Mobile Games are Living Organisms, Too

Armin Ronacher
Bruno Garcia

Armin Ronacher

Director of Engineering

Bruno Garcia

Engineering Manager

What do we do?

Application Performance Monitoring

We build a service (sentry) to monitor applications. In short: we tell you when your app runs slow or crashes.

In practical terms it means we **develop an SDK** to be embedded into an application and we **build and operate a service** that receives these crash and performance reports.

Agenda

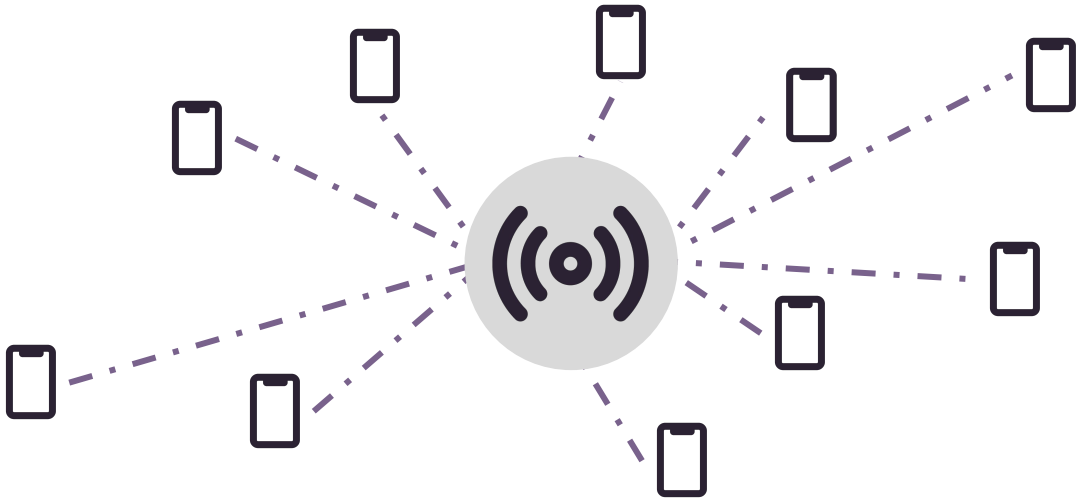
1. What we mean by mobile game
2. What this has to do with organisms
3. Stories from the Real World

Mobile Game

(in the context of this talk)

A Mobile Game

In the context of this talk a mobile game refers to any game **with a large installation base** of **independent devices** that are able to **communicate to external services** but can **operate independently** of these services.



In Concrete Terms

Large installation base:

Measured in terms of concurrent devices or users

Independent devices:

That means outside of the developer's general control

Communicate to External Services:

Devices check in with central services (config update, metric etc.)

Operate independently:

Game can run offline, does not require central services.

