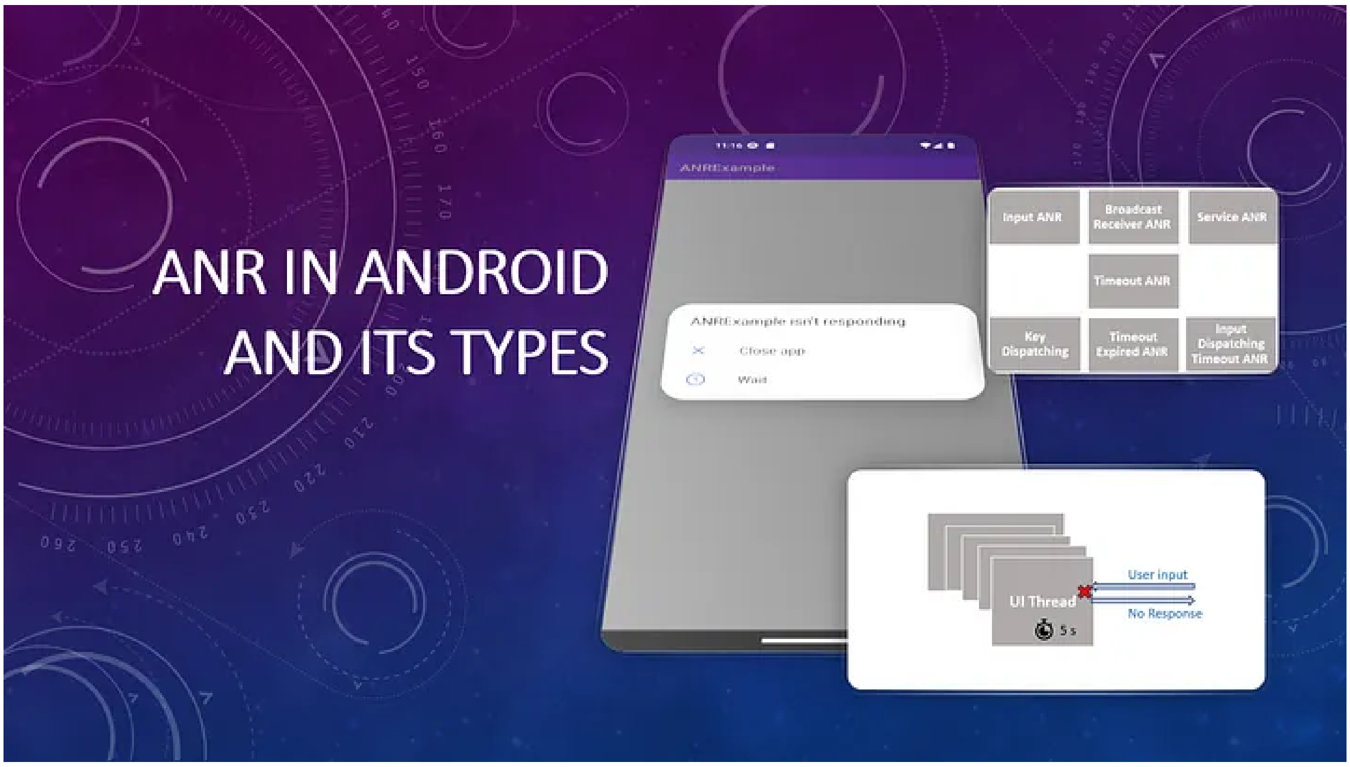
ANR in Android & its types

What is ANR in Android & different types of ANR issues can occur in the Android Sub-System

