



# STRANGER THINGS

## The Forces That Disrupt Netflix

our world

ACROBAT



FLEA



parallel world

# A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

--Leslie Lamport

our world

ENGINEER

~~ACROSS~~



FLEA



computing

~~parallel world~~



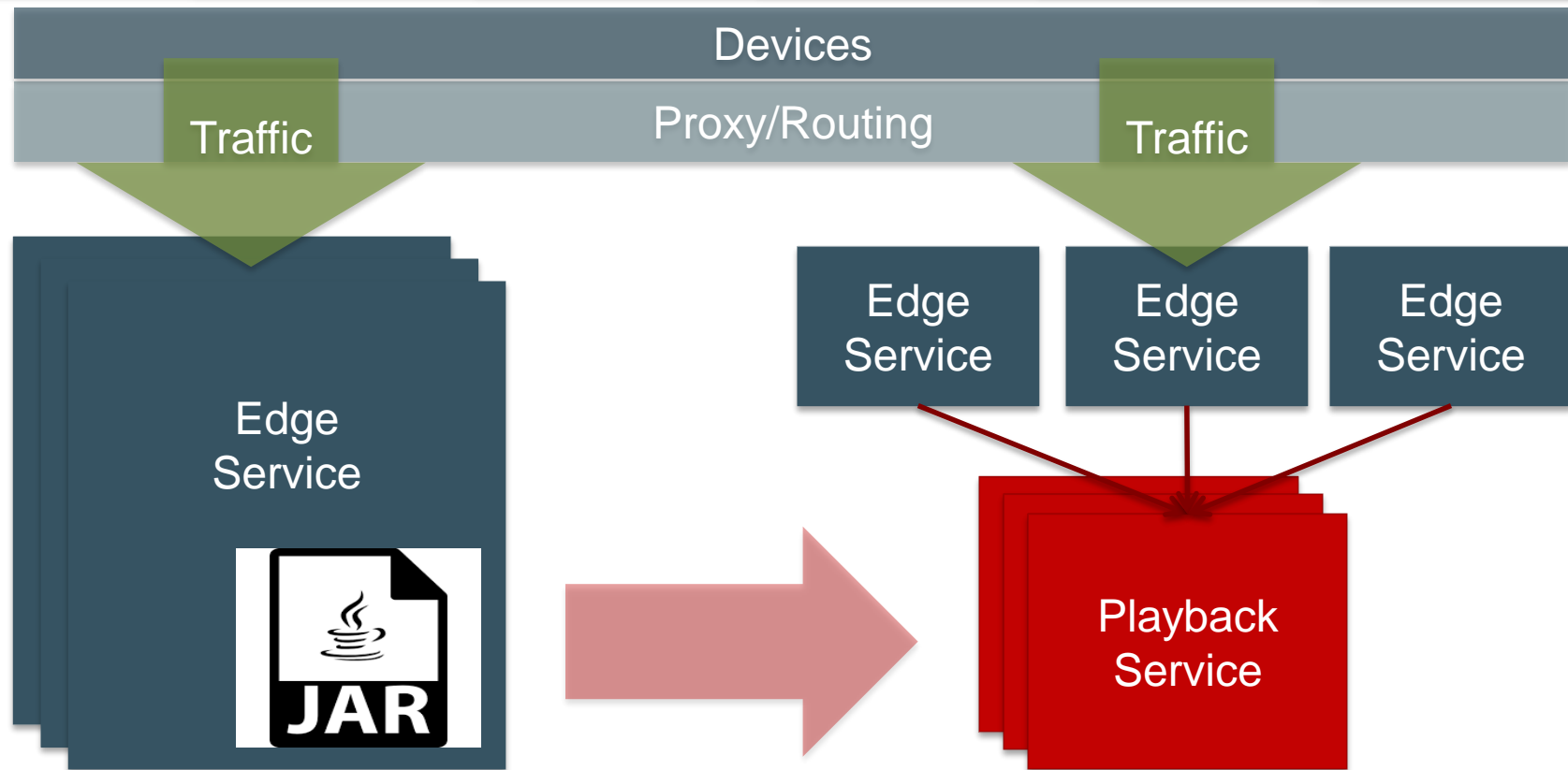
PROLOGUE

---

DISTRIBUTED SYSTEMS

# NETFLIX

# DECOMPOSING THE MONOLITH





**# Distributed  
systems are  
different because  
they fail often.**

--Jeff Hodges

# TABLE OF CONTENTS

---

## FORCES AT WORK

### **CHAPTER 1: THE WEIRD DATA IN THE CATALOG**

- Metadata impacts on availability

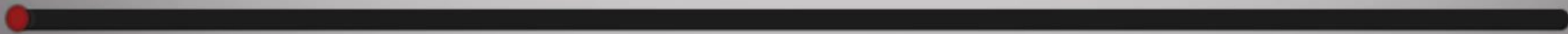
### **CHAPTER 2: THE VANISHING OF CRITICAL SERVICES**

- Crashing services and cascading failures

### **CHAPTER 3: THE THROTTLE**

- Latency spikes and the impact of fallbacks

# NETFLIX

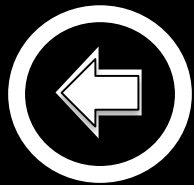


48:21



Stranger Things Season 1: Ep. 1 Chapter One: The Vanishing Of Will Byers





Whoops, something went wrong...

**Netflix Streaming Error**

We're having trouble playing this title right now. Please try again later or select a different title.

# CHAPTER ONE

---

## THE WEIRD DATA IN THE CATALOG



**holly trabel**  
@hollyberryfleur



 Follow

So it was wide spread [#netflixdown](#)

**Janet Montgomery** @jayrmony

Netflix just went down when I have one more episode of Stranger Things. I'm filled with real sadness....