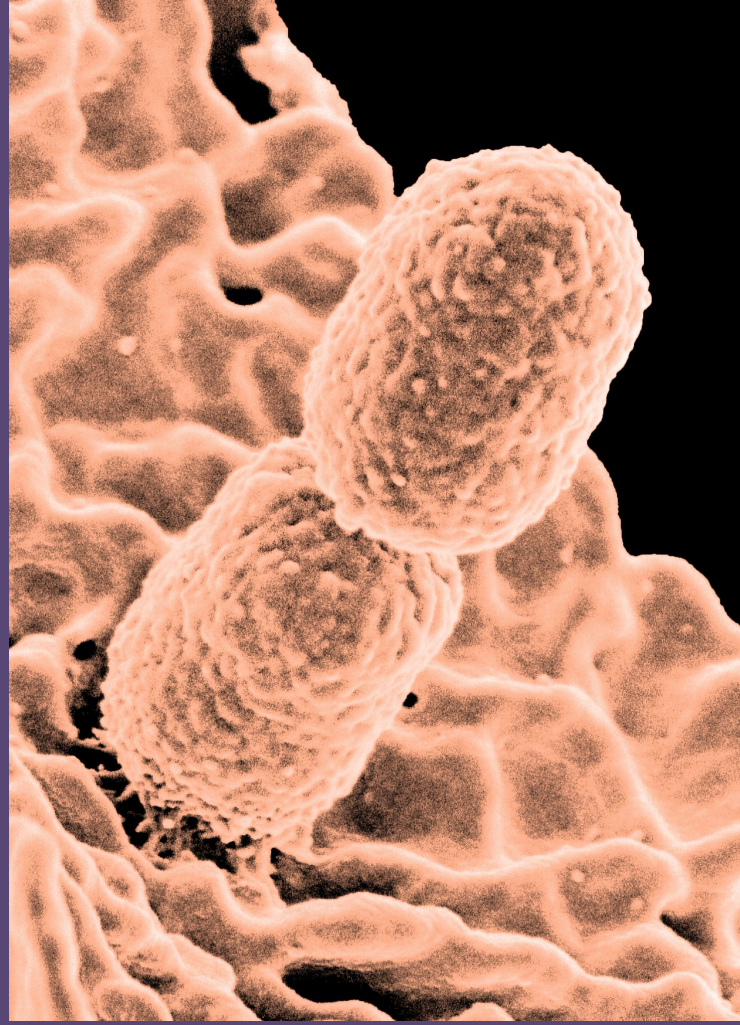
This also applies to

- Console and PC games with a large installation base
- A range of in-browser experiences

You are building a distributed system and you might not realize it

except you do not control your devices and some of the devices are awful

What does this have to do with organisms?



Emergent Behavior

When devices talk to central services they can indirectly influence each other.

The emergent behavior can be hard to understand.

Behavior is no longer deterministic.

Distributed Killswitch

- Once emergent behavior is in progress, controlling it can be tricky due to the distributed nature

- Old affected clients might not have a functioning kill-switch yet and are dependent on the server restoring the service to normal levels

Touchpoints

Service -> Device

1. Application on start loads a config packet from central service
2. Failure upon parsing -> crash on startup

Service -> Device -> Service

1. Application upon loading main menu fetches update file
2. Encounters network error: retries
3. Partial outage causes all active players in main menu to reload
4. Increases server load worsening the issue

Service -> Device -> Service -> Device

1. Application upon loading main menu fetches config file
2. Encounters network error: retries
3. Partial outage causes all active players in main menu to reload
4. Increases server load worsening the issue
5. Load balancer produces an error response
6. Game accidentally parses error response
7. Game persists broken config

Device -> Service -> Device

1. User inputs a null byte into a profile info
2. Other players joining into that player's game crash

Real World Stuff

The Distributed Queue

A few years ago, in Sentry SDK Land

1. Sentry iOS SDK installs signal handler
2. Upon crash persist crash file to “disk”
3. Upon reload, try to send crash. If successful -> delete, if failed: keep

what’s the problem?

