**Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer** e-ISSN: 2548-964X
Vol. 9, No. 5, Mei 2025, hlm. x-x http://j-ptiik.ub.ac.id

# Analisis Performa *Networking Library* Retrofit, Ktor Client, Okhttp Dan Volley Pada Pengembangan Aplikasi Perangkat Bergerak Berbasis Android

**Dzikry Aji Santoso[1], Djoko Pramono[2], Aryo Pinandito[3]**

Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]dzikryaji@student.ub.ac.id, [2]djoko.jalin@ub.ac.id, [3]aryo@ub.ac.id

**Abstrak**

Pertumbuhan pengembangan aplikasi Android mendorong kebutuhan akan komunikasi data yang cepat dan efisien melalui berbagai *networking library* seperti Retrofit, Ktor Client, OkHttp, dan Volley. Penelitian ini membandingkan kinerja keempat *library* dalam proses *data fetching* menggunakan HTTP POST dan GET. Pengukuran meliputi *response time*, *memory usage*, serta *network usage* (*transmitted* dan *received*), dengan skenario permintaan data bervariasi dari 100 hingga 10.000. Analisis dilakukan melalui uji normalitas, uji beda, serta analisis rata-rata dan median. Hasil menunjukkan perbedaan signifikan dalam *response time*, *memory usage*, dan data *transmitted*. OkHttp unggul dalam *response time*, memberikan waktu respons tercepat. Retrofit menunjukkan efisiensi pada *memory usage*, sementara Ktor Client terbaik dalam *transmitted network usage* dan tidak ditemukan perbedaan signifikan pada *received network usage* dari keempat *library*. Secara keseluruhan, Retrofit menonjol sebagai pilihan terbaik berkat efisiensinya pada *memory usage* serta performa yang baik pada *response time* dan *transmitted network usage*. Meski tidak unggul pada semua aspek, kinerja Retrofit yang konsisten menjadikannya solusi optimal untuk pengembangan aplikasi Android.

**Kata kunci**: *networking library, android, retrofit, ktor client, Okttp, volley, data fetching.*

**Abstract**

*The growth of Android application development drives the need for fast and efficient data communication through various networking libraries such as Retrofit, Ktor Client, OkHttp, and Volley. This study compares the performance of these four libraries in data fetching processes using HTTP POST and GET. Measurements include response time, memory usage, and network usage (transmitted and received), with data request scenarios ranging from 100 to 10,000. The analysis was conducted through normality tests, difference tests, as well as mean and median analysis. The results indicate significant differences in response time, memory usage, and transmitted data. OkHttp excels in response time, providing the fastest responses. Retrofit demonstrates efficiency in memory usage, while Ktor Client performs best in transmitted network usage. No significant differences were found in received network usage among the four libraries. Overall, Retrofit stands out as the best choice due to its efficiency in memory usage and solid performance in response time and transmitted network usage. Although it does not lead in all aspects, Retrofit's consistent performance makes it an optimal solution for Android application development.*

**Keywords**: *networking library, android, retrofit, ktor client, Okttp, volley, data fetching.*

## 1. PENDAHULUAN

Penggunaan aplikasi perangkat berbasis Android di Indonesia terus meningkat seiring dengan bertambahnya kebutuhan akan akses data yang cepat dan responsif melalui jaringan internet. Pada awal tahun 2024, jumlah pengguna internet di Indonesia mencapai 185,3 juta orang (Kemp, 2024), dengan lebih dari 80% perangkat bergerak di Indonesia menggunakan sistem operasi Android (StatCounter Global Stats, 2024). Kondisi ini menciptakan peluang besar sekaligus tantangan bagi pengembang aplikasi untuk menghadirkan layanan yang berkualitas tinggi dan optimal, terutama dalam hal kinerja aplikasi yang sangat bergantung pada

komunikasi dengan server melalui jaringan internet.

Dalam pengembangan aplikasi Android, pemilihan *networking library* menjadi salah satu aspek yang krusial. *Networking library* berfungsi sebagai penghubung antara aplikasi dan server untuk melakukan pengambilan maupun pengiriman data. Performa *networking library* yang dipilih memiliki dampak langsung terhadap pengalaman pengguna, terutama dalam hal waktu respons, penggunaan memori, dan konsumsi data. Hal ini menjadi semakin penting mengingat koneksi internet di Indonesia yang tidak merata, terutama di daerah dengan keterbatasan kecepatan atau kuota data, sehingga aplikasi harus mampu menggunakan sumber daya secara efisien agar tetap responsif.

Salah satu *library* networking yang banyak digunakan oleh pengembang Android adalah Retrofit, yang dikembangkan oleh Square. Retrofit adalah *HTTP Client library* yang populer untuk pengembangan aplikasi Android, yang dikenal karena kemudahan penggunaan dan fitur-fiturnya (Belkhir et al., 2019). Keunggulan utama Retrofit adalah kemudahan penggunaan, di mana pengembang dapat mendeklarasikan endpoint API sebagai interface, lalu Retrofit secara otomatis mengonversi respons JSON ke dalam objek data menggunakan converter seperti Gson atau Moshi. Selain itu, Retrofit mendukung berbagai metode HTTP dan fitur canggih seperti interceptor, retry, dan caching.