Narendra Harny · Follow

Published in Make Android

7 min read · Dec 17, 2023

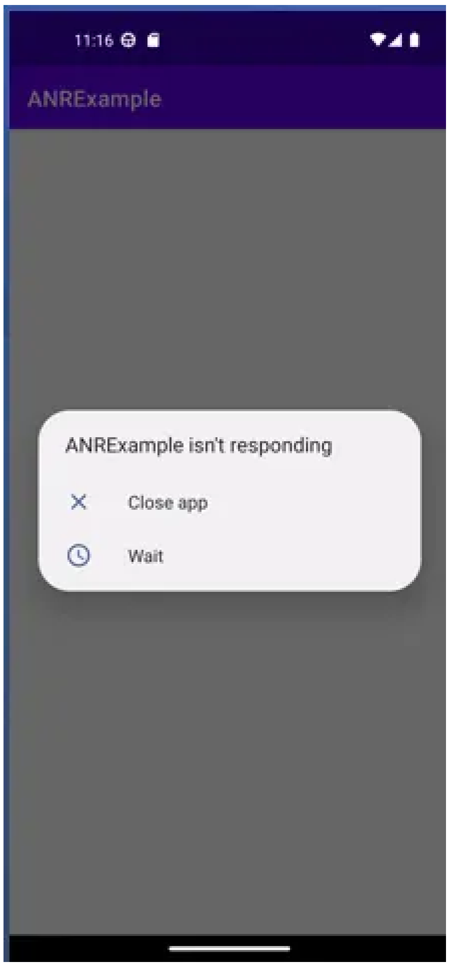


Android Not Responding

Android Sub-system is vast in terms of concepts, One aspect of understanding Android OS is understanding the Framework of Android like Media, radio, IME, Managers, Service and all. ANR is one of the subjects which is very important to understand to create a seamless user experience for Android Users.

What is ANR?

ANR stands for Application not responding!! It means the User is operating the Android device and there is no response of touch and click, It looks like a hang or stuck which will end with either an ANR Dialogue or Application crash.



ANR dialogue

As per the user experience, ANR is one of the worst problems which an application can offer to the user for getting frustrated while doing something important.

In general, It is most important while programming the applications that, everything that we implement should not block the UI or create a bad user experience is the primary focus.

Everything that the user does on their device is more important than anything else. So there is zero space for compromising if anything interrupts the user. Asking to close the application or wait for some time for recovery, is what ANR does in between user interactions and this is how ANR can impact the User.

Technically, What is ANR as a Problem in Android-Subsystem?

From an Android programmer's perspective, the ANR is considered a severe issue so is for users too. ANR is directly related to the mess done while programming the

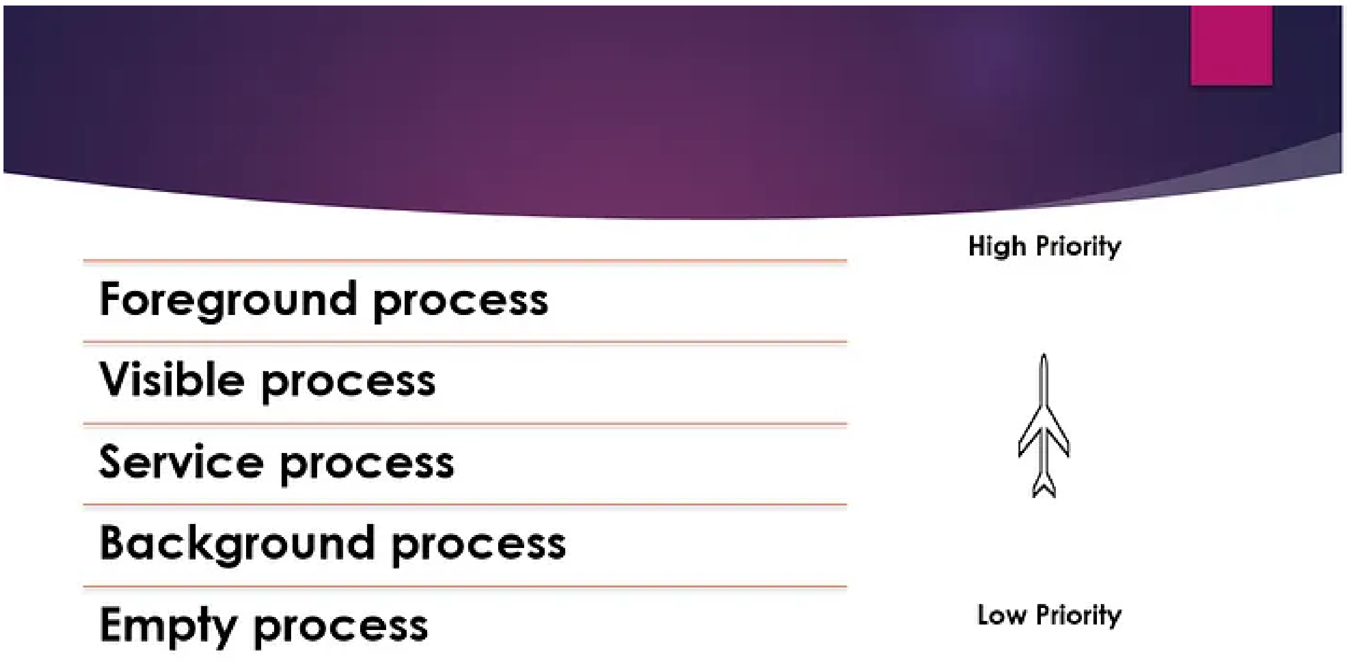
Application.