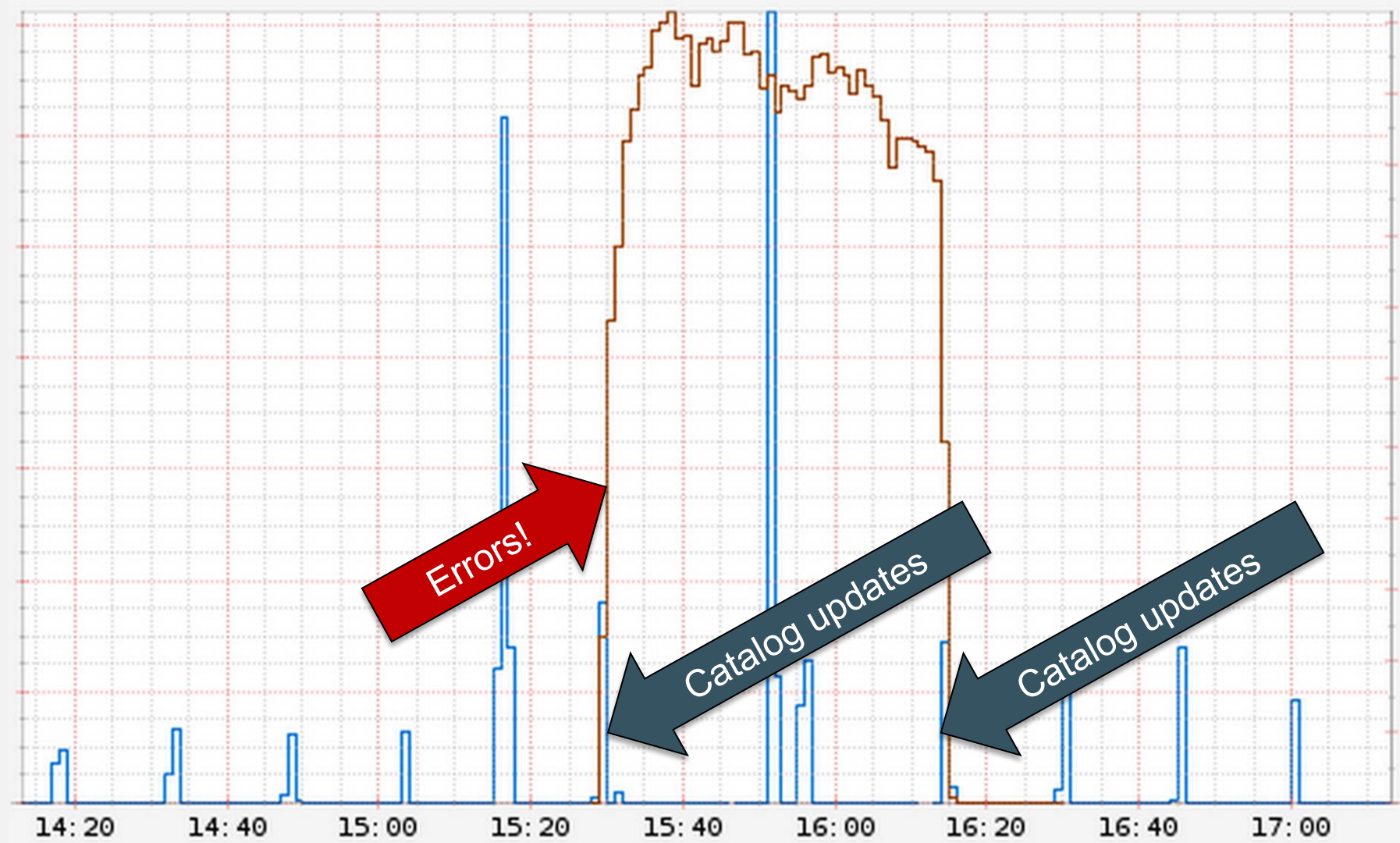
2:50 PM

 Indiana, USA



45 MINUTES!!

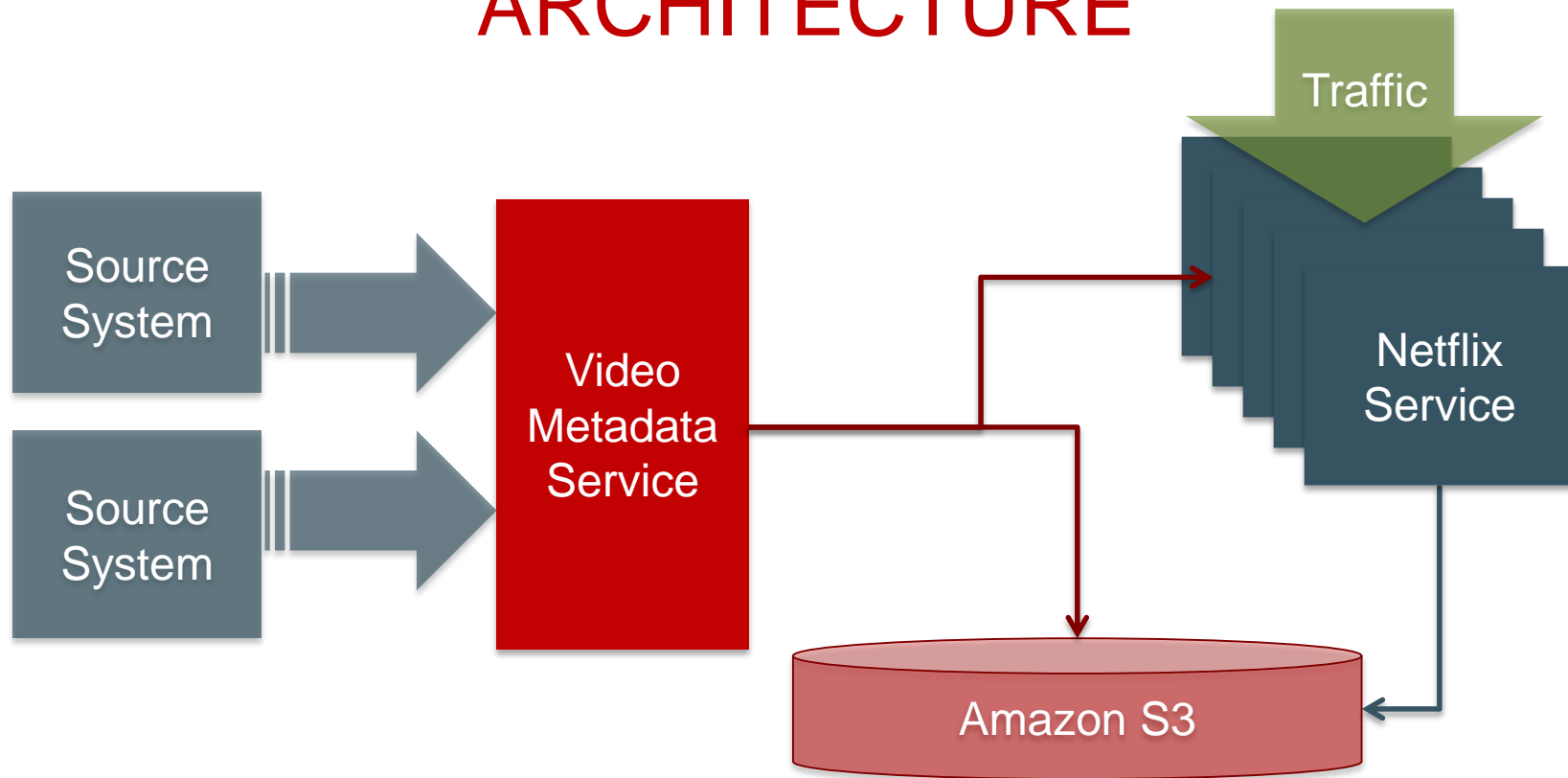






# VIDEO METADATA ARCHITECTURE

---



Netflix  
Playback  
Service



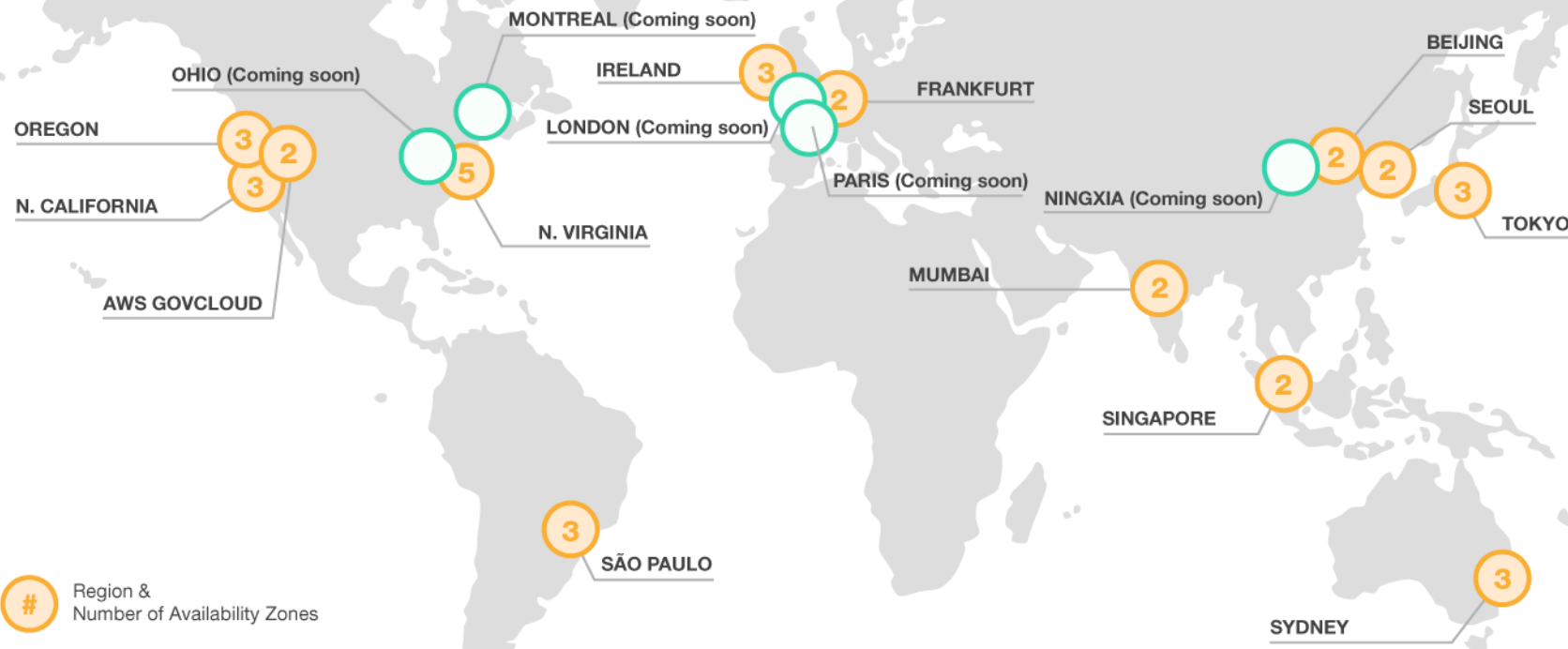
Amazon S3

```
{  
    String msg = "This should never  
happen!";  
    throw new IllegalStateException(msg);  
}
```