Alternatif lain yang banyak digunakan, terutama untuk aplikasi berbasis Kotlin, adalah Ktor Client. Ktor Client, yang dikembangkan oleh JetBrains, merupakan bagian dari framework Ktor yang berfokus pada operasi jaringan dari sisi klien. Ktor Client memungkinkan pengembang untuk membuat HTTP request dan mengelola response dari server dengan fleksibilitas tinggi (Jetbrains, 2024). Keunggulan utama Ktor Client adalah fleksibilitas dan dukungannya untuk pengembangan multiplatform menggunakan Kotlin Multiplatform. Library ini menawarkan DSL (Domain-Specific Language) yang bersih dan terstruktur untuk membangun request HTTP. Selain itu, Ktor Client memiliki integrasi yang mendalam dengan coroutine di Kotlin sehingga memberikan efisiensi, struktur kode yang bersih, kemudahan dalam mengelola error, dan kemampuan untuk menangani banyak operasi asinkron secara bersamaan.

*Library* lain yang sering digunakan adalah OkHttp, yang juga dikembangkan oleh Square. OkHttp memungkinkan komunikasi HTTP yang efisien, mempercepat pemuatan konten, dan menghemat bandwidth, menjadikannya ideal untuk aplikasi yang membutuhkan penghematan data dan respons cepat (Square, 2019). Keunggulan OkHttp adalah performanya yang sangat cepat dan efisien karena menyediakan kontrol granular terhadap request dan respons. OkHttp mendukung fitur tingkat lanjut seperti koneksi HTTP/2, multiplexing, caching disk, dan timeout granular.

Selain ketiga *library* tersebut, ada juga Volley, yang dikembangkan oleh Google. *Library* Volley sendiri merupakan *library* yang paling banyak digunakan dalam aplikasi yang dibuat oleh Google (Lachgar et al., 2018). Volley memiliki keunggulan dalam menangani request bersamaan (*concurrent*) dan mendukung caching bawaan, menjadikannya cocok untuk aplikasi yang membutuhkan banyak operasi jaringan seperti sinkronisasi data atau pengambilan gambar. Library ini juga menawarkan kemudahan untuk menangani respons dalam format string, JSON, atau image.

Dalam memilih *networking library*, beberapa aspek performa perlu diperhatikan, salah satunya adalah *response time*. *Response time* mengacu pada waktu yang dibutuhkan oleh aplikasi untuk menerima respons dari server setelah melakukan permintaan HTTP. Aspek ini sangat penting karena *response time* secara langsung memengaruhi pengalaman pengguna (Zhou et al., 2016).

Selain *response time*, *memory usage* juga dapat digunakan dalam mengukur performa *networking library*. *Memory usage* adalah jumlah memori yang digunakan oleh aplikasi saat beroperasi, termasuk saat melakukan permintaan HTTP melalui *library*. Efisiensi dalam pengelolaan *memory usage* sangat penting, karena responsivitas dan efisiensi aplikasi perangkat bergerak sangat bergantung pada kemampuan aplikasi untuk mengelola *memory usage* secara optimal (Ghag, 2024).

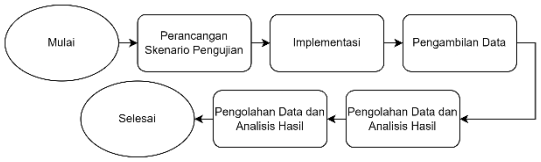
Terakhir, terdapat *network usage* yang dapat digunakan dalam mengukur performa *networking library*. *Network usage* merujuk pada jumlah data yang dikirimkan dan diterima

oleh aplikasi saat berkomunikasi dengan server. Penggunaan jaringan yang efisien tentunya dapat berpengaruh dalam aplikasi perangkat bergerak karena sebagian besar pengguna bergantung pada koneksi internet yang mungkin terbatas, baik dari segi kecepatan maupun kuota data. Selain itu, penggunaan *network usage* yang tidak efisien dapat memengaruhi konsumsi energi perangkat (Rosen et al., 2024).

Penelitian ini membandingkan performa empat library networking, yaitu Retrofit, Ktor Client, OkHttp, dan Volley, berdasarkan pertimbangan teknis, fitur, dan relevansi terhadap kebutuhan pengembangan aplikasi Android modern. Keempat library ini dipilih tidak hanya karena popularitasnya, tetapi juga karena masing-masing memiliki karakteristik unik yang relevan untuk kebutuhan pengembangan aplikasi dengan berbagai skenario. Penelitian ini diharapkan dapat membantu pengembang aplikasi Android dalam memilih *library* yang tepat sesuai dengan kebutuhan, terutama terkait aspek kecepatan, efisiensi penggunaan data, dan pengelolaan sumber daya perangkat. Dengan mengetahui networking library yang memiliki performa terbaik dalam hal response time, memory usage, dan network usage, dapat membantu pengembang aplikasi perangkat bergerak berbasis Android dalam memilih library networking yang tepat untuk pengembangan aplikasi mereka khususnya untuk proses data fetching dengan menggunakan metode POST dan GET.

