Cloud Patterns: Part 1

Systems Analysis & Design

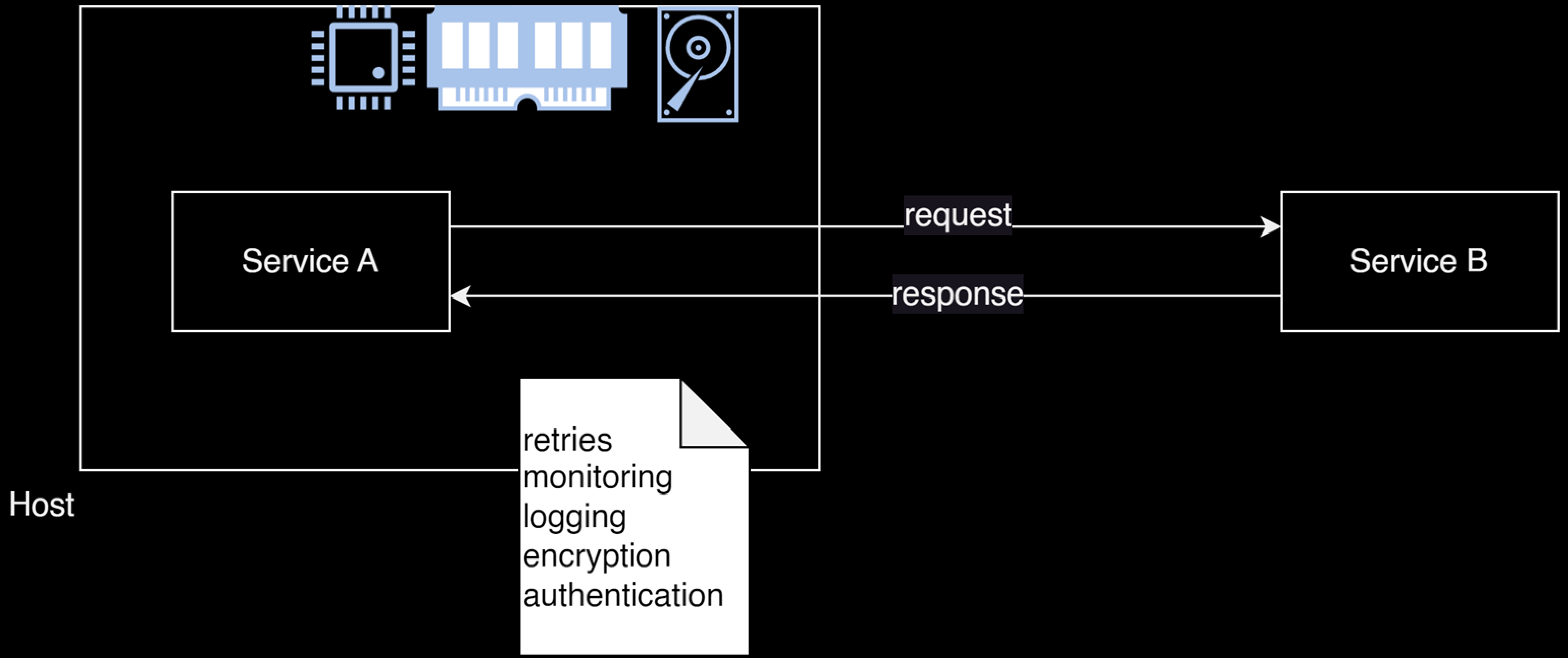
Learning Objectives

By the end of this session, you will have acquired the following information:

- Ambassador
- Anti-Corruption Layer
- Backends for Frontends
- Retry
- Circuit Breaker
- Bulkhead
- Dead Letter Queue
- Rolling Deployment
- Blue-Green Deployment
- Canary Deployment
- Scatter-Gather
- CQRS