# MITIGATION BLAST RADIUS

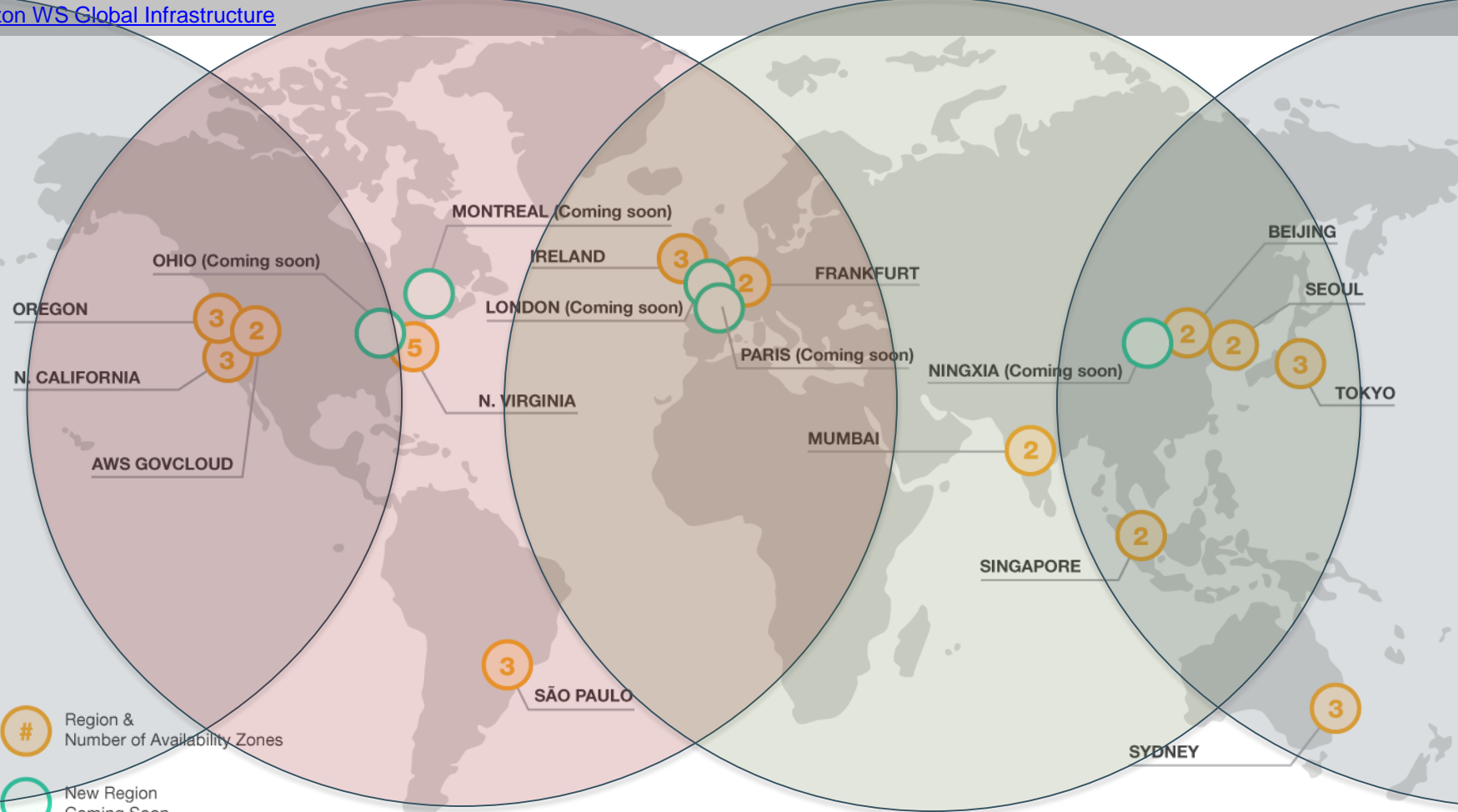
---



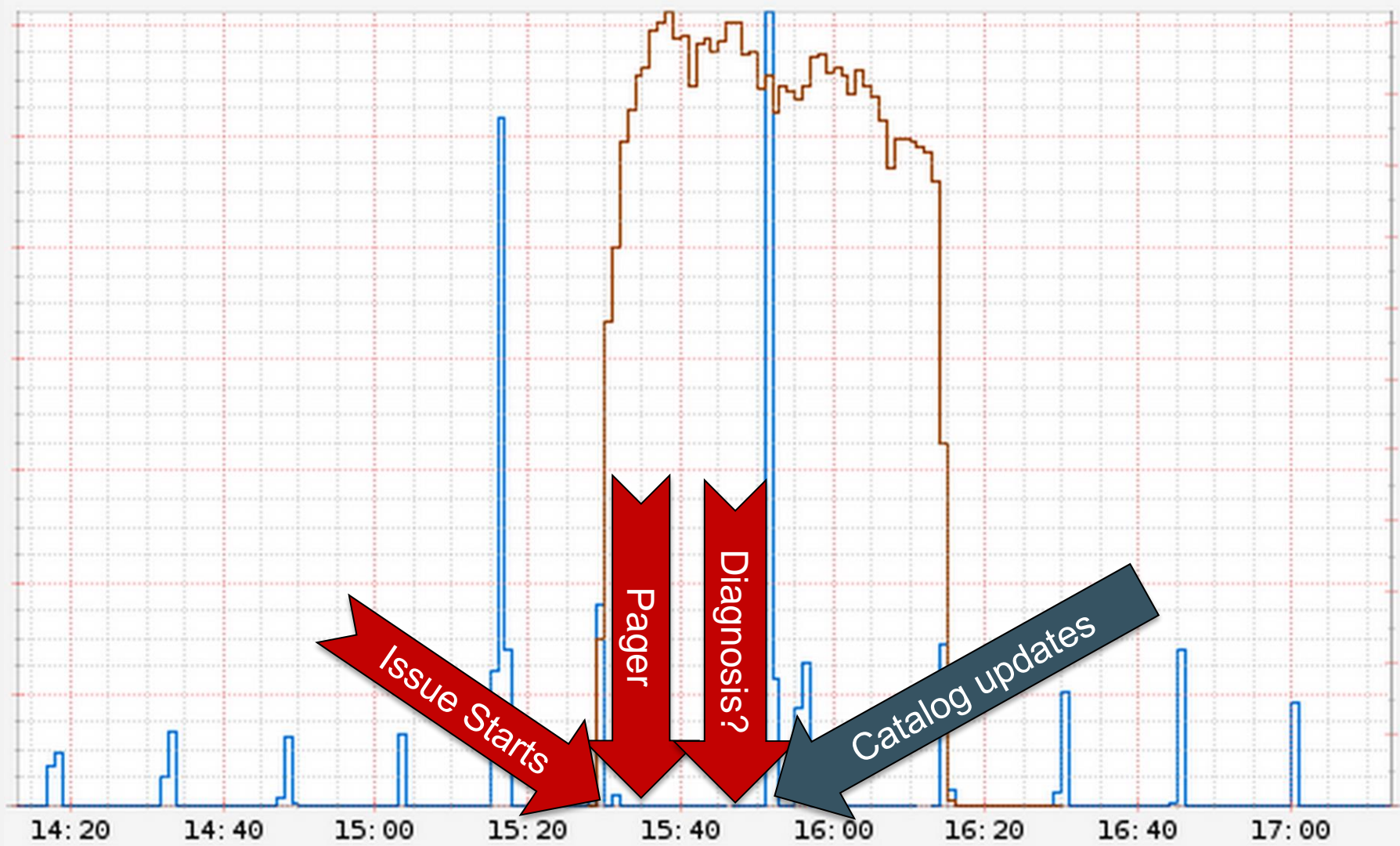


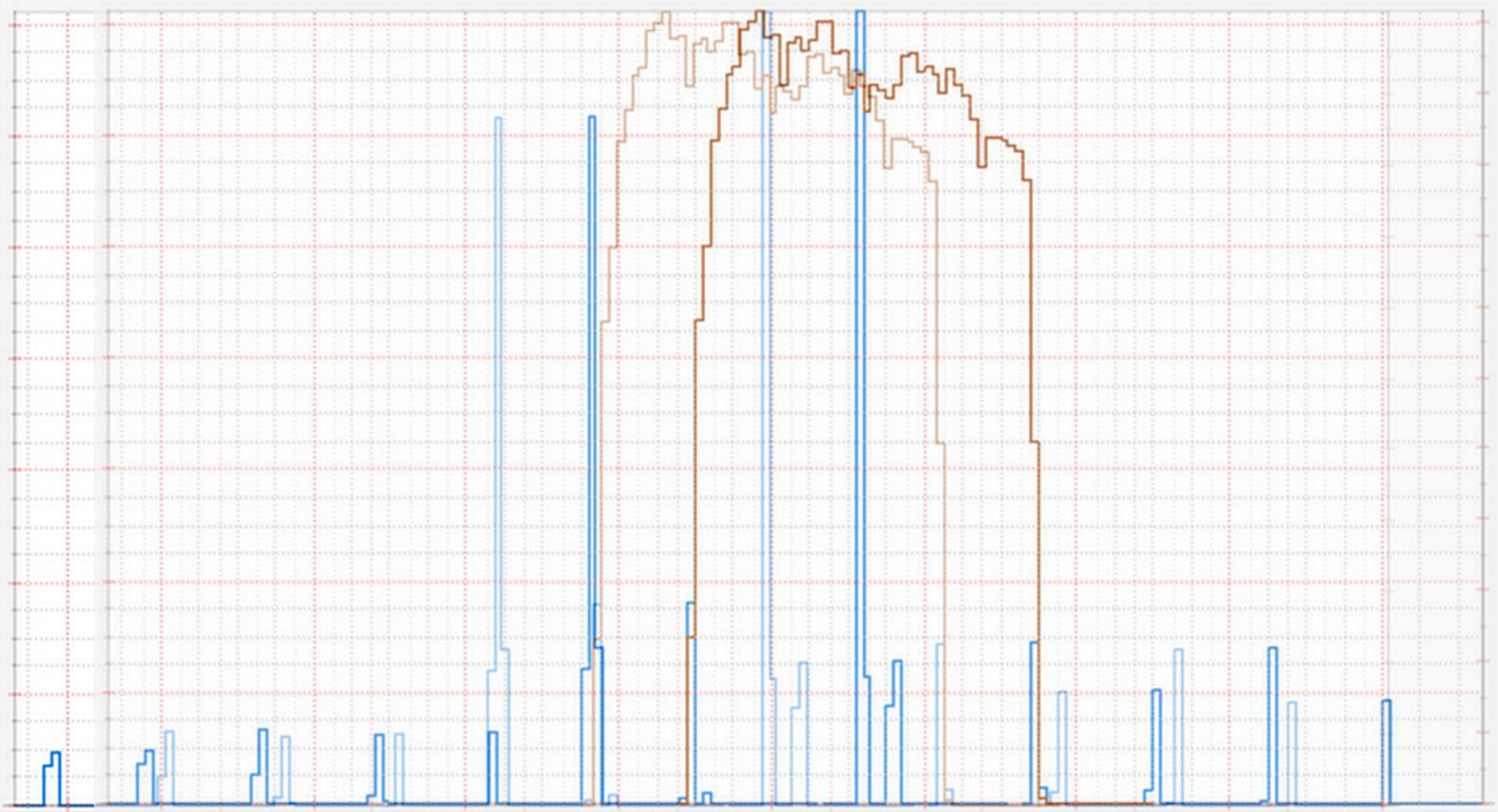
# Region & Number of Availability Zones

○ New Region Coming Soon



# STAGGERED ROLLOUT







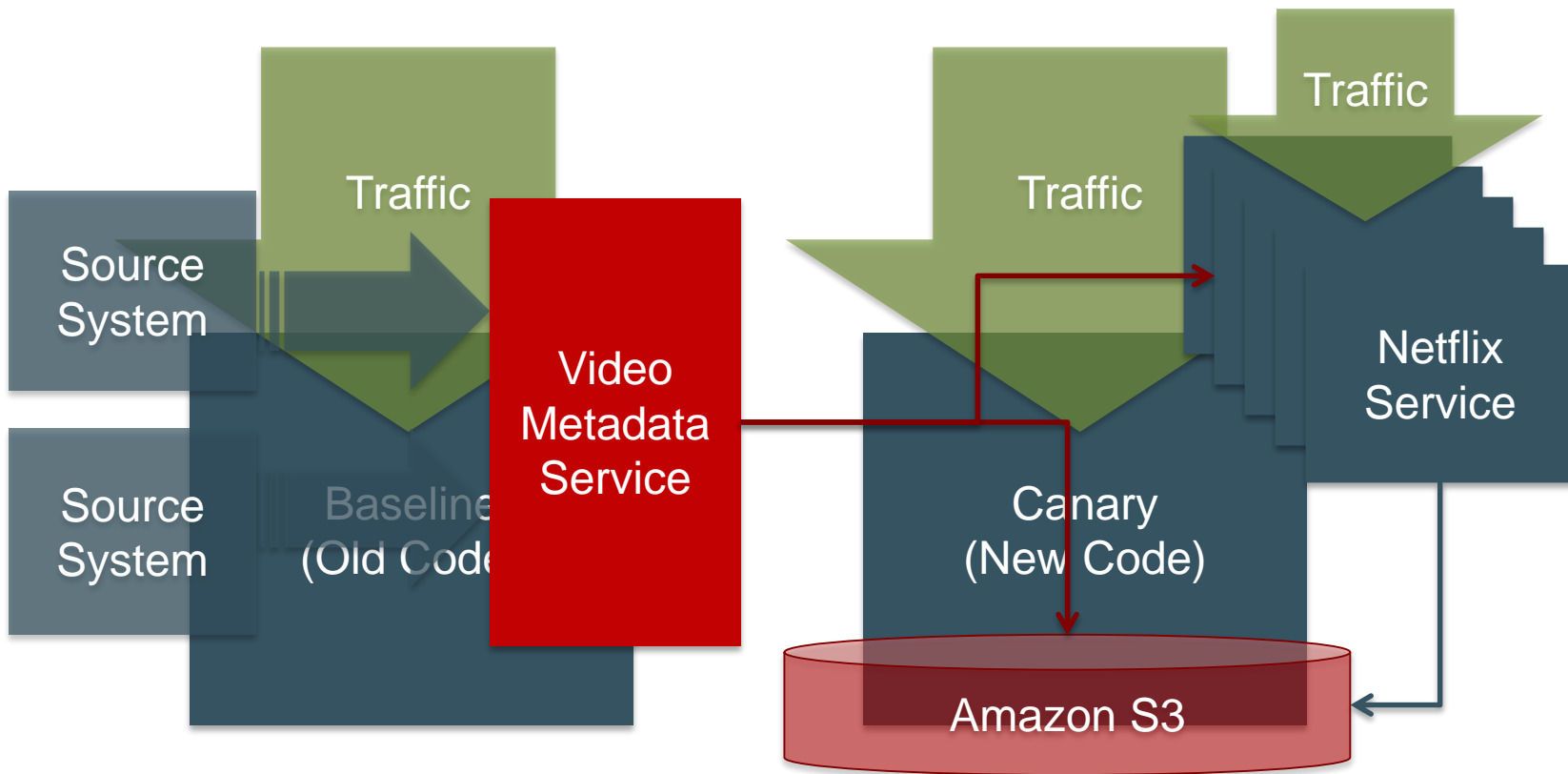
# PREVENTION CANARIES

---



# TRADITIONAL CANARY

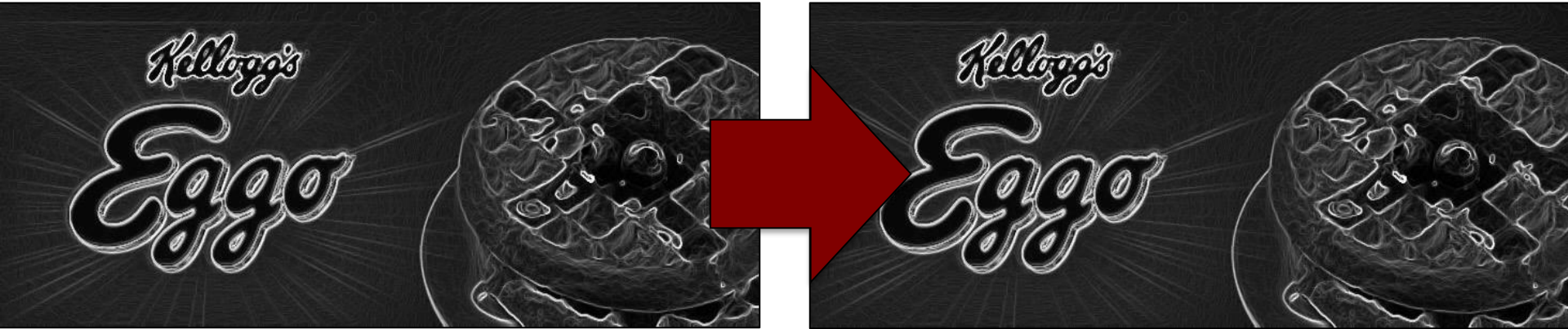
---



# CONSISTENCY

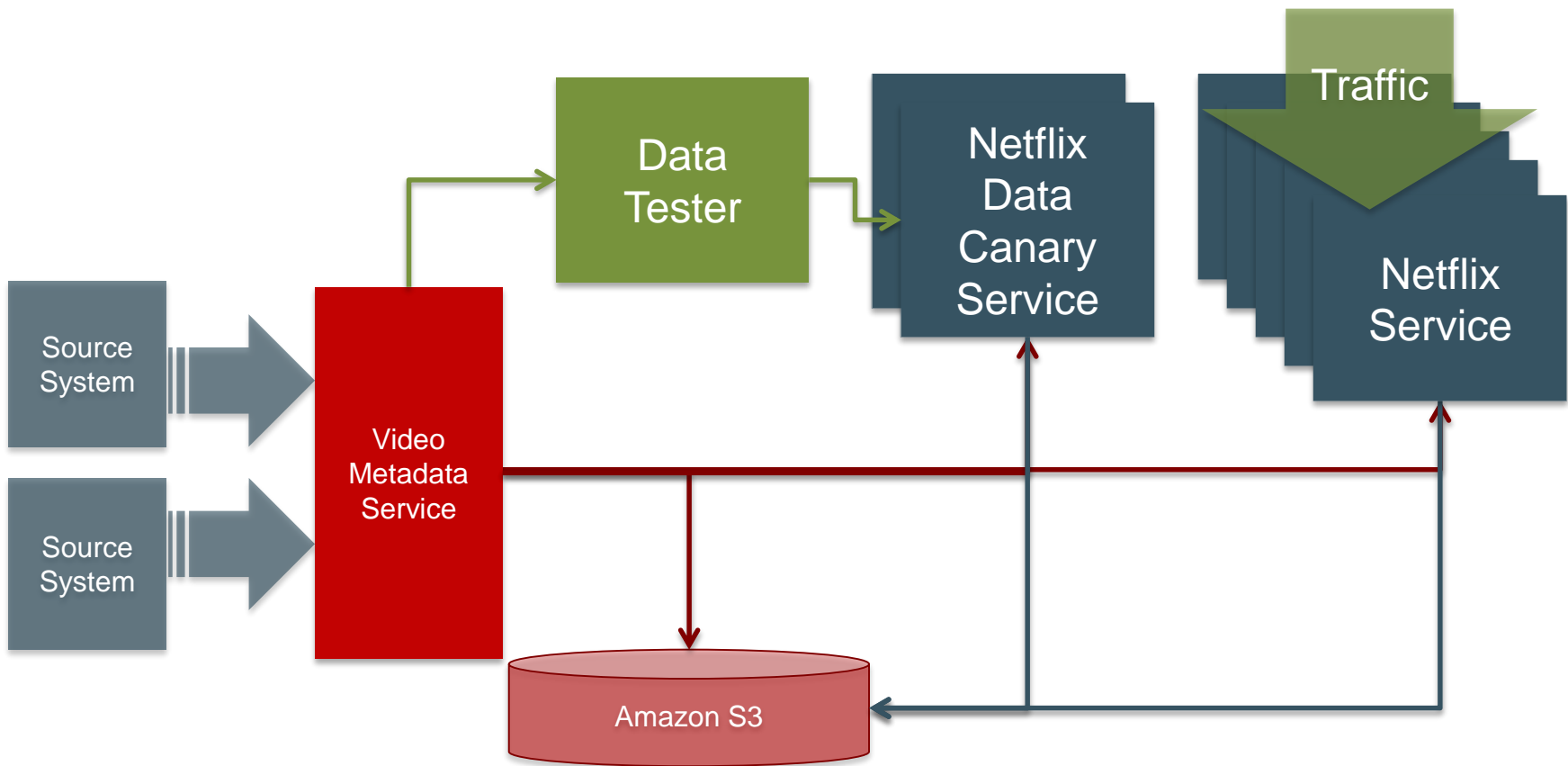
---

## VALID STATE TRANSITIONS



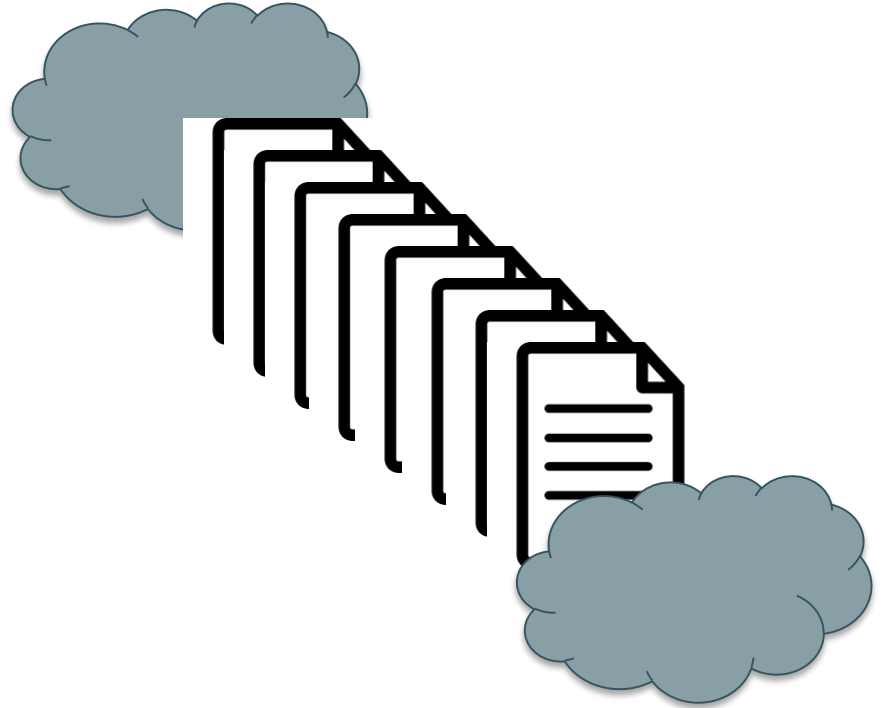
# DATA CANARY

---



# SEEING RETURNS

---



Verify consistency prior to  
applying state changes.

...one tool is a data canary.