## 2. METODOLOGI PENELITIAN

Tahapan penilitian yang dilakukan dapat dilihat dalam Gambar 1




Gambar 1. Metode Penelitian

### 3.1. Perancangan Skenario Pengujian

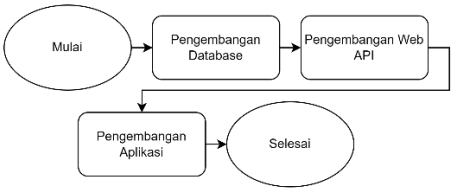
Tahap pertama dalam metodologi penelitian ini adalah merancang skenario pengujian yang bertujuan untuk menjawab rumusan masalah. Skenario pengujian dalam penelitian ini dirancang untuk menguji response time, memory usage, dan network usage dari penggunaan library Retrofit, Ktor Client, OkHttp dan Volley dalam pengembangan aplikasi perangkat bergerak berbasis Android. Pengujian difokuskan pada proses data fetching menggunakan HTTP Method GET dan POST, dengan jumlah data yang bervariasi, yaitu 100, 1000, 5000, dan 10000 row. Studi kasus yang digunakan dalam penelitian ini adalah The Movie Database (TMDb), oleh karena itu kasus uji difokuskan pada pengambilan data movie. Detail kasus uji dapat dilihat pada Tabel 1.

Tabel 1 Detail Kasus Uji

| No | Kode | Keterangan |
|---|---|---|
| 1 | P100 | Mengambil 100 *data row* movie menggunakan POST request. |
| 2 | P1000 | Mengambil 1000 *data row* movie menggunakan POST request. |
| 3 | P5000 | Mengambil 5000 *data row* movie menggunakan POST request. |
| 4 | P10000 | Mengambil 10000 *data row* movie menggunakan POST request. |
| 5 | G100 | Mengambil 100 *data row* movie menggunakan GET request. |
| 6 | G1000 | Mengambil 1000 *data row* movie menggunakan GET request. |
| 7 | G5000 | Mengambil 5000 *data row* movie menggunakan GET request. |
| 8 | G10000 | Mengambil 10000 *data row* movie menggunakan GET request. |

### 3.2. Implementasi

Pada tahap ini, dilakukan pengembangan database, Web API, serta aplikasi yang akan digunakan sebagai lingkungan uji. Tahap implementasi ini disesuaikan dengan skenario pengujian yang telah dirancang sebelumnya untuk memastikan bahwa skenario pengujian dapat dijalankan dengan baik. Alur lengkap dari tahap implementasi dapat dilihat dalam Gambar 2.




Gambar 2. Alur Tahap Implementasi

### 3.3. Pengambilan Data

Pada penelitian ini, pengambilan data dilakukan dengan mengikuti skenario yang telah dirancang sebelumnya menggunakan aplikasi yang dikembangkan selama tahap implementasi. Proses pengambilan data dibantu oleh Android

Studio Profiler dan juga Logcat. Android Studio Profiler adalah alat yang tersedia di Android Studio untuk memantau performa aplikasi secara real-time. Logcat adalah alat yang digunakan untuk menampilkan log sistem dan aplikasi di Android.

Penggunaan kedua alat tersebut dalam penelitian ini bertujuan untuk mengukur beberapa aspek penting dari performa aplikasi, yaitu *response time*, *memory usage*, dan *network usage*. Pengukuran *response time* dilakukan untuk mengetahui seberapa cepat aplikasi dapat merespons permintaan data yang dikirimkan ke server. Selanjutnya, *memory usage* diukur untuk mengevaluasi seberapa banyak memori yang digunakan oleh aplikasi selama operasi jaringan berlangsung. Terakhir, *network usage* diukur untuk menilai efisiensi penggunaan data saat aplikasi berkomunikasi dengan server.

### 3.4. Pengolahan Data dan Analisis Hasil

Pada tahap ini, data yang diperoleh dari pengambilan data diolah dan dianalisis untuk mendapatkan informasi yang relevan. Langkah ini bertujuan menjawab pertanyaan penelitian dan membandingkan kinerja empat library yang diuji, yaitu Retrofit, Ktor Client, OkHttp, dan Volley. Analisis dilakukan dengan uji normalitas untuk menentukan distribusi data serta uji beda untuk mengetahui perbedaan signifikan antar *library*.

Uji normalitas menggunakan metode Shapiro-Wilk karena jumlah sampel relatif kecil, yaitu 30 untuk setiap skenario pengujian. Uji ini menentukan apakah data berdistribusi normal, yang memengaruhi jenis uji beda yang digunakan. Jika data berdistribusi normal, analisis dilakukan menggunakan Uji ANOVA. Jika tidak, digunakan Uji Kruskal-Wallis, yang merupakan metode non-parametrik. Hasil analisis ini menjadi dasar untuk penarikan kesimpulan mengenai performa keempat library.

