Discussion-4 What are the different methods for making network requests in Android and when should each be used?

Discussion Topic:

What are the different methods for making network requests in Android and when should each be used?

My Post:

Hello Class,

For this discussion, I chose the following topic: What are the different methods for making network requests in Android and when should each be used?
I chose this topic because I'm still fairly new to API calls and Android development, and it's a good way to become more familiar with them.

Developing an Android app frequently involves implementing some data fetching functionality and sending data to servers within the application (Coditive, 2024). This often means making network requests asynchronously and handling network errors. When using a combination of Kotlin and Jetpack Compose, several popular HTTP (Hypertext Transfer Protocol) clients can be used to perform network requests. This post explores three of the most commonly used clients: Retrofit, Volley, and OkHttp, and their strengths and weaknesses, and discusses when to use each one

Clients for Network Requests

The most popular HTTP clients for making network requests while using Kotlin and Jetpack Compose are Retrofit, Volley, and OkHttp. HTTP client is a software that allows applications to communicate with web servers and APIs over the internet. It handles network requests, responses, and connections within the HTTP protocol system, including managing headers, methods (GET, POST, PUT, DELETE).

Retrofit

Retrofit was developed by Square, and it is a type-safe HTTP client (Kostadinov, 2024). It is an abstraction that allows making API calls through declarative interfaces and handles parsing JSON into Java/Kotlin objects using libraries like Gson or Moshi. The table below lists and describes different aspects and features of Retrofit.

Table 1
Retrofit Aspect and Features

Feature/Aspect	Description
Name	Retrofit
Developer	Square
Type	Type-safe HTTP client for Android and Java
Industry Standard	Widely regarded as the industry standard for Android networking

Strengths	Type-safety (API endpoints defined as interfaces with annotations) Ease of use (abstracts low-level HTTP details) Excellent documentation and community support Built-in support for OkHttp (connection pooling, GZIP compression) Automatic JSON parsing (Gson, Moshi) Support for Kotlin Coroutines (asynchronous requests)
Weaknesses	Less flexible than OkHttp (for fine-grained control) Can be overkill for simple requests
When to Use	When you need a simple and readable way to define API requests When your team is already familiar with Retrofit When you require out-of-the-box features like JSON parsing and request retries When you have a complex API with many endpoints

Note: The table lists various aspects and features of the Retrofit HTTP client for Kotlin and Java. Data from several sources (Square, n.d.a; GeeksForGeeks, 2025; Kramer, 2024; Kostadinov, 2024; Anna, 2024)

Volley
Volley is another HTTP client developed by Google, it was designed in 2013 to make networking easier and faster in Android apps (Vartika02, 2025). It is well-suited for applications making frequent, small network requests. The table below lists and describes different aspects and features of Volley.

Table 2
Volley Aspect and Features

Feature/Aspect	Description
Name	Volley
Developer	Google
Type	HTTP library for Android networking
Industry Standard	Popular for lightweight Android networking, especially for small, frequent requests
Strengths	Lightweight and efficient (optimized for small to medium-sized network operations with a small memory footprint) Request queuing and prioritization (manages queues and prioritizes requests to handle multiple operations efficiently) Built-in caching (robust in-memory and on-disk caching to reduce network requests and improve performance) Request cancellation (allows cancellation of unneeded requests, e.g., when users navigate away)
Weaknesses	Not suitable for large downloads (holds responses in memory during parsing, unsuitable for streaming or large files) Less versatile than OkHttp or Retrofit (fewer features and customization options despite some flexibility)

When to Use	When you need to make frequent, small network requests When caching is crucial for your app's performance When you need a simple library with built-in request queuing and prioritization When you need to cancel requests easily

Note: The table lists various aspects and features of the Retrofit HTTP client for Kotlin and Java. Data from several sources (Google, n.d.; Vartika02, 2025; AbhiAndroid, n.d.)

OkHttp

OkHttp was developed by Square, it is a powerful and efficient HTTP client for Kotlin and Java, it was used as the foundation for both Retrofit and Volley (Gouda, 2024). It can also be used directly, providing more control on network requests than Retrofit and Volley. The table below lists and describes different aspects and features of OkHttp.

Table 3

OkHttp Aspect and Features

Feature/Aspect	Description
Name	OkHttp
Developer	Square
Type	Powerful and efficient HTTP client
Foundation For	Retrofit and Volley
Usage	Can be used directly for maximum control over network requests
Strengths	High performance and efficiency (minimizes latency, maximizes throughput) Connection pooling and GZIP compression (reduces latency and bandwidth) Flexibility and customization (fine-grained control over headers, timeouts, etc.) HTTP/2 Support (multiplexing, header compression, server push) WebSocket Support (real-time, bidirectional communication) Interceptors (modify requests and responses)
Weaknesses	More verbose than Retrofit (manual request building and response parsing) Steeper learning curve (understanding HTTP concepts and OkHttp's API)
When to Use	When maximum control over network operations is needed When implementing custom features not provided by Retrofit or Volley When comfortable with a more verbose and hands-on approach to networking

Note: The table lists various aspects and features of the OkHttp HTTP client for Kotlin and Java. Data from several sources (Square, n.d.b; Gouda, 2024; Baeldung, n.d.)

