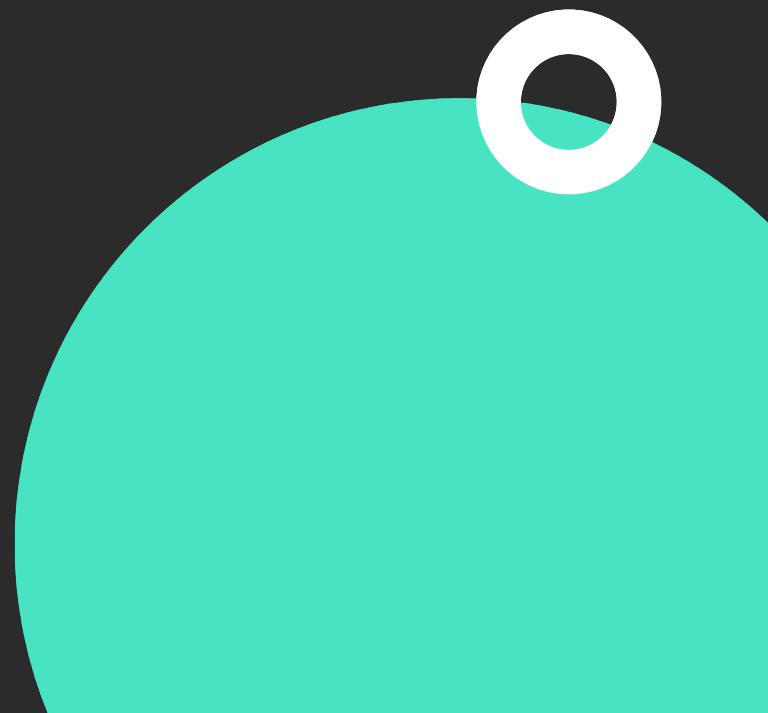




Fixing common causes of ANRs in Android



Jamie Lynch
Software Engineer at Embrace
 @fractalwrench
 in/fractalwrench



Codelab Overview

- How does Android trigger ANRs in apps?
- How can you identify and fix ANRs in a real-world app?
- How can you fix ANRs in production?





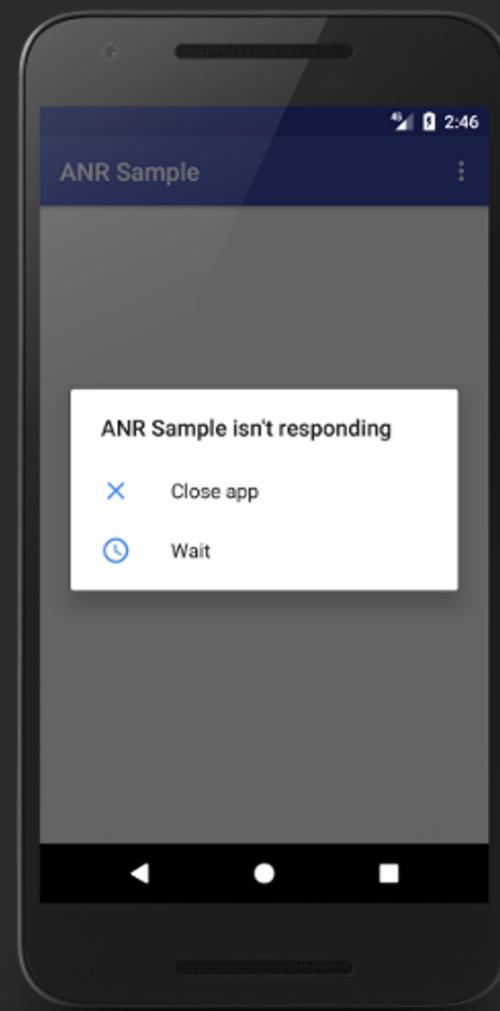
How does Android trigger ANRs in apps?

Up Next



How does Android trigger ANRs in apps?

- ANRs happen when the App is Not Responding (to the system).
- ‘Not Responding’ means the Android OS waited too long for the app to perform a task.
- Usually, ANRs are triggered by main thread blockages.
 - But this is not always the case!