### 3.5. Kesimpulan

Tahap ini merupakan tahap akhir dari penelitian yang bertujuan untuk menarik kesimpulan berdasarkan hasil pengolahan dan analisis data yang telah dilakukan sebelumnya. Kesimpulan ini akan menjawab pertanyaan penelitian mengenai perbandingan performa empat *networking library*, yaitu Retrofit, Ktor Client, OkHttp dan Volley, dalam pengembangan aplikasi Android. Melalui pengujian yang mencakup aspek *response time*, *memory usage*, dan penggunaan memori, akan ditentukan *library* mana yang menunjukkan performa lebih baik dalam skenario pengujian yang telah dirancang. Hasil penelitian ini diharapkan dapat memberikan wawasan kepada pengembang dalam memilih *networking library* yang paling sesuai untuk kebutuhan pengembangan aplikasi berbasis Android, sehingga mereka dapat membuat keputusan yang lebih tepat berdasarkan performa keempat *library* tersebut.

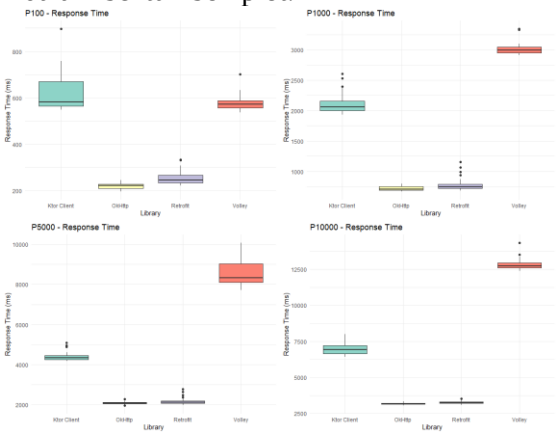
## 3. ANALISIS DAN HASIL

### 4.1. Pengambilan Data

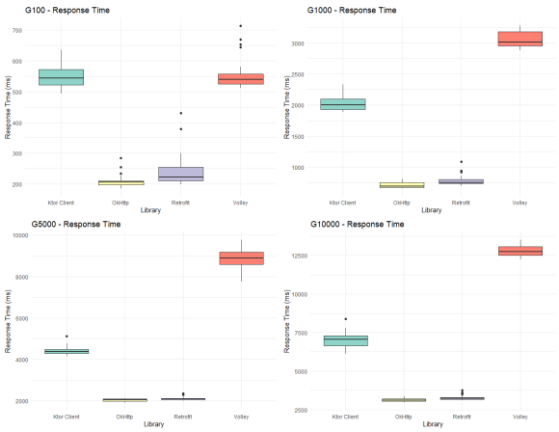
Pengambilan data dilakukan berdasarkan kasus uji yang telah dirancang sebelumnya, di mana setiap skenario diulang sebanyak 30 kali untuk setiap *library*. Data terkait *response time* dan *network usage* dicatat menggunakan log aplikasi. Pengukuran *network usage* dilakukan dengan memanfaatkan `TrafficStats` untuk menghitung data *transmitted* dan received network usage. Pengukuran *response time* dilakukan dengan mencatat selisih waktu sebelum dan sesudah permintaan menggunakan `System.currentTimeMillis()`. Sementara itu, *memory usage* selama operasi jaringan dipantau menggunakan Android Studio Profiler.

#### 4.1.1 Data *Response Time*

Pengambilan data *response time* dilakukan dengan mengukur selisih waktu antara sebelum dan sesudah proses fetching data. Pengukuran waktu dilakukan menggunakan fungsi `System.currentTimeMillis()` yang dicatat sebelum dan setelah proses *data fetching*. Adapun hasil pengambilan data *response time* yang diperoleh dapat dilihat pada Gambar 3 dan 4 dalam bentuk box plot.



