BUILDING RELIABLE SERVICES AT NRK TV

@EINARWH

NRK TV


SKAM

AGENDA

WHAT IS THE SERVICE?
WHAT IS RELIABILITY?
RELIABILITY ENGINEERING
STRESS TESTS
TOOLS & TECHNIQUES

WHAT IS THE SERVICE?

WHAT IS RELIABILITY?

RELIABILITY ENGINEERING

STRESS TESTS

TOOLS & TECHNIQUES

WHAT IS THE SERVICE?

WHAT IS RELIABILITY?

RELIABILITY ENGINEERING

STRESS TESTS

TOOLS & TECHNIQUES

WHAT IS THE SERVICE?

WHAT IS RELIABILITY?

RELIABILITY ENGINEERING

STRESS TESTS

TOOLS & TECHNIQUES

WHAT IS THE SERVICE?

WHAT IS RELIABILITY?

RELIABILITY ENGINEERING

STRESS TESTS

TOOLS & TECHNIQUES

WHAT IS THE SERVICE?
WHAT IS RELIABILITY?
RELIABILITY ENGINEERING
STRESS TESTS
TOOLS & TECHNIQUES

WHAT IS THE SERVICE?

TV STREAMING SERVICE

TV STREAMING APP

CLIENT - SERVER

PLAY THE GAME OF

HUNT THE MEDIA MANIFEST

CONTENT DELIVERY NETWORK

FOR THE PETABYTES

OUR API IS A METADATA API

ALMOST EXCLUSIVELY READS

WRITES FOR PROGRESS AND ANALYTICS

TRAFFIC BURSTS

BIG LAUNCH = BLACK FRIDAY

WHAT IS A RELIABLE SERVICE?

WHAT DOES IT MEAN TO BE RELIABLE?

ROBUST AND RESILIENT

WHAT DOES IT MEAN TO BE ROBUST?

WHAT DOES IT MEAN TO BE RESILIENT?

ROBUST


* CROWD CHEERING *

RESILIENT

Tubthumping

Chumbawamba

I get knocked down, but I get up again
You are never gonna keep me down
I get knocked down, but I get up again
You are never gonna keep me down
I get knocked down, but I get up again
You are never gonna keep me down
I get knocked down, but I get up again
You are never gonna keep me down

ROBUST AND RESILIENT ARE NOT BOOLEANS

ROBUST AND RESILIENT

ARE NOT EVEN PERCENTAGES

ROBUST AGAINST

CERTAIN KINDS AND AMOUNTS OF STRESS

RESILIENT UNDER CERTAIN KINDS OF BAD CONDITIONS

NON-FUNCTIONAL REQUIREMENTS

SPECIFIC AND CONTEXTUAL

DEPENDS ON THE SERVICE

BUSINESS REQUIREMENTS

«GOOD ENOUGH»

TYPICALLY NOT SPECIFIED

«BEST EFFORT»

BEST EFFORT IS A GIVEN

BEST EFFORT

ISN’T AUTOMATICALLY

GOOD ENOUGH

SOMETIMES

GOOD ENOUGH

MUST BE

REALLY QUITE GOOD

WE MIGHT NOT REACH IT THROUGH BEST EFFORT ALONE

MINDSET

MOST DEVELOPERS ARE CONSCIENTOUS

COMPILE TIME MINDSET

«THE CODE IS THE TRUTH»

WHAT TRUTH?

THE TRUTH ABOUT WHAT?

WHAT HAPPENS IN PRODUCTION?

RUNTIME ENVIRONMENT

AVAILABLE RESOURCES

EXTERNAL DEPENDENCIES

THE RUNTIME IS DYNAMIC

THE RUNTIME IS PARTIALLY OUT OF YOUR CONTROL

THE RUNTIME IS UNKNOWNABLE

RUNTIME BEHAVIOR IS NOT PURELY ANALYTICAL

CODE

+

ENVIRONMENT

RUNTIME MINDSET

WE CAN’T FIGURE IT ALL OUT IN OUR HEADS

SYSTEMS THINKING

HYPOTHESES AND VALIDATION

BEING ON-CALL DOES WONDERS FOR THE RUNTIME MINDSET

HOW COMPLEX SYSTEMS FAIL

RICHARD COOK

RELIABILITY AT NRK TV

IMPROVEMENT START WITH A CRISIS

OUR BEST EFFORT WASN’T GOOD ENOUGH

HOW TO IMPROVE BEYOND BEST EFFORT?

HOW CAN WE TELL IF IT IS GETTING BETTER OR WORSE?

WE NEED FEEDBACK

ENGINEERING

=

BEST EFFORT + FEEDBACK

FROM AD HOC TO STRATEGIC RELIABILITY WORK

SERVICE LEVEL OBJECTIVE (SLO)

SERVICE LEVEL INDICATOR (SLI)

ERROR RATE

LATENCY

THROUGHPUT

CAVEATS

METRICS CAN BE HARMFUL

METRICS CAN BE ABUSED

METRICS CAN BE GAMED

UNDERMINES THE PURPOSE

REQUIRES TRUST

BOTTOM-UP

SELF-IMPOSED BY TEAM

DEFINED BY TEAM

OWNED BY TEAM

NO REWARDS

NO COMPETITION

AN SLO SHOULD BE APPROPRIATE

INVESTING IN MEETING A TOO-STRICT SLO IS WASTE

NO VANITY METRICS

WHAT ARE SLOS FOR?

GUIDE DECISIONS

TO HELP COMMUNICATE

TO VALIDATE IMPROVEMENTS

TO AVOID REGRESSIONS

HOW TO DEFINE AN SLO

UNCOMFORTABLE!

UNCOMFORTABLE!
(IT MUST BE GOOD!)

FORCE DECISIONS

MAKE IT CONCRETE

SLO Channel Playback endpoints

Uptime: 99.95%

Measured behind gateway/cache.

Measuring interval: 1 minute.

SLO applies to loads of 10 to 50.000 req/min.

SLO applies in time interval 06:00 - 01:00.

	5xx errors	Latency (99 percentile)
PS Playback API Channel - tv/live	< 0.01 %	< 200 ms
PS Playback API Channel - playback/channel/metadata	< 0.01 %	< 100 ms
PS Playback API Channel - playback/channel/manifest	< 0.01 %	< 200 ms

