HYSTRIX

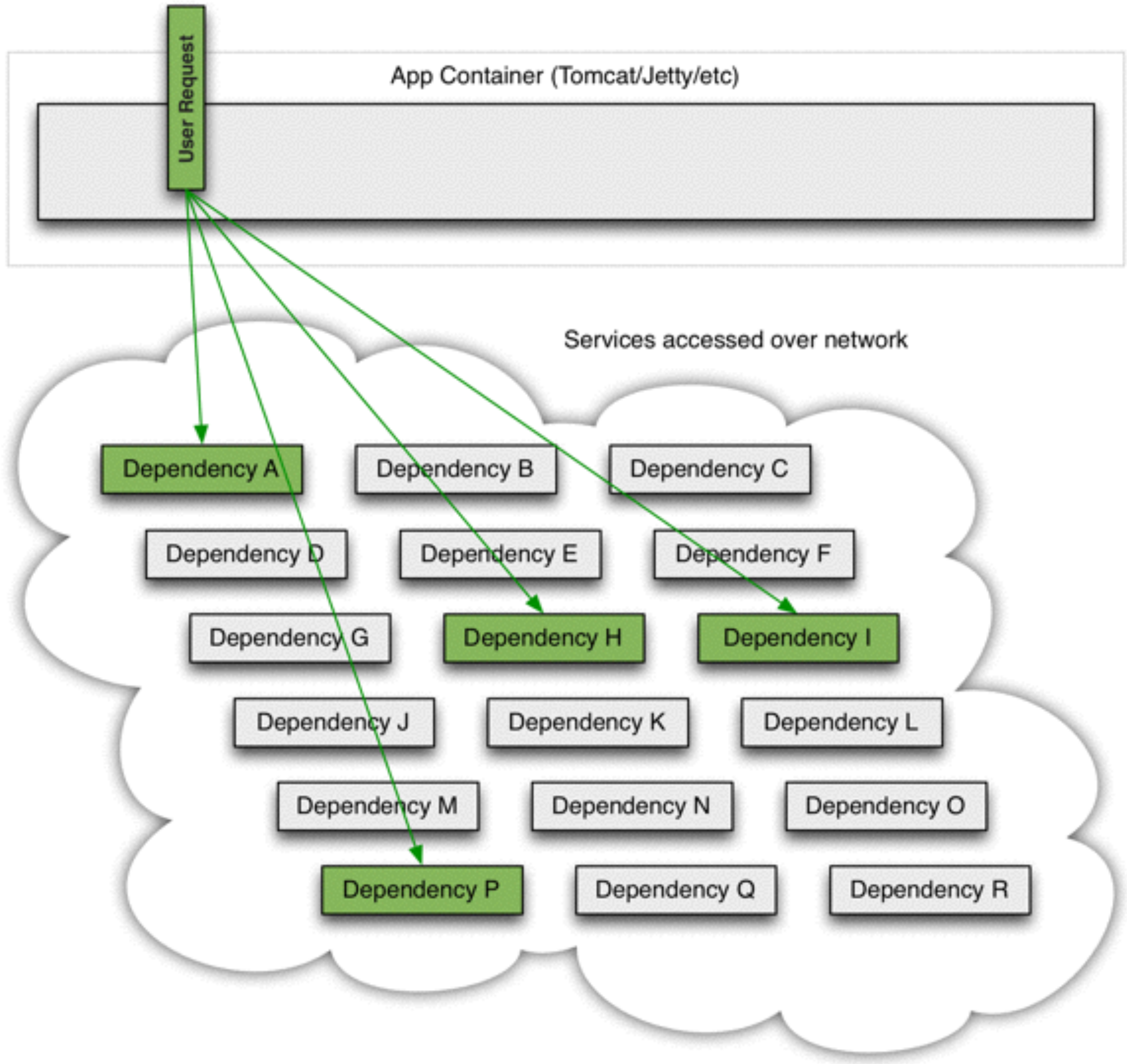
DEFEND YOUR APP

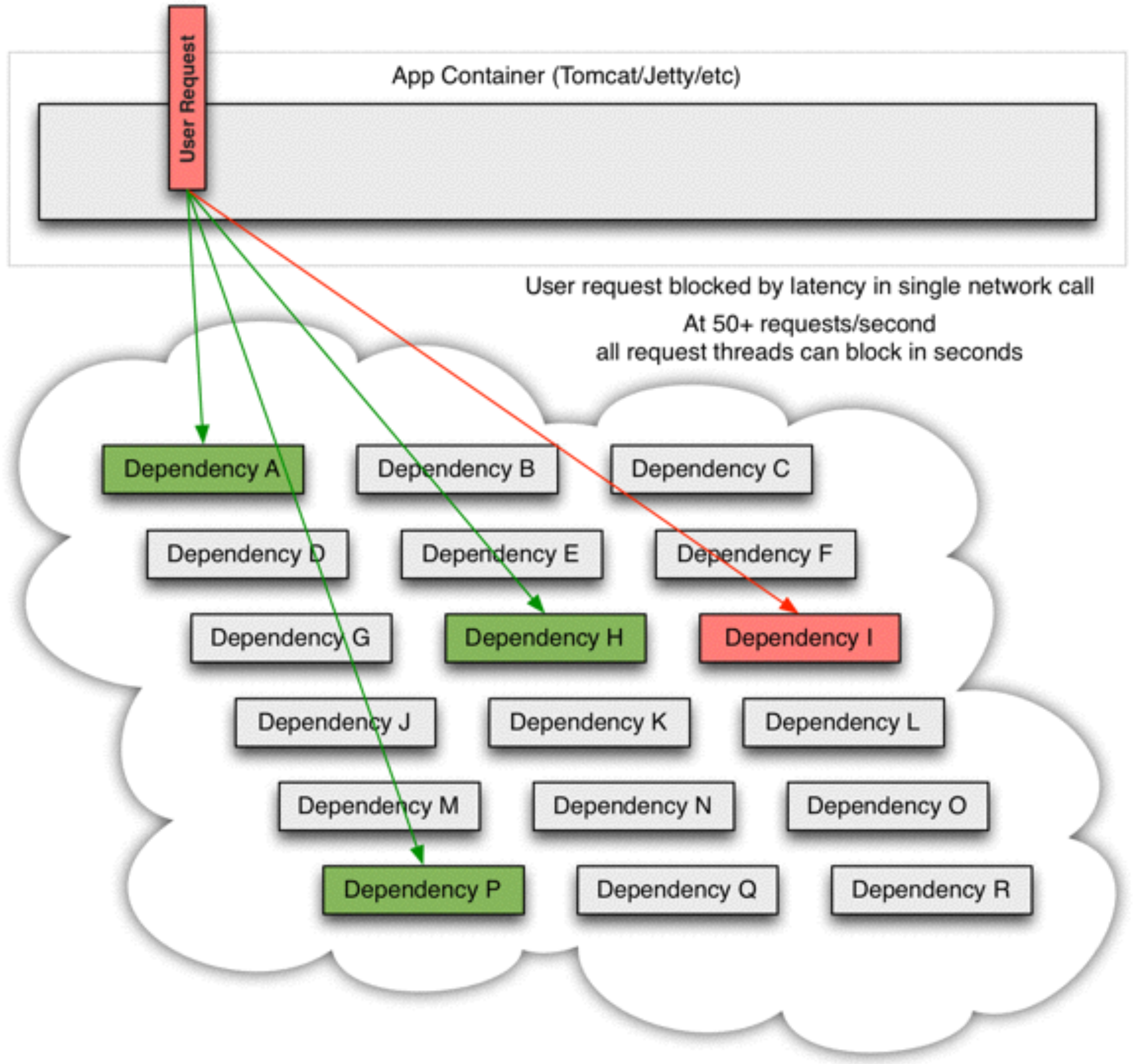
LIAOKAILIN

WHAT IS HYSTRIX

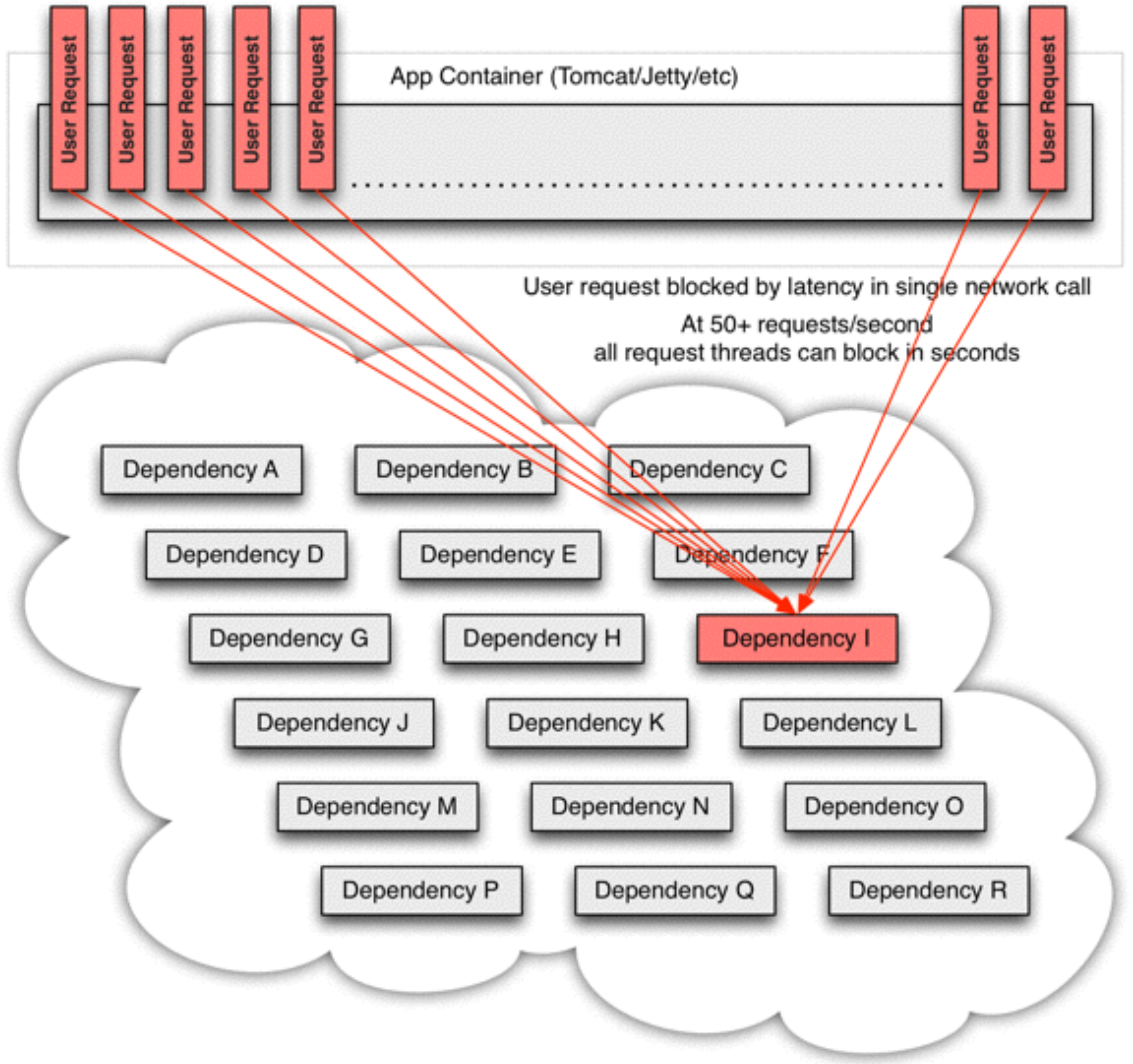
WITH SERVICE ISOLATION, CIRCUIT BREAKER AND FALLBACK
FOR CONTROL DEPENDENCIES SERVICE LATENCY OR FAULT

WHAT HYSTRIX SOLVE





What Hystrix Solve

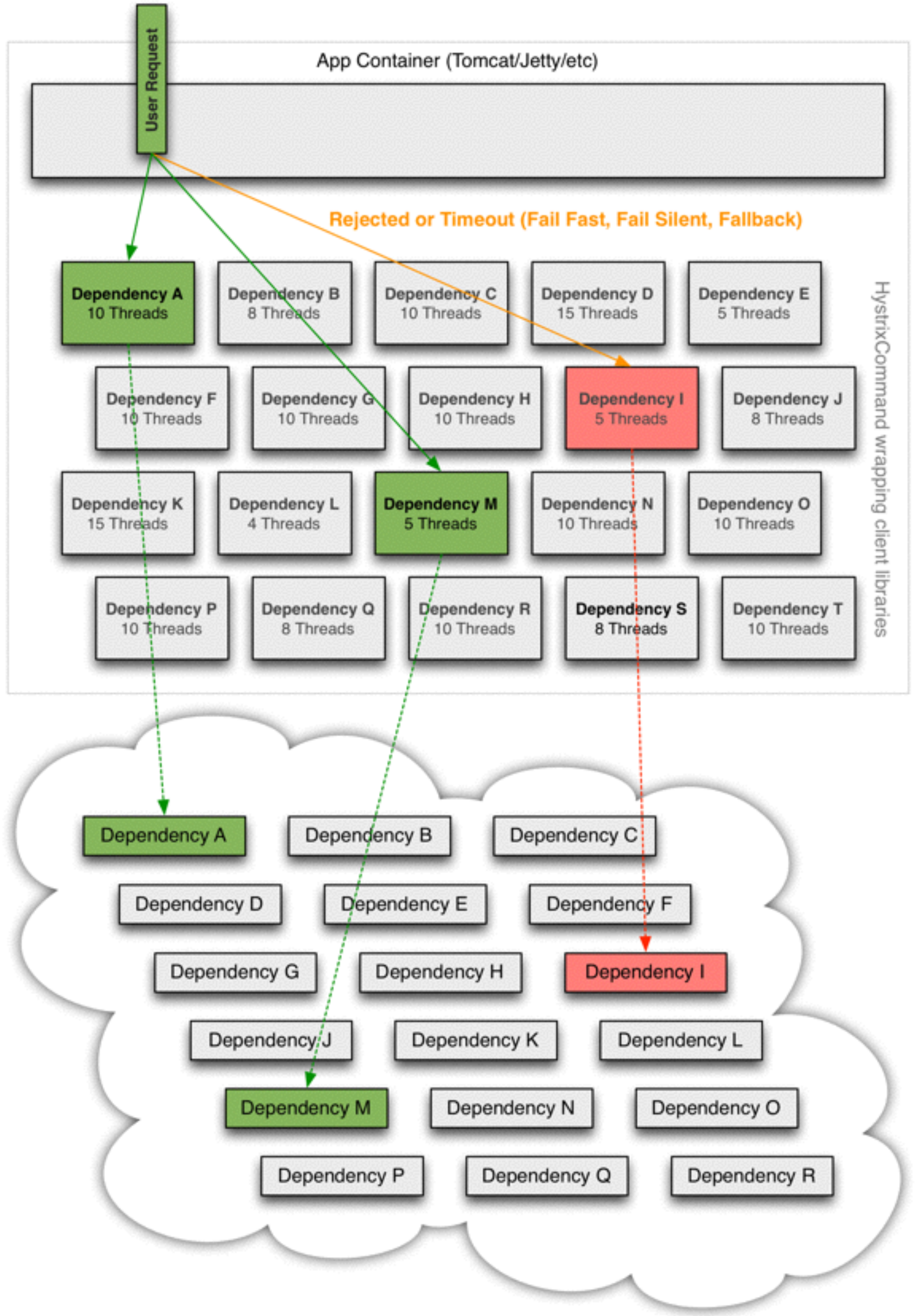


WHAT HYSTRIX SOLVE

For example, for an application that depends on 30 services where each service has 99.99% uptime, here is what you can expect:

> $99.99^{30} = 99.7\%$ uptime
>
> 0.3% of 1 billion requests = 3,000,000 failures
>
> 2+ hours downtime/month even if all dependencies have excellent uptime

WHAT HYSTRIX SOLVE



HOW TO USE

- EXTENDS HYSTRIXCOMMAND
- EXTENDS HYSTRIXOBSERVABLECOMMAND

```
<dependency>
     <groupId>com.netflix.hystrix</groupId>
     <artifactId>hystrix-core</artifactId>
     <version>${hystrix.version}</version>
 </dependency>
```