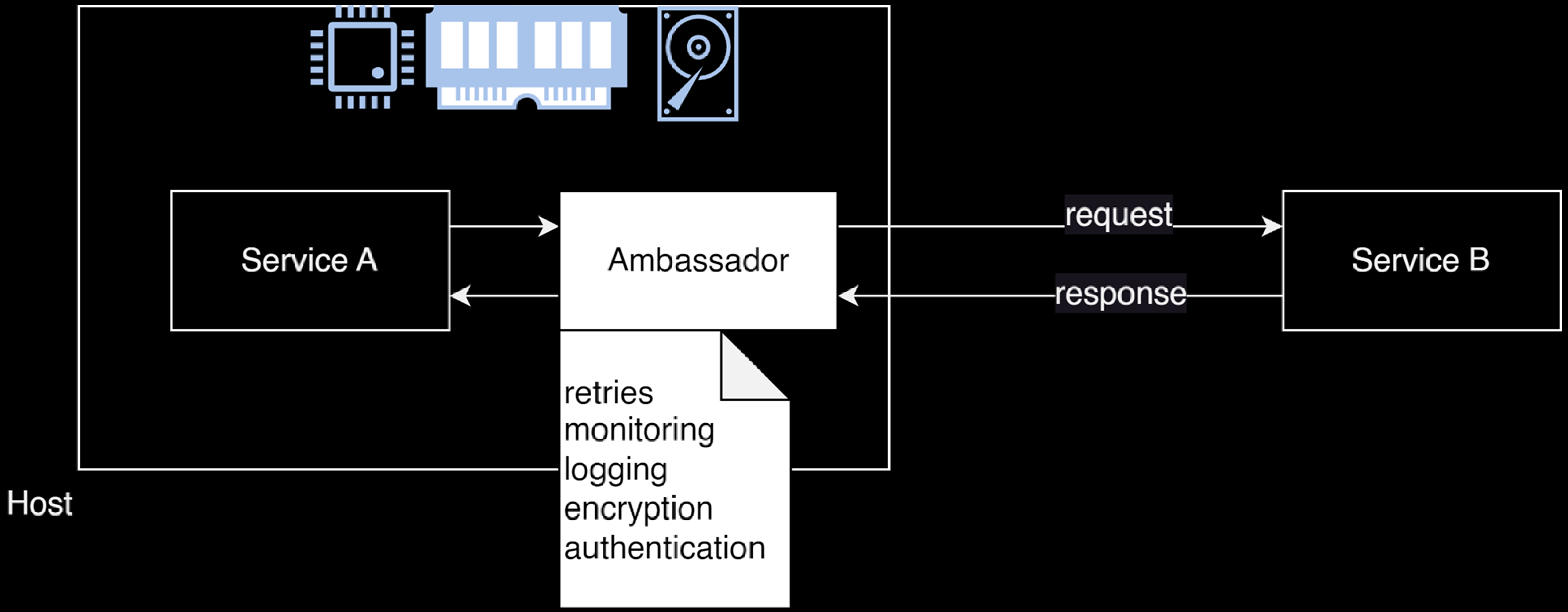
Ambassador: Problem

- We have to implement infrastructure-related features:
  - monitoring
  - service discovery
  - logging
  - encryption
- What if Service A is a legacy service and we are unable to change its source code?
- What if we have multiple services, each developed in a different programming language?



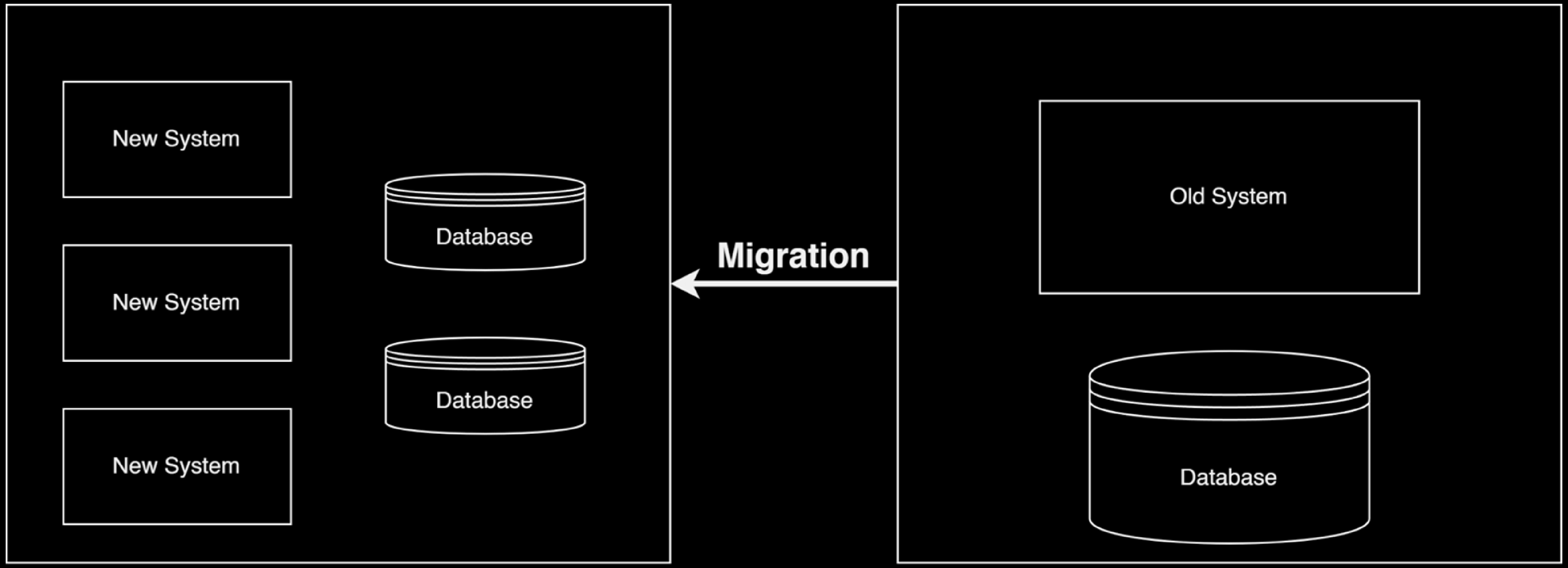
Ambassador: Solution

- Service meshes like Consul or Istio embody this concept at a production level.
- In the Kubernetes environment, we deploy the Ambassador within the same pod that houses the service container.