START MEASURING
CHOOSE OBJECTIVES
OBSERVE
REVISE/INVEST

=> START MEASURING <=

CHOOSE OBJECTIVES

OBSERVE

REVISE/INVEST

START MEASURING

=> CHOOSE OBJECTIVES <=

OBSERVE

REVISE/INVEST

START MEASURING

CHOOSE OBJECTIVES

=> OBSERVE <=

REVISE/INVEST

START MEASURING

CHOOSE OBJECTIVES

OBSERVE

=> REVISE/INVEST <=

START MEASURING

CHOOSE OBJECTIVES

=> OBSERVE <=

REVISE/INVEST

START MEASURING

CHOOSE OBJECTIVES

OBSERVE

=> REVISE/INVEST <=

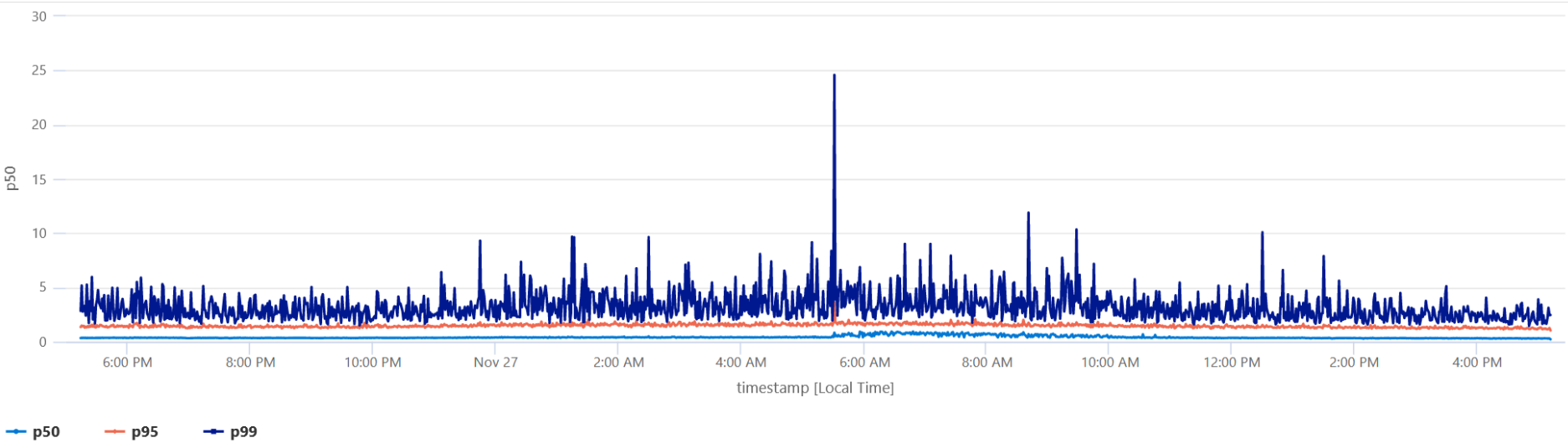
OBSERVATION

DASHBOARDS

```
requests
| where customDimensions.applicationName == "ps-playback-api-channel"
| where url contains "metadata/channel"
| summarize percentiles(duration, 50, 95, 99) by bin(timestamp, 1m)
| project
    timestamp,
    p50=percentile_duration_50,
    p95=percentile_duration_95,
    p99=percentile_duration_99
| render timechart
```

Results | Chart | Display time (UTC+01:00)

Completed. Showing results from the last 24 hours. 00:01.2 1,439 records



REPORTS

Weekly SLO Uptime report for playback/manifest/channel

SLO report for the week 11/15/2021 12:00:08 AM - 11/22/2021 12:00:08 AM

SLO latency 99% pr min: < 200 ms
SLO error rate pr min: < 0.01%

SLO applies to the time interval 06:00 - 01:00

SLO uptime: 99.95%
SLO downtime budget: 4 minutes.

SLO Status: ✔

Uptime: 99.99%

Downtime: 1 minutes.
Downtime (latency): 1 minutes.
Downtime (error rate): 0 minutes.

Additional downtime outside SLO hours: 1 minutes.

NOTIFICATIONS?

SHOULD WE SET UP ALERTS FOR SLO VIOLATIONS?

ALERTS SHOULD MEAN

ACTION NECESSARY

IT’S NOT A HEALTH CHECK

ARE WE RELIABLE NOW?

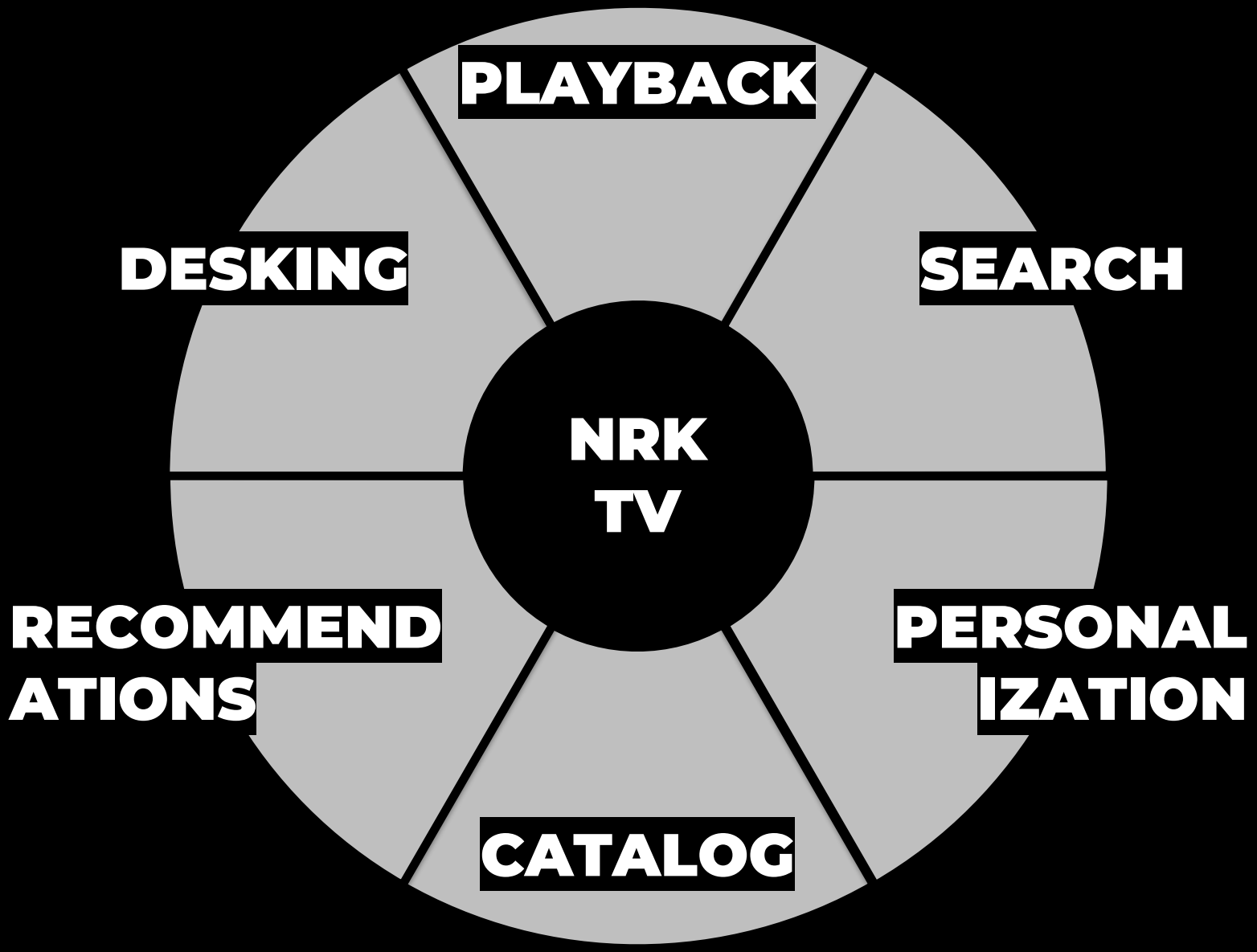
OF COURSE NOT!

