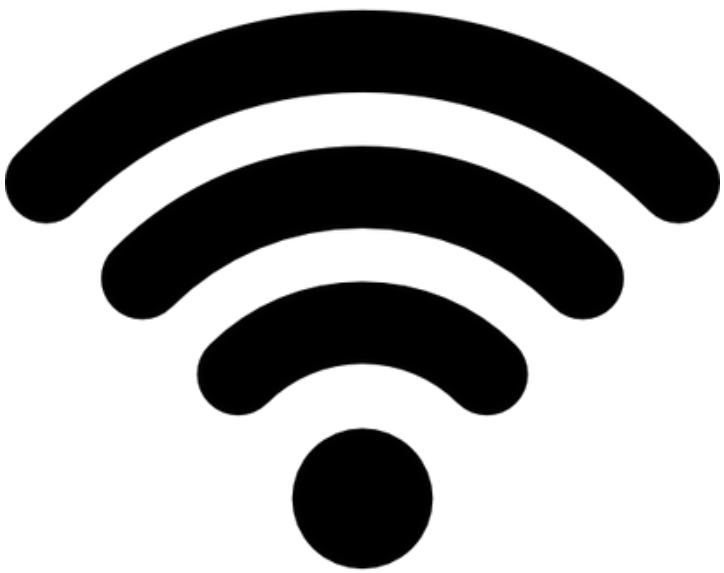
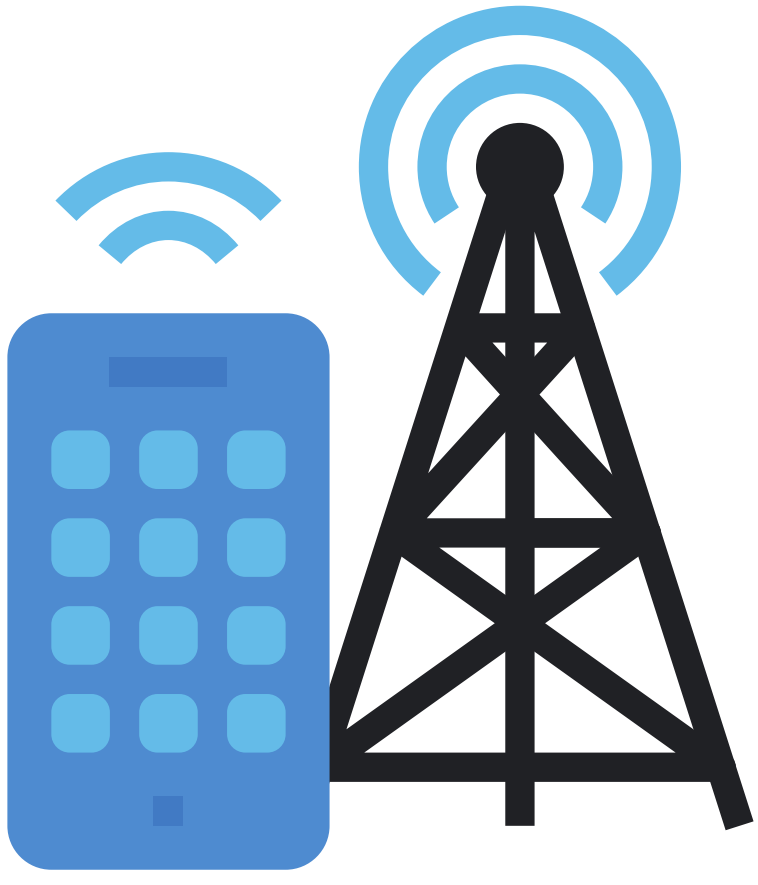
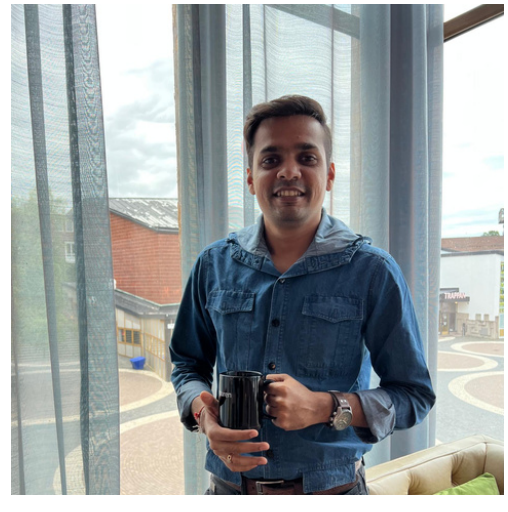