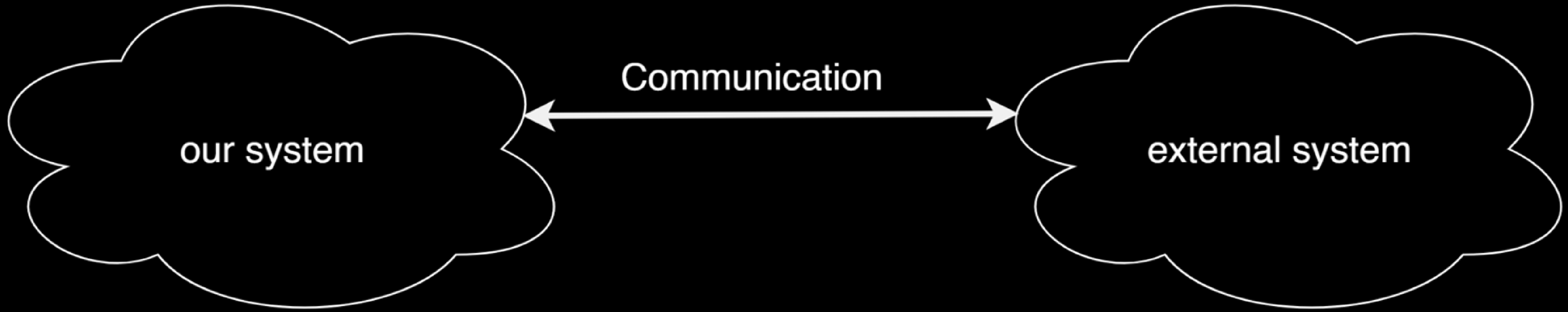
Anti-Corruption Layer: Problem 1

- During migration, we may need to interface with the legacy system and data.
- The legacy system may employ an outdated protocol.

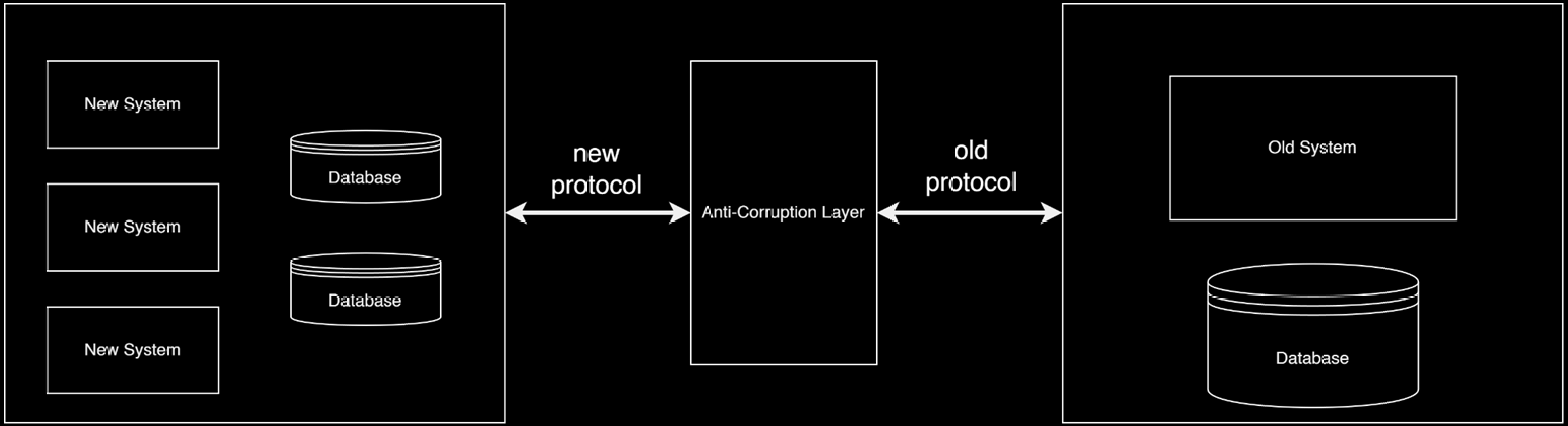


Anti-Corruption Layer: Problem 2

- We may need to engage with external systems, beyond our control, that could be using an outdated protocol.