HOW TO USE

```
public class CommandHelloWorld extends HystrixCommand<String> {

 private final String name;

 public CommandHelloWorld(String name) {
     super(HystrixCommandGroupKey.Factory.asKey("ExampleGroup"));
     this.name = name;
 }

 @Override
 protected String run() {
     // a real example would do work like a network call here
     return "Hello " + name + "!";
 }
}
```

must not be null

HOW TO USE

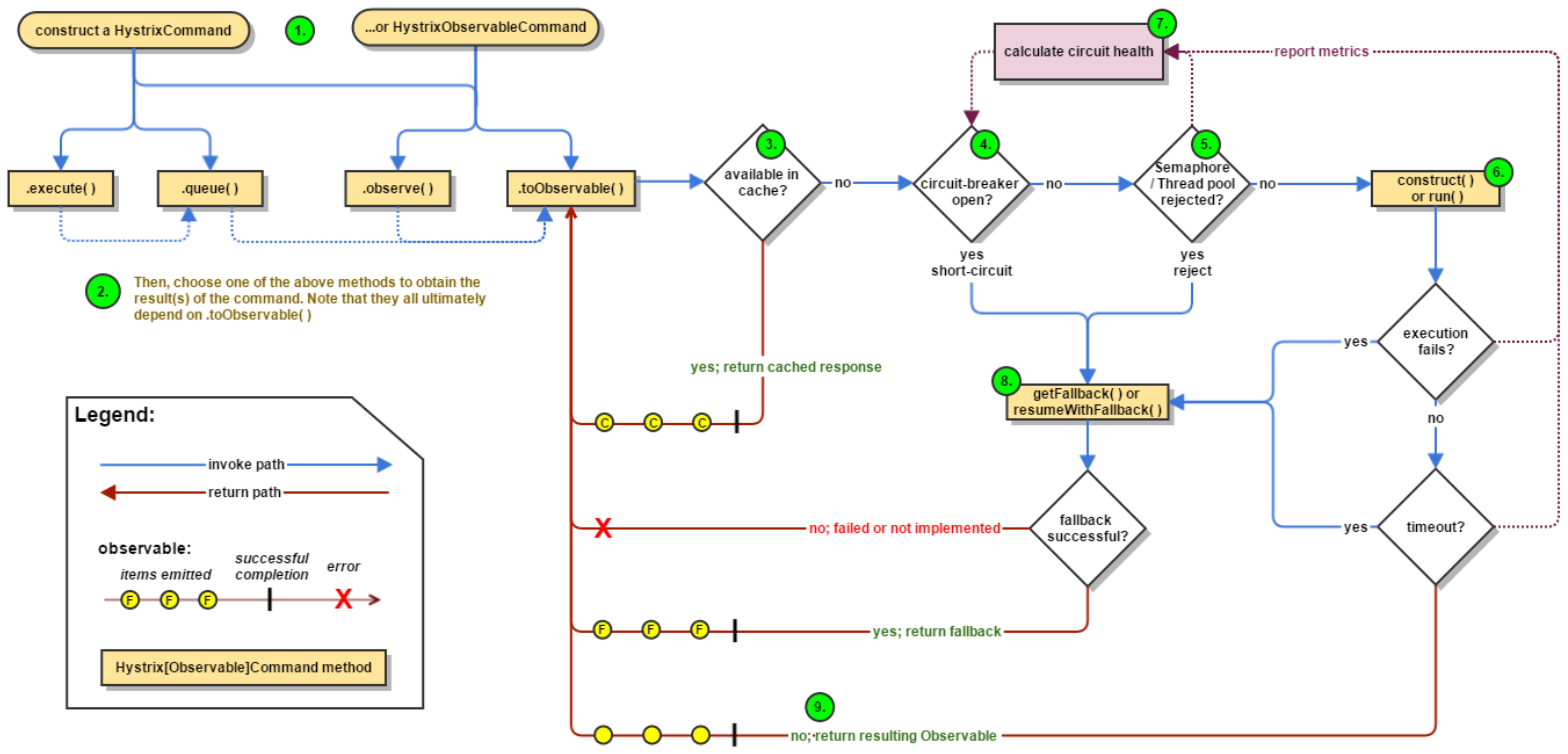
```
public class CommandHelloWorld extends HystrixObservableCommand<String> {

    private final String name;

    public CommandHelloWorld(String name) {
        super(HystrixCommandGroupKey.Factory.asKey("ExampleGroup"));
        this.name = name;
    }
    @Override
    protected Observable<String> construct() {
        return Observable.create(new Observable.OnSubscribe<String>() {
            @Override
            public void call(Subscriber<? super String> observer) {
            }
         } );
    }
}
```

How to use

Execute the command

```
K                     value   = command.execute();
Future<K>      fValue  = command.queue();
Observable<K> ohValue = command.observe();
Observable<K> ocValue = command.toObservable();
```

FLOW CHART


construct a HystrixCommand
1.
...or HystrixObservableCommand
.execute( )
.queue( )
.observe( )
.toObservable( )
2.
Then, choose one of the above methods to obtain the result(s) of the command. Note that they all ultimately depend on .toObservable( )
3.
available in cache?
no
4.
circuit-breaker open?
no
5.
Semaphore / Thread pool rejected?
no
6.
construct( ) or run( )
7.
calculate circuit health
report metrics
yes short-circuit
yes reject
yes; return cached response
execution fails?
yes
no
8.
getFallback( ) or resumeWithFallback( )
fallback successful?
no; failed or not implemented
timeout?
yes
yes; return fallback
9.
no; return resulting Observable
Legend:
invoke path
return path
observable:
items emitted
successful completion
error
Hystrix[Observable]Command method

CONFIG

- ARCHAIUS

  config.properties -Darchaius.configurationSource.additionalUrls

- CONSTRUCTOR

  ```
  Setter.withGroupKey().andCommandKey().andThreadPoolKey().andCommandPropert
  iesDefaults().andThreadPoolPropertiesDefaults()
  ```

CONFIG

```
# 线程池大小
hystrix.threadpool.default.coreSize=10
# 排队线程数量阈值，达到时拒绝
hystrix.threadpool.default.queueSizeRejectionThreshold=5
#最大排队长度。默认-1，使用SynchronousQueue。其他值则使用 LinkedBlockingQueue。
如果要从-1换成其他值则需重启
hystrix.threadpool.default.maxQueueSize=-1
#command线程执行超时时间，默认1s
hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds=1000
#当在配置时间窗口内达到此数量的失败后，进行短路。默认20个10秒钟内至少请求20次，熔断器才发挥起作用
hystrix.command.default.circuitBreaker.requestVolumeThreshold=20
#短路多久后开始尝试是否恢复，默认5s
hystrix.command.default.circuitBreaker.sleepWindowInMilliseconds=5000
#出错百分比阈值，当达到此阈值后，开始短路。默认50%
hystrix.command.default.circuitBreaker.errorThresholdPercentage=50
#是否开启HystrixRequestLog。默认开启
hystrix.command.default.requestLog.enabled=true
```

CONFIG

```
########## Fallback 配置
#调用线程允许请求HystrixCommand.GetFallback()的最大数量，默认10。超出时将会有异常抛出
hystrix.command.default.fallback.isolation.semaphore.maxConcurrentRequests=10
# isolation策略。默认是THREAD，线程模式
hystrix.command.default.execution.isolation.strategy=THREAD
# 是否启用fallback。默认开启
hystrix.command.default.fallback.enabled=true
# 是否允许断路。默认开启
hystrix.command.default.circuitBreaker.enabled=true
##########  使用Threadpool时的配置 ##########
#command的执行知否需要有超时时间，默认开启，开启后timeoutInMilliseconds有意义
hystrix.command.default.execution.timeout.enabled=true
#command如果超时是否可以被终止，默认可以
hystrix.command.default.execution.isolation.thread.interruptOnTimeout=true
```

CONFIG

```
########## 使用信号量时的配置 ##########
#允许的并发command数量，如果超出，新的command会被拒绝。默认10个
hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests=10
#a property to allow forcing the circuit open (stopping all requests)
#是否强制开启熔断器阻断所有请求,默认:false,不开启
hystrix.command.default.circuitBreaker.forceOpen=false
#是否允许熔断器忽略错误,默认false, 不开启
hystrix.command.default.circuitBreaker.forceClosed=false
#统计滚动的时间窗口10s 每个bucket统计1s内数据
hystrix.command.default.metrics.rollingStats.timeInMilliseconds=10000
#统计窗口的Buckets的数量,默认:10个,每秒一个Buckets统计
hystrix.command.default.metrics.rollingStats.numBuckets=10
hystrix.command.default.metrics.rollingPercentile.enabled=true
hystrix.command.default.metrics.rollingPercentile.timeInMilliseconds=true
```