# CHAPTER TWO

---

## THE VANISHING OF CRITICAL SERVICES





Ernesto Valle

@Ernesto\_V\_E



Follow

Retweeted Bryony Anne Harris  
(@BryonyHarris2):

netflix is down i am dying [#netflixdown](#)  
[fb.me/Mqrs0zls](https://fb.me/Mqrs0zls)



LIKE

1



3:45 PM



1

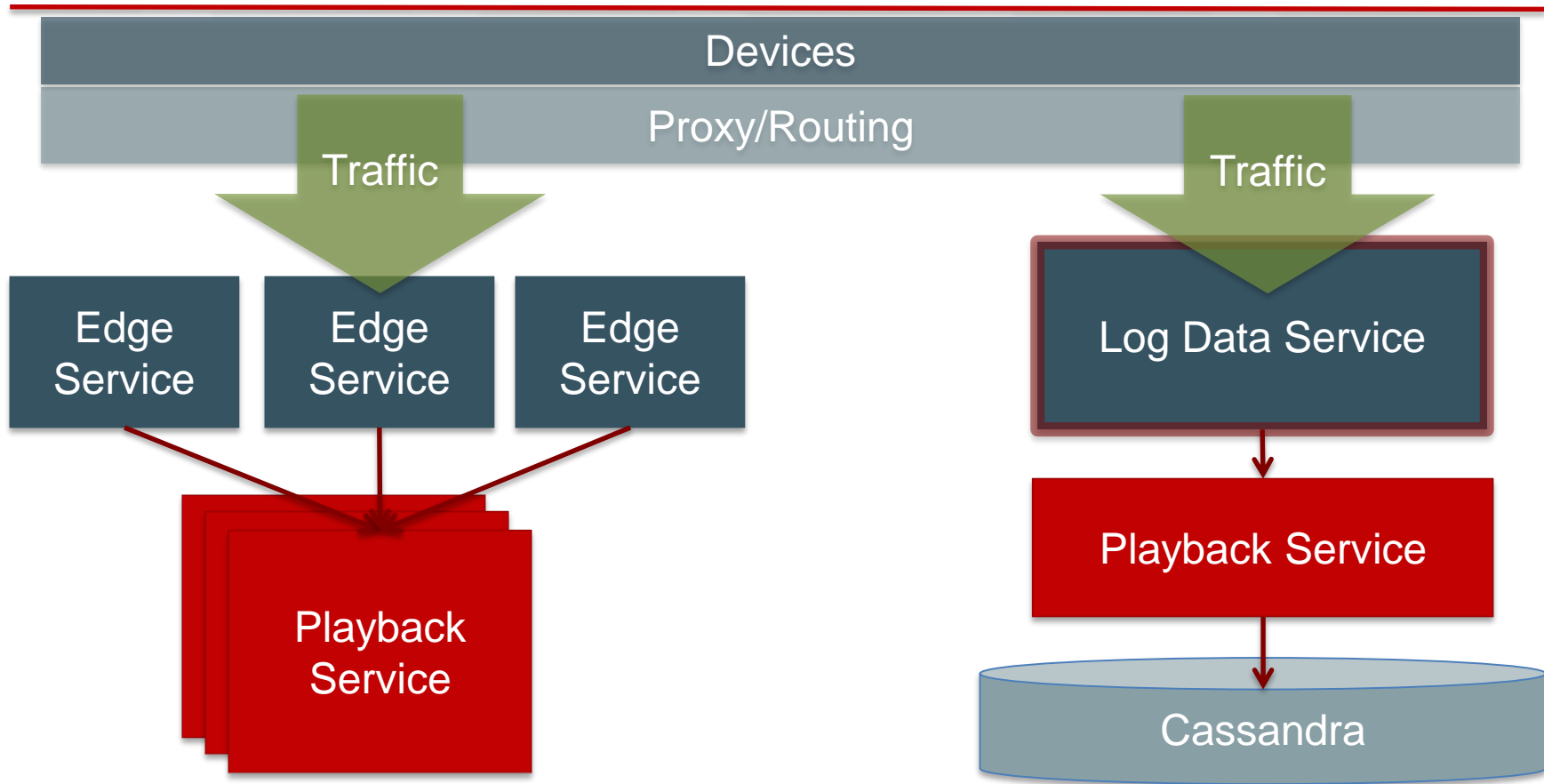


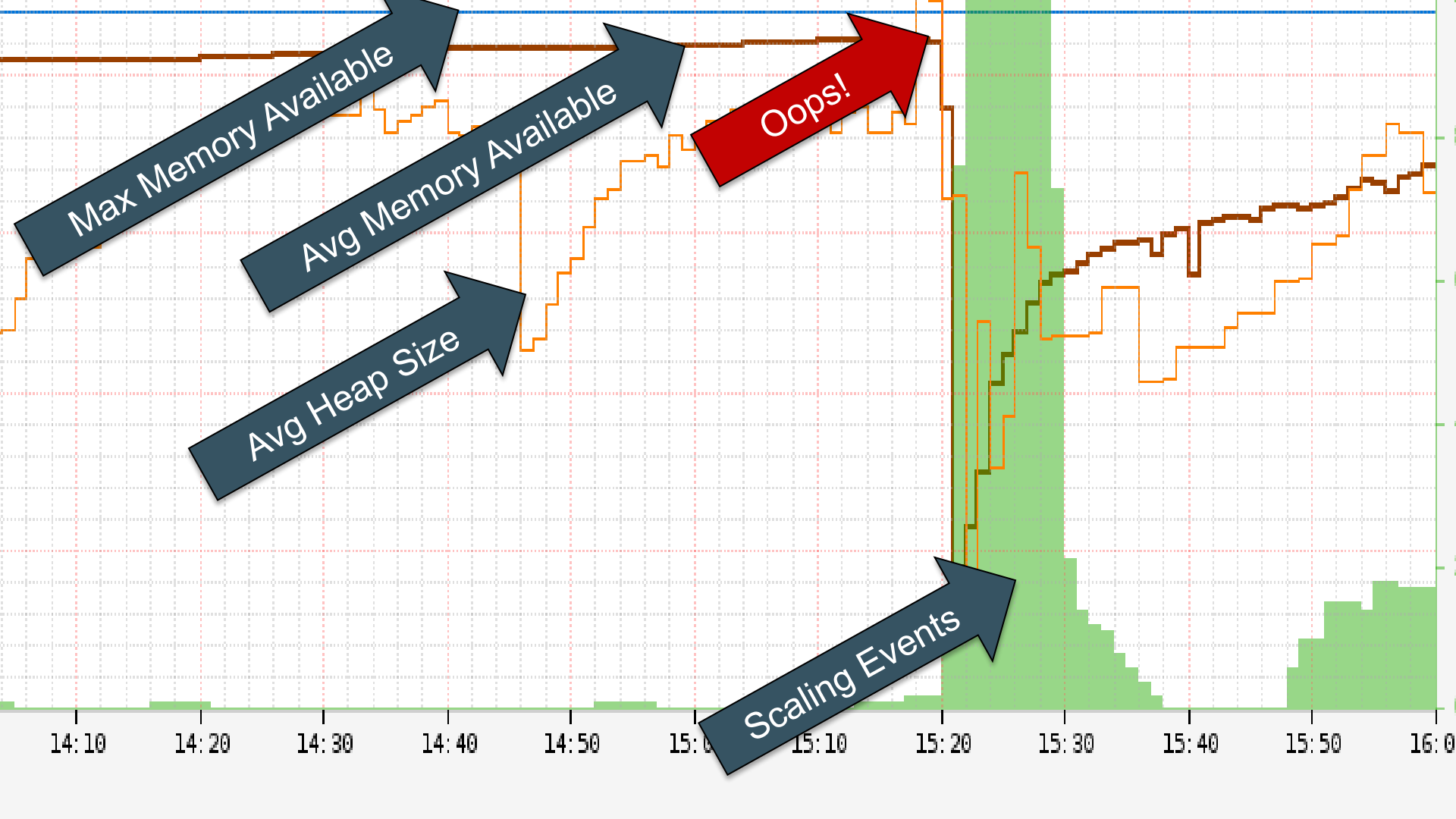


# A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.

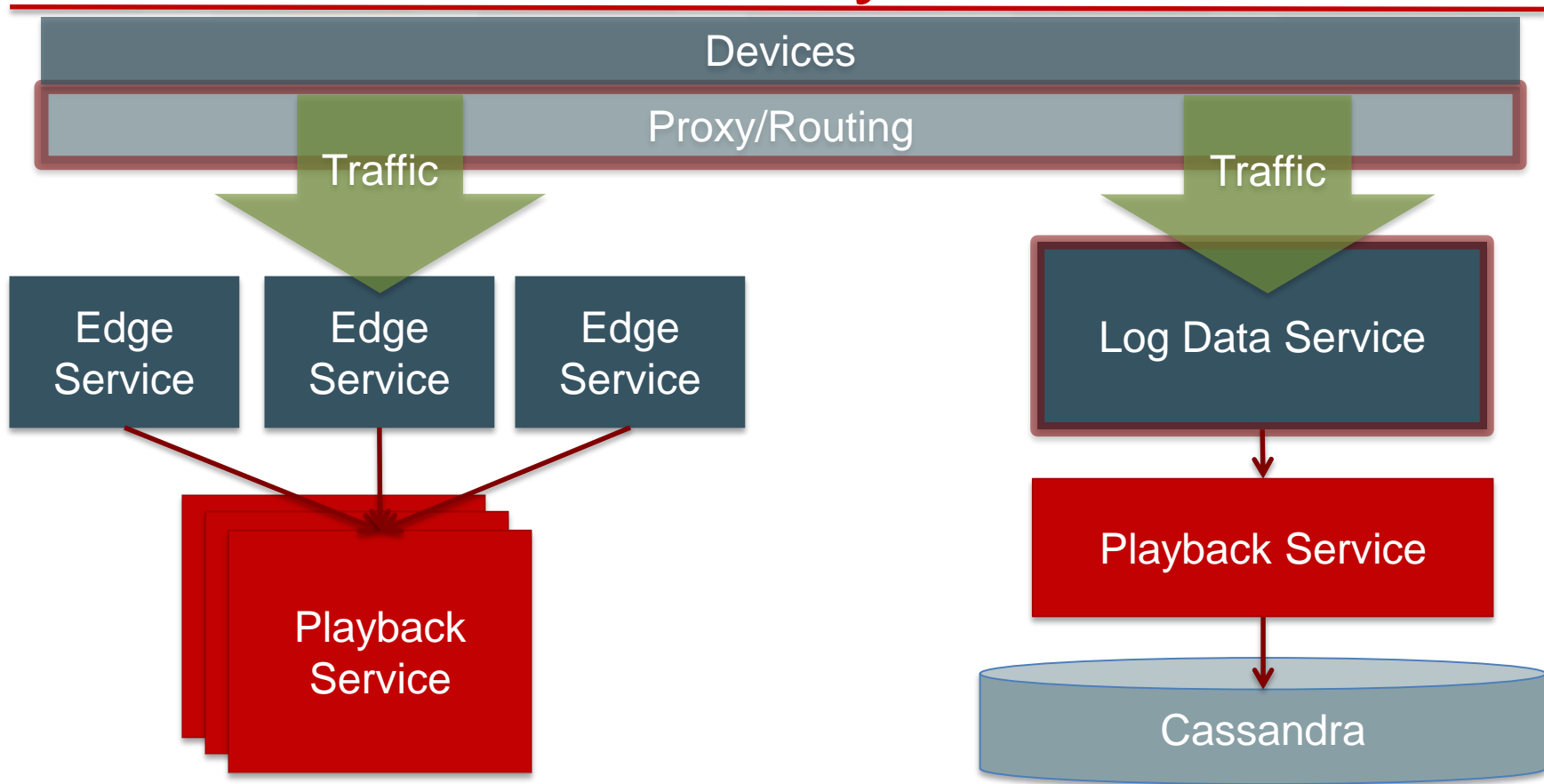
--Leslie Lamport

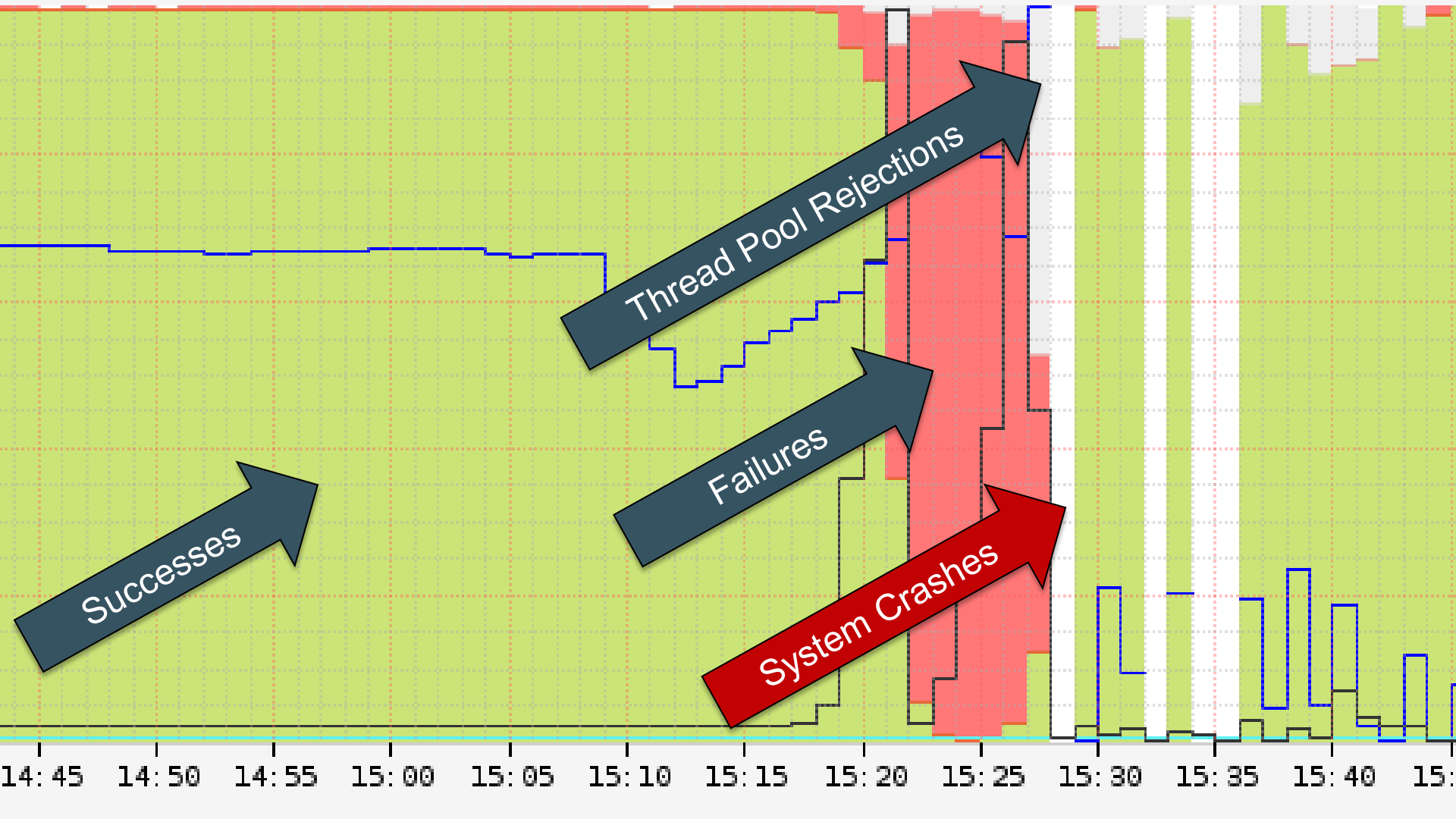
# LOG DATA



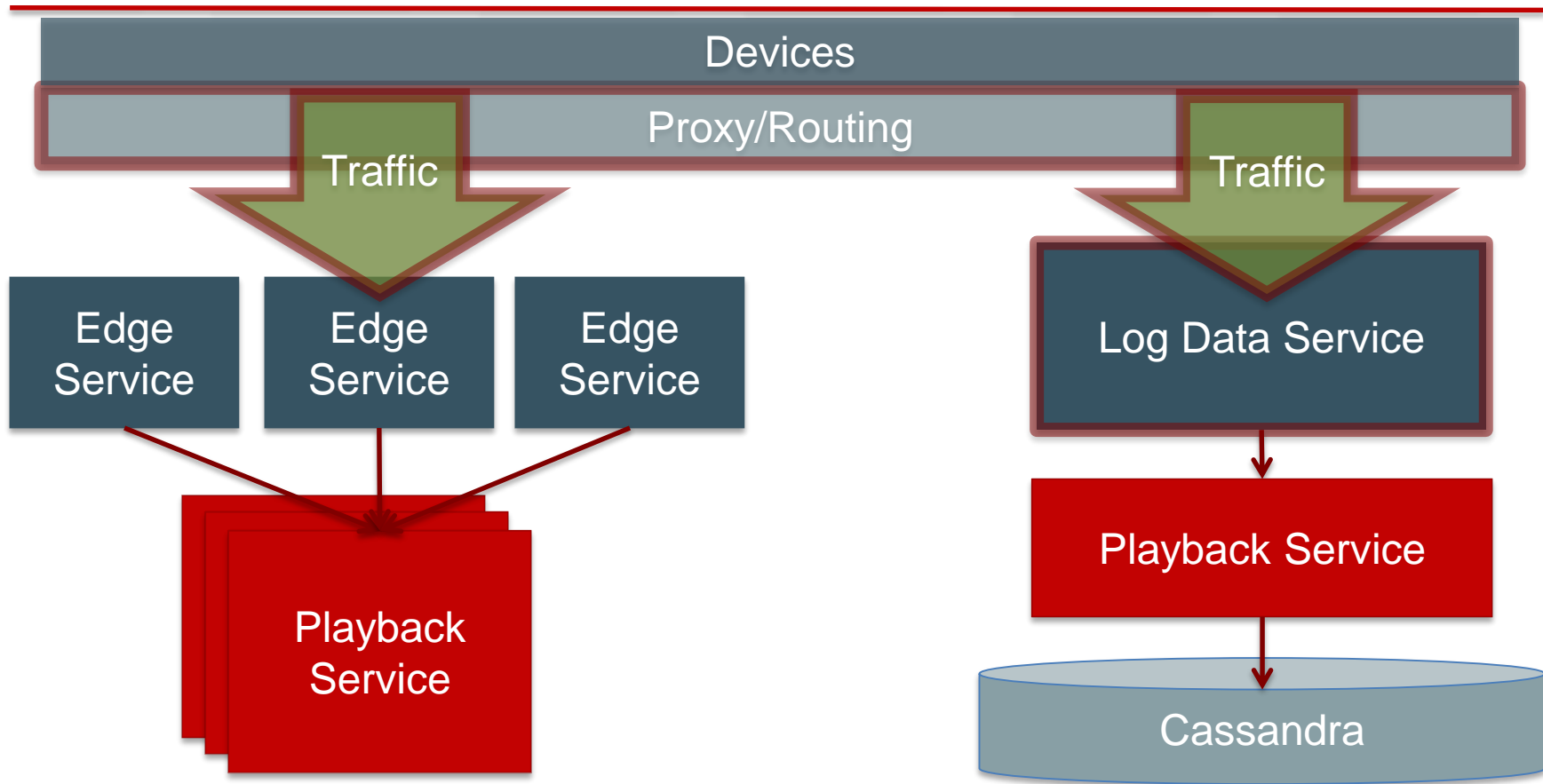


# Proxy





# CASCADING FAILURE





Log Data Service



Playback Service



Cassandra

```
{  
    
}
```

```
  throw new OutOfMemoryError();
```



PREVENTION

---

MANAGING RESOURCE CONSTRAINTS



# REDUCE SURFACE AREA

---



# 1

## SO MANY JARS!!

```
+ com.netflix.governator:governator-test-junit: (-),
+ com.netflix.hollow:hollow: (-),
+ com.netflix.hystrix:hystrix:hystrix-core: (-),
+ com.netflix.hystrix:hystrix:hystrix-metrics-event-stream: (-),
+ com.netflix.hystrix:hystrix:hystrix-serialization: (-),
+ com.netflix.hystrix:hystrix:hystrix-servo-metrics-publisher: (-),
+ com.netflix.ip-shadow:ipshadow-ip-module-rxnetty: (-),
+ com
  + netflix:netflix: (-),
+ com
  + avro:avro: (-),
+ com
  + avro:avro-compiler: (-),
+ com
  + ch.qos.logback:logback-classic: (-),
+ com
  + ch.qos.logback:logback-core: (-),
+ com
  + com.netflix.governator:governator-test-junit: (-),
+ com
  + com.netflix.hollow:hollow: (-),
+ com
  + com.netflix.hystrix:hystrix-core: (-),
+ com
  + com.netflix.hystrix:hystrix-metrics-event-stream: (-),
+ com
  + com.netflix.hystrix:hystrix-serialization: (-),
+ com
  + com.netflix.hystrix:hystrix-servo-metrics-publisher: (-),
+ com
  + com.netflix.ip-shadow:ipshadow-ip-module-rxnetty: (-),
+ com
  + com.netflix.ip-shadow:ipshadow-ip-rabbit: (-),
+ com
  + com.netflix.ip-shadow:ipshadow-rxnetty: (-),
+ com
  + com.netflix.ip-shadow:ipshadow-rxnetty-contexts: (-),
+ com
  + com.netflix.ip-shadow:ipshadow-rxnetty-spectator: (-),
+ com
  + com.netflix.jaxrs:netflix-jaxrs-extensions-avro: (-),
+ com
  + com.netflix.jaxrs:netflix-jaxrs-extensions-exceptions-mapper: (-),
+ com
  + com.netflix.karyon:karyon-core: (-),
+ com
  + com.netflix.karyon:karyon-eureka: (-),
+ com
  + com.netflix.karyon:karyon-api: (-),
+ com
  + com.netflix.karyon:karyon2-admin: (-),
+ com
  + com.netflix.karyon:karyon2-admin-eureka-plugin: (-),
+ com
  + com.netflix.karyon:karyon2-admin-web: (-),
+ com
  + com.netflix.mal:mal-core: (-),
+ com