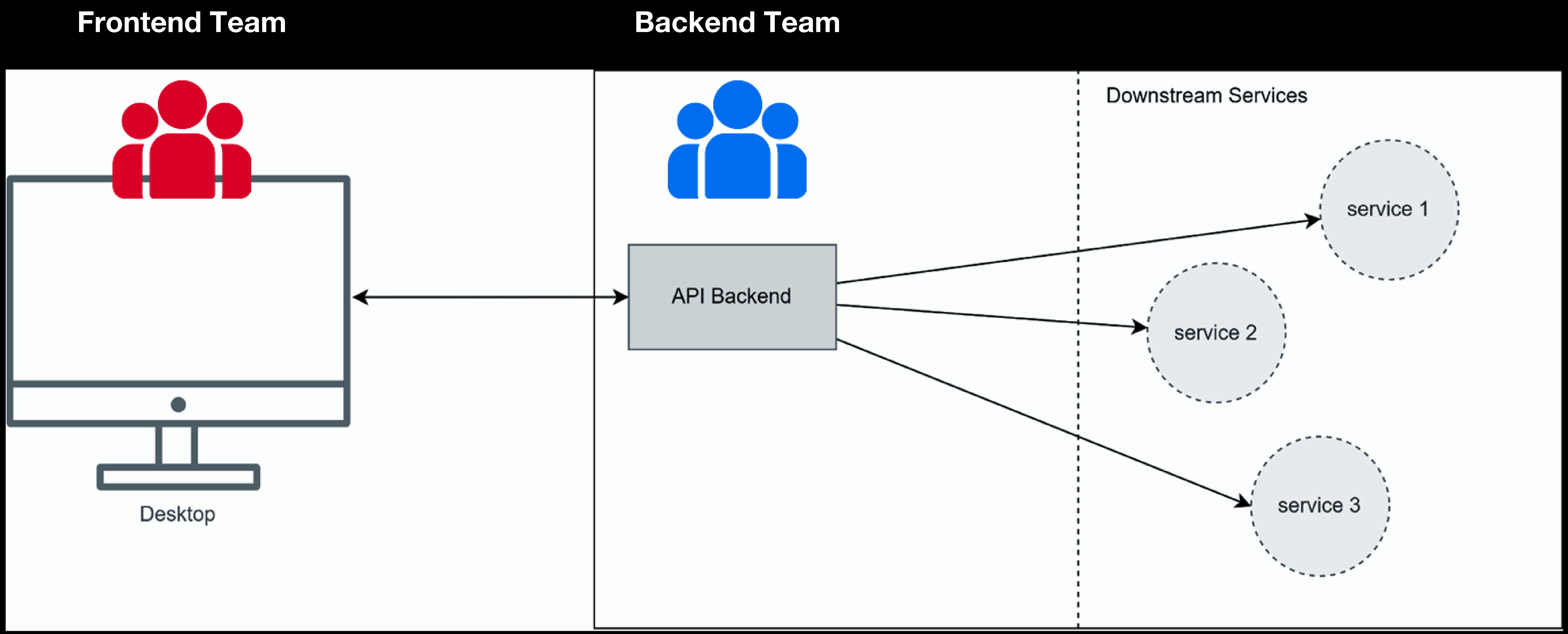


Anti-Corruption Layer: Solution

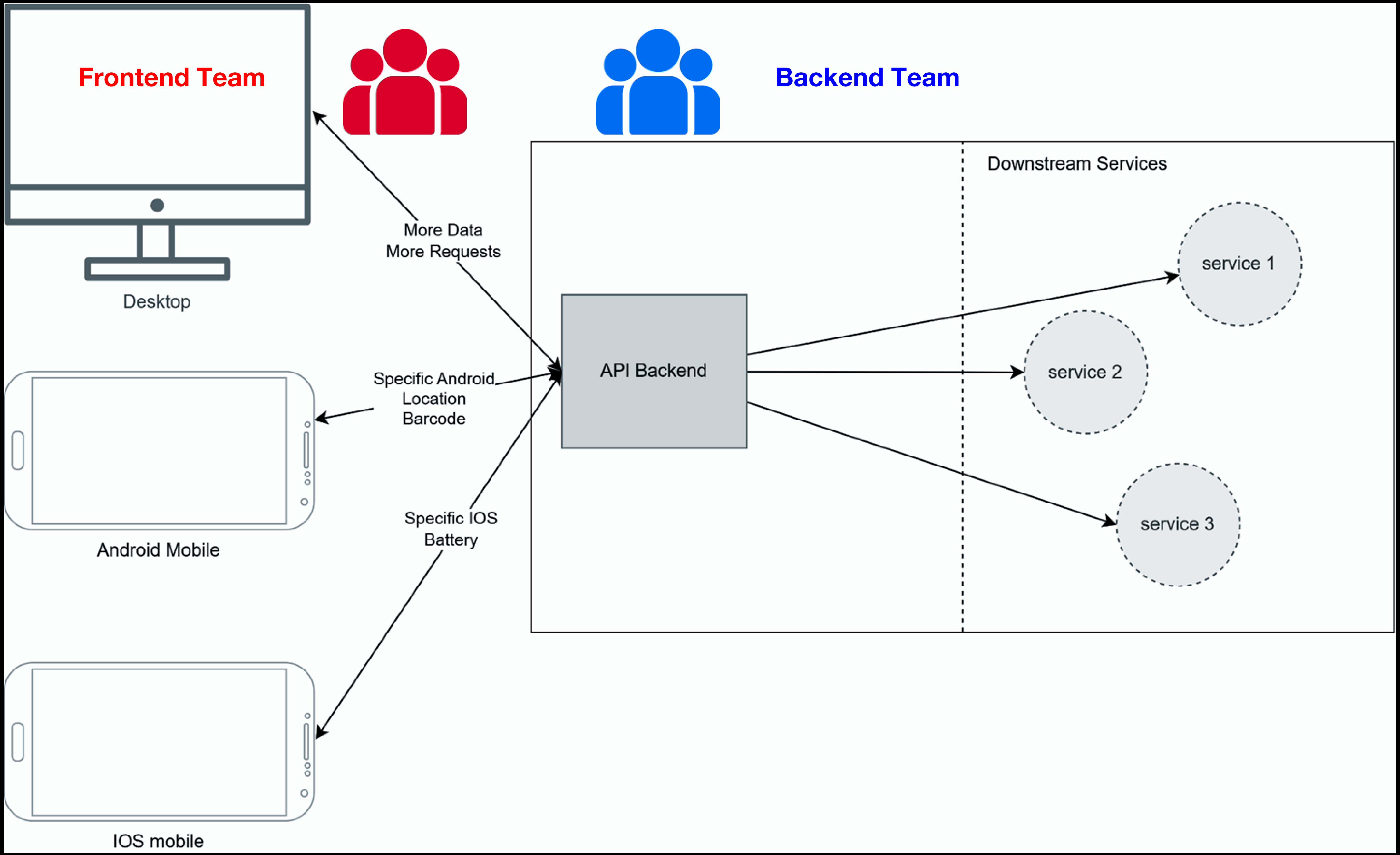


Backends for Frontends: Problem

