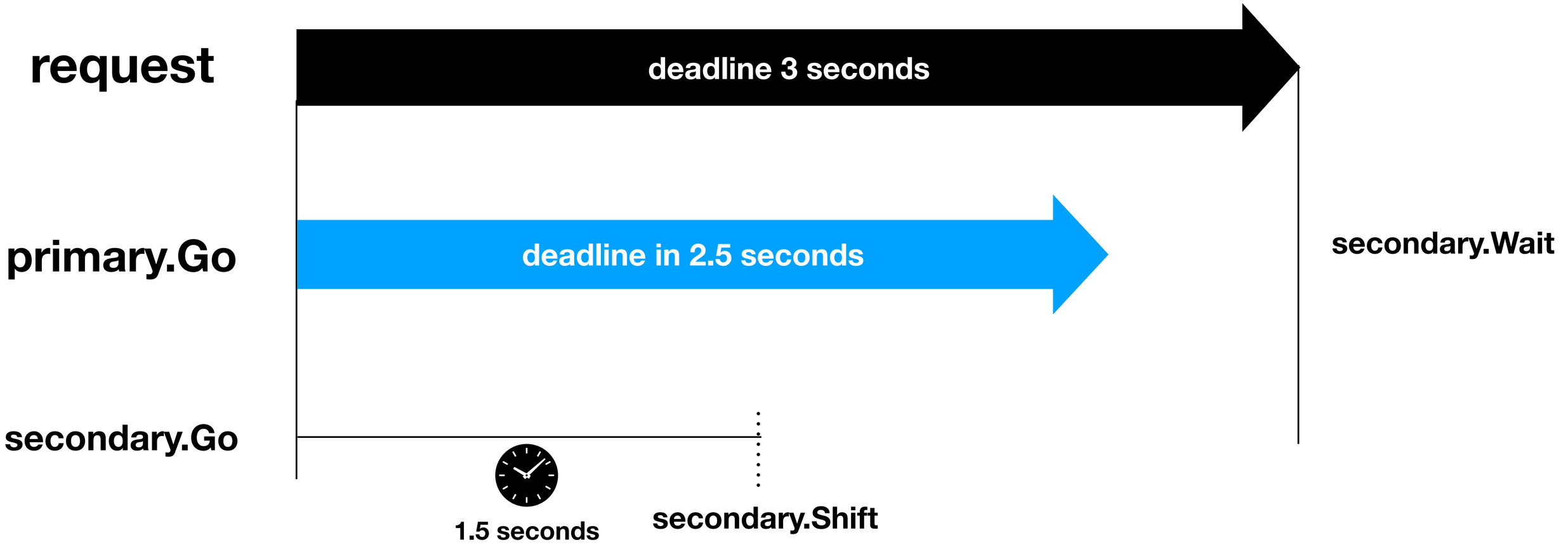
```
func (*Primary) Go(Func)
func (*Primary) Wait() bool


func (*Secondary) Go(Func)
func (*Secondary) Wait() bool
func (*Secondary) Shift()
```

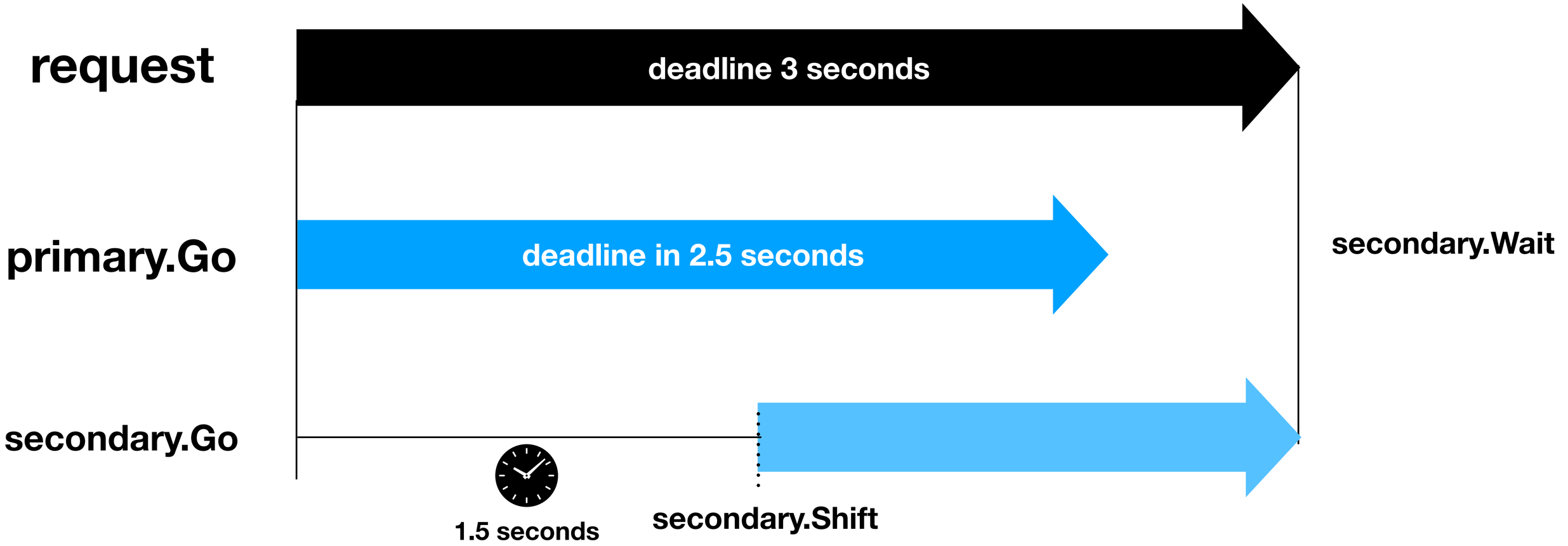

request
deadline 3 seconds


request
deadline 3 seconds
primary.Go
deadline in 2.5 seconds


request
deadline 3 seconds
primary.Go
deadline in 2.5 seconds
secondary.Go
1.5 seconds


request
deadline 3 seconds
primary.Go
deadline in 2.5 seconds
secondary.Wait
secondary.Go
1.5 seconds


request
deadline 3 seconds
primary.Go
deadline in 2.5 seconds
secondary.Wait
secondary.Go
1.5 seconds
secondary.Shift


request
deadline 3 seconds
primary.Go
deadline in 2.5 seconds
secondary.Wait
secondary.Go
1.5 seconds
secondary.Shift

Demo

“Anything that can go wrong will go wrong.”

–Murphy's law

@regeda

https://github.com/regeda/go.fallback

Use fallbacks!


@regeda


https://github.com/regeda/go.fallback