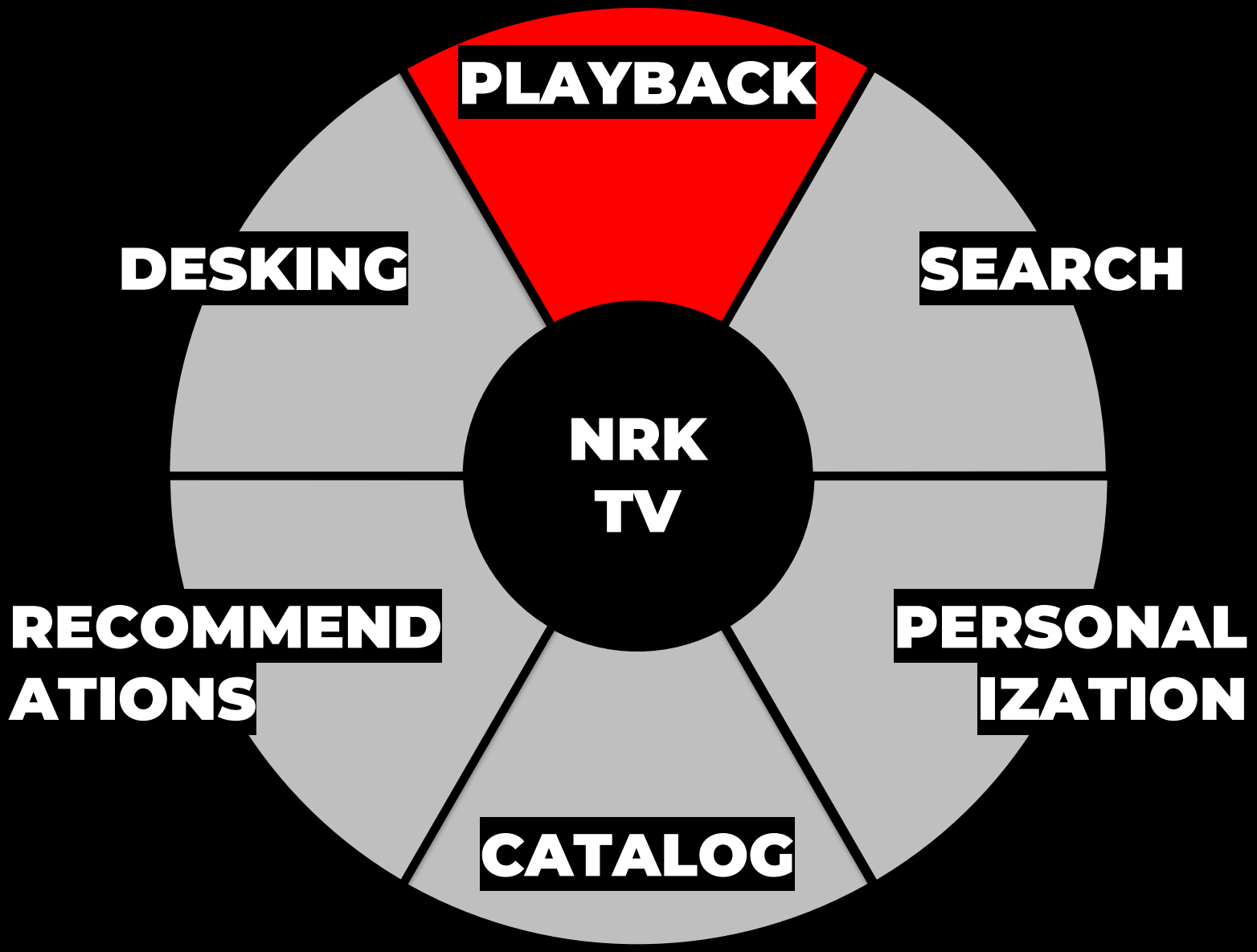
WE HAVE FEEDBACK

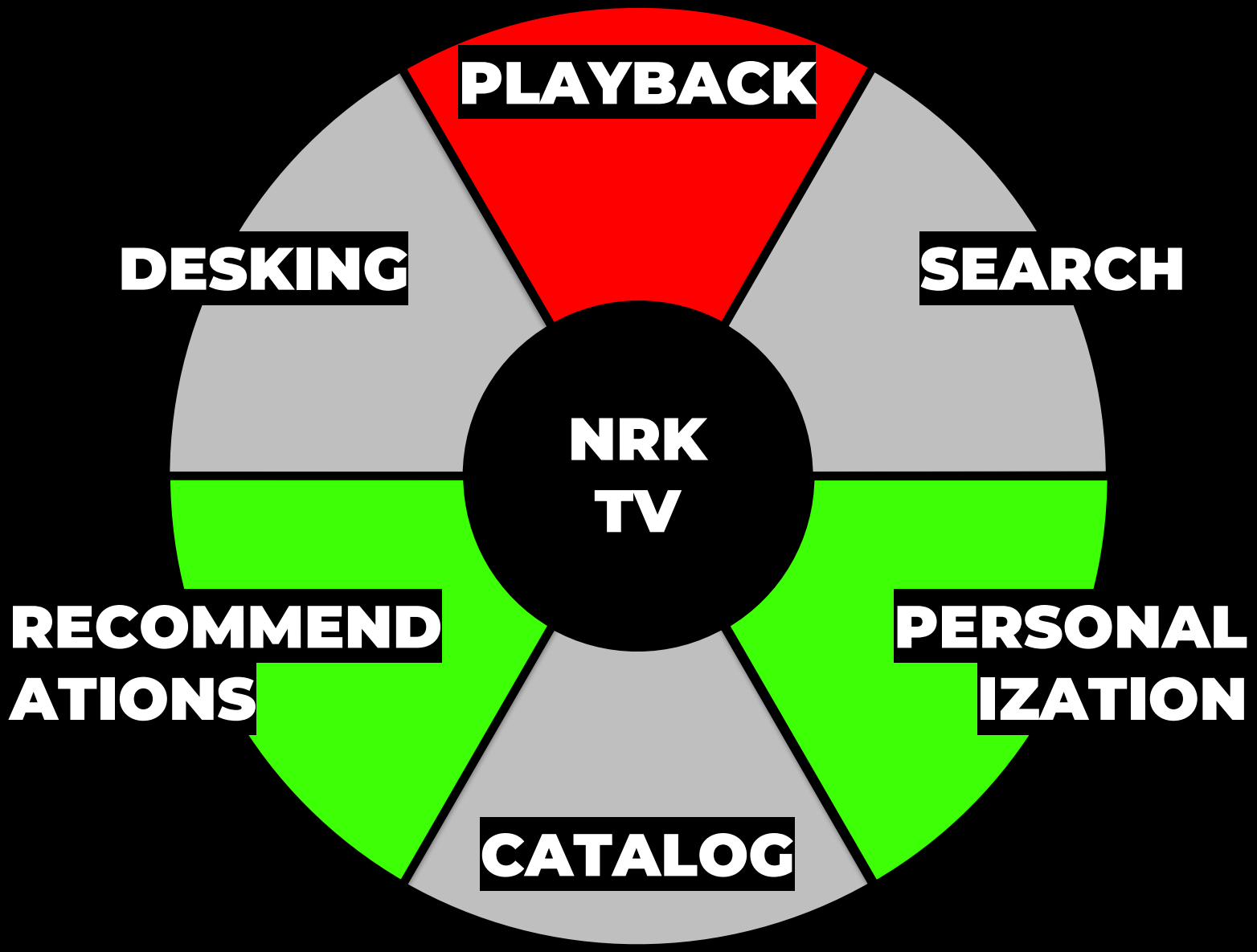
WE CAN BEGIN WORKING

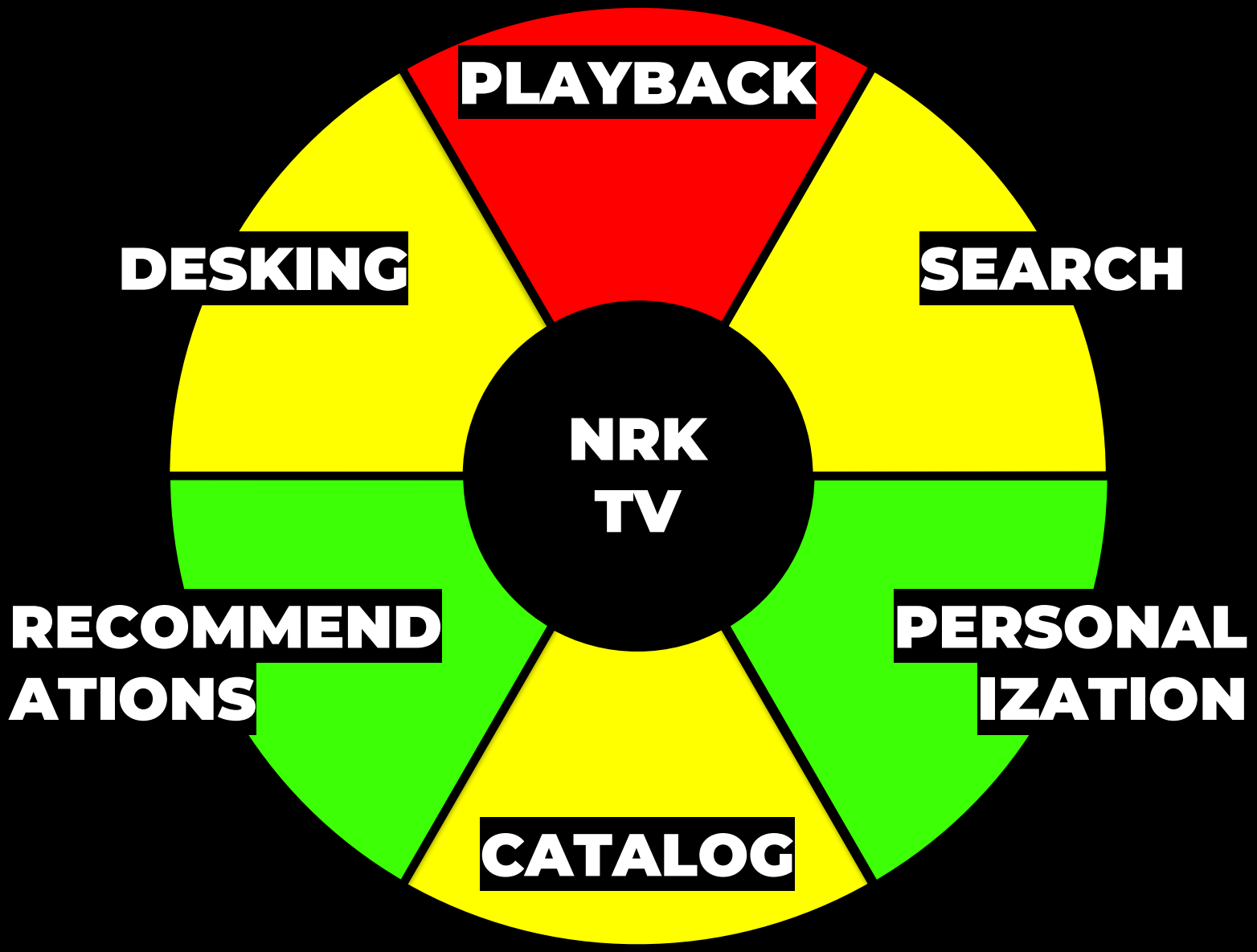
ARCHITECTING FOR RELIABILITY

RED AND GREEN SERVICES


PLAYBACK
DESKING
SEARCH
NRK
TV
RECOMMEND
ATIONS
PERSONAL
IZATION
CATALOG


PLAYBACK
DESKING
SEARCH
NRK
TV
RECOMMEND
ATIONS
PERSONAL
IZATION
CATALOG


PLAYBACK
DESKING
SEARCH
NRK
TV
RECOMMEND
ATIONS
PERSONAL
IZATION
CATALOG


PLAYBACK
SEARCH
PERSONAL IZATION
CATALOG
RECOMMEND ATIONS
DESKING
NRK TV

SEPARATE SLOS

SEPARATE RELIABILITY MEASURES

INDEPENDENT SERVICES

FAILURE ISOLATION

PARTIAL FAILURE

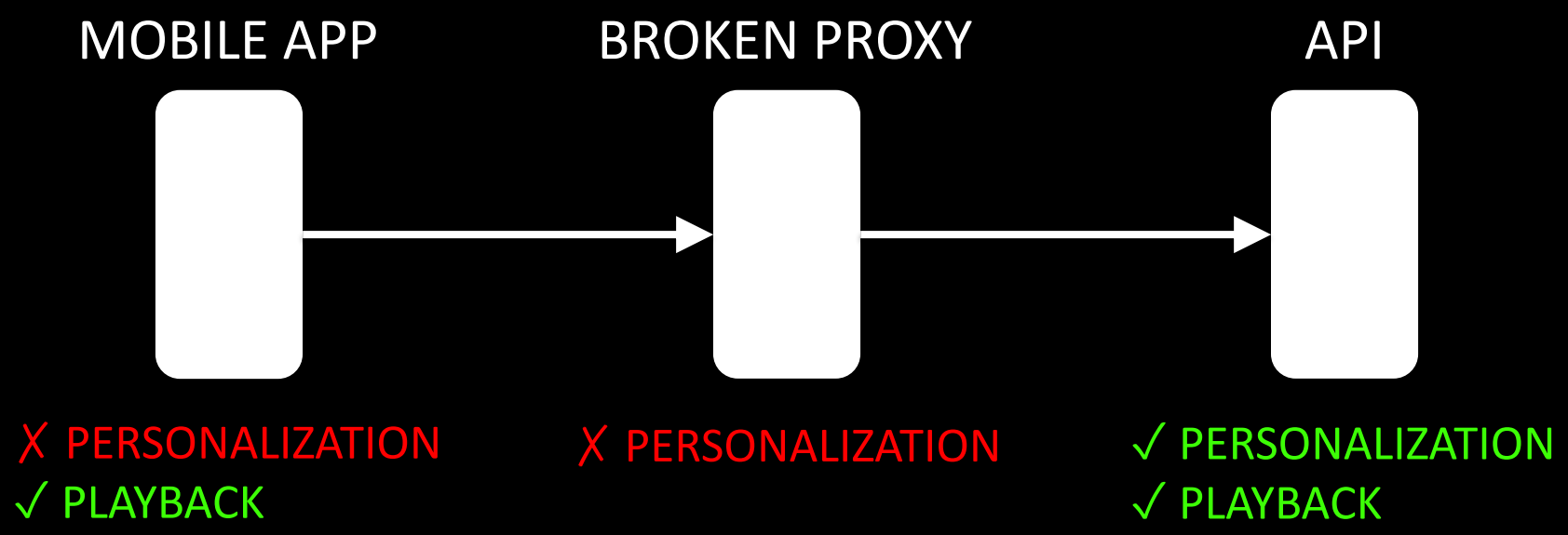
PARTIAL SUCCESS

GRACEFUL DEGRADATION

VALIDATION

BROKEN PROXY

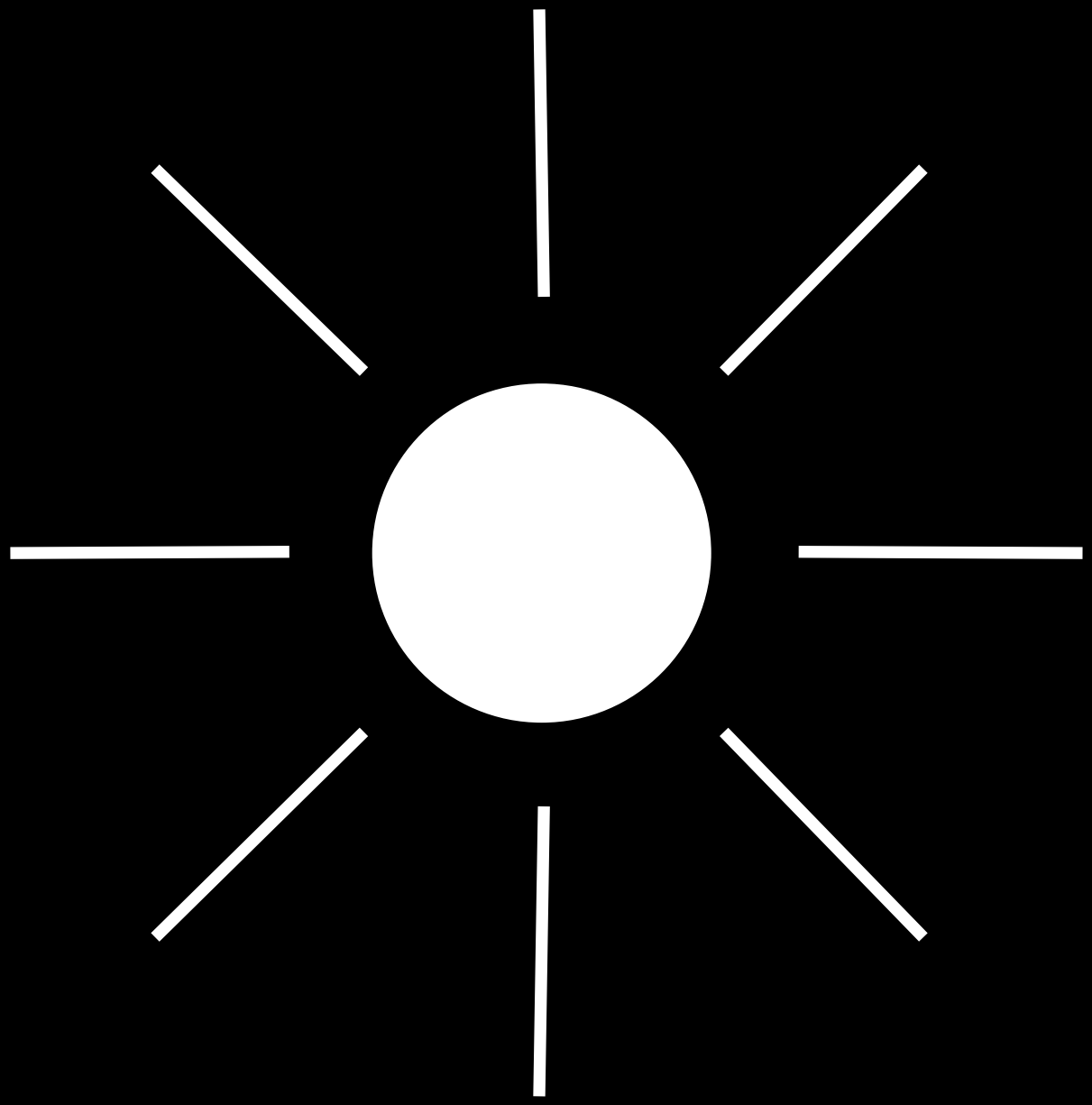