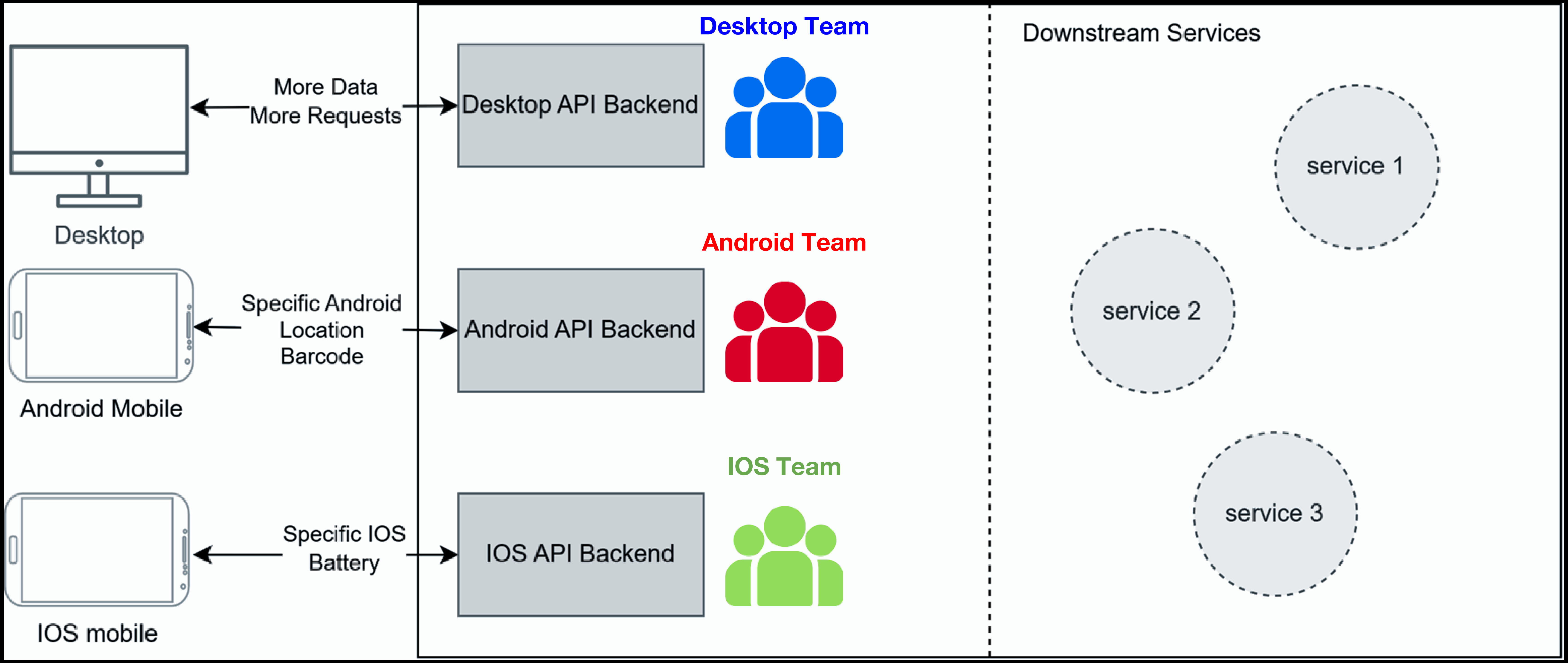
- For interfacing with multiple downstream services, an API backend is required.