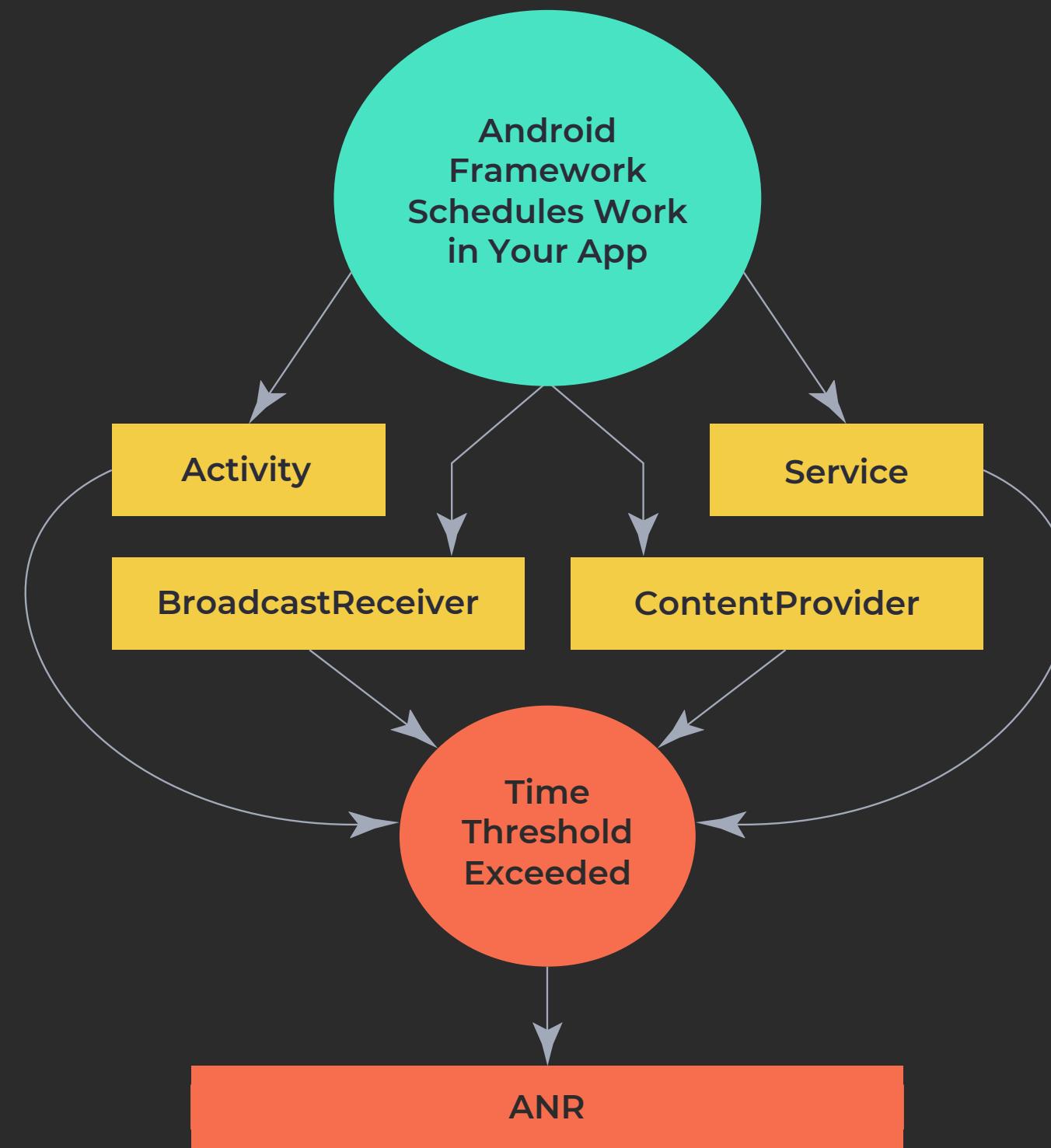


How does Android trigger ANRs in apps?

- The Android OS schedules work in all its components, and ANRs can occur in any component that cannot complete work within a set amount of time.

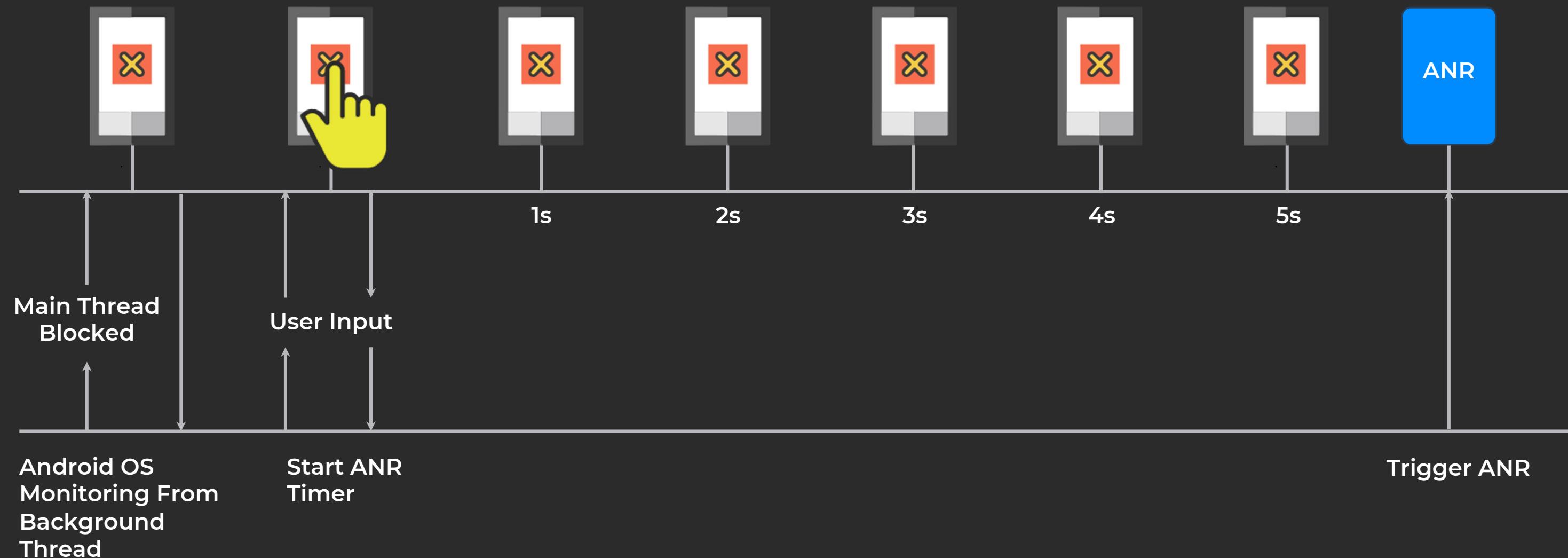
- Android components:

- Activity
- Service
- BroadcastReceiver
- ContentProvider



How does Android trigger ANRs in apps?