Gambar 3. Box Plot *Response Time* Kasus Uji P100, P1000, P5000 dan P10000

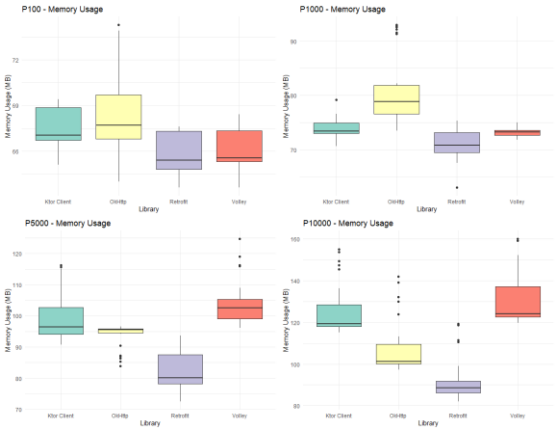



Gambar 4. Box Plot *Response Time* Kasus Uji G100, G1000, G5000 dan G10000

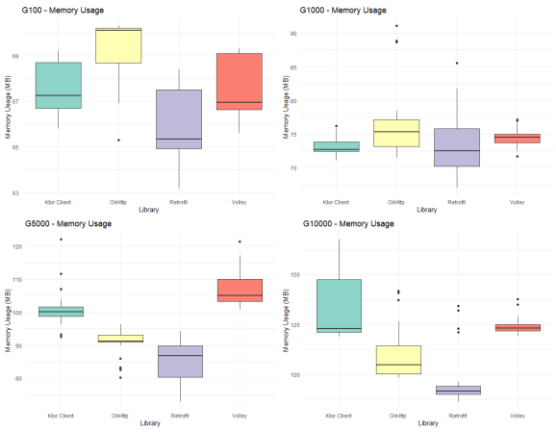
Secara keseluruhan, *response time* meningkat seiring dengan bertambahnya data yang diambil. Selian itu, hasil pengujian *response time* pada berbagai kasus uji (P100, P1000, P5000, P10000, G100, G1000, G5000, dan G10000) juga menunjukkan bahwa OkHttp secara konsisten menunjukkan kinerja terbaik dengan rata-rata dan median *response time* terendah pada semua kasus uji. Hal ini didukung oleh rentang data yang sempit pada *box plot*, menunjukkan stabilitas performa yang tinggi. Retrofit berada di urutan kedua dengan performa yang tidak jauh berbeda. Sementara itu, Ktor Client dan Volley cenderung memiliki rata-rata dan median *response time* yang lebih tinggi, dengan Volley menunjukkan variabilitas performa yang signifikan berdasarkan rentang data yang luas pada *box plot*.

### 4.1.2 Data *Memory Usage*

Pengambilan data *memory usage* dilakukan dengan melihat *memory usage* selama proses *data fetching*. Pengukuran dari *memory usage* ini dilakukan dengan bantuan Android Studio Profiler. Adapun hasil pengambilan data *memory usage* yang diperoleh dapat dilihat pada Gambar 5 dan 6 dalam bentuk box plot.




Gambar 5. Box Plot *Memory Usage* Kasus Uji P100, P1000, P5000 dan P10000




Gambar 6. Box Plot *Memory Usage* Kasus Uji G100, G1000, G5000 dan G10000

Secara keseluruhan, *memory usage* meningkat seiring dengan bertambahnya jumlah data yang diambil. Pada data kecil (P100, G100, P1000, dan G1000), perbedaan rata-rata *memory usage* antar *library* relatif kecil, tetapi menjadi lebih signifikan pada data besar (P5000, P10000, G5000, dan G10000). Meskipun OkHttp kurang efisien dalam penggunaan memori pada data kecil dibandingkan Ktor Client dan Volley, performanya lebih unggul pada data besar jika dibandingkan dengan kedua *library* tersebut. Namun demikian, secara keseluruhan, Retrofit tetap menjadi *library* yang paling efisien dalam penggunaan memori, baik untuk data kecil maupun besar.

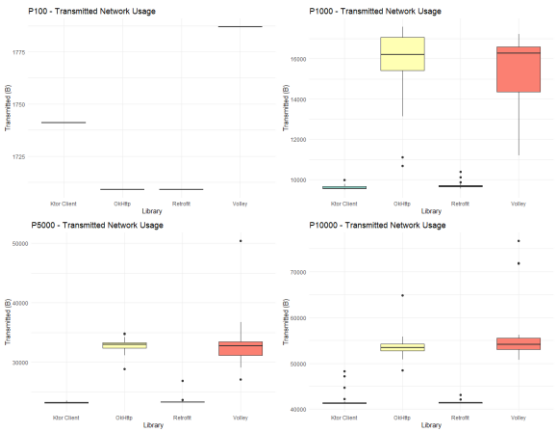
### 4.1.3 Data *Network Usage*

Pengambilan data *network usage* dilakukan dengan memanfaatkan kelas bawaan Android SDK yaitu `TrafficStats`. Kelas ini berfungsi untuk mengukur berapa banyak data yang dikirim dan diterima oleh perangkat. Pengambilan data jaringan dibagi menjadi dua

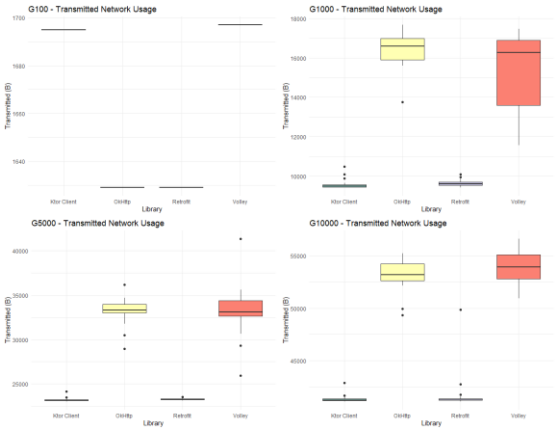
yaitu *transmitted* dan *received*. *Transmitted network usage* merujuk pada jumlah data yang dikirimkan dari perangkat ke jaringan, sementara *received network usage* menggambarkan jumlah data yang diterima perangkat dari jaringan.

#### 4.1.3.1 Data *Transmitted Network Usage*

Pengambilan data *transmitted network usage* dilakukan dengan memanfaatkan fungsi `getUidTxBytes()` pada kelas `TrafficStats`, yang menggunakan UID dari aplikasi yang diperoleh melalui `android.os.Process.myUid()`. Pengukuran dilakukan dengan mencatat nilai sebelum dan setelah proses data fetching. Hasil pengambilan data *transmitted network usage* yang diperoleh dapat dilihat pada Gambar 7 dan 8 dalam bentuk box plot.