  + com.netflix.mebius:mebius-graphic-resolution-rules: (-),
+ com
  + com.netflix.netflix-commons:netflix-commons-util: (-),
+ com
  + com.netflix.netflix-commons:netflix-eventbus: (-),
+ com
  + com.netflix.netflix-commons:netflix-eventbus-bridge: (-),
+ com
  + com.netflix.netflix-commons:netflix-infix: (-),
+ com
  + com.netflix.netflix-commons:netflix-statistics: (-),
+ com
  + com.netflix.nfsgraph:nfs-graph: (-),
+ com
  + com.netflix.nfschlep:nfschlep-core: (-),
+ com
  + com.netflix.nfschlep:nfschlep-kafka: (-),
+ com
  + com.netflix.nfschlep:nfschlep-kafka-consumer: (-),
+ com
  + com.netflix.numerus: Numerus: (-),
+ com
  + com.netflix.pythess:pythess-api: (-),
+ com
  + com.netflix.pythess:pythess-core: (-),
+ com
  + com.netflix.ribbon:ribbon-core: (-),
+ com
  + com.netflix.ribbon:ribbon-eureka: (-),
+ com
  + com.netflix.ribbon:ribbon-httpclient: (-),
+ com
  + com.netflix.ribbon:ribbon-loadbalancer: (-),
+ com
  + com.netflix.runtime:health-api: (-),
+ com
  + com.netflix.runtime:health-core: (-),
+ com
  + com.netflix.runtime:health-guice: (-),
+ com
  + com.netflix.runtime:health-integrations: (-),
+ com
  + com.netflix.runtime:runtime-lifecycle: (-),
+ com
  + com.netflix.runtime:runtime-swagger-core: (-),
+ com
  + com.netflix.runtime:runtime-swagger-lifecycle: (-),
+ com
  + com.netflix.rxjava:rxjava-core: (-),
+ com
  + com.netflix.schlep:schlep-core: (-),
+ com
  + com.netflix.schlep:schlep-eureka: (-),
+ com
  + com.netflix.schlep:schlep-kafka: (-),
+ com
  + com.netflix.schlep:schlep-kafka-consumer: (-),
+ com
  + com.netflix.schlep:schlep-sqs: (-),
+ com
  + com.netflix.servo:servo-apache: (-),
+ com
  + com.netflix.servo:servo-atlas: (-),
```

Keep Only  
Dependencies  
which are  
Necessary

```
+ com.netflix.ribbon:ribbon-core: (-),
+ com.netflix.ribbon:ribbon-eureka: (-),
+ com.netflix.ribbon:ribbon-httpclient: (-),
+ com.netflix.ribbon:ribbon-loadbalancer: (-),
+ com.netflix.runtime:health-api: (-),
+ com.netflix.runtime:health-core: (-),
+ com.netflix.runtime:health-guice: (-),
+ com.netflix.runtime:health-integrations: (-),
+ com.netflix.runtime:runtime-lifecycle: (-),
+ com.netflix.runtime:runtime-swagger-core: (-),
+ com.netflix.runtime:runtime-swagger-lifecycle: (-),
+ com.netflix.rxjava:rxjava-core: (-),
+ com.netflix.schlep:schlep-core: (-),
+ com.netflix.schlep:schlep-eureka: (-),
+ com.netflix.schlep:schlep-kafka: (-),
+ com.netflix.schlep:schlep-kafka-consumer: (-),
+ com.netflix.schlep:schlep-sqs: (-),
+ com.netflix.servo:servo-apache: (-),
+ com.netflix.servo:servo-atlas: (-),
```