Almost all the applications run as a separate process when launched. Once a memory is occupied through a process, it can only be reclaimed by the system when needed otherwise it will be allocated to the running process until explicitly killed by the user.

Applications do not control the Process Lifecyle of itself!! Android System rules over the life of an application. There might be several parameters to play a fair game with all processes on the ground of Android but, the most important rule is **“What application doing is important for the audience”** If it is performing a User Interaction task then important and has a higher priority to be alive upfront more then everything else.

Android Process and their Priority

As mentioned above Android-Subsystem is the one responsible for controlling the process lifecycle, not the process itself. So Android has its conceptual-based priorities for the process running in the system. A higher-priority process will be the first in a row to use the system resource and be alive and executing operations all the time.

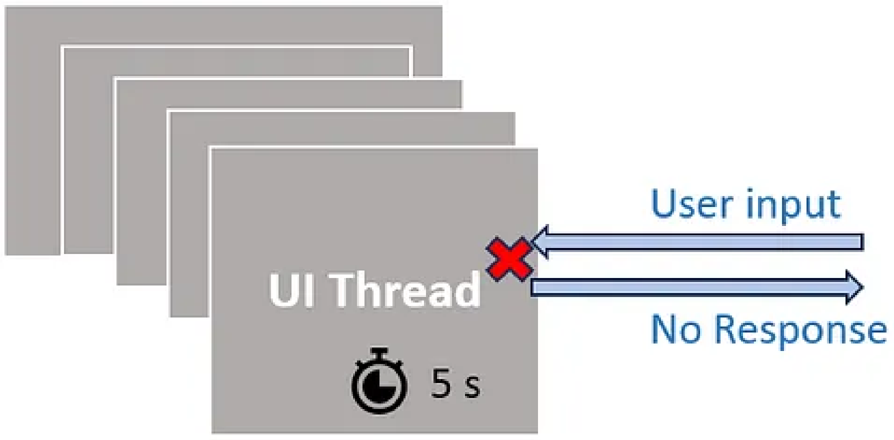


Why does ANR occur?