Backends for Frontends: Problem

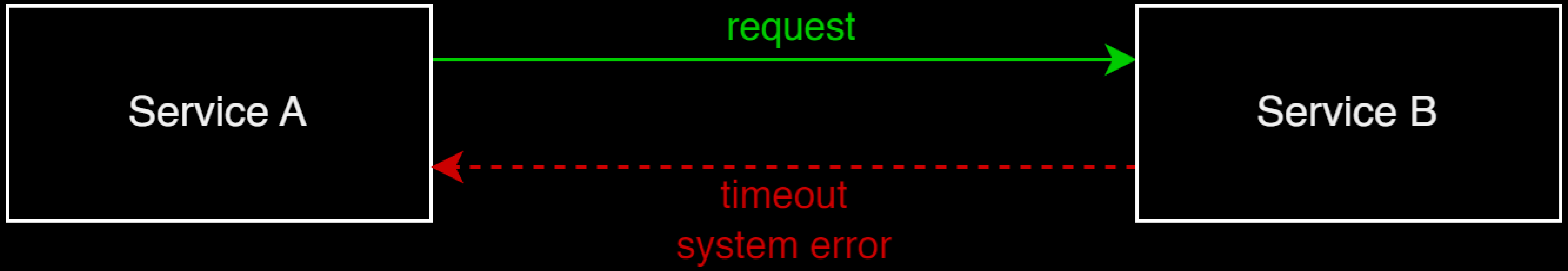


Backends for Frontends: Solution



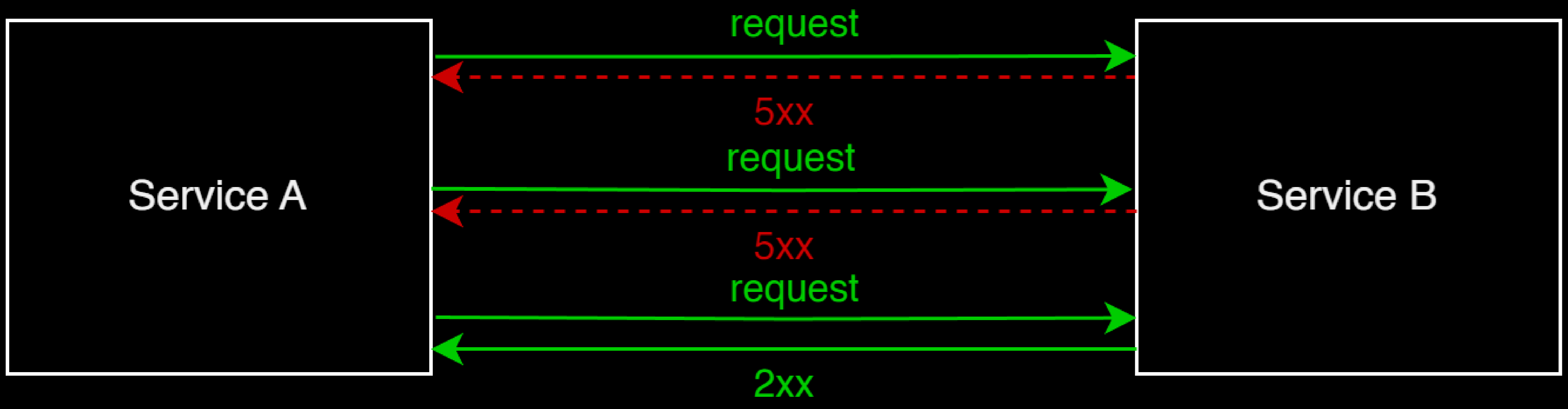
Retry: Problem

- The fault is temporary and short-lived.
- The service has the potential to recover from the fault and return to normal operations soon



Retry: Solution

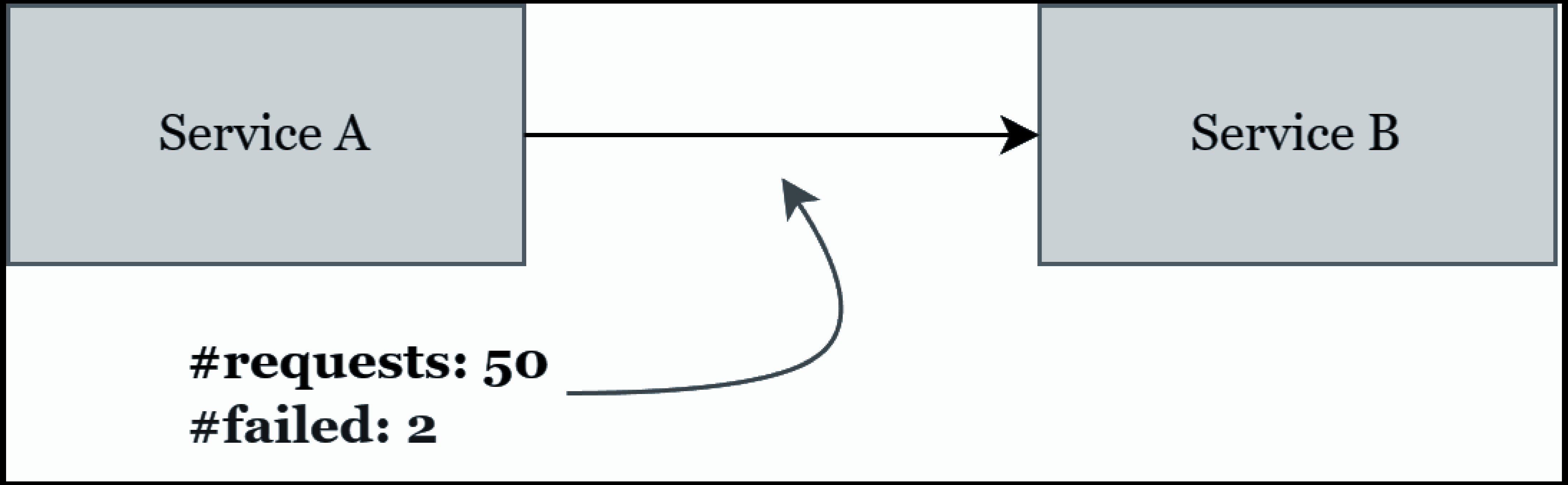
- Idempotency in the invoked operation is important.
- What should be the time intervals for sending requests to the service: Fixed, Incremental, or Exponential?
- How many times should the request be sent?