- The most common ANR is when input dispatch takes >5s to process events.
- Android has one UI/main thread, so performing non-UI work can block the UI.
- This is a bad user experience because a user might click a button, wait for 5s, then see nothing change in the app.



How does Android trigger ANRs in apps?

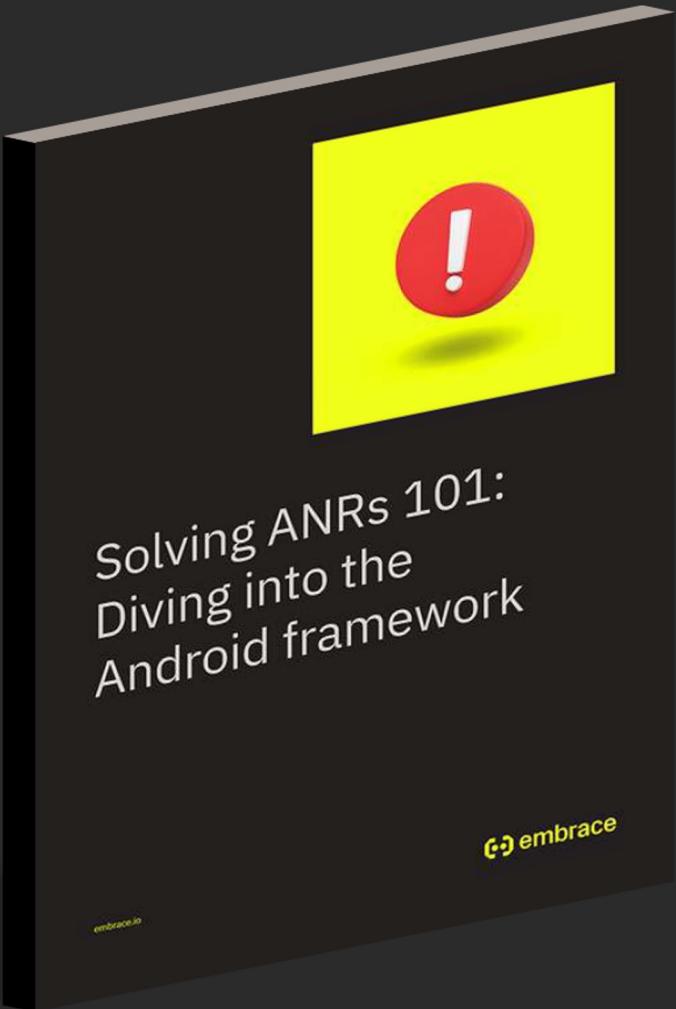
- The input dispatch queue being blocked is the most common (and user-facing) problem.
- However, there are other ways to trigger ANRs:
 - `BroadcastReceiver` taking >10s in `onReceive()`
 - `Service` taking >20s to start
 - Improper use of system APIs
- **Whenever the Android OS schedules work in an Activity/Service/BroadcastReceiver/ContentProvider in your app and it takes too long, you risk triggering an ANR.**



How does Android trigger ANRs in apps?

- There are dozens of places where an ANR can be triggered.
- We've covered the key ones here.
- If you're interested in the full story, check out Embrace's eBook, "Solving ANRs 101".

<https://get.embrace.io/solving-anrs-101/>





How can you identify and fix ANRs in a real-world app?

Up Next



How can you identify and fix ANRs in a real-world app?

- Let's identify performance problems that cause ANRs in a real app.
- We'll open the code repo and solve the following problems:
 - The main thread being blocked for >5s by one operation
 - The main thread being blocked for >5s by many operations
 - Deadlock between the main thread and another thread
 - The main thread waiting synchronously for a Future
 - A `BroadcastReceiver` with a slow `onReceive()` call
 - A foreground service that didn't call `onForeground()`



Main thread blocked >5s by one operation

- If the main thread is blocked for >5s while an input event is in the input dispatch queue, an ANR will be triggered.
- This can often be caused by one long operation, such as writing a very large file to disk.