MOBILE APP
API
✓ PERSONALIZATION
✓ PLAYBACK
✓ PERSONALIZATION
✓ PLAYBACK


MOBILE APP
BROKEN PROXY
API
✗ PERSONALIZATION
✓ PLAYBACK
✗ PERSONALIZATION
✓ PERSONALIZATION
✓ PLAYBACK

THIS IS RESILIENCE

WHAT ABOUT ROBUSTNESS?

MEETING SLO



MEETING SLO

ON A BAD DAY

MEETING SLO
UNDER STRESS

EXPERIMENTS

STRESS TESTS

TOOLS

LOAD TESTING

RESILIENCE LIBRARY

FAULT INJECTION LIBRARY

FEATURE TOGGLING

PURE LOAD TESTS

VERIFY SCALABILITY

AUTOSCALE?

LOAD TESTS WITH INTRODUCED FAULTS

SIMULATE
FAILURE

WHAT CAN FAIL?

FAILING DEPENDENCY

HTTP CALL

DATABASE

SECONDARY FAILURES

EXHAUST LIMITED RESOURCES

HOW LONG DOES IT TAKE?

ASSUMPTION

FAST FAILURE

REALITY

SLOW FAILURE

STRATEGIES & REMEDIES

WHAT TO DO ABOUT FAILURE?

RELIABILITY PATTERNS

TIMEOUT