CONFIG

```
# default to 6 buckets (10 seconds each in 60 second window)
hystrix.command.default.metrics.rollingPercentile.numBuckets=6

# default to 100 values max per bucket
hystrix.command.default.metrics.rollingPercentile.bucketSize=100

# 计算成功或失败率的频率的快照 default to 500ms ，时间不易设置太短
hystrix.command.default.metrics.healthSnapshot.intervalInMilliseconds=500

hystrix.command.default.requestCache.enabled=true
```

CONFIG

- HystrixXXXProperties

  HystrixCommandProperties

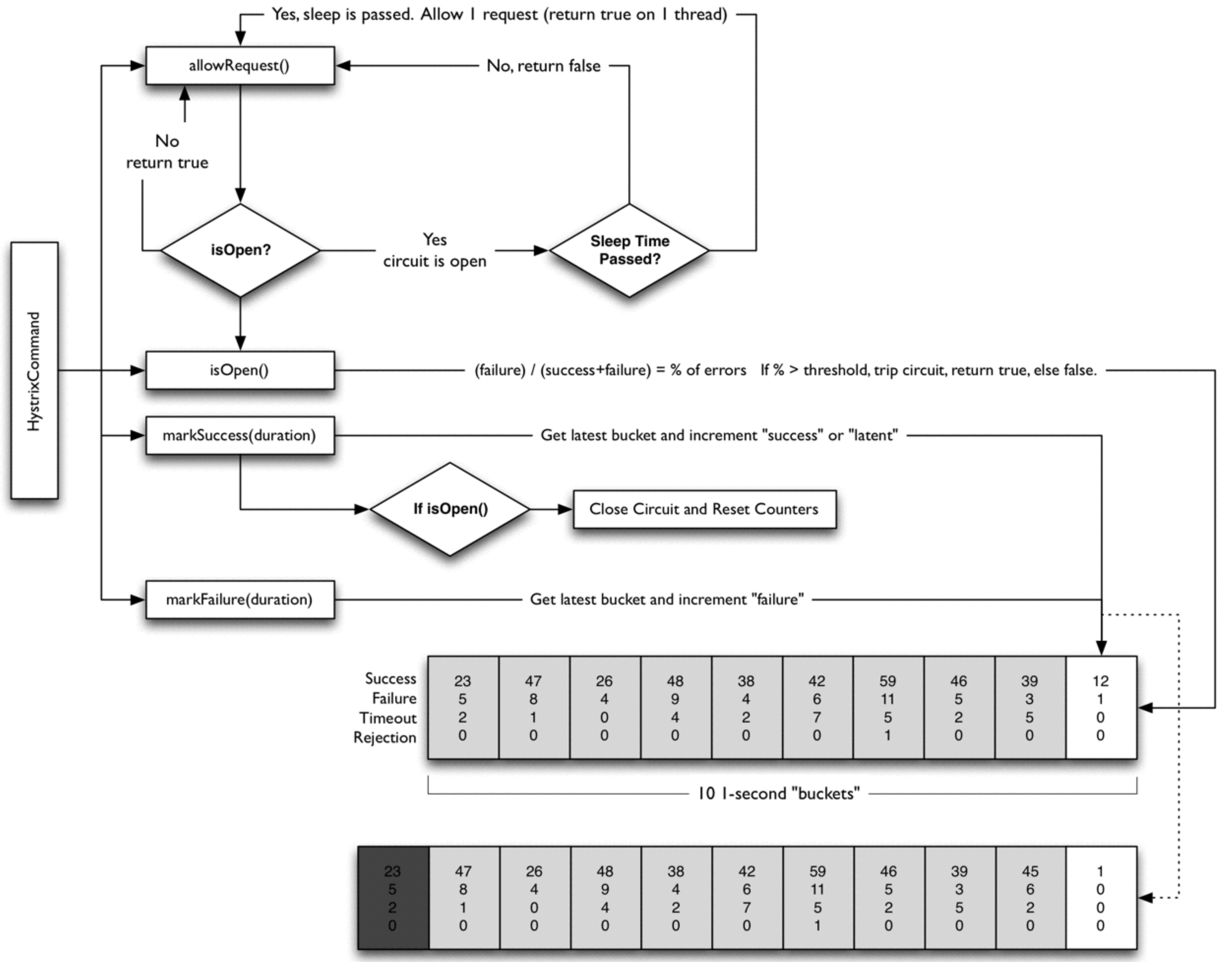
  HystrixThreadPoolProperties

  HystrixCollapserProperties

CACHE

- Default enable
- Override getCacheKey method

CIRCUIT BREAKER



Yes, sleep is passed. Allow 1 request (return true on 1 thread)
allowRequest()
No, return false
No
return true
isOpen?
Yes
circuit is open
Sleep Time
Passed?
HystrixCommand
isOpen()
(failure) / (success+failure) = % of errors If % > threshold, trip circuit, return true, else false.
markSuccess(duration)
Get latest bucket and increment "success" or "latent"
If isOpen()
Close Circuit and Reset Counters
markFailure(duration)
Get latest bucket and increment "failure"
10 1-second "buckets"

Success	23	47	26	48	38	42	59	46	39	12
Failure	5	8	4	9	4	6	11	5	3	1
Timeout	2	1	0	4	2	7	5	2	5	0
Rejection	0	0	0	0	0	0	1	0	0	0

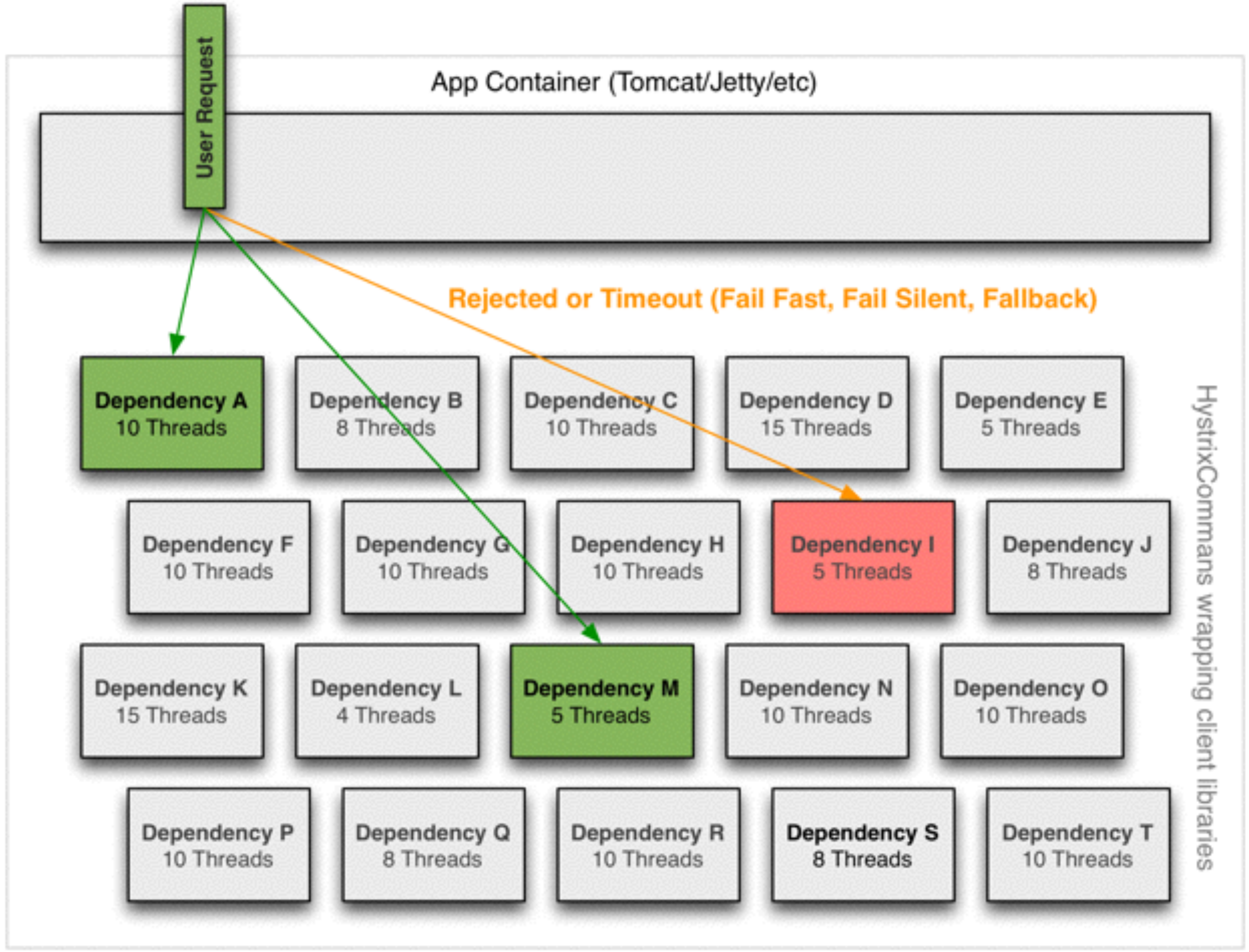
23	47	26	48	38	42	59	46	39	45	1
5	8	4	9	4	6	11	5	3	6	0
2	1	0	4	2	7	5	2	5	2	0
0	0	0	0	0	0	1	0	0	0	0