Now let's go to the code in the repo and fix the problem!

```
fun downloadBinaryResources():  
    ByteArray {  
    Thread.sleep(10000)  
    return ByteArray(0)  
}
```



Main thread blocked >5s by many operations

- Lots of small tasks add up over time, and these can breach the 5s threshold.
- Here are common examples of operations that cause performance problems:
 - Loading 3rd party SDKs
 - I/O on the main thread
 - JSON deserialization
 - Waiting on other threads

Now let's go to the code in the repo and fix the problem!

```
private fun performWork() {  
    val aJson = readFromDisk("a.json")  
    val a = deserializeJson(aJson)  
  
    val bJson = readFromDisk("b.json")  
    val b = deserializeJson(bJson)  
  
    val cJson = readFromDisk("c.json")  
    val c = deserializeJson(cJson)  
  
    val dJson = readFromDisk("d.json")  
    val d = deserializeJson(dJson)  
}
```



Deadlock between main thread and another thread

- An ANR can be triggered if the main thread and another thread are both waiting to acquire a lock (or similar shared resource).

Now let's go to the code in the repo and fix the problem!

```
fun triggerDeadlock() {  
    backgroundExecutor.submit {  
        writeToFileWithLock("Wrote to file from  
background thread")  
  
        handler.post {  
            writeToFileWithLock("Wrote to file  
from main thread")  
        }  
    }  
  
    fun writeToFileWithLock(contents: String) {  
        lock.lock()  
        // forget to release the lock  
    }  
}
```



Main thread waiting synchronously for a Future

- Waiting on a background thread to finish can block the main thread.
- A good example of this is calling `Future.get()`.

Now let's go to the code in the repo and fix the problem!

```
private fun performWork() {  
    val client = NetworkApiClient()  
    val imageFuture = client.downloadImage()  
    val image = imageFuture.get()  
}  
  
private class NetworkApiClient {  
    fun downloadImage(): Future<*> {  
        return executor.submit(Callable {  
            // simulate network requests  
            Thread.sleep(10000)  
            ByteArray(0)  
        })  
    }  
}
```



A BroadcastReceiver with a slow onReceive() call

- If a BroadcastReceiver takes longer than 10s in `onReceive()`, an ANR will be triggered.
- This happens regardless of whether anything is in the input dispatch queue.

Now let's go to the code in the repo and fix the problem!

```
override fun onReceive(  
    context: Context?,  
    intent: Intent?  
) {  
    performLongOperation()  
}  
  
private fun performLongOperation() {  
    Thread.sleep(12000)  
}
```



A foreground service that didn't call onForeground()

- If a foreground service doesn't call `onForeground()` within 10s, an ANR will be triggered.
- A crash will also be triggered.

Now let's go to the code in the repo and fix the problem!

```
class AnrForegroundService :  
Service() {  
  
    override fun onCreate() {  
        super.onCreate()  
    }  
}
```





How can you fix ANRs in production?

Up Next



How can you fix ANRs in production?

- ANRs are hard to fix.
- The Android OS can trigger ANRs from dozens of different places, with different rules.
- Real apps can have several activities/services/broadcasts all performing work at the same time.
- ANRs are probabilistic - they are less likely to happen on higher-end devices that devs tend to use.
- Most solutions focus on recording a single stack trace at one point in time.

The question becomes: **How can you fix ANRs in production apps?**





Helping engineers
•• build better
mobile experiences



See all the details, clearly

Full user session play-by-play

Start Time	Type	Details
	(Foreground)	< Sessions 2 of 5 >
> 22:22:51	Network	200 OK POST data.emb-api.com (322ms) data.emb-api.com/v1/log/events
22:22:52	Device	Low memory warning
22:22:53	Enter View	com.enflick.android.ShopAround.activities.phoneNumberSelection.PhoneNumberAreaCodeFragment (3s 423ms)
22:23:00	Tap	com.enflick.android.tn2ndLine:id/next_button (60, 1572)
> 22:23:01	Moment	StartTime - Normal (224ms) Less than 5s to complete
> 22:23:01	Network	200 OK POST data.emb-api.com (314ms) data.emb-api.com/v1/log/events
22:23:01	Enter View	com.enflick.android.ShopAround.activities.MainActivity (28s)
> 22:23:02	Network	200 OK POST data.emb-api.com (296ms) data.emb-api.com/v1/log/events
22:23:02	Breadcrumb	BatteryLevel: 0.33
22:23:02	ANR Interval	App entered ANR interval (1s 548ms)
22:23:30	End	App ended in crash Stack Trace SIGTRAP Trace/breakpoint trap
		0 libmonochrome.so 0xcf760000 + 29940852
		1 libmonochrome.so

Auto-collect everything to find true root causes

- User ID look-up (PII compliant)
- OOTB data capture
- Easy integrations
- Seamless pivot to exact line of code and stack trace or see high level user impact



Surface the issues that matter

High-fidelity, structured, and correlated data that helps you focus

Crashes List										
Score	Events	Sessions	Devices	Users	Description	App State	First Seen	Last Seen	Versions	Status
High	405	0.07%	146	0.11%	getenv /system/lib/arm/nb/...	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Snoozed
High	405	0.07%	146	0.11%	getenv com.xhy.lila.ui.me.Co...	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Open
Medium	405	0.07%	146	0.11%	java.lang.NullPointerException com.clevetap.android...	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Ignored
Low	405	0.07%	146	0.11%	Java_io_agora_rtm_jni_Agora... com.xhy.lila.ui.explore...	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Resolved
Low	405	0.07%	146	0.11%	android:TableOfContentThread... libstagefight.so	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Open
Low	405	0.07%	146	0.11%	getenv /system/lib/arm/nb/li...	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Snoozed
Low	405	0.07%	146	0.11%	getenv com.xhy.lila.ui.me.Coi...	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Open
Low	405	0.07%	146	0.11%	java.lang.NullPointerException com.clevetap.android...	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Ignored
Low	405	0.07%	146	0.11%	Java_io_agora_rtm_jni_Agora... com.xhy.lila.ui.explore...	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Resolved
Low	405	0.07%	146	0.11%	android:TableOfContentThread... libstagefight.so	<div style="width: 100%; background-color: #4CAF50; height: 10px;"></div>	Nov 9, 2020	Nov 9, 2020	7.0.0 - 7.0.4	Open