RETRY

CIRCUIT BREAKER

...

SOME QUESTIONS

DO WE NEED THE RESULT?

HOW BADLY DO WE NEED IT?

DEFAULT VALUE?

CACHED VALUE?

CACHING

HTTP CACHE

DESIGNING FOR CACHEABILITY

BÅRD'S

TIME EXPANSION ALGORITHM

BÅRD'S

TIME INDEPENDENCE PRINCIPLE

TIME-DEPENDENT TITLES

REFRESH TO SEE IF THE TITLE CHANGED?

THIS IS STUPID

TITLES CHANGE VERY RARELY

THIS IS COMPLETELY DETERMINISTIC

GENERATE A TABLE

FROM	TO	TITLE
03.12.2021 00:00:00	04.12.2021 00:00:00	I dag
04.12.2021 00:00:00	05.12.2021 00:00:00	I går
05.12.2021 00:00:00	08.12.2021 00:00:00	Fredag
		3. desember

TIME-DEPENDENT TITLES

NO LONGER INFLUENCE CACHEABILITY

WHAT TO DO ABOUT FAILURE?

HYPOTHESIS

VALIDATION

CASE

LINEAR CHANNEL PLAYBACK API

WHY IS THE LINEAR CHANNEL PLAYBACK API IMPORTANT?