2

# LIMIT “MAGIC”



3

# ADD KILL SWITCHES

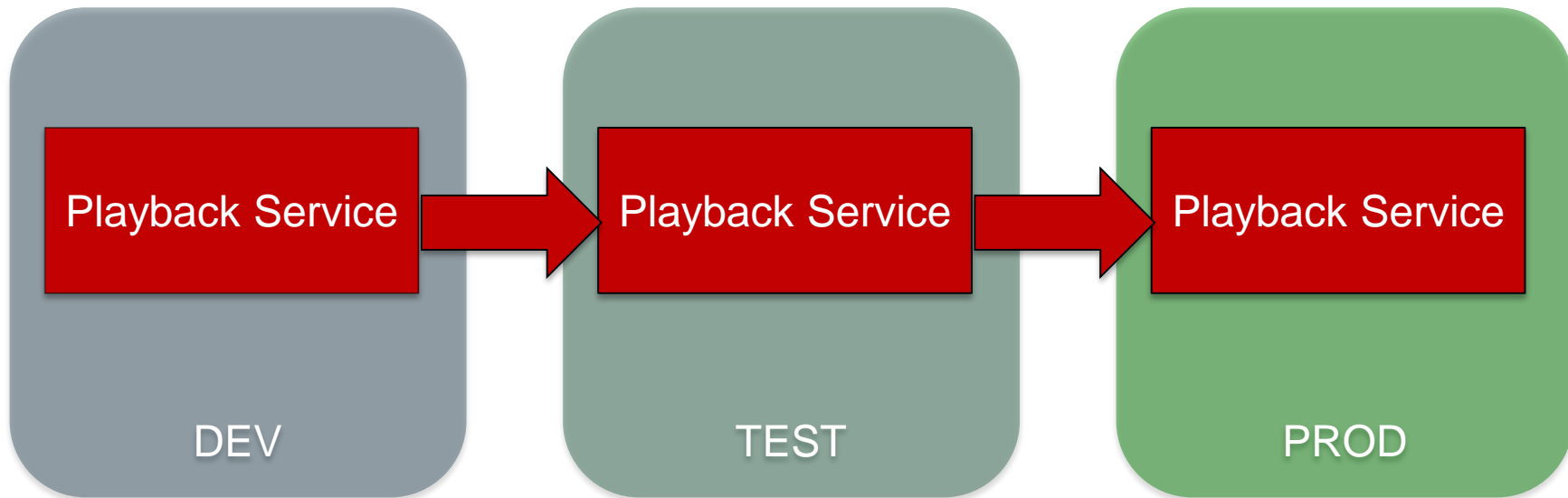
---



# 4

## FAVOR IMMUTABILITY

---





Proxy/Routing

Traffic

Log Data  
Service

Playback  
Service

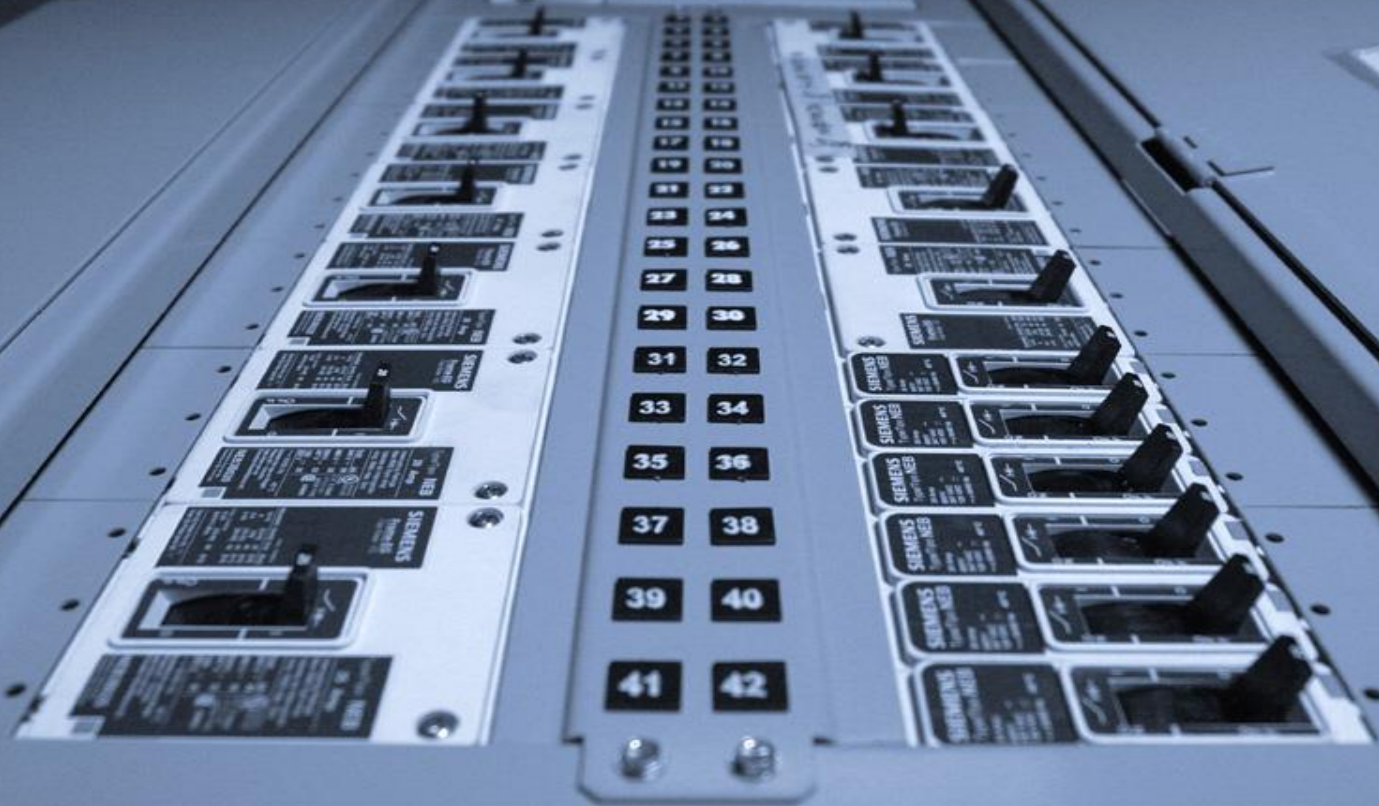
Cassandra

```
try {  
    remoteService.call();  
} catch ( Throwable t ) {  
    //Oops!  
    System.exit(1);  
}
```

# MITIGATION

---

# CIRCUIT BREAKERS





**HYSTRIX**

DEFEND YOUR APP

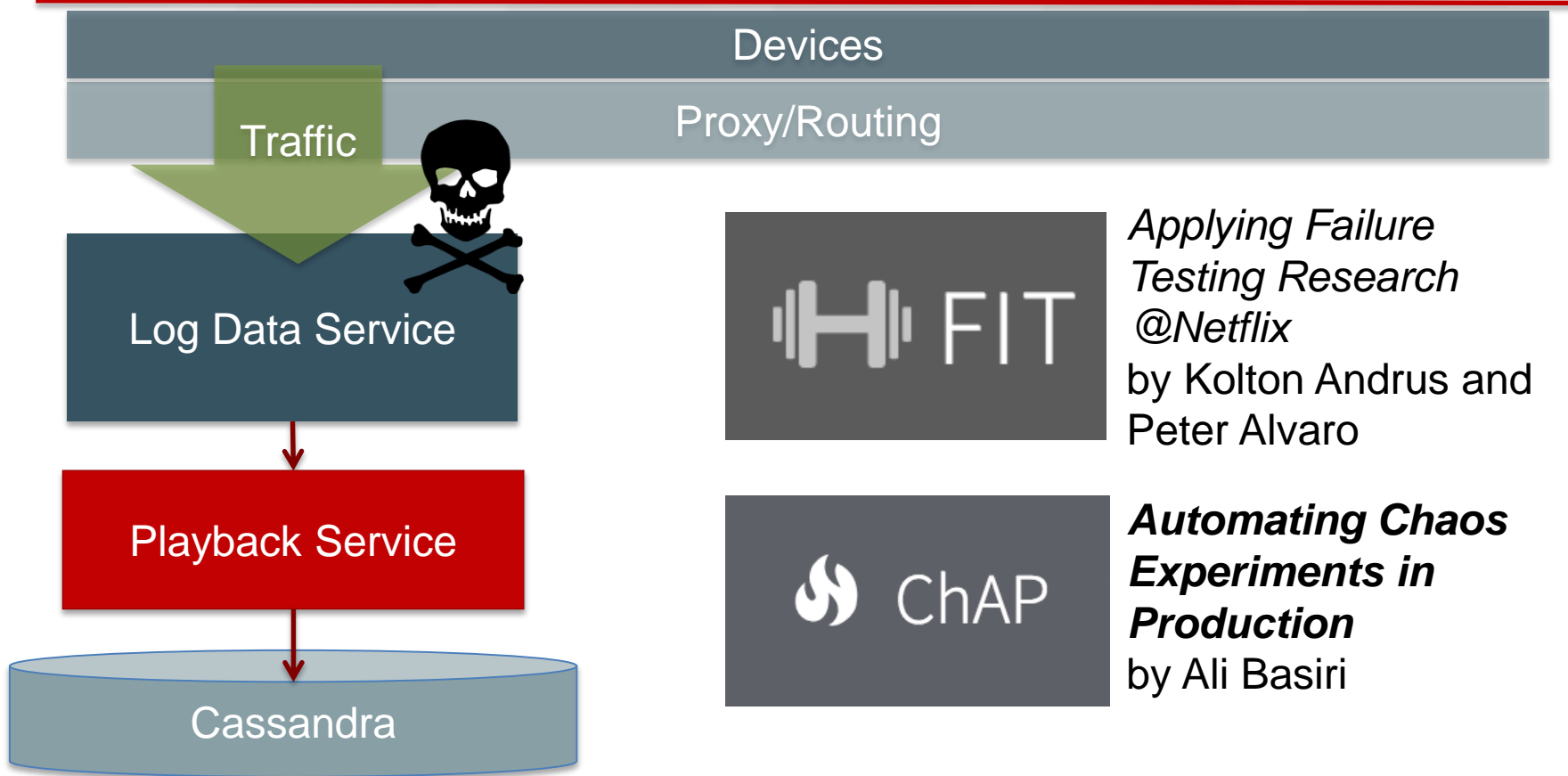




# FAILURE TESTING

---

# FAILURE TESTING



*Applying Failure  
Testing Research  
@Netflix*

by Kolton Andrus and  
Peter Alvaro

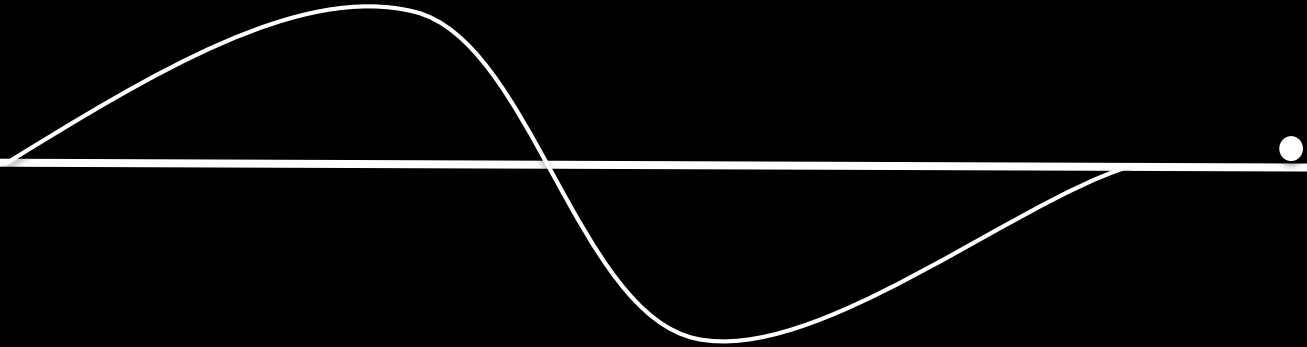


***Automating Chaos  
Experiments in  
Production***

by Ali Basiri

Manage resource constraints by  
reducing surface area.

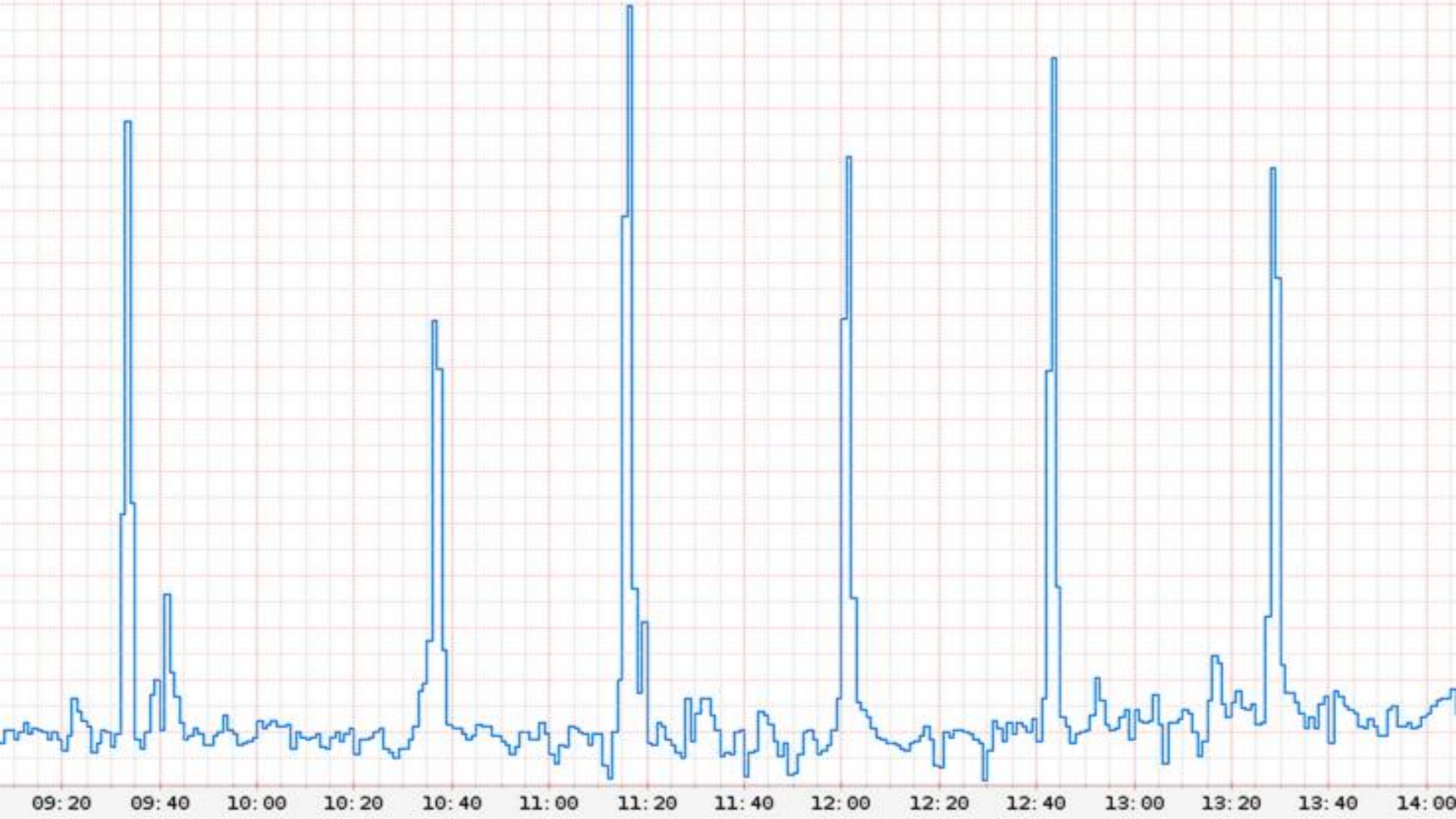
Leverage circuit breakers and  
rigorously test failures.