Topic for discussion:

Android: Monitor Network Connectivity



Presented by
Rohan Pambhar
(Android Engineer)



Preface

- Most android applications use an internet connection, and as android developers, we always implement a mechanism that handles the internet connection and the application state based on the internet connectivity.

- We mostly struggle to handle this mechanism to make it work seamlessly without affecting so much of our android code.

- So, here in this post, I'll be sharing a proper way to handle the internet connection state in application and update your UI accordingly

Connectivity Manager class

- The ConnectivityManager provides an API that enables you to request that the device connects to a network based on various conditions, including device capabilities and data transport options.

- The callback implementation provides information to your app about the device's connection status and the capabilities of the currently connected network.

- The API lets you determine whether the device is currently connected to a network that satisfies your app’s requirements.

Easy to implement Connectivity Manager

- **Configure a network request**
  - Declare a **NetworkRequest** that describes your app's network connection needs. The following code creates a request for a network connection to the internet and uses a Wi-Fi or cellular connection for the transport type.

```
val networkRequest = NetworkRequest.Builder()
        .addCapability(NetworkCapabilities.NET_CAPABILITY_INTERNET)
        .addTransportType(NetworkCapabilities.TRANSPORT_WIFI)
        .addTransportType(NetworkCapabilities.TRANSPORT_CELLULAR)
        .build()
```

  - Use **NET_CAPABILITY_NOT_METERED** to determine whether the connection is expensive.

- **Configure a network callback**
    - When you register the NetworkRequest with the ConnectivityManager, you must implement a **NetworkCallback** to receive notifications about changes in the connection status and network capabilities.
    - The most commonly implemented functions in the **NetworkCallback** include the following:
        - **onAvailable()**
        - **onLost()**
        - **onCapabilitiesChanged()**

```
private val networkCallback = object : ConnectivityManager.NetworkCallback() {
    // network is available for use
    override fun onAvailable(network: Network) {
        super.onAvailable(network)
    }

    // Network capabilities have changed for the network
    override fun onCapabilitiesChanged(
            network: Network,
            networkCapabilities: NetworkCapabilities
    ) {
        super.onCapabilitiesChanged(network, networkCapabilities)
        val unmetered = networkCapabilities.hasCapability(NetworkCapabilities.NET_CAPABILITY_
    }

    // lost network connection
    override fun onLost(network: Network) {
        super.onLost(network)
    }
}
```

- **Register for network updates**
    - After you declare the NetworkRequest and NetworkCallback, use the **requestNetwork()** or **registerNetworkCallback()** functions to search for a network to connect from the device that satisfies the NetworkRequest. The status is then reported to the NetworkCallback.

```
val connectivityManager = getSystemService(ConnectivityManager::class.java)
connectivityManager.requestNetwork(networkRequest, networkCallback)
```

Important Note:

- Make sure you register a callback once in the application's lifecycle. So I suggest registering the event in the **onCreate()** method of the Application class.

- And then, you can also notify your activities and fragments using **LiveData**.

- For that, create a ‘Singleton’ class with a ‘MutableLivedata’ of type ‘Boolean’ to maintain its instance throughout the application's lifecycle.

- See the code snippet given below.

```
object AppNetworkManager {

    private val _activeNetworkStatus = MutableLiveData<Boolean>()
    val activeNetworkStatus : LiveData<Boolean> = _activeNetworkStatus

    fun setNetworkStatus(connectivityStatus: Boolean) {
        if (Looper.myLooper() === Looper.getMainLooper()) {
            _activeNetworkStatus.setValue(connectivityStatus)
        } else {
            _activeNetworkStatus.postValue(connectivityStatus)
        }
    }

    fun getNetworkConnectivityStatus(): LiveData<Boolean> {
        return activeNetworkStatus
    }
}
```


it Agenturen