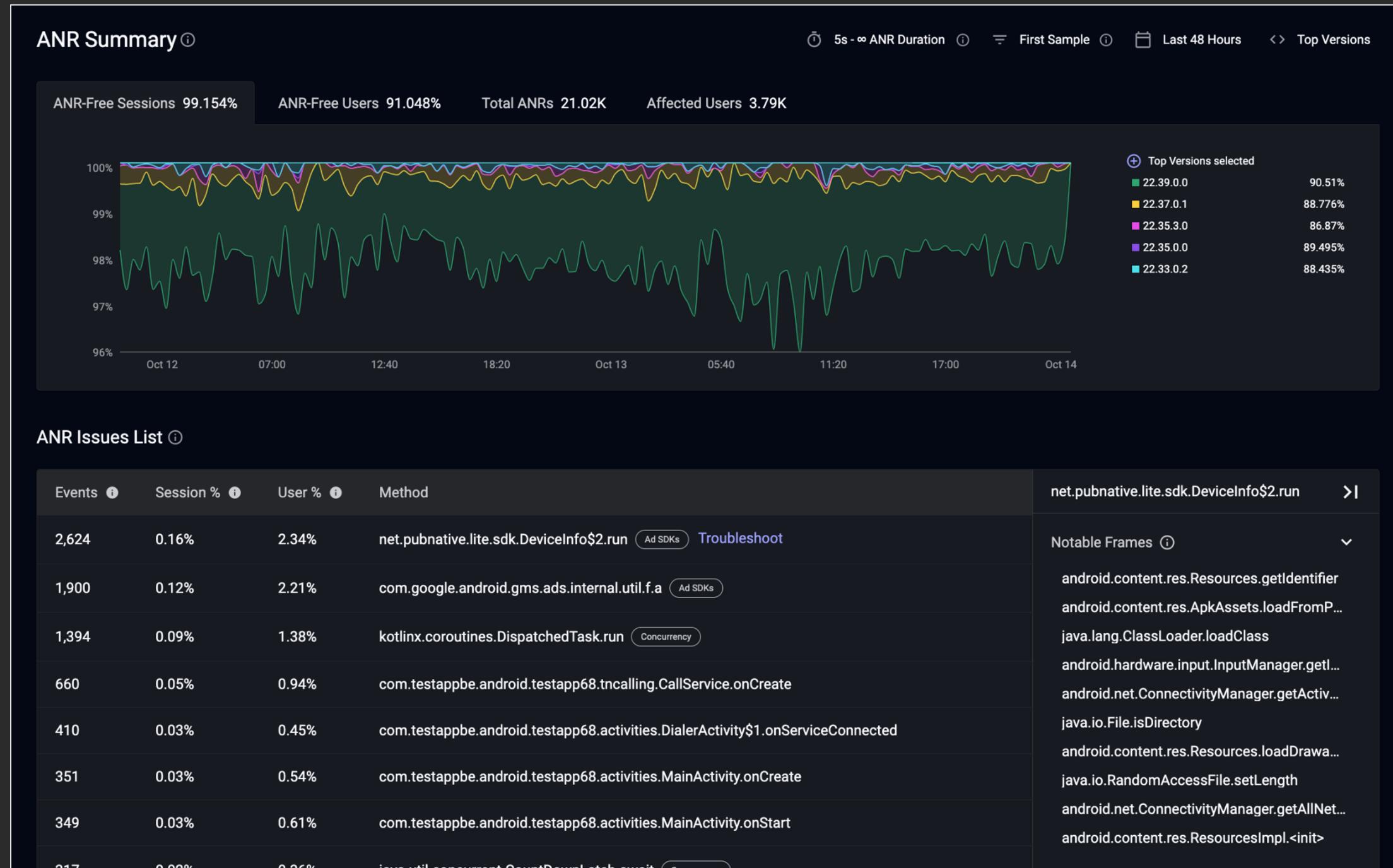
Confidently pinpoint where to focus first with a high-fidelity, structured, and correlated view.