LOAD TESTS WITH INJECTED FAILURES

INJECTING FAILURE WITH SIMMY

CONTROLLING FAILURE CONFIG WITH UNLEASH

Search

channel-blob-storage-inject-fault

Last minute

By name

0%

ps-playback-api-channel-blob-storage-inject-fault-always

Inject fault in all calls to blob storage for the channels.json blob using Simmy.

0%

ps-playback-api-channel-blob-storage-inject-fault-low-rate

Inject fault in calls to blob storage for the channels.json blob using Simmy (low injection rate).

0%

ps-playback-api-channel-blob-storage-inject-fault-moderate-rate

Inject fault in calls to blob storage for the channels.json blob using Simmy (moderate injection rate).

0%

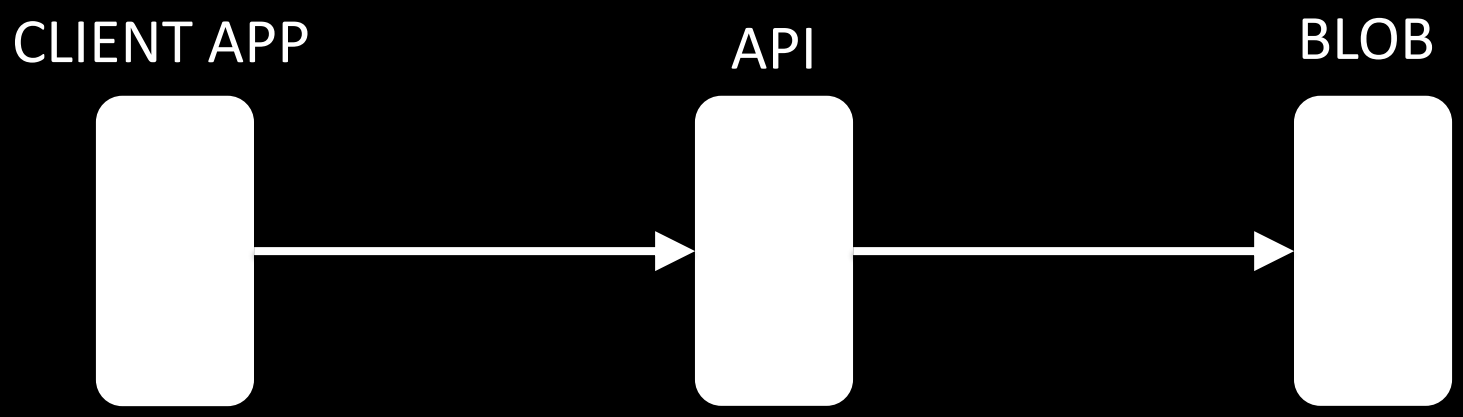
ps-playback-api-channel-blob-storage-inject-fault-severe-rate

Inject fault in calls to blob storage for the channels.json blob using Simmy (severe injection rate).

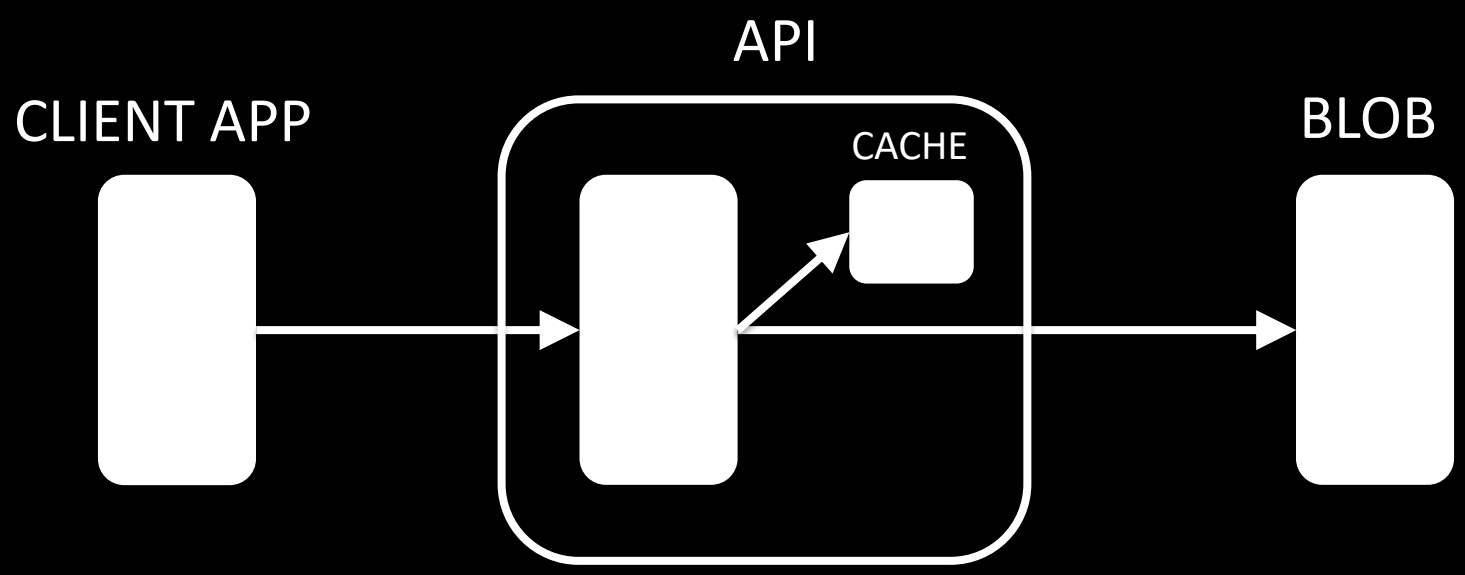
0%

ps-playback-api-channel-blob-storage-inject-fault-terrible-rate

Inject fault in calls to blob storage for the channels.json blob using Simmy (terrible injection rate).