Each client has its pros and its cons. When choosing the best client for the application developers need to consider factors such as the complexity of the project, the frequency of the requests, the size of the requests, and control vs. convenience to select the client that fits best with those needs. The table below compares the clients by key features and characteristics.

Table 4
HTTP Client Comparison

Feature	Retrofit	Volley	OkHttp
Type-safety	Yes	No	No
Ease of use	High	High	Moderate
Flexibility	Moderate	Moderate	High
Performance	High	High (for small requests)	High
Caching	Yes (via OkHttp)	Yes	Yes
Request cancellation	Yes	Yes	Yes
Coroutines support	Yes	No	No
HTTP/2 support	Yes (via OkHttp)	No	Yes
WebSocket support	No	No	Yes
Ideal use cases	Complex APIs, large projects	Frequent small requests, simple APIs	Maximum control, custom features

Note: The table below compares the clients by key features and characteristics.

No matter what client is implemented within an application, network requests should never be executed on the main threads. Doing so can result in the app freezing and poor user experience. Best practices dictate the use of asynchronous mechanisms such as Kotlin Coroutines to avoid blocking the app's main thread. Kotlin Coroutine is a concurrency design pattern that simplifies code executing asynchronously (Android Developers, n.d.). The design provides several advantages including its lightweight nature, built-in cancellation support, and improved memory management.

In addition to implementing asynchronous mechanisms for executing network requests, it is essential to handle network errors properly to create an overall robust application and a good user experience. Strategies such as robust network exception handling, thorough testing, retry mechanisms, and user feedback should be implemented and designed when integrating network request mechanisms within the application.

To summarize, when developing an Android app with Kotlin and Jetpack Compose, common HTTP clients for integrating network requests are Retrofit, Volley, and OkHttp. Choosing the right HTTP client depends on the specific needs of the project. Regardless of the chosen library (client), developers need to implement asynchronous execution for network requests by using designs such as Kotlin Coroutines. This is essential for not freezing the app during network calls and for a good user experience. Additionally, implementing network error handling is also essential for building a robust application that can use network request mechanisms effectively. Therefore, understanding how to integrate HTTP clients with asynchronous programming and error handling is crucial for designing a successful Android development that incorporates network requests.

-Alex

References:

AbhiAndroid (n.d.). *Volley tutorial with example In Android Studio*. AbhiAndroid. https://abhiandroid.com/programming/volley#gsc.tab=0

Android Developers (n.d.). *Kotlin coroutines on Android*. Android. https://developer.android.com/kotlin/coroutines

Anna. (2024, November 16). *Retrofit in Android*. Medium. https://medium.com/@anna972606/retrofit-in-android-15fa724a8fa6

Baeldung (n.d.). *A guide to OkHttp*. Baeldung. https://www.baeldung.com/guide-to-okhttp

Coditive. (2024, November 24). *Understanding retrofit: Simplifying android networking*. Medium. https://medium.com/@coditive/understanding-retrofit-simplifying-android-networking-ef37f72f9cb8

Kramer, N. (2024, May 14). Retrofit tutorial for Android beginners. Daily.dev. https://daily.dev/blog/retrofit-tutorial-for-android-beginners

Kostadinov, D. (2024, December 11). *When to use Retrofit and when to use KTOR: a guide for Android developers*. Medium. https://proandroiddev.com/when-to-use-retrofit-and-when-to-use-ktor-a-guide-for-android-developers-918491dcf69a

GeeksForGeeks (2025, February 18). *Introduction to Retrofit in Android*. GeeksForGeeks. https://www.geeksforgeeks.org/introduction-retofit-2-android-set-1/

Google (n.d.). *Volley*. GitHub. https://google.github.io/volley/

Gouda, M. (3034, December 5). *Comprehensive guide to OkHttp for Java and Kotlin*. ScrpFly. https://scrapfly.io/blog/guide-to-okhttp-java-kotlin/

Square (n.d.a). *Retrofit*. GitHub. https://square.github.io/retrofit/.

Square (n.d.b). *OkHttp*. GitHub. https://square.github.io/okhttp/

Vartika02 (2025, January 6). *Volley library in Android*. GeeksForGeeks. https://www.geeksforgeeks.org/volley-library-in-android/