- ANR groupings using common problematic methods
- Crash groupings based on most relevant stack frame
- ML-powered crash correlations and scoring
- Alerting and tracing for revenue impacting sessions



Optimize beyond crashes

See every non-fatal issue to perfect your user's experience



**99% crash-free isn't enough.
Achieve flawless performance
with deep visibility into non-
fatal errors.**

- ANR stack traces captured at 1s and every 100ms thereafter
- Failing network call insights in every session timeline
- Error tracking for handled exceptions, info logs and warnings
- View app exits and OOM insights that signal poor experiences

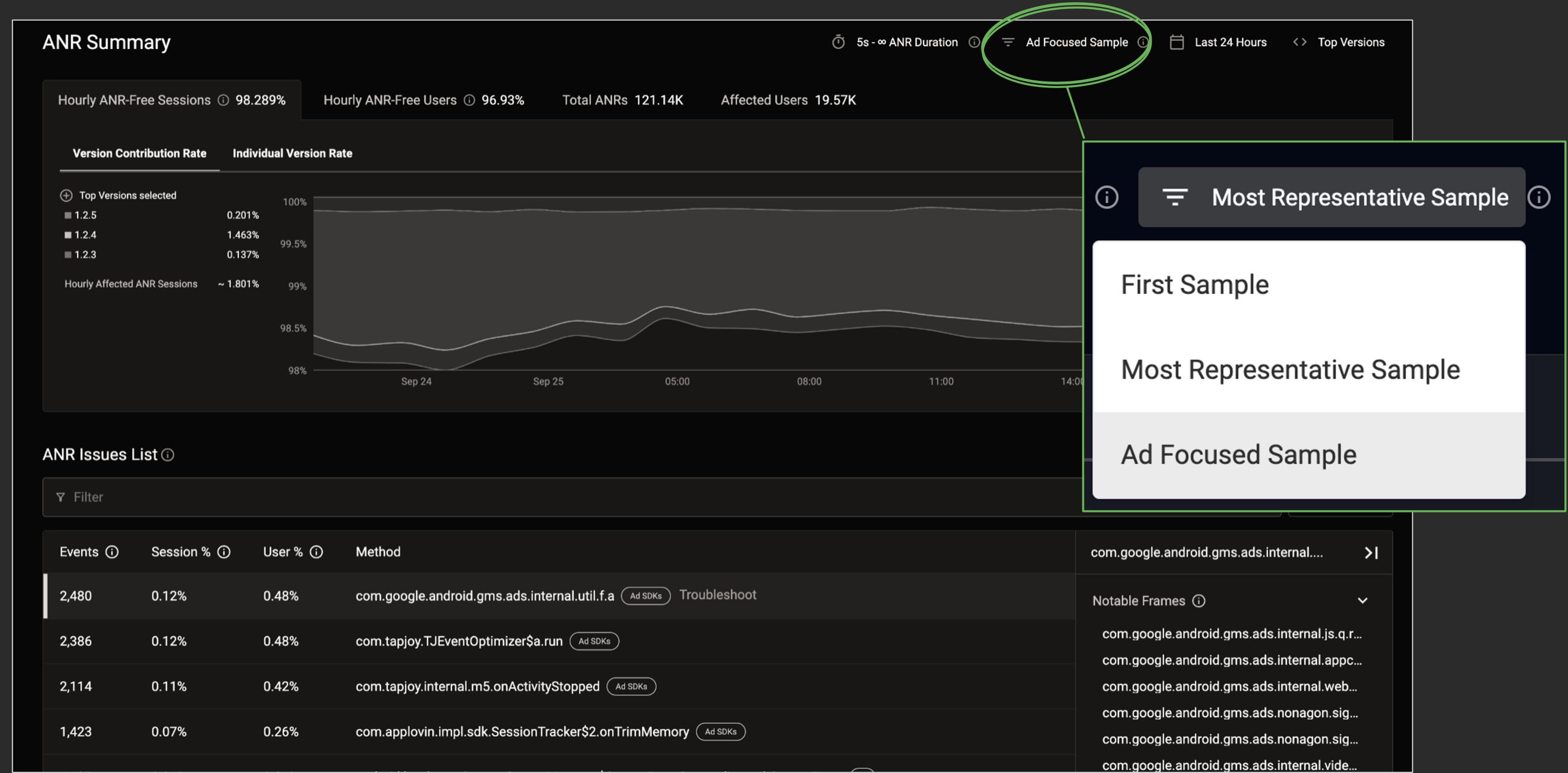