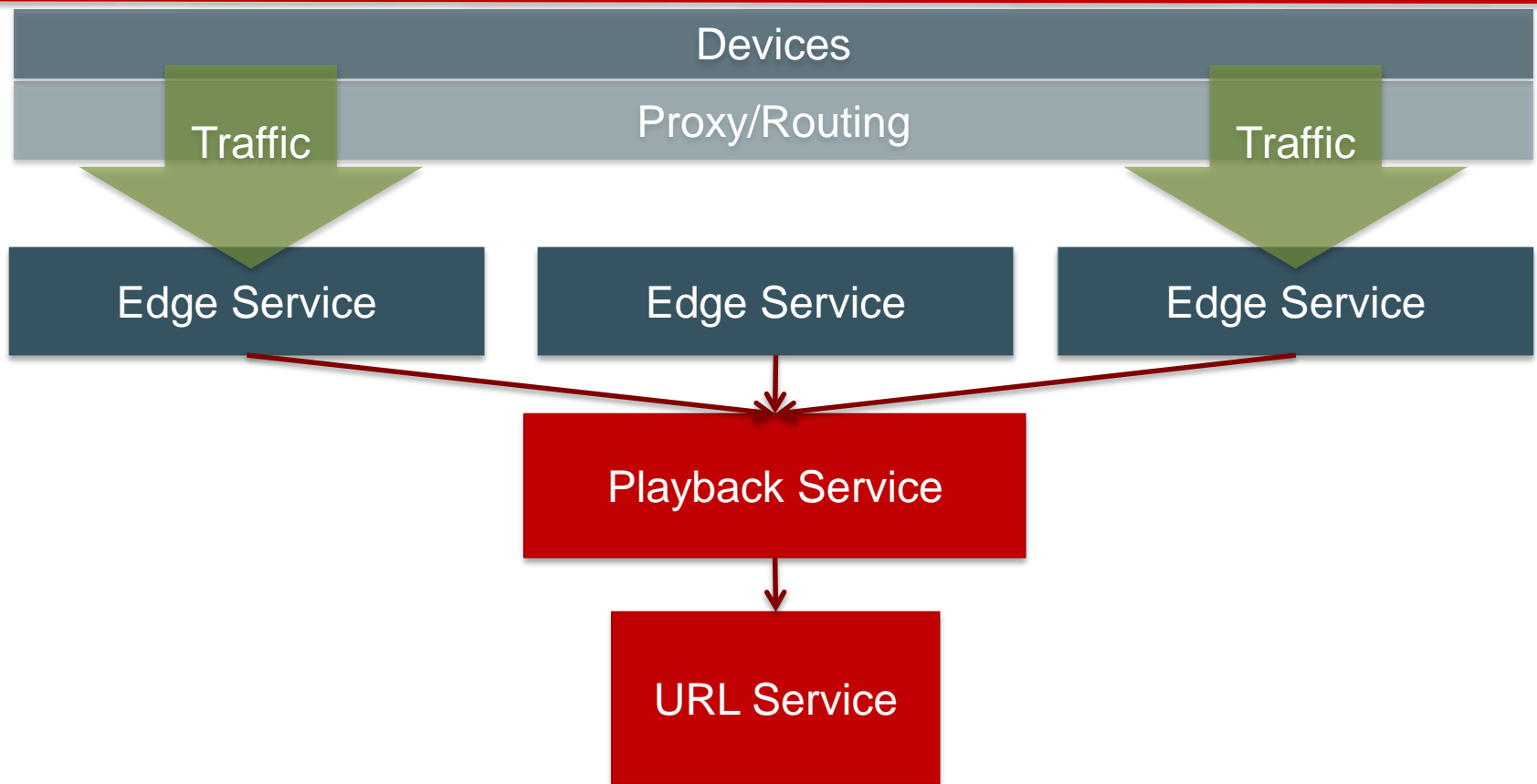
# CHAPTER THREE

---

## THE THROTTLE



# PLAYBACK ARCHITECTURE



# NETFLIX CLIENT JARS

---

## Playback Service

### URL Client

Service Discovery

RPC

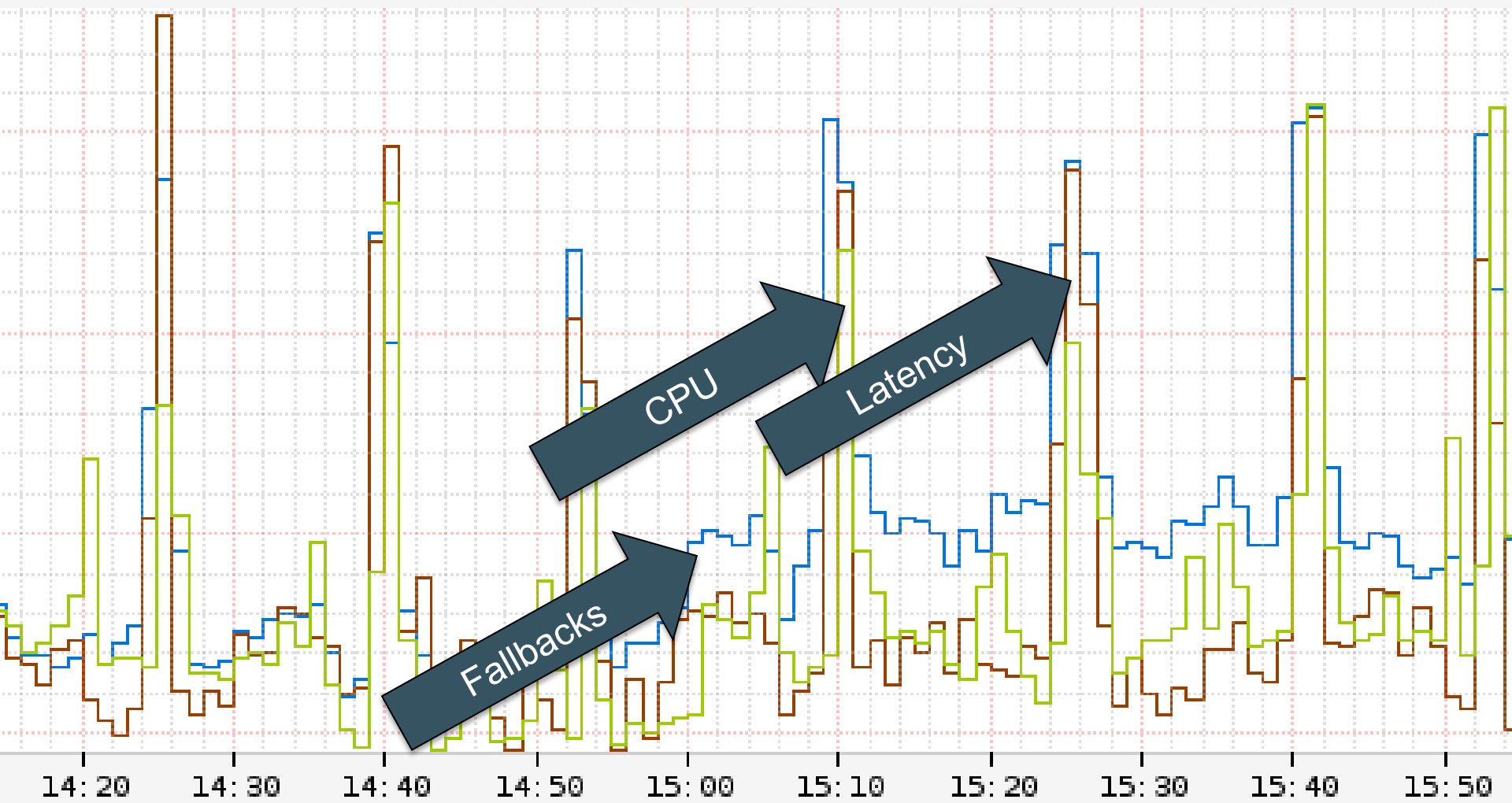
Metrics

Retries and Timeouts

Circuit-breakers and Fallbacks



URL  
Service

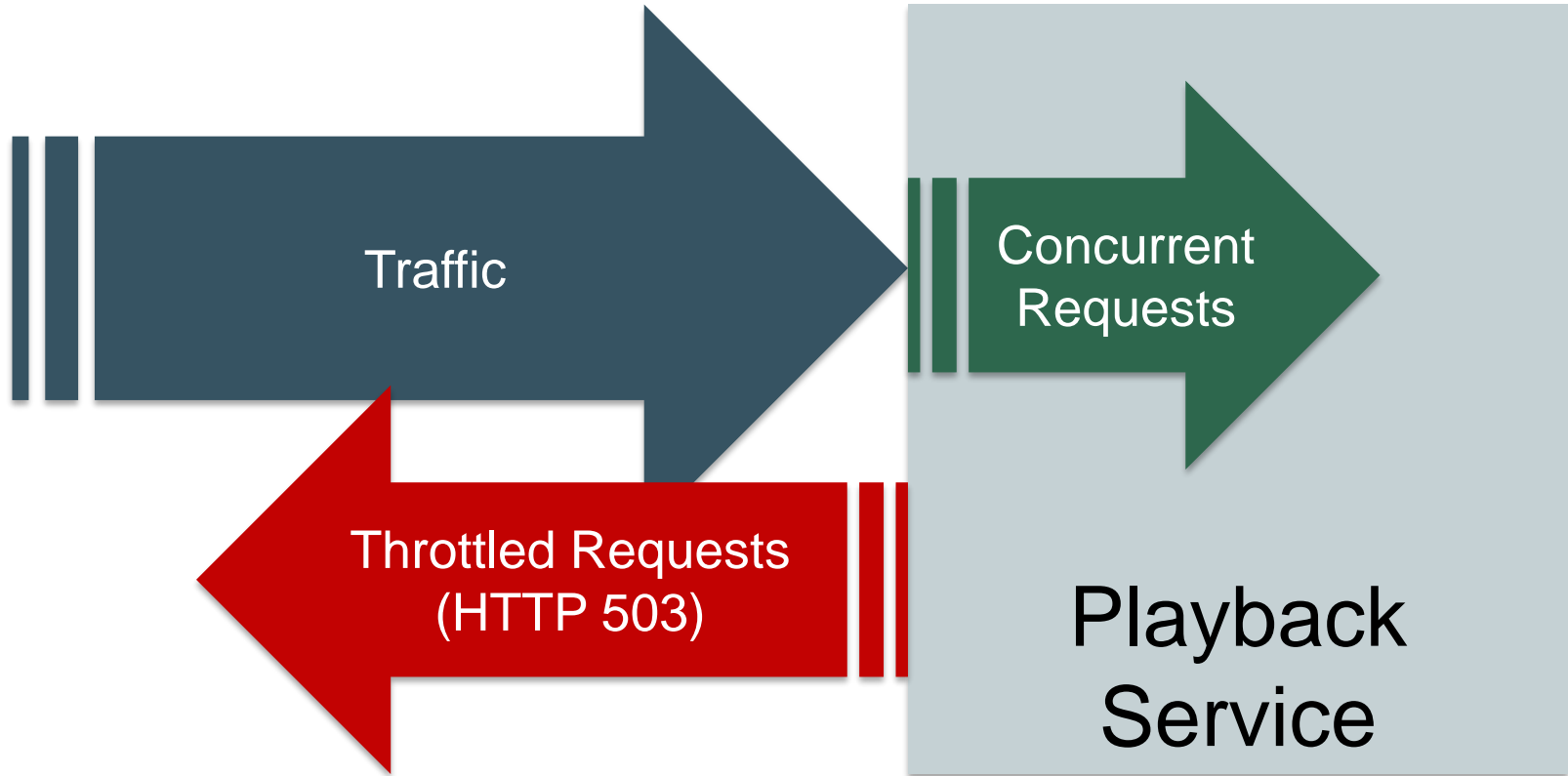


14:20 14:30 14:40 14:50 15:00 15:10 15:20 15:30 15:40 15:50



# THROTTLING

---





**Svenn Amish**

@amishschool



Follow

With Netflix down I had to make small talk with my kids with questions like "how was school?" and "what's your name again?"