Gambar 7. Box Plot *Transmitted Network Usage* Kasus Uji P100, P1000, P5000 dan P10000

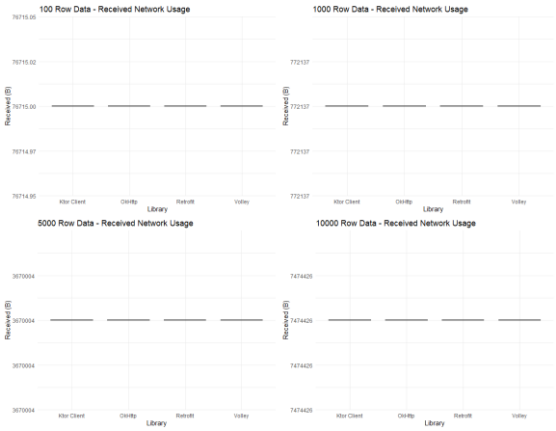



Gambar 8. Box Plot *Transmitted Network Usage* Kasus Uji G100, G1000, G5000 dan G10000

Secara keseluruhan, *transmitted network usage* cenderung meningkat seiring dengan bertambahnya jumlah data yang diambil dalam setiap kasus uji baik rata-rata maupun median. Hasil pengujian menunjukkan bahwa Retrofit dan Ktor Client secara konsisten memiliki rata-rata dan median *transmitted network usage* yang lebih rendah dibandingkan dengan OkHttp dan Volley. Ktor Client mencatatkan rata-rata dan median *transmitted network usage* terendah pada hampir semua kasus uji, kecuali pada kasus uji P100 dan G100. Selain itu, pada kasus uji P100 dan G100, semua data yang diperoleh dari masing-masing *library* identik, dengan nilai yang sama persis seperti rata-ratanya, yang terlihat pada box plot berupa garis.

#### 4.1.3.2 Data *Received Network Usage*

Pengambilan data untuk *received network usage* dilakukan dengan menggunakan fungsi `getUidRxBytes()` dari kelas `TrafficStats`, yang memanfaatkan UID aplikasi yang diperoleh melalui `android.os.Process.myUid()`. Proses pengukuran dilakukan dengan mencatat nilai sebelum dan setelah proses data fetching. Hasil pengambilan data *received network usage* yang diperoleh dapat dilihat pada Gambar 9 dalam bentuk box plot.




Gambar 9. Box Plot *Received Network Usage*

Hasil menunjukkan bahwa seluruh data yang diambil dari keempat *library* pada setiap kasus uji memiliki nilai identik tanpa variasi. Pada kasus uji P100 dan G100 dengan jumlah data row 100, semua nilai identik sebesar 76715 byte.. Fenomena serupa terlihat pada kasus uji P1000 dan G1000 dengan jumlah data row 1000, di mana semua nilai identik sebesar 772137 byte. Untuk kasus uji dengan jumlah data lebih besar, seperti P5000 dan G5000, semua nilai yang diperoleh adalah 3670004 byte. Demikian pula, pada kasus uji P10000 dan G10000 dengan jumlah data 10000, semua nilai identik sebesar 7474426 byte.

## 4.2 Pengolahan Data dan Analisis Hasil

### 4.2.1 Uji Normalitas

Uji normalitas data dilakukan menggunakan uji Shapiro-Wilk pada perangkat lunak R Studio. Uji ini bertujuan untuk menentukan apakah data berdistribusi normal atau tidak. Hasil dari uji ini bertujuan untuk menentukan metode analisis statistik yang akan digunakan dalam uji beda. Jika data terbukti berdistribusi normal, maka metode uji beda yang sesuai adalah ANOVA, yang dirancang untuk data dengan distribusi normal. Sebaliknya, jika data tidak berdistribusi normal, maka analisis akan dilakukan menggunakan Kruskal-Wallis, yang lebih cocok untuk data dengan distribusi non-parametrik. Adapun hipotesis uji uji Shapiro-Wilk dengan tingkat signifikansi sebesar 0,05 adalah sebagai berikut:

- Hipotesis Nol ($H_0$): Jika nilai probabilitas (p-value) > tingkat signifikansi ($\alpha$), maka populasi data terdistribusi secara normal.
- Hipotesis Alternatif ($H_1$): Jika nilai probabilitas (p-value) ≤ tingkat signifikansi ($\alpha$), maka populasi data terdistribusi secara tidak normal.

Pada data *received network usage*, hasil menunjukkan nilai yang selalu identik untuk setiap permintaan dengan jumlah request data yang sama. Oleh karena itu, uji normalitas dan uji beda tidak dilakukan pada data ini. Sebaliknya, untuk data *transmitted network usage* pada kasus uji P100 dan G100, meskipun uji normalitas tidak dilakukan, uji beda tetap dilaksanakan dengan asumsi bahwa data tidak berdistribusi normal. Uji Kruskal-Wallis dipilih sebagai metode analisis yang sesuai untuk kasus uji P100 dan G100.