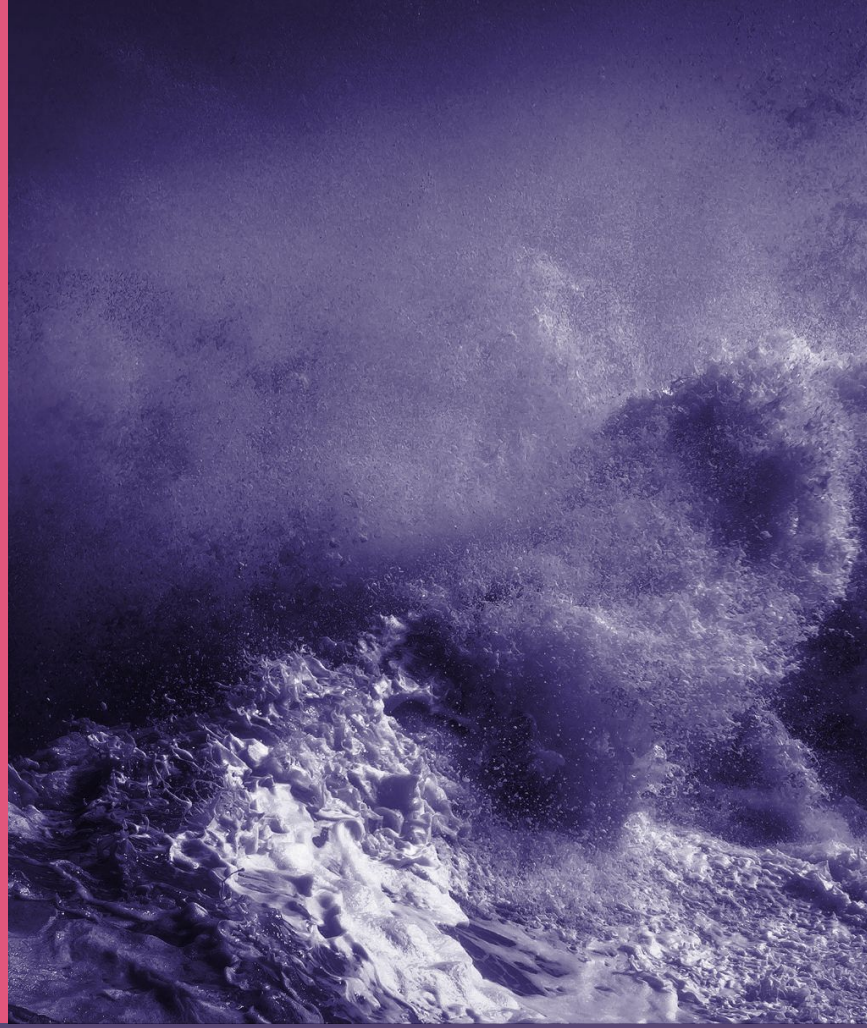
A slow death

- Crash reports sent from oldest to newest
- Failures are not dropped
- As time goes on, some customers only get **weeks old crash reports**
- Fleet of devices are caching **older and older crashes**
- Exaggerated by running into **rate limits** upon submission

Updates take time

- We patched the client to delete old reports, but it takes time to update
- Changed the server to lie about accepting reports for mobile clients
- Within a few days the backlog of ancient crash reports cleared

The Storm



Abuse Protection vs Back-Pressure Control

When clients send too much data, our ingestion system replies with 429

Includes Retry-After header that tells SDKs for how long they must slow down

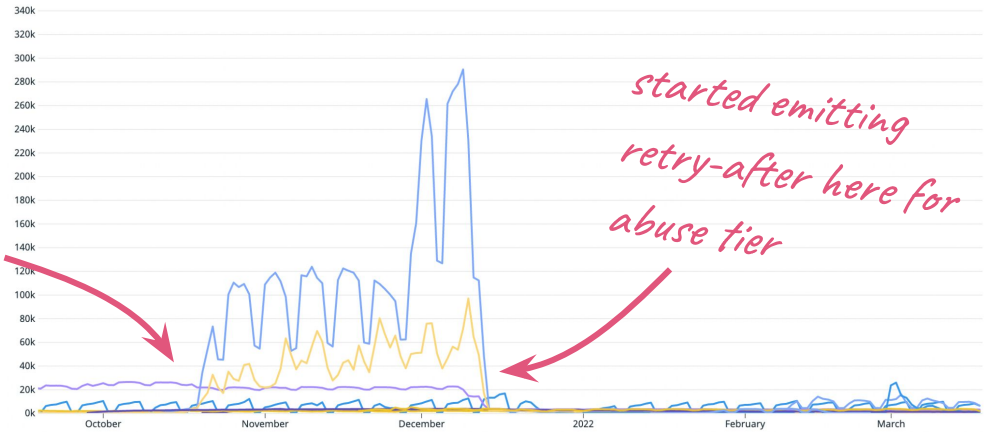
Our global load balancer has an abuse protection

A chrome extension started pushing through the abuse level

When Abuse Protection Worsens

Abuse limit protection did not reply with 429

As traffic from an abuse project was creeping up, the moment it crashed through the abuse limit the Retry-After header disappeared and the traffic was going up faster and faster



Perspective

Our protection pushed one single project into sending a **magnitude more traffic** past the point of rejection than the rest of all projects combined.

