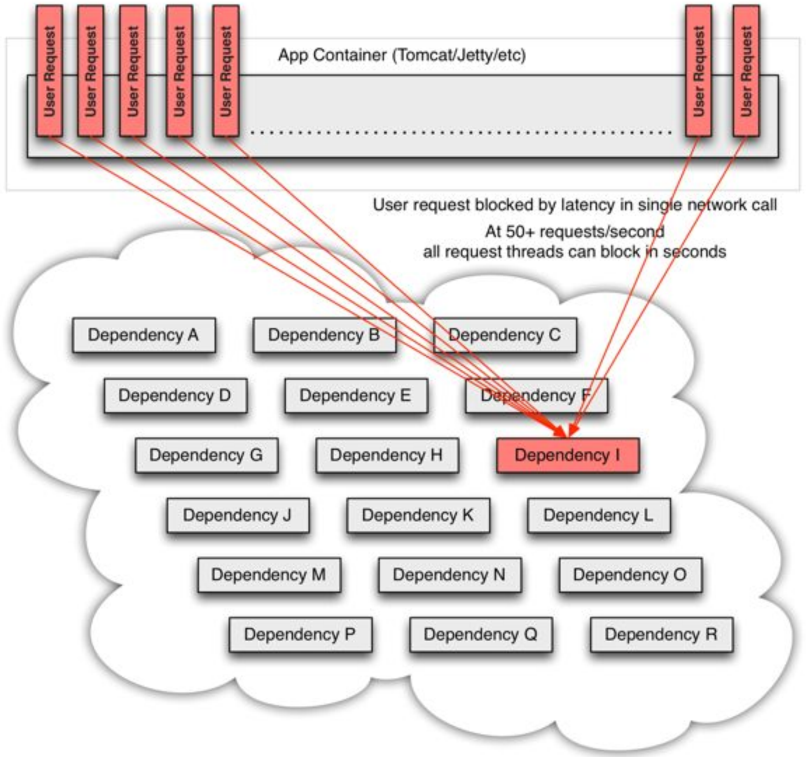
Pivotal®

Circuit Breaker Pattern

Problem - Failure Cascades to Caller

- *Slow Responses* may contribute to *Blocked Threads*
- Calling application's thread pool is exhausted waiting on misbehaving dependency. This is a *Chain Reaction* induced by *Blocked Threads*
- Failure cascades to caller, this results in *Cascading Failure*
- It may be better to *Fail Fast*



Problem

- There is a large percentage or probability of failures
- Impact to system results in *Cascading Failure* or *Dog Piles*
- Does it make sense to torture your system further?

Solution - Fail Fast

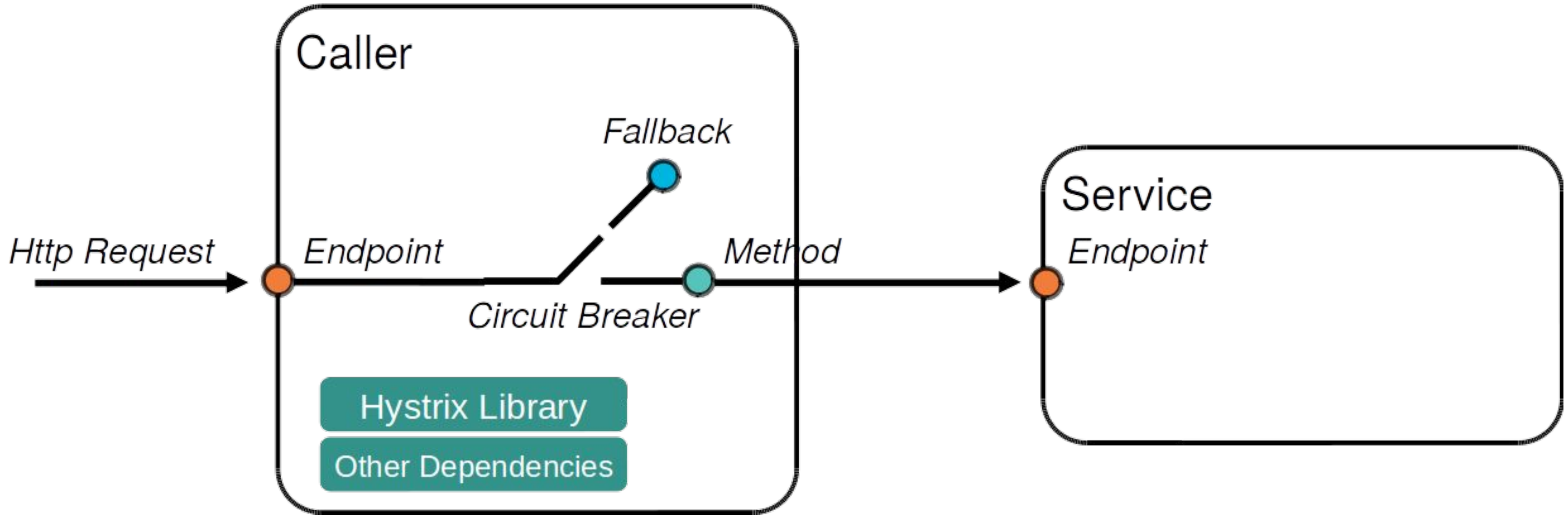
- Avoid or prevent executions in an already unstable part of the system
- This solution will require special handling, monitoring and tools