Tabel 3 Hasil Uji Normalitas Pada *Response Time*

| Kode | Retrofit | Ktor Client | OkHttp | Volley |
|---|---|---|---|---|
| P100 | 0.00028 | 1.551e-05 | 0.76134* | 4.55e-05 |
| P1000 | 8.44e-06 | 5.170e-05 | 0.00278 | 1.66e-05 |
| P5000 | 1.34e-05 | 3.390e-05 | 0.63535* | 0.00136 |
| P10000 | 0.05062* | 0.07934* | 0.91866* | 0.00013 |
| G100 | 1.87e-06 | 0.14199* | 6.8e-05 | 4.2e-06 |
| G1000 | 4.59e-06 | 0.00133 | 0.01118 | 0.00399 |
| G5000 | 0.00044 | 0.00155 | 0.03713 | 0.77592* |
| G10000 | 749e-05 | 0.52160* | 0.16470* | 0.30018* |

Tabel 4 Hasil Uji Normalitas Pada *Memory Usage*

| Kode | Retrofit | Ktor Client | OkHttp | Volley |
|---|---|---|---|---|
| P100 | 0.00289 | 0.00685 | 0.12922* | 0.00046 |
| P1000 | 0.21055* | 0.17929* | 3.392e-05 | 0.1623* |
| P5000 | 0.03488 | 0.00030 | 1.275e-07 | 0.00136 |
| P10000 | 5.88e-06 | 8.933e-06 | 1.184e-06 | 2.7e-05 |
| G100 | 0.002589 | 0.00892 | 2.717e-06 | 6.3e-05 |
| G1000 | 0.049131 | 0.00165 | 9.597e-06 | 0.6664* |
| G5000 | 0.008001 | 3.477e-05 | 0.00023 | 0.00723 |
| G10000 | 5.82e-08 | 1.235e-05 | 6.896e-05 | 9.2e-05 |

Tabel 5 Hasil Uji Normalitas Pada *Transmitted Network Usage*

| Kode | Retrofit | Ktor Client | OkHttp | Volley |
|---|---|---|---|---|
| P100 | - | - | - | - |
| P1000 | 3.783e-06 | 5.1284e-05 | 2.4e-05 | 2.67e-05 |
| P5000 | 8.18e-11 | 0.25351* | 0.00160 | 4.25e-07 |
| P10000 | 4.81e-09 | 9.03e-10 | 2.6e-06 | 5.7e-08 |
| G100 | - | - | - | - |
| G1000 | 0.003838 | 1.24e-08 | 0.01567 | 0.00053 |
| G5000 | 0.294229* | 6.36e-07 | 0.00613 | 0.00149 |
| G10000 | 5.61e-11 | 2.44e-08 | 0.01620 | 0.57955* |

Hasil uji normalitas untuk data *response time*, *memory usage* dan *transmitted network usage* dapat dilihat pada Tabel 3, 4, dan 5. Beberapa data p-value pada tabel-tabel tersebut ditandai dengan tanda bintang (*), yang menunjukkan bahwa data tersebut berdistribusi normal. Namun, secara keseluruhan, semua kasus uji dianalisis menggunakan uji beda Kruskal-Wallis.

### 4.2.1 Uji Beda

Hasil uji normalitas menunjukkan bahwa seluruh kasus uji akan dianalisis menggunakan uji beda Kruskal-Wallis. Uji ini dilakukan menggunakan bahasa pemrograman R pada

perangkat lunak RStudio. Uji beda Kruskal-Wallis menghasilkan nilai p-value yang menjadi indikator adanya perbedaan signifikan. Adapun hipotesis uji beda Kruskal-Wallis dengan tingkat signifikansi sebesar 0,05 adalah sebagai berikut:

- Hipotesis Nol ($H_0$): Jika nilai probabilitas (p-value) > tingkat signifikansi ($\alpha$), maka tidak ada perbedaan yang signifikan.
- Hipotesis Alternatif ($H_1$): Jika nilai probabilitas (p-value) ≤ tingkat signifikansi ($\alpha$), maka terdapat perbedaan yang signifikan.

Tabel 6 Hasil Uji Beda Pada *Response Time*

| Kode | Uji Beda | *p-value* |
|---|---|---|
| P100 | Kruskal-Wallis | < 2.2e-16* |
| P1000 | Kruskal-Wallis | < 2.2e-16* |
| P5000 | Kruskal-Wallis | < 2.2e-16* |
| P10000 | Kruskal-Wallis | < 2.2e-16* |
| G100 | Kruskal-Wallis | < 2.2e-16* |
| G1000 | Kruskal-Wallis | < 2.2e-16* |
| G5000 | Kruskal-Wallis | < 2.2e-16* |
| G10000 | Kruskal-Wallis | < 2.2e-16* |

Tabel 7 Hasil Uji Beda Pada *Memory Usage*

| Kode | Uji Beda | *p-value* |
|---|---|---|
| P100 | Kruskal-Wallis | 2.562e-07* |
| P1000 | Kruskal-Wallis | 2.234e-15* |
| P5000 | Kruskal-Wallis | < 2.2e-16* |
| P10000 | Kruskal-Wallis | < 2.2e-16* |
| G100 | Kruskal-Wallis | 3.795e-11* |
| G1000 | Kruskal-Wallis | 0.0005385* |
| G5000 | Kruskal-Wallis | < 2.2e-16* |
| G10000 | Kruskal-Wallis | 3.109e-14* |