Embrace captures multiple ANR samples

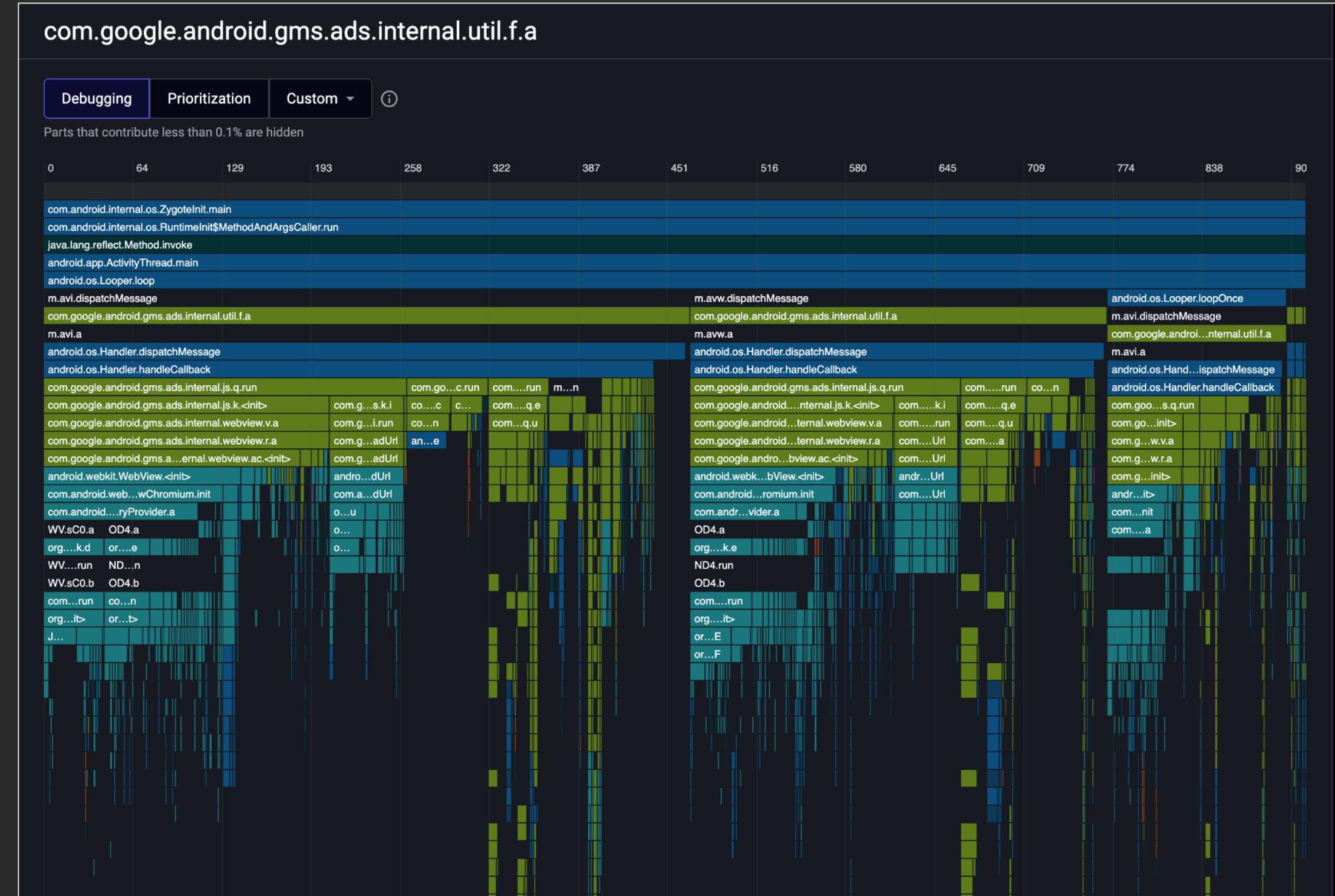
23:16:02	Breadcrumb	Interstitial not ready event
> 23:16:08	Network	200 OK POST some.adnetwork.com (114ms) some.adnetwork.com/fetch
> 23:16:36	Network	200 OK POST my.domain.com (586ms) my.domain.com/event
▼ 23:16:37	ANR Interval	ANR interval App entered ANR interval (5s) Started 23:16:37 Ended 23:16:43 Duration 5s
	Relevant Methods in Samples	Samples
	java.util.concurrent.Semaphore.tryAcquire	39 Concurrency
	java.util.concurrent.locks.AbstractQueuedSynchro...	39 Concurrency
	java.util.concurrent.locks.AbstractQueuedSynchro...	39 Concurrency
	java.util.concurrent.locks.LockSupport.parkNanos	39 Concurrency