CLIENT APP
API
BLOB

ORIGINAL DESIGN



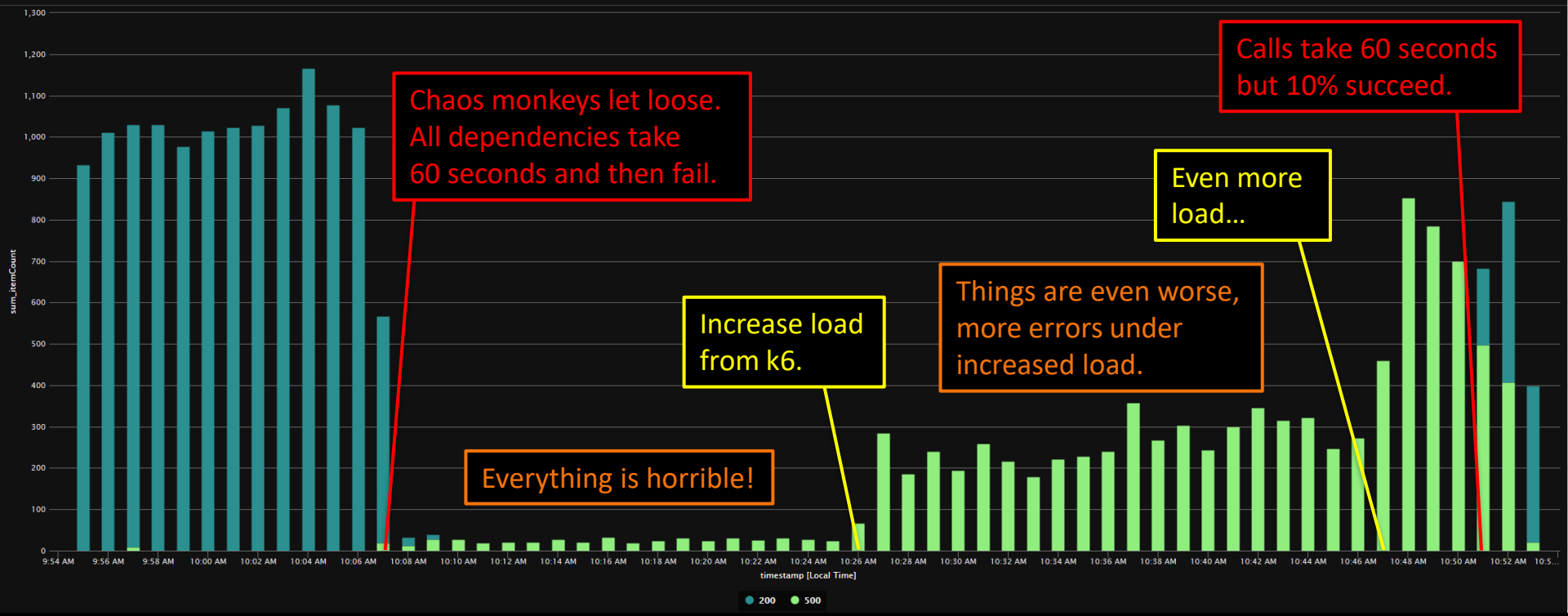
EXPERIMENT

100%	**ps-playback-api-channel-blob-storage-inject-fault-always** Inject fault in all calls to blob storage for the channels.json blob using Simmy.	default
0%	**ps-playback-api-channel-blob-storage-inject-fault-low-rate** Inject fault in calls to blob storage for the channels.json blob using Simmy (low injection rate).	default
0%	**ps-playback-api-channel-blob-storage-inject-fault-moderate-rate** Inject fault in calls to blob storage for the channels.json blob using Simmy (moderate injection rate).	default
0%	**ps-playback-api-channel-blob-storage-inject-fault-severe-rate** Inject fault in calls to blob storage for the channels.json blob using Simmy (severe injection rate).	default
0%	**ps-playback-api-channel-blob-storage-inject-fault-terrible-rate** Inject fault in calls to blob storage for the channels.json blob using Simmy (terrible injection rate).	default
100%	**ps-playback-api-channel-blob-storage-inject-latency-always** Inject latency in all calls to blob storage for the channels.json blob using Simmy.	default
0%	**ps-playback-api-channel-blob-storage-inject-latency-low-rate** Inject latency in calls to blob storage for the channels.json blob using Simmy (low injection rate).	default
0%	**ps-playback-api-channel-blob-storage-inject-latency-moderate-rate** Inject latency in calls to blob storage for the channels.json blob using Simmy (moderate injection rate).	default
0%	**ps-playback-api-channel-blob-storage-inject-latency-severe-rate** Inject latency in calls to blob storage for the channels.json blob using Simmy (severe injection rate).	default
0%	**ps-playback-api-channel-blob-storage-inject-latency-terrible-rate** Inject latency in calls to blob storage for the channels.json blob using Simmy (terrible injection rate).	default
100%	**ps-playback-api-channel-blob-storage-latency-long-duration** Any latency injected in calls to blob storage for the channels.json blob should be of long duration.	default
0%	**ps-playback-api-channel-blob-storage-latency-medium-duration** Any latency injected in calls to blob storage for the channels.json blob should be of medium duration.	default
0%	**ps-playback-api-channel-blob-storage-latency-short-duration** Any latency injected in calls to blob storage for the channels.json blob should be of short duration.	default

```
requests
| where customDimensions.applicationName == "ps-playback-api-channel"
| where cloud_RoleName == "ps-playback-api-channel-preprod-ne"
| where url contains "playback/metadata/channel" or url contains "playback/manifest/channel"
| summarize sum(itemCount) by bin(timestamp, 1m), resultCode
| render barchart
```

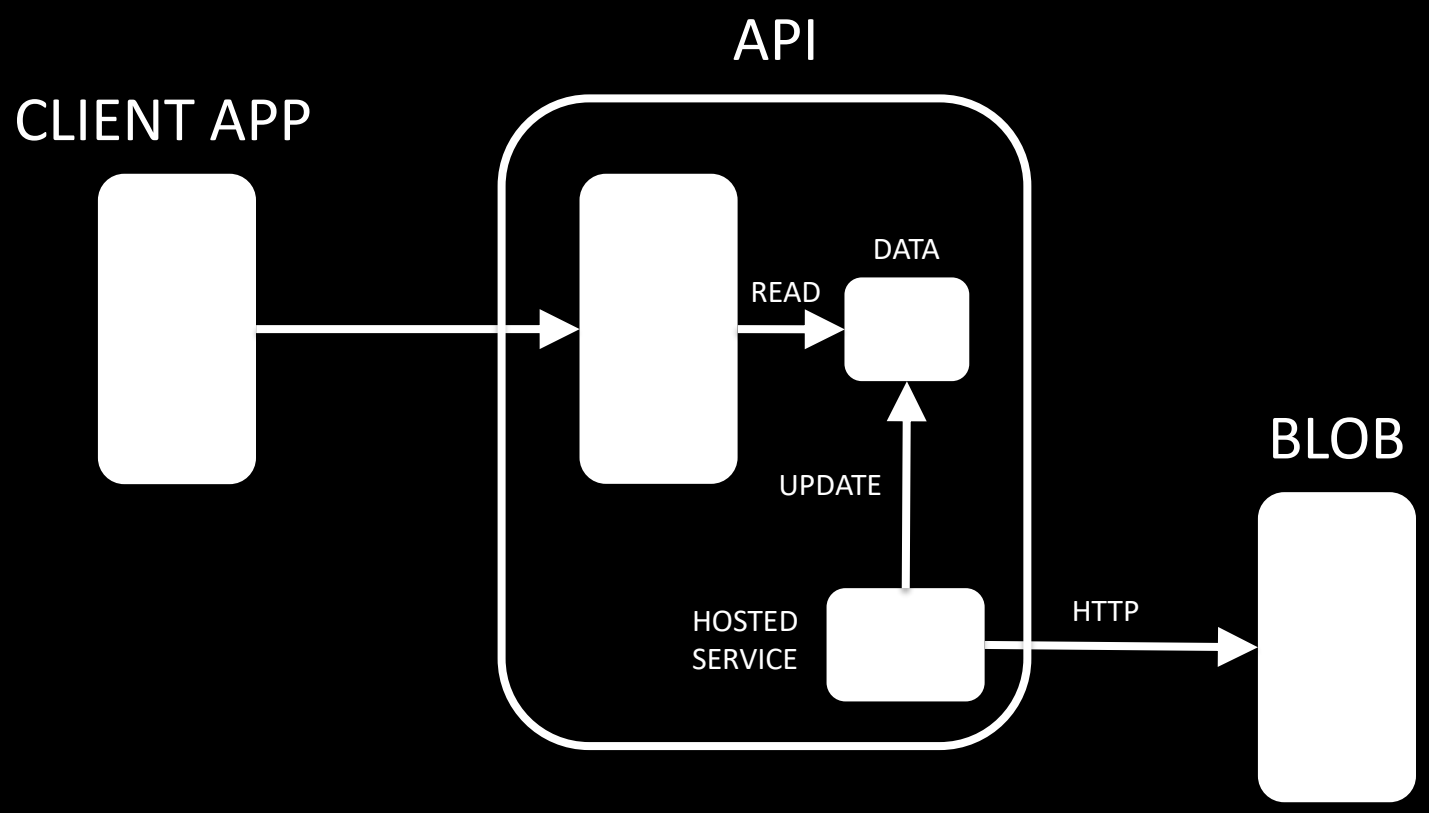
Results | Chart | Stacked Column | timestamp | sum_itemCount | resultCode | Sum | Display time (UTC+02:00)

Completed. Showing results from the last hour. 00:00:03.327 66 records



BUT STILL...