On "getLatestBucket" if the 1-second window is passed a new bucket is created, the rest slid over and the oldest one dropped.

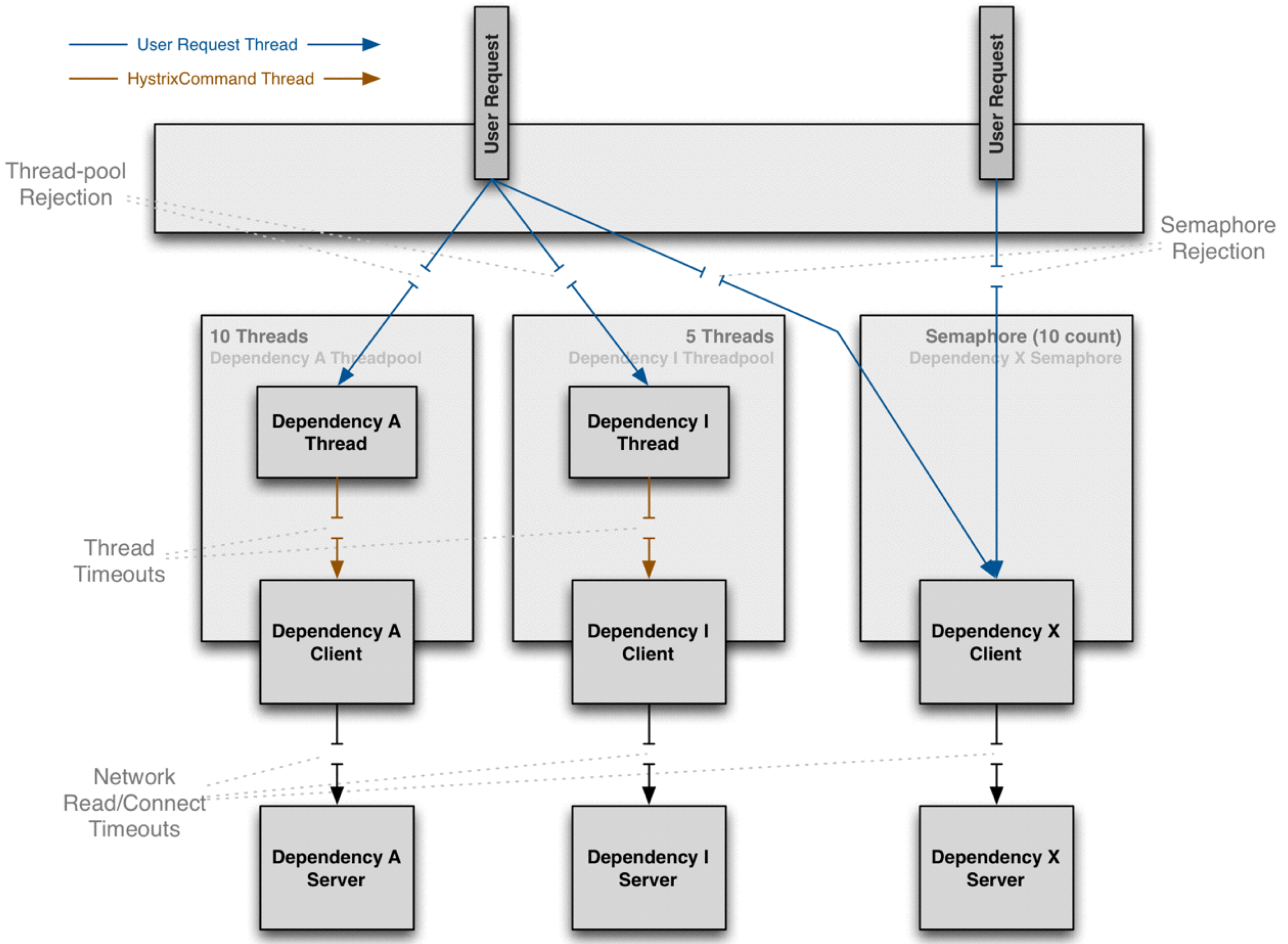
Isolation

- Thread
- Semaphores

ISOLATION

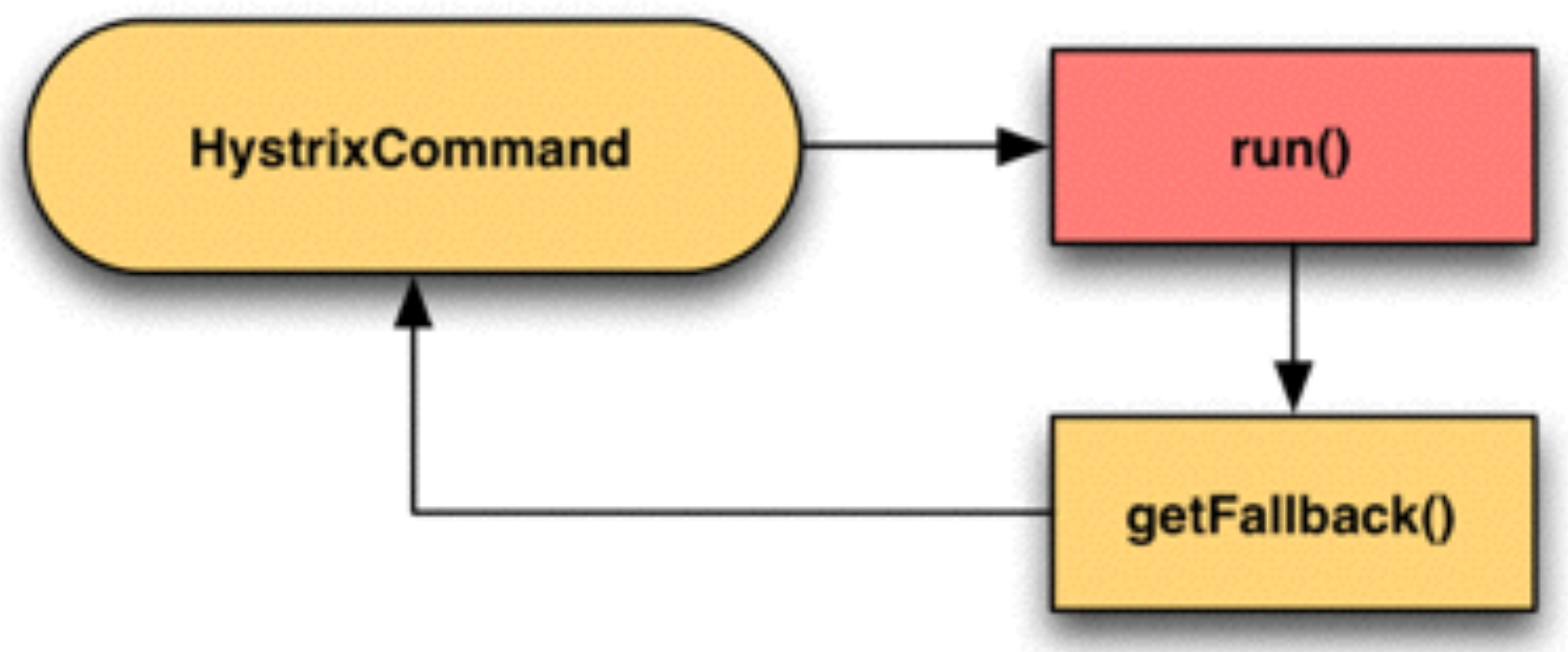

User Request
App Container (Tomcat/Jetty/etc)
Rejected or Timeout (Fail Fast, Fail Silent, Fallback)
Dependency A 10 Threads
Dependency B 8 Threads
Dependency C 10 Threads
Dependency D 15 Threads
Dependency E 5 Threads
Dependency F 10 Threads
Dependency G 10 Threads
Dependency H 10 Threads
Dependency I 5 Threads
Dependency J 8 Threads
Dependency K 15 Threads
Dependency L 4 Threads
Dependency M 5 Threads
Dependency N 10 Threads
Dependency O 10 Threads
Dependency P 10 Threads
Dependency Q 8 Threads
Dependency R 10 Threads
Dependency S 8 Threads
Dependency T 10 Threads
HystrixCommans wrapping client libraries

ISOLATION

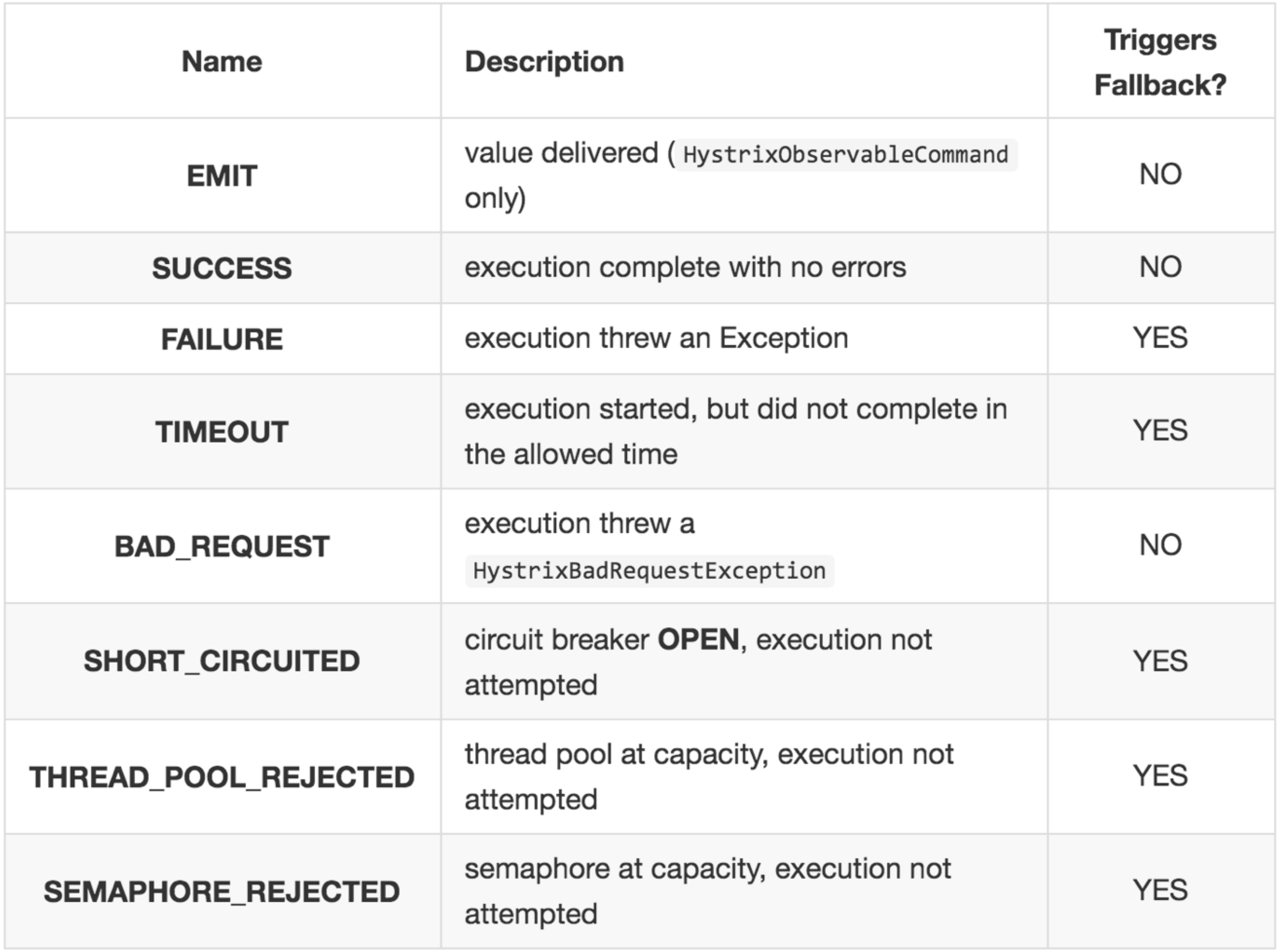

User Request Thread
HystrixCommand Thread
User Request
User Request
Thread-pool Rejection
Semaphore Rejection
10 Threads
Dependency A Threadpool
5 Threads
Dependency I Threadpool
Semaphore (10 count)
Dependency X Semaphore
Dependency A Thread
Dependency I Thread
Thread Timeouts
Dependency A Client
Dependency I Client
Dependency X Client
Network Read/Connect Timeouts
Dependency A Server
Dependency I Server
Dependency X Server

FALLBACK

- HystrixCommand/getFallback
- HystrixObservableCommand/resumeWithFallback



FALLBACK



Name	Description	Triggers Fallback?
EMIT	value delivered (`HystrixObservableCommand` only)	NO
SUCCESS	execution complete with no errors	NO
FAILURE	execution threw an Exception	YES
TIMEOUT	execution started, but did not complete in the allowed time	YES
BAD_REQUEST	execution threw a `HystrixBadRequestException`	NO
SHORT_CIRCUITED	circuit breaker **OPEN**, execution not attempted	YES
THREAD_POOL_REJECTED	thread pool at capacity, execution not attempted	YES
SEMAPHORE_REJECTED	semaphore at capacity, execution not attempted	YES