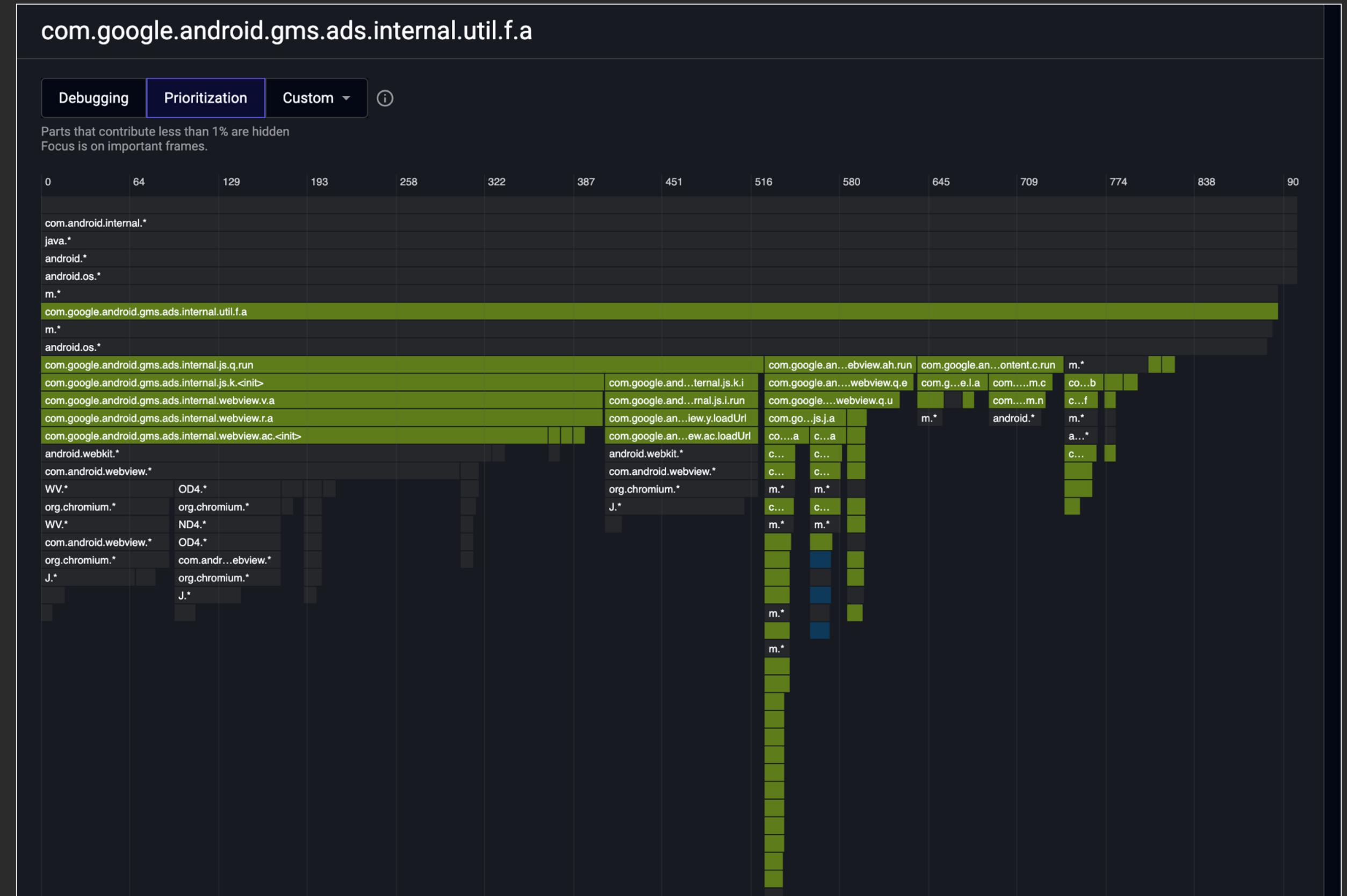
Embrace provides multiple views of the data



Embrace stitches stack traces into flame graphs



Embrace eliminates noise to help you prioritize



Time for a demo!



Summary

- There are dozens of places the Android OS triggers ANRs, and usually the main thread is blocked.
- There are various ways that the main thread could be blocked by performance problems in your app.
- Embrace shows the whole story for production ANRs by showing everything that happened in flame graphs.
- We covered several approaches for writing code that prevents ANRs in your app.

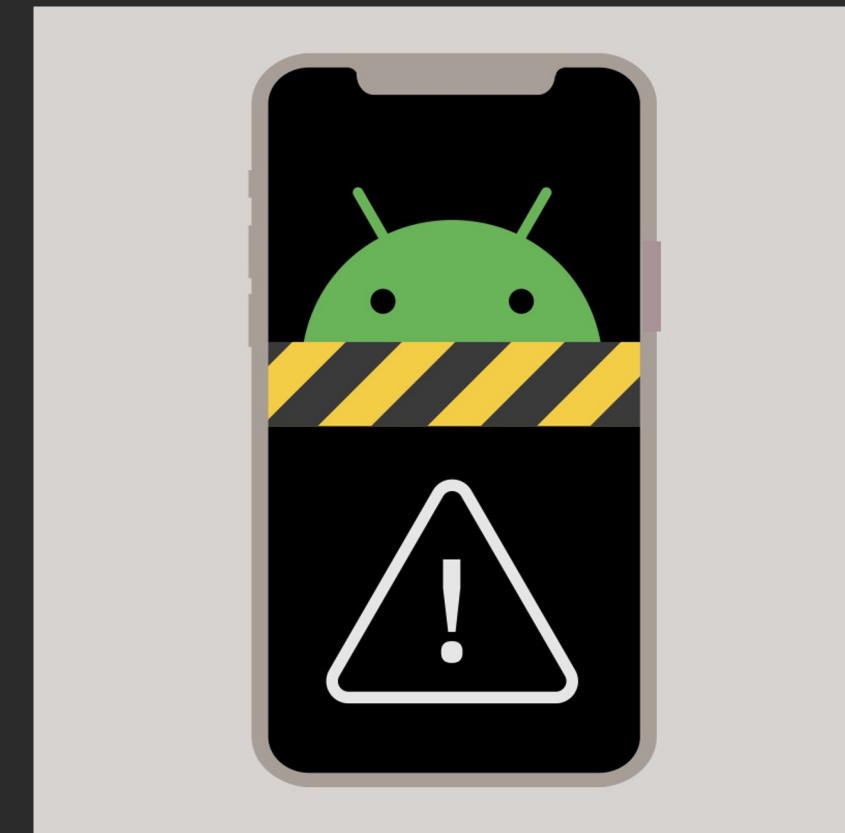


Learn more



Jamie Lynch
Software Engineer at
Embrace

jamie.lynch@embrace.io



www.embrace.io