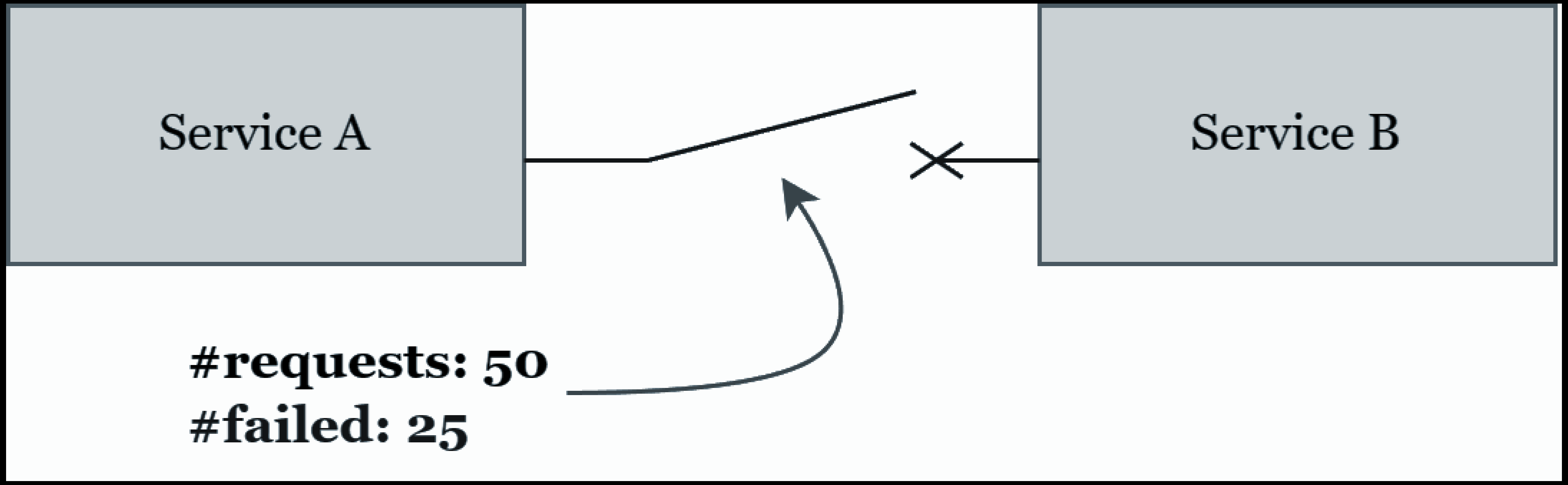


Circuit Breaker: Problem

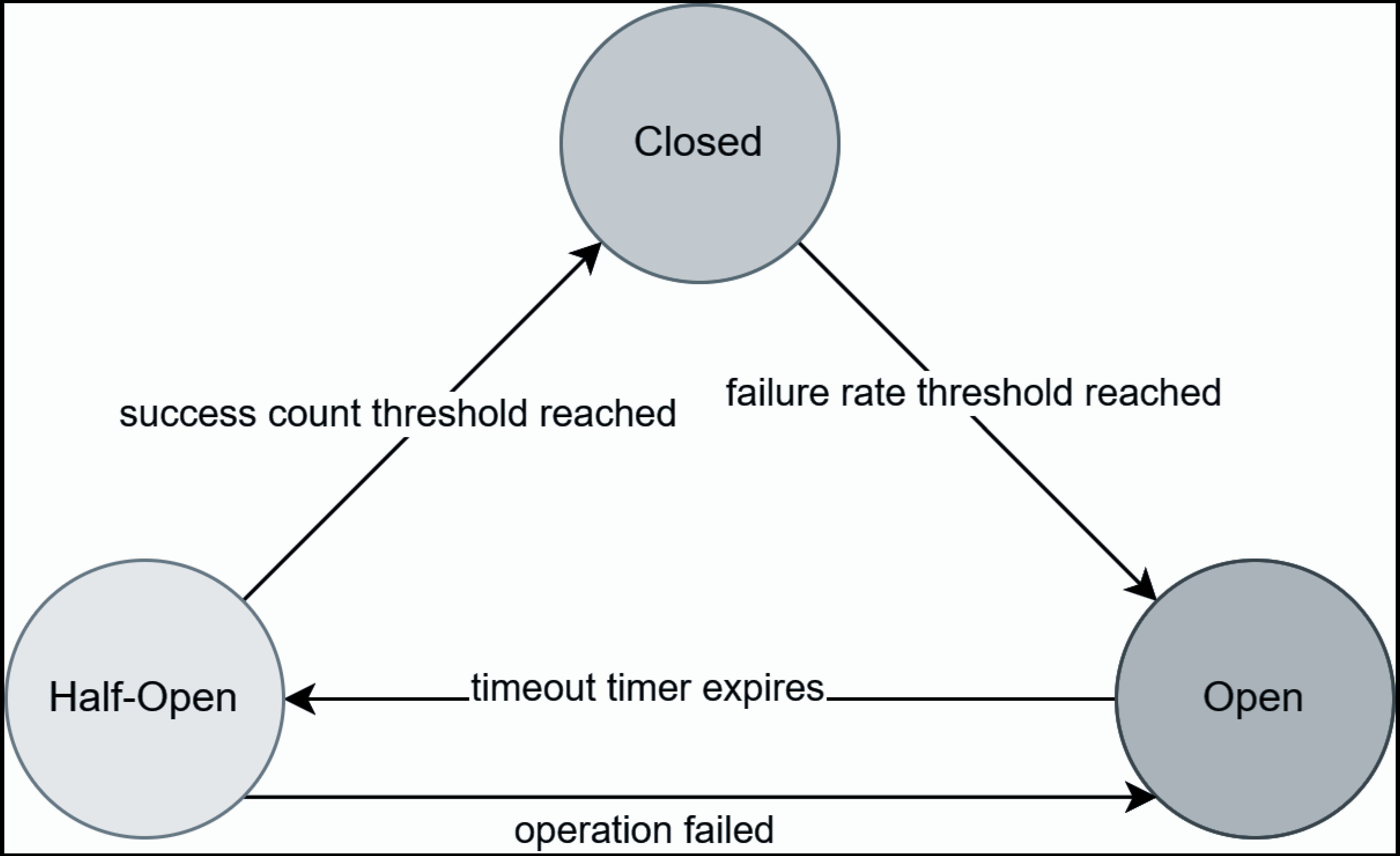
- We operate under the assumption that faults are temporary when using the retry pattern. But what happens if they’re not?
- Optimistic view: even if one request fails, the subsequent one will succeed.
- Pessimistic view: if a series of requests have failed, the following one will fail as well.