Dashboard

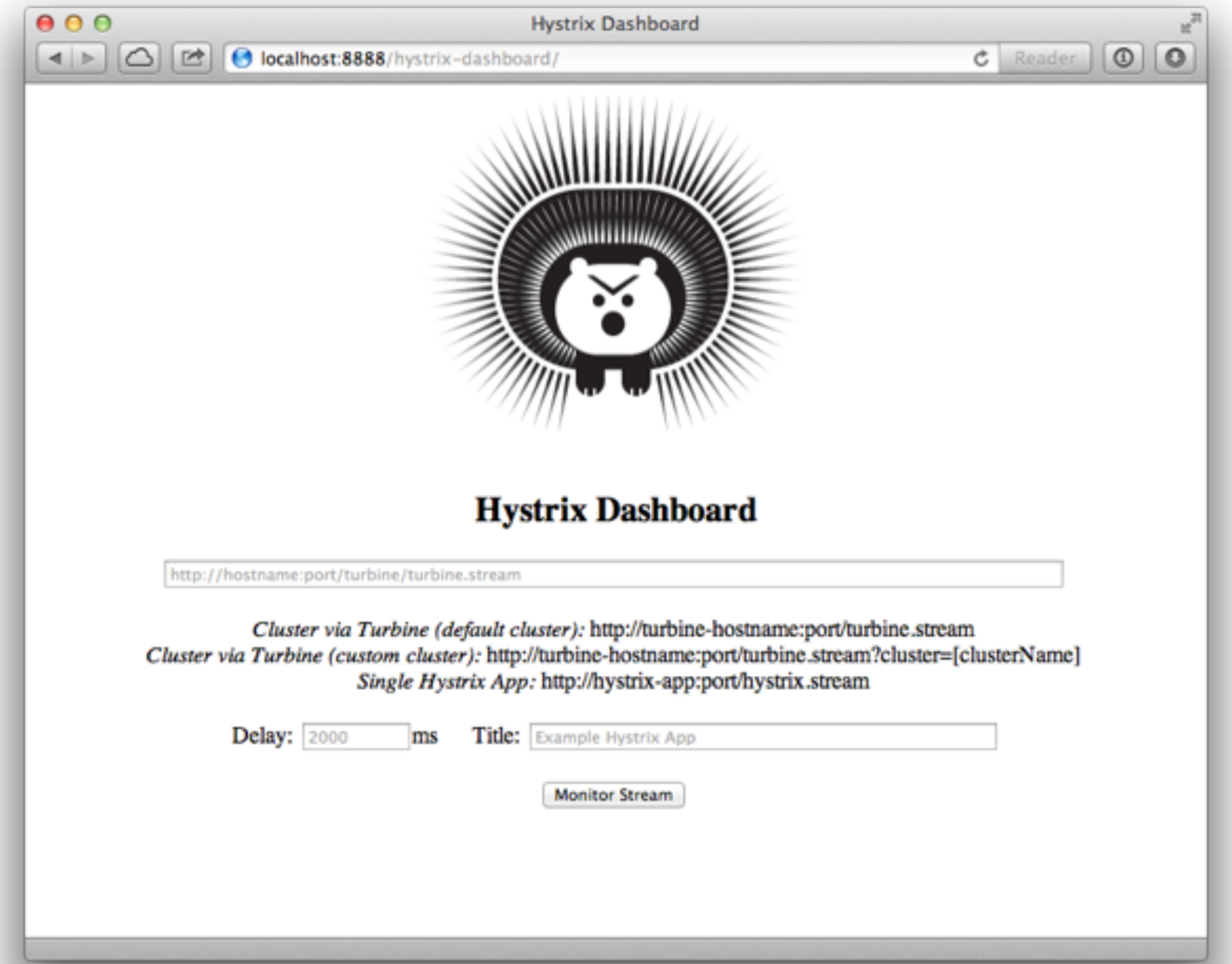
```xml
<dependency>
     <groupId>com.netflix.hystrix</groupId>
     <artifactId>hystrix-metrics-event-stream</artifactId>
     <version>${hystrix.version}</version>
 </dependency>
```

```java
@Configuration
public class HystrixConfig {
    @Bean
    public HystrixMetricsStreamServlet getHystrixMetricsStreamServlet() {
        return new HystrixMetricsStreamServlet();
    }
    @Bean
    public ServletRegistrationBean registration(HystrixMetricsStreamServlet filter) {
        ServletRegistrationBean registration = new ServletRegistrationBean(filter, new String[0]);
        registration.setEnabled(true);
        registration.addUrlMappings(new String[] { "/hystrix.stream" });
        return registration;
    }
}
```

DASHBOARD

Download hystrix-dashboard-#.#.#.war

Install it in a servlet container such as Apache Tomcat 7



Hystrix Dashboard

localhost:8888/hystrix-dashboard/

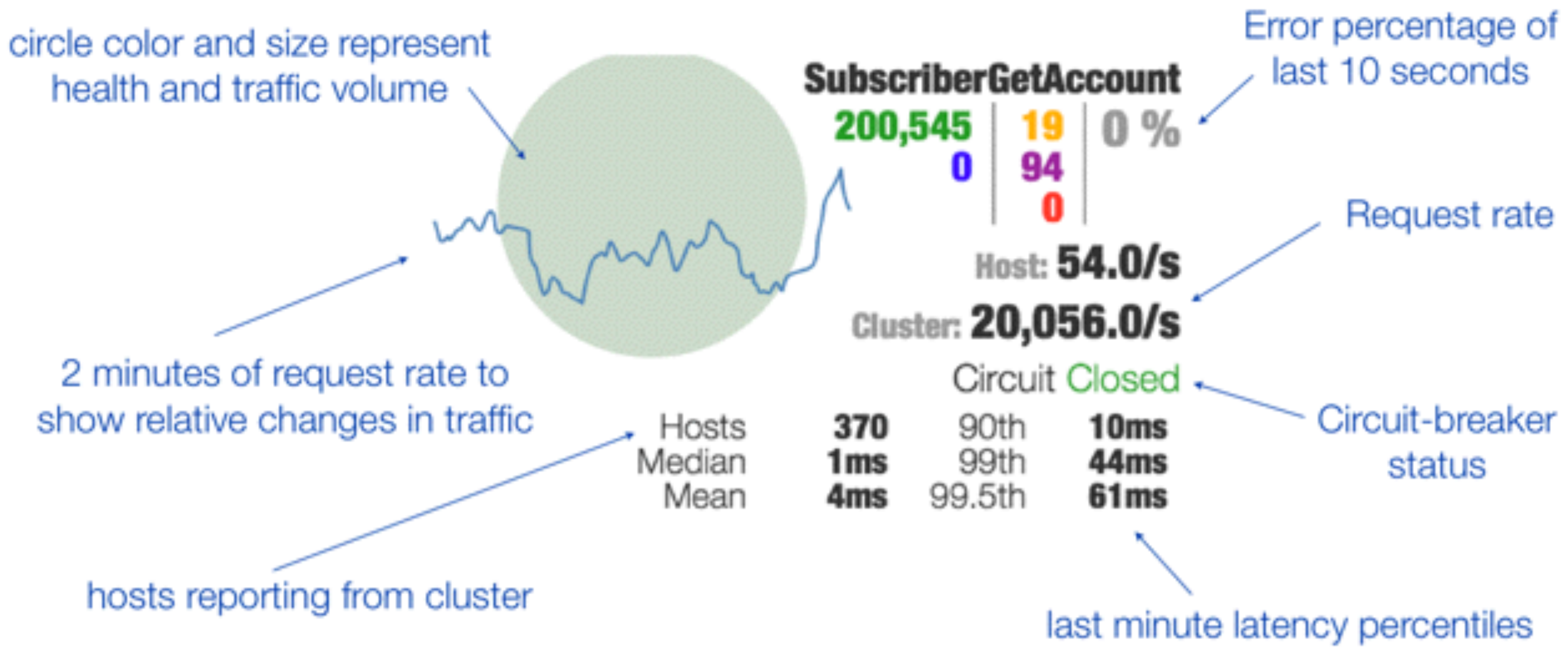
Hystrix Dashboard

http://hostname:port/turbine/turbine.stream

Cluster via Turbine (default cluster): http://turbine-hostname:port/turbine.stream
Cluster via Turbine (custom cluster): http://turbine-hostname:port/turbine.stream?cluster=[clusterName]
Single Hystrix App: http://hystrix-app:port/hystrix.stream

Delay: 2000 ms Title: Example Hystrix App

DASHBOARD


circle color and size represent health and traffic volume
SubscriberGetAccount
200,545
19
0 %
0
94
0
Error percentage of last 10 seconds
Request rate
Host: 54.0/s
Cluster: 20,056.0/s
2 minutes of request rate to show relative changes in traffic
Circuit Closed
Circuit-breaker status
Hosts 370
Median 1ms
Mean 4ms
90th 10ms
99th 44ms
99.5th 61ms
hosts reporting from cluster
last minute latency percentiles

Rolling 10 second counters with 1 second granularity

Successes	200,545	19	Thread timeouts
Short-circuited (rejected)	0	94	Thread-pool Rejections
		0	Failures/Exceptions