Above I showed the Android Subsystem Process priorities, So we should understand that process priorities clearly →

> *By https://developer.android.com/ A **foreground process** is required for what the user is currently doing. Various application components can cause its containing process to be considered foreground in different ways. A process is considered to be in the foreground if any of the following conditions hold:*
>
> *To know more about processes and app lifecycle click here*

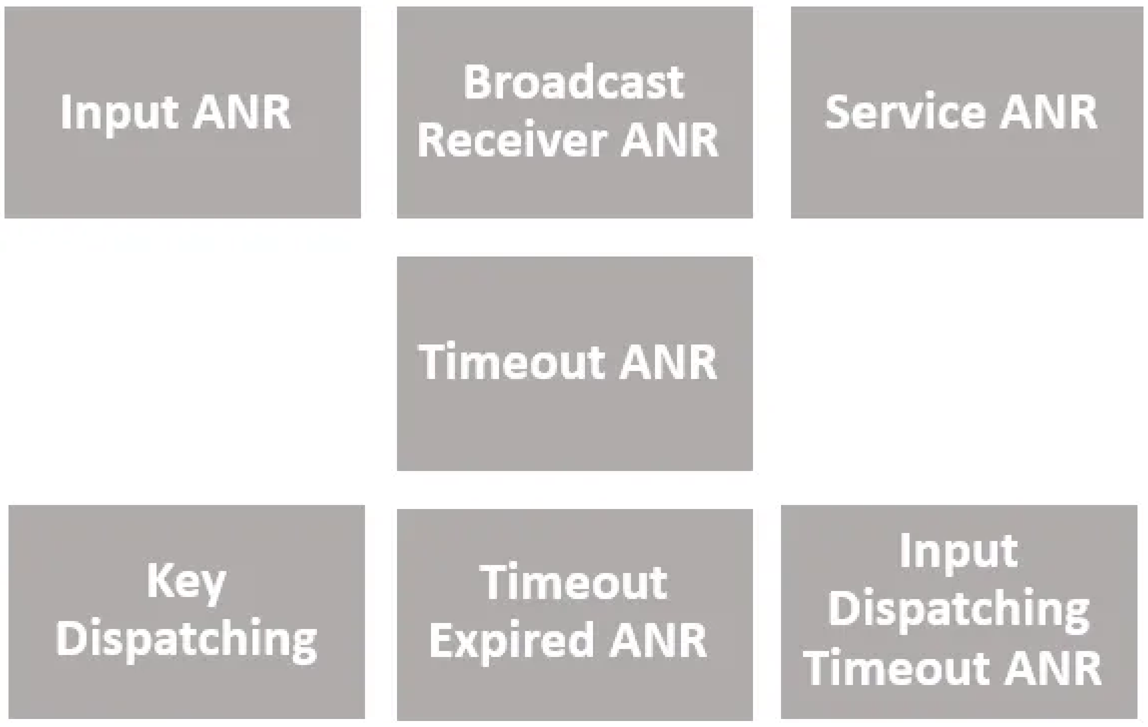
ANR (Application Not Responding) error occurs when a process takes a long time on the UI thread delaying user input processing. If this continues for more than five seconds, the system displays an ANR dialogue box.



Android's application layer provides guidelines and suggestions on how to program Android applications. Deviating from these guidelines can negatively impact the entire system and result in ANR (Application Not Responding) issues. Several reasons can cause ANR on Android devices, including long-running operations, blocked UI threads, deadlocks, inefficient code, memory issues, hardware resource constraints, delayed UI rendering, network issues, and external factors like incoming calls or system-level interruptions. These reasons are based on the core concepts of operating systems and applications and can lead to unresponsiveness in the Android sub-system.

Types of ANRs in Android

In the context of Android, ANR and its types are more related to the Android subsystem components. Therefore, instead of understanding it generically, we can directly relate it to the Android components mentioned below:



Possible reasons for ANRs

I have explained each reason with an example scenario when these types of ANR can occur and why!!

Input ANR

When the main thread or UI thread is blocked for an extended period(5 seconds +) and as a result, the user interface becomes unresponsive, and an ANR dialogue may appear to inform the user of the issue. This typically happens when a time-consuming operation, such as network or database access, is performed on the main thread.

To avoid ANRs, it's recommended to move time-consuming tasks to background threads or manage them asynchronously.