Exaggerated by this SDK not having a fallback retry-after compared to other SDKs.

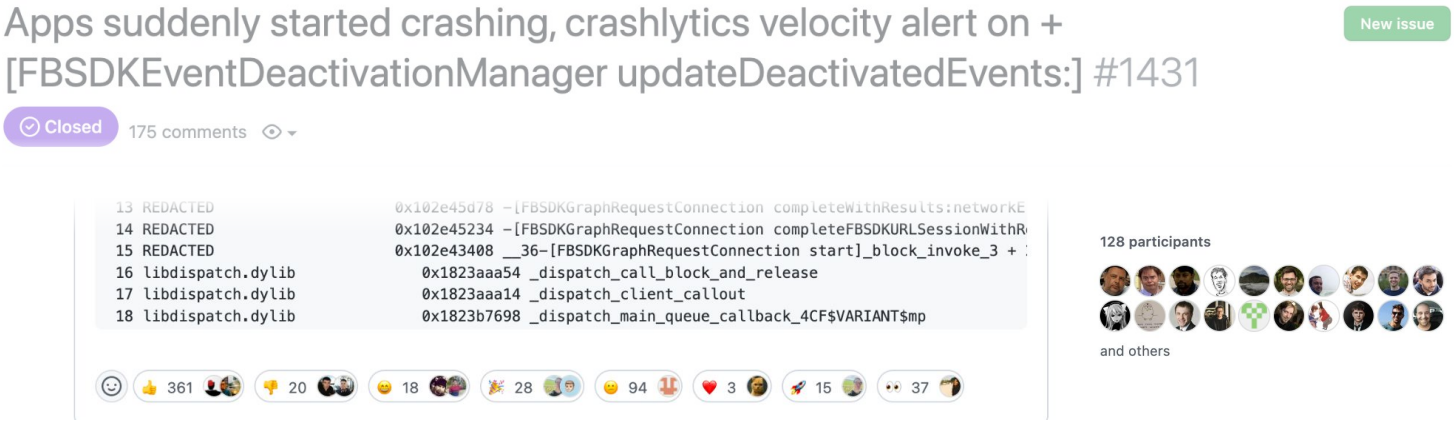
The Outage

Centralized User Frustration

1. Facebook Authentication library fetches config from server
2. Server serves malformed config
3. Applications using this library crash globally

Knock-on Effects

1. Lots of confused users take to support of apps using the library
2. Lots of crash reports from different apps come to crash reporting services
3. Many user restarts of apps increase traffic to services queried on startup



UGC of Death

User Generated Content Can be Dangerous

Profile info is frequently denormalized, cached and synched to other players.

Format strings are messy and easy to misuse.

A user with manipulated information that is synchronized to other players can cause crashes.

Left 4 Dead Corrupted Sprays

Example: in Left 4 Dead users can use custom sprays (images) that are placed as textures.

When other players encounter such sprays their game client crashes.

The act of sharing a game session with such a griefing player can cause disruption.

Towards Replication

Replication is worse. Replication happens when the questionable payload can spread between game sessions.

This can even happen for bugs: World of Warcraft's Corrupted Blood incident.

A status change was able to spread out of a restricted game session via player pets. Caused the status effect to spread like a virus. **Required multiple fixes and a month to fully remove**.

Weird Corruption

The Fleet is Bizarre

The larger the deployment, the more weird devices show up.

- Completely wrong on-device clocks
- Bizarre memory corruption
- Malware interfering

Gibberish breaks Product Features

- A string identifying the release was randomly corrupted
- Created garbage release identifiers on the server

Outline

- 429 backoff vs queue (backpressure, load shedding)
  - Offline caching + retry - Sentry 2017 SDK bug?
  - Behavior caused by 429 in general
- Facebook Sign-In
- “Infected” but non replicating player
- Worms
- State machine stuck, JSON bad, refresh JSON every frame
- Android is its own organism. Due to fragmentation and custom builds
  - Thinking about those garbage data from China. Hacks in games that break the apks. Rooted devices

Questions?