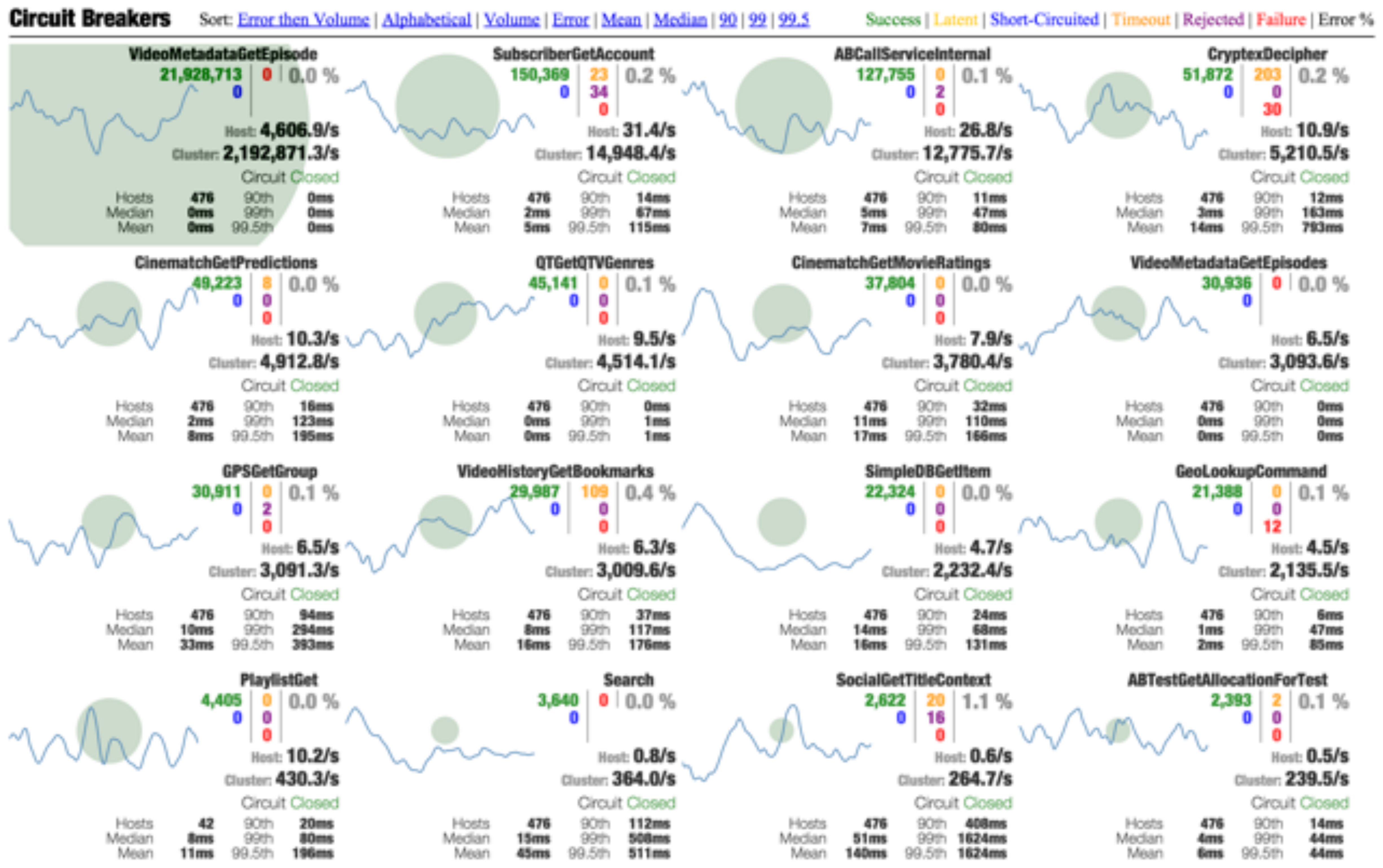
DASHBOARD

Download turbine-web-1.0.0.war
Install in servlet container such as Apache Tomcat 7

```
turbine-web-1.0.0
├── META-INF
└── WEB-INF
    ├── classes
    |   ├── StartTurbineServer.class
    |   ├── config.properties
    |   └── log4j.properties
    ├── lib
    └── web.xml
```

DASHBOARD

```
turbine.aggregator.clusterConfig=configcenter
turbine.ConfigPropertyBasedDiscovery.configcenter.instances=10.0.80.60,10.0.41.13
turbine.instanceUrlSuffix.configcenter=:8080/configcenter-web/hystrix.stream
```



Circuit Breakers Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5 Success | Latent | Short-Circuited | Timeout | Rejected | Failure | Error %

```
VideoMetadataGetEpisode
21,928,713 | 0 | 0.0 %
0
Host: 4,606.9/s
Cluster: 2,192,871.3/s
Circuit Closed
Hosts   476  90th   0ms
Median  0ms  99th   0ms
Mean    0ms  99.5th 0ms

SubscriberGetAccount
150,369 | 23 | 0.2 %
0 | 34
0
Host: 31.4/s
Cluster: 14,948.4/s
Circuit Closed
Hosts   476  90th   14ms
Median  2ms  99th   67ms
Mean    5ms  99.5th 115ms

ABCallServiceInternal
127,755 | 0 | 0.1 %
0 | 2
0
Host: 26.8/s
Cluster: 12,775.7/s
Circuit Closed
Hosts   476  90th   11ms
Median  5ms  99th   47ms
Mean    7ms  99.5th 80ms

CryptexDecipher
51,872 | 203 | 0.2 %
0 | 0
30
Host: 10.9/s
Cluster: 5,210.5/s
Circuit Closed
Hosts   476   90th   12ms
Median  3ms   99th   163ms
Mean    14ms  99.5th 793ms

CinematchGetPredictions
49,223 | 8 | 0.0 %
0 | 0
0
Host: 10.3/s
Cluster: 4,912.8/s
Circuit Closed
Hosts   476  90th   16ms
Median  2ms  99th   123ms
Mean    8ms  99.5th 195ms

QTGetQTVGenres
45,141 | 0 | 0.1 %
0 | 0
0
Host: 9.5/s
Cluster: 4,514.1/s
Circuit Closed
Hosts   476  90th   0ms
Median  0ms  99th   1ms
Mean    0ms  99.5th 1ms

CinematchGetMovieRatings
37,804 | 0 | 0.0 %
0 | 0
0
Host: 7.9/s
Cluster: 3,780.4/s
Circuit Closed
Hosts   476   90th   32ms
Median  11ms  99th   110ms
Mean    17ms  99.5th 166ms

VideoMetadataGetEpisodes
30,936 | 0 | 0.0 %
0
Host: 6.5/s
Cluster: 3,093.6/s
Circuit Closed
Hosts   476  90th   0ms
Median  0ms  99th   0ms
Mean    0ms  99.5th 0ms

GPSGetGroup
30,911 | 0 | 0.1 %
0 | 2
0
Host: 6.5/s
Cluster: 3,091.3/s
Circuit Closed
Hosts   476   90th   94ms
Median  10ms  99th   294ms
Mean    33ms  99.5th 393ms

VideoHistoryGetBookmarks
29,987 | 109 | 0.4 %
0 | 0
0
Host: 6.3/s
Cluster: 3,009.6/s
Circuit Closed
Hosts   476   90th   37ms
Median  8ms   99th   117ms
Mean    16ms  99.5th 176ms

SimpleDBGetItem
22,324 | 0 | 0.0 %
0 | 0
0
Host: 4.7/s
Cluster: 2,232.4/s
Circuit Closed
Hosts   476   90th   24ms
Median  14ms  99th   68ms
Mean    16ms  99.5th 131ms

GeoLookupCommand
21,388 | 0 | 0.1 %
0 | 0
12
Host: 4.5/s
Cluster: 2,135.5/s
Circuit Closed
Hosts   476  90th   6ms
Median  1ms  99th   47ms
Mean    2ms  99.5th 85ms

PlaylistGet
4,405 | 0 | 0.0 %
0 | 0
0
Host: 10.2/s
Cluster: 430.3/s
Circuit Closed
Hosts   42    90th   20ms
Median  8ms   99th   80ms
Mean    11ms  99.5th 196ms

Search
3,640 | 0 | 0.0 %
0
Host: 0.8/s
Cluster: 364.0/s
Circuit Closed
Hosts   476   90th   112ms
Median  15ms  99th   508ms
Mean    45ms  99.5th 511ms

SocialGetTitleContext
2,622 | 20 | 1.1 %
0 | 16
0
Host: 0.6/s
Cluster: 264.7/s
Circuit Closed
Hosts   476    90th   408ms
Median  51ms   99th   1624ms
Mean    140ms  99.5th 1624ms

ABTestGetAllocationForTest
2,393 | 2 | 0.1 %
0 | 0
0
Host: 0.5/s
Cluster: 239.5/s
Circuit Closed
Hosts   476  90th   14ms
Median  4ms  99th   44ms
Mean    6ms  99.5th 44ms
```

SUMMARY

- CONTROL LATENCY AND FAILURE FROM DEPENDENCIES
- STOP CASCADING FAILURES
- FAIL FAST AND RAPIDLY RECOVER
- FALLBACK AND GRACEFULLY DEGRADE WHEN POSSIBLE
- ENABLE NEAR REAL-TIME MONITORING

THANKS