Solution - Circuit Breaker

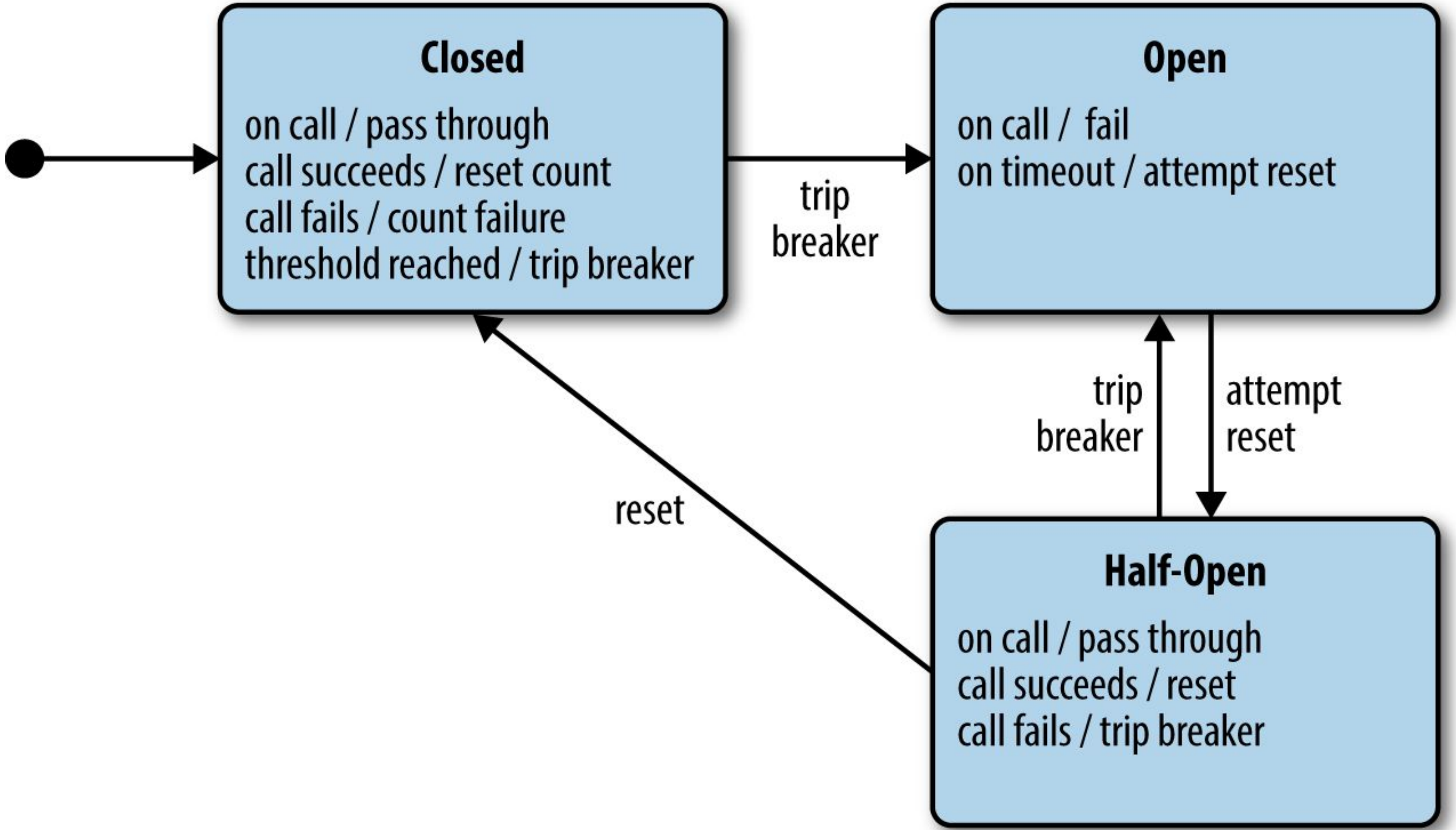
- Similar in concept to a residential or industrial electrical circuit breaker.
- Goal is to protect the “client”...
- ...But has side effect of partially or fully isolating faulty downstream dependencies
- Open State - primary path not allowed to execute
- Closed State - primary path allowed to execute
- Half-Open State - Protection algorithm allows small percentage requests to execute and monitor for success or failure
    - Failures keep the circuit open
    - Successes reset the circuit to close state

Circuit Breaker

isolates calls to other services



Circuit Breaker Lifecycle


Closed
on call / pass through
call succeeds / reset count
call fails / count failure
threshold reached / trip breaker
trip breaker
Open
on call / fail
on timeout / attempt reset
trip breaker
attempt reset
reset
Half-Open
on call / pass through
call succeeds / reset
call fails / trip breaker