RETWEETS

8

LIKES

26



9:50 PM

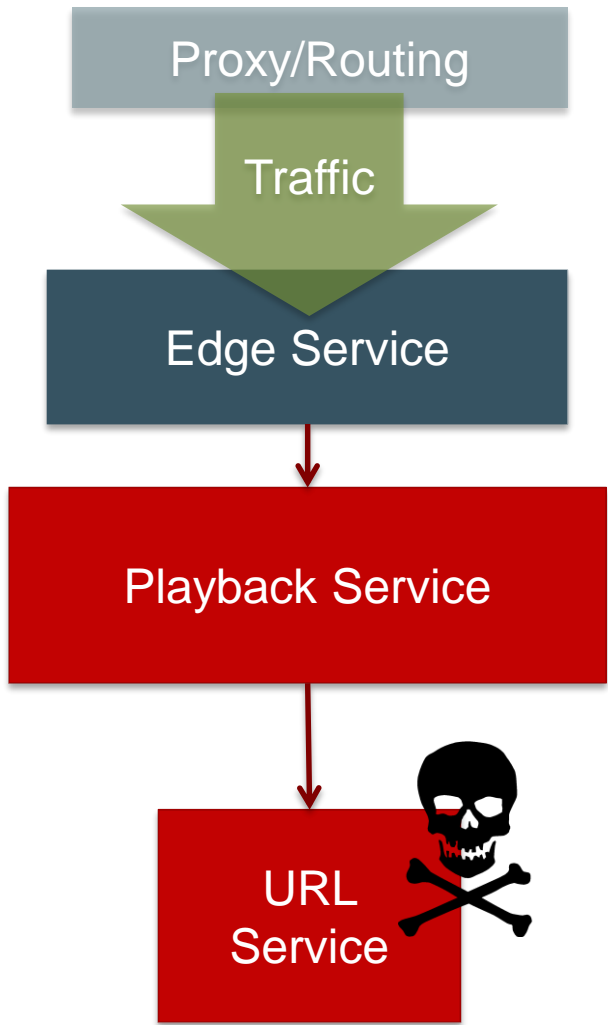


8



26





```
}  
  System.gc();  
}
```

# NETFLIX CLIENT JARS

---

## Playback Service

### URL Client

Service Discovery


RPC

Metrics

Retries and  
Timeouts

Circuit-breakers

Heavy  
Fallback



URL  
Service

# FALLBACK TESTING

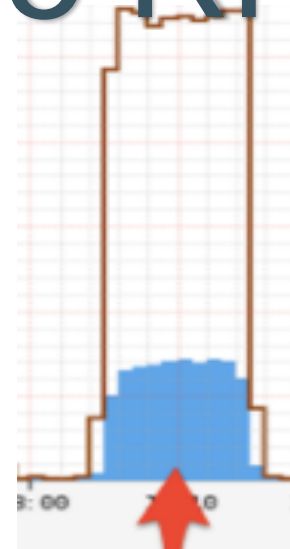
---

No fallback, CPU held at 90%



58 RPS

15 RPS

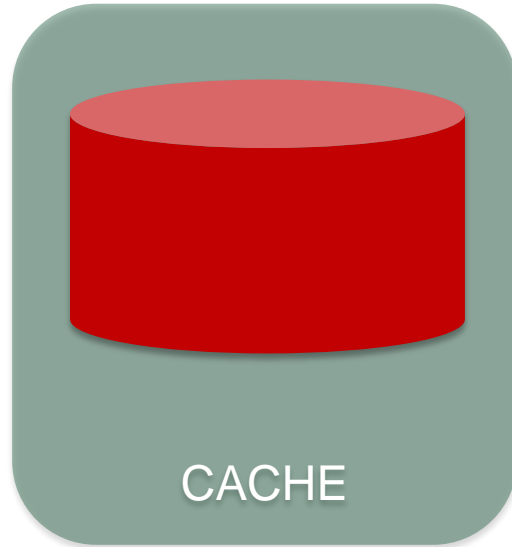
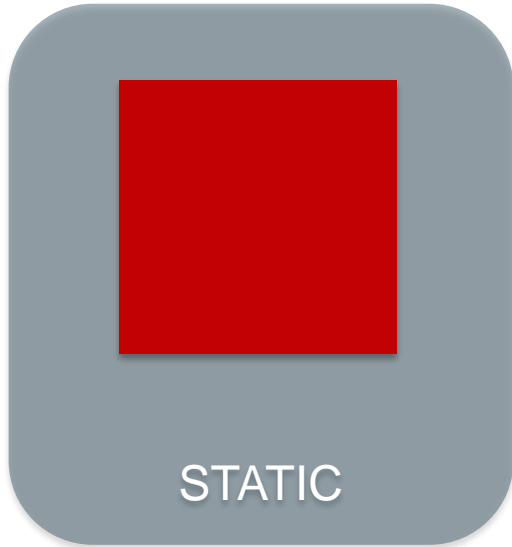


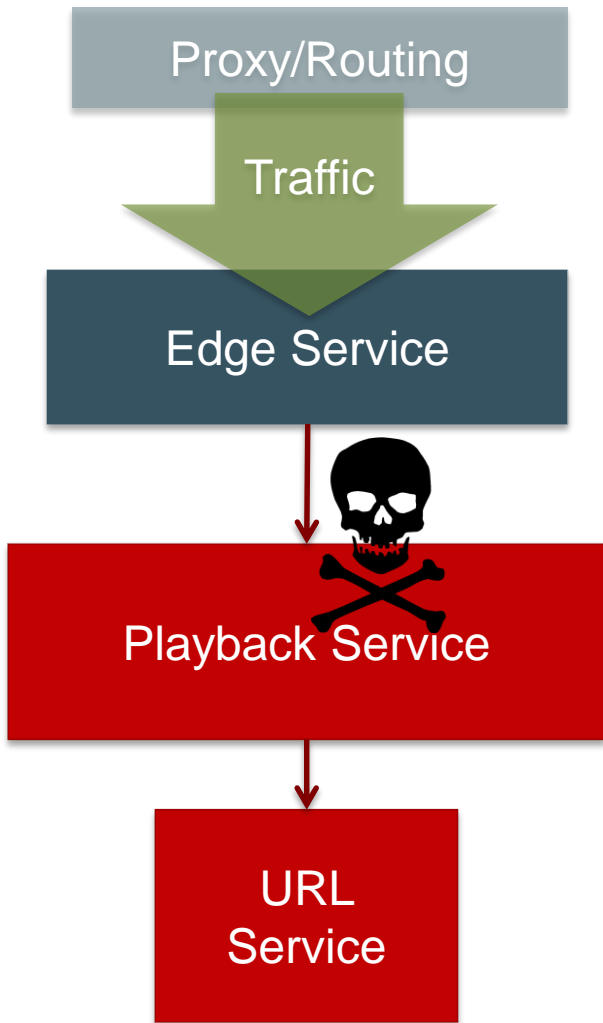
With 100% Fallback,  
CPU held at 90%

Siege: <https://github.com/JoeDog/siege>

# SELECTING FALLBACKS

---





```
}
```

```
return Response  
    .status(503)  
    .build();
```

```
}
```

# REQUEST BUCKETING

---

**MIDLAND RAILWAY**  
THESE **B**UCKETS MUST BE KEPT  
FULL OF **C**LEAN **W**ATER, AND USED  
ONLY IN CASE OF **F**IRE.  
IN SEVERE FROST THEY MUST  
BE EMPTIED AND ARRANGED CLOSE  
TO A WATER TAP, UNLESS THEY  
CAN BE TEMPORARILY KEPT IN A  
CONVENIENT PLACE WHERE THE  
WATER WILL NOT FREEZE.  
DERBY, JAN 1909 BY ORDER.

## CRITICAL

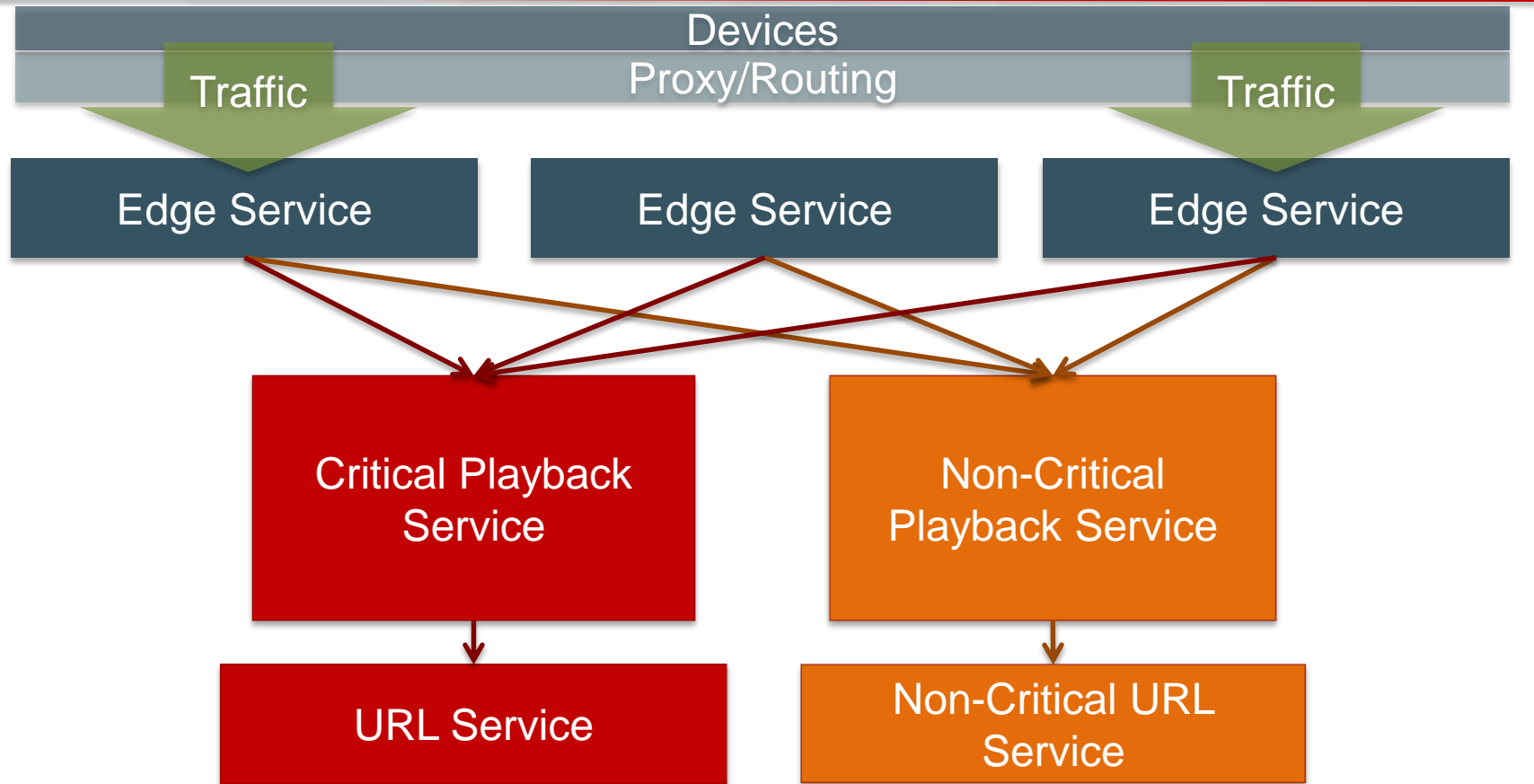
Customer  
Streaming  
Impact

## NON-CRITICAL

Experience or  
Performance  
Impact



# APPLICATION SHARDING



# CRITICAL

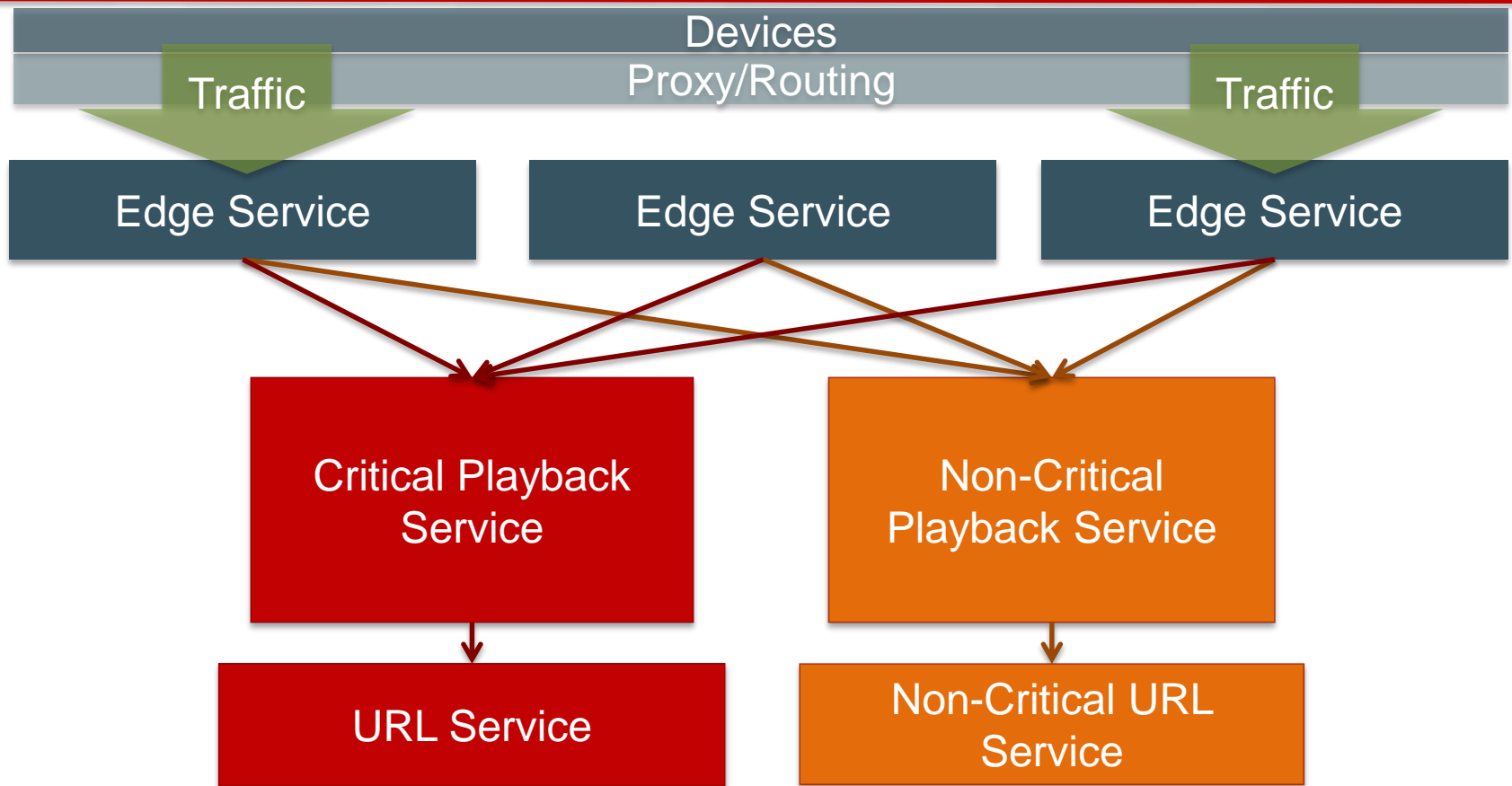
---



# NON-CRITICAL



# APPLICATION SHARDING



No heavy fallbacks!!

Fallbacks should be light and fast.

Shard your application based on  
operational characteristics.



# EPILOGUE

---

## KEY TAKEAWAYS

# KEY TAKEAWAYS

---

## CHAPTER 1: THE WEIRD DATA IN THE CATALOG

- Verify consistency prior to applying state changes.
- One tool is a data canary.

## CHAPTER 2: THE VANISHING OF CRITICAL SERVICES

- Manage resource constraints by reducing surface area.
- Leverage circuit breakers and rigorously test failures.

## CHAPTER 3: THE THROTTLE

- No heavy fallbacks!! Fallbacks should be light and fast.
- Shard your application based on operational characteristics.

Plan to fail.



The unexpected will happen.





PARTING THOUGHT  

---

~~DISTRIBUTED SYSTEMS~~  
SOCIAL

Questions?



Haley Tucker