Circuit Breaker: Solution



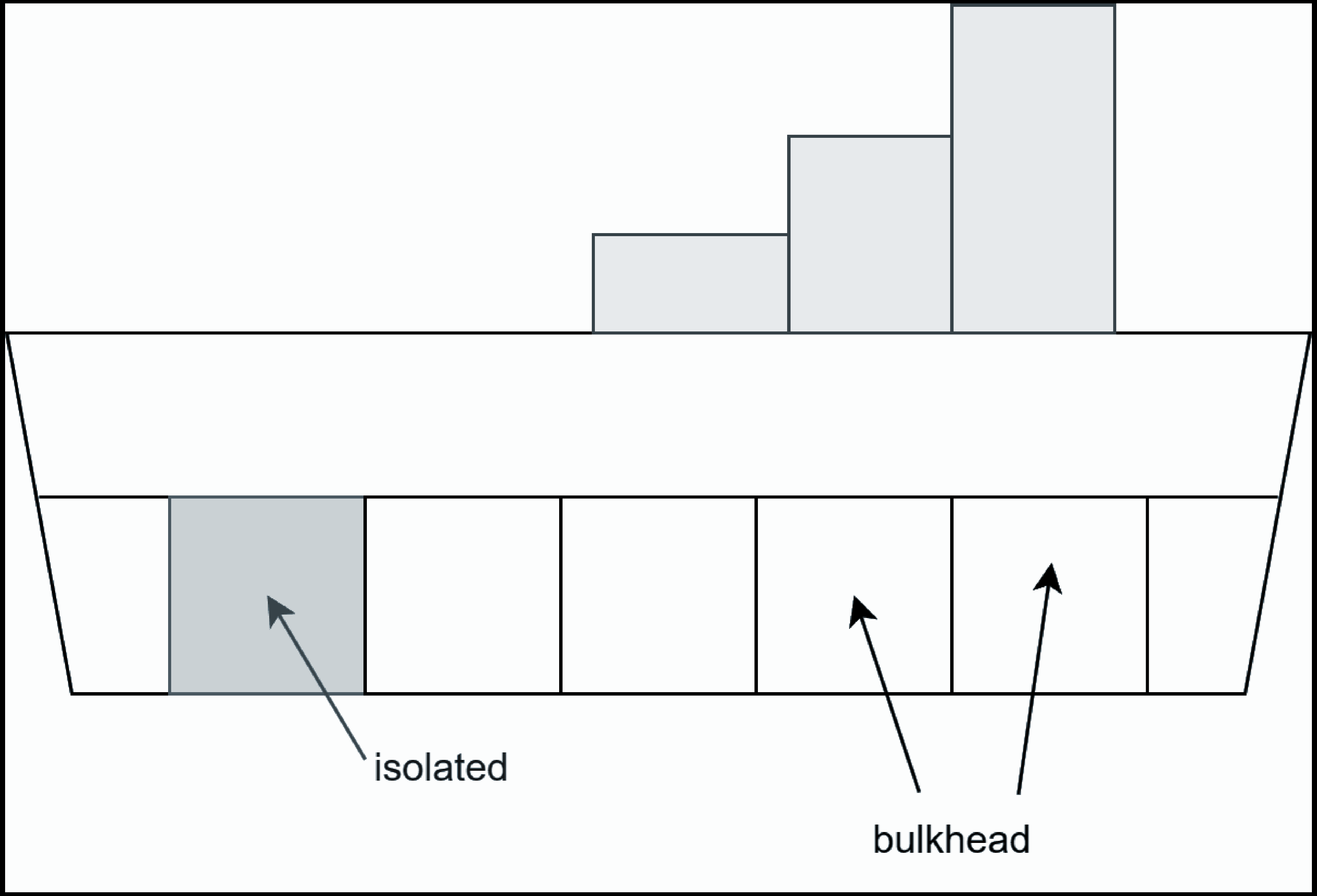


Circuit Breaker: Solution


Closed
failure rate threshold reached
Open
timeout timer expires
Half-Open
success count threshold reached
operation failed

Bulkhead

Using excessive resources of one service can negatively impact other services.



Bulkhead: Kubernetes example

```
apiVersion: v1
kind: pod
metadata:
    name: nginx-reverse-proxy
spec:
    containers:
        - name: nginx-reverse-proxy-container
          image: nginx
          resources:
            requests:
                memory: "512Mi"
                cpu: "1"
            limits:
                memory: "1024Mi"
                cpu: "2"
```

Further Resources

- youtube.com/@golemcourse