Broadcast Receiver ANR

When a Broadcast Receiver's "onReceive" method takes too much time to execute due to any possible reason. This issue can occur if time-consuming operations are performed within the "onReceive" method on the main thread, which can block the UI thread and result in an ANR.

To avoid ANRs in this context, it is recommended to delegate lengthy tasks to background threads or use services for such operations to ensure that the "onReceive" method completes quickly and keeps the application responsive.

Service ANR

Sometimes, an Android service can cause an Application Not Responding (ANR) error if it performs time-consuming operations on the main thread. This issue can occur if the onStartCommand or onHandleIntent method of the service involves blocking tasks such as heavy computations or network requests.

To avoid Application Not Responding (ANR) errors caused by services, it is important to perform resource-intensive operations on separate threads using components such as IntentService or AsyncTask. This ensures that the main thread remains free to respond to user actions.

Key Dispatching Timeout ANR

Sometimes, the system may fail to deliver a crucial event to the intended application component within a certain timeframe. This situation can occur due to lengthy operations or a deadlock in the event handling code, which can ultimately cause the application to stop responding.

When a UI component, such as an Activity or Fragment, is responsible for handling key events on the main thread and the process takes too long, it can cause a Key Dispatching Timeout ANR.

To avoid unresponsive applications, it is recommended to either move time-consuming tasks to background threads or to handle them asynchronously. By doing so, the main thread remains free to process user input quickly.

Input Dispatching Timeout ANR

Input Dispatching Timeout ANR in Android occurs when the system fails to process an input event within a specific time frame, making the app unresponsive to user interactions. Also when a huge amount of UI Update is happening in a short period and if it is not possible to process the UI update faster then the system can throw the ANR.

I witnessed the input dispatching timeout reason for ANR comes with heavy UI animations also so we need to carefully implement the UI animations-related functionality.

Developers can avoid this by offloading time-consuming tasks to background threads or using asynchronous mechanisms to keep the main thread responsive.

Conclusion!!

> *Time out of 5 seconds on the UI Thread is Over!!!*

After discussing the reasons mentioned above, it is evident that any task that blocks the UI thread for more than 5 seconds will result in an Application Not Responding (ANR) error.

The cause of the ANR can vary, but it is mostly due to one of the reasons mentioned earlier. Once we identify the type of ANR, it becomes easier to determine the appropriate resolution for it.

As per Android OS standards, here are some common reasons for ANR!!

The application is currently experiencing slow operations that require Input/Output (I/O) on the main thread. Moreover, it is also performing extensive computations on the same thread.

The main thread is getting blocked by a synchronous binder call to another process, which is taking an extended amount of time to return.

Another issue is that the main thread is waiting for a synchronized block for a long operation that is happening on another thread. Finally, the main thread is in a deadlock with another thread, either within the same process or through a binder call.

This situation is more severe than just waiting for a long operation to finish and can result in a system crash.

So after discussing all the things about ANR, I think the simplest way to prevent ANR is to be aware and attentive to what code we are writing to be performed on the main Thread, If we think twice about it while writing the code then maybe the software probably will not suffer by ANR and performance related problems.

. . .

Thanks for Reading!

Please comment with questions and suggestions!! If this blog is helpful for you then hit Please hit the clap! Follow on Medium, Connect on LinkedIn, Follow on Insta and send emails if any discussion is needed on the same topic on copy email.

Please Follow & subscribe to Make Android, Publication by Narendra K H for prompt updates on Android platform-related blogs.

Thank you!

Make Android

Make Android your writing space into the world of Android. Uncover the future of technology through our in-depth...

medium.com

Android | Android App Development | AndroidDev | Anr | Android App Developers

Follow

Written by Narendra Harny

256 Followers · Writer for Make Android

Write On, Android AOSP | Android OS | Android Application | Python | DevOps | Java | C++ | Kotlin | Shell | Linux | Android Auto | IVI

More from Narendra Harny and Make Android