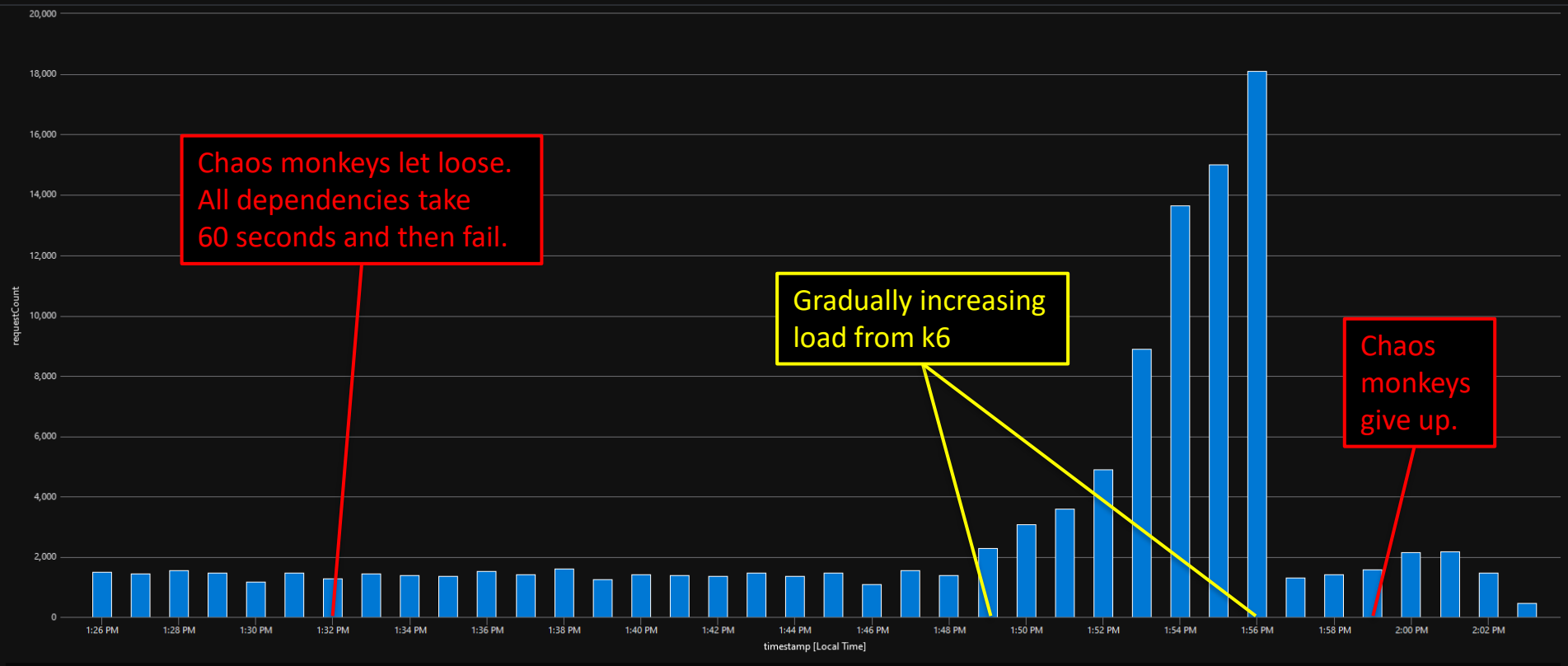
REVISED DESIGN



```
requests
| where timestamp > ago(40m)
| where customDimensions.applicationName == "ps-playback-api-channel"
| where cloud_RoleName == "ps-playback-api-channel-preprod-ne"
| where url contains "playback/metadata/channel" or url contains "playback/manifest/channel"
| summarize requestCount=sum(itemCount) by bin(timestamp, 1m), resultCode
| render columnchart
```

Results | Chart | Display time (UTC+01:00)

Completed 00:00.3 38 records



200

INJECTED FAULTS MADE IRRELEVANT

IS IT ROBUST?

TRICK QUESTION!

ROBUST ISN’T A BOOLEAN!

VERY ROBUST AGAINST BLOB DOWNLOAD PROBLEMS

NOT ROBUST AGAINST AZURE GOING DOWN

THE CASE STUDY IS A SIMPLE SCENARIO

MANY OTHER SCENARIOS ARE MUCH HARDER

TRADE-OFFS CAN BE MORE PAINFUL

ONLY AS GOOD AS THE EXPERIMENTS

«I HAVEN’T BEEN ABLE TO MAKE IT FAIL»

SUMMARY

RELIABILITY ENGINEERING

REQUIRES FEEDBACK

THE RIGHT MECHANISMS ARE CONTEXT-DEPENDENT

EXPLOIT CONTEXT TO FIND GOOD SOLUTIONS

ALWAYS VALIDATE

RUN EXPERIMENTS

LISTEN TO THE FEEDBACK

Monday Room 6 09:00 - 17:00 (UTC+01) 2 Days

Building and testing resilient services over HTTP

Users' patience with services not working is gradually decreasing as the quality of services online is improving. With current trends of moving services to the cloud and building smaller and network-intensive services, meeting these expectations can be challenging for us developers. We want to be able to build services that we can run confidently despite partial failures and outages.

Venue: Rebel .NET

In this workshop, you will learn how to simulate latency and failures in your web application and how to add strategies to deal with this. We will learn how changing the different parameters change the behavior of our application under load, and what trade-offs we ultimately must make. Because we build on HTTP and TCP/IP, we will also have fun sessions where we dig deeper into details in hands-on sessions to get a better understanding of the foundations on which we build.

Bjørn Einar Bjartnes

Bjørn Einar is a .NET developer and architect working at 4Subsea. He has a background from automation systems in the energy sector and has for the last 6 years worked for NRK TV as a backend developer and architect in the streaming service. His main topics of interest are domain driven design and functional programming. Keeping NRK TVs services up and running has been the main driver for his interest in resilient architecture. In the search for robustness, he has also learned to love HTTP.

Linkedin

Roger Hoem-Martinsen

Roger is a full-stack test developer with a varied background and a strong passion for building quality software solutions. In recent years, he has specialized in software testing. He started his career 15 years ago as a R&D engineer in signal processing, developing solutions for underwater communications. He has also worked with jet plane modems, gas detection systems, signal systems for railway, wireless condition monitoring systems and streaming services.

https://github.com/bjartnes/bounded-disturbances