Tabel 8 Hasil Uji Beda Pada *Transmitted Network Usage*

| Kode | Uji Beda | *p-value* |
|---|---|---|
| P100 | Kruskal-Wallis | < 2.2e-16* |
| P1000 | Kruskal-Wallis | < 2.2e-16* |
| P5000 | Kruskal-Wallis | < 2.2e-16* |
| P10000 | Kruskal-Wallis | < 2.2e-16* |
| G100 | Kruskal-Wallis | < 2.2e-16* |
| G1000 | Kruskal-Wallis | < 2.2e-16* |
| G5000 | Kruskal-Wallis | < 2.2e-16* |
| G10000 | Kruskal-Wallis | < 2.2e-16* |

Hasil uji beda untuk data *response time*, *memory usage*, dan *transmitted network usage* dapat dilihat pada Tabel 4.10, 4.11, dan 4.12. Pada tabel-tabel tersebut, data p-value yang menunjukkan adanya perbedaan signifikan antar kelompok ditandai dengan tanda bintang (*). Analisis ini menunjukkan bahwa seluruh kasus uji memiliki perbedaan yang signifikan antar kelompok.

## 4. KESIMPULAN

Kesimpulan yang dapat diambil dari hasil uji beda yang dilakukan adalah bahwa terdapat perbedaan signifikan antara keempat *library* yang diuji dalam hal *response time*, *memory usage*, dan *transmitted network usage*. Berdasarkan analisis rata-rata dan median, hasil pengujian menunjukkan bahwa OkHttp unggul dalam hal *response time*, di mana *library* ini memberikan waktu respons yang lebih cepat dibandingkan dengan Ktor Client, Retrofit, dan Volley pada seluruh kasus uji.

Pada *memory usage*, Retrofit menunjukkan penggunaan memori yang lebih efisien pada seluruh kasus uji. Ini berarti Retrofit lebih hemat dalam penggunaan memori dibandingkan dengan ketiga library lainnya pada berbagai skenario uji, yang tentunya penting untuk aplikasi dengan keterbatasan memori. Pada *network usage*, hasilnya terbagi menjadi dua kategori yaitu *transmitted* dan *received network usage*.

Pada *transmitted network usage*, Ktor Client menunjukkan penggunaan jaringan yang lebih efisien pada hampir seluruh kasus uji, kecuali pada kasus uji P100 dan G100. Ini menunjukkan bahwa Ktor Client dapat mengelola pengiriman data lebih efisien pada sebagian besar skenario uji. Namun, untuk *received network usage*, hasil pengujian menunjukkan bahwa seluruh data yang diambil

dari keempat library pada setiap kasus uji memiliki nilai yang identik, tanpa adanya variasi meskipun dilakukan pengambilan data yang sama beberapa kali. Hal ini menunjukkan bahwa pemilihan library tidak mempengaruhi *received network usage*.

Secara keseluruhan, Retrofit dapat dianggap sebagai *library* yang paling baik meskipun tidak menjadi yang terbaik dalam hal *response time* dan *transmitted network usage*. Pada *response time*, Retrofit berada di urutan kedua setelah OkHttp. Hal yang sama juga berlaku pada *transmitted network usage*, di mana Retrofit berada di urutan kedua setelah Ktor Client. Dengan demikian, meskipun Retrofit tidak berada di urutan pertama dalam dua aspek tersebut, kinerjanya yang baik pada aspek *response time* dan *transmitted network usage* menjadikannya pilihan yang sangat baik secara keseluruhan.

## 5. DAFTAR PUSTAKA


Belkhir, A., Abdellatif, M., Tighilt, R., Moha, N., Gueheneuc, Y.-G., & Beaudry, E. (2019). An observational study on the state of REST API uses in Android mobile applications. *2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. https://doi.org/10.1109/mobilesoft.2019.00020

Bhade, J., & Yadav, H. (2019). Evaluation of Android Networking Libraries. In *International Journal of Scientific Research & Engineering Trends* (Vol. 5).

Ghag, O. M. (2024). Optimizing app memory usage in Android smartphones. *International Journal of Science and Research (IJSR)*, *13*(1), 464–466. https://doi.org/10.21275/sr24104104037

Jetbrains. (2024). *Create a client application: KTOR*. Ktor Help. https://ktor.io/docs/client-create-new-application.html

Kemp, S. (2024). *Digital 2024: Indonesia - DataReportal – global digital insights*. DataReportal. https://datareportal.com/reports/digital-2024-indonesia

Lachgar, M., Benouda, H., & Elfirdoussi, S. (2018). Android Rest Apis: Volley vs retrofit. *2018 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, *22*, 1–6. https://doi.org/10.1109/isaect.2018.8618824

McKight, P. E., & Najab, J. (2010). Kruskal-Wallis Test. *The Corsini Encyclopedia of Psychology*, 1–1. https://doi.org/10.1002/9780470479216.corpsy0491

Mohd Razali, N., & Bee Wah, Y. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. In *Journal of Statistical Modeling and Analytics* (Vol. 2).

Rosen, S., Nikravesh, A., Guo, Y., Mao, Z. M., Qian, F., & Sen, S. (2015). Revisiting network energy efficiency of Mobile Apps. *Proceedings of the 2015 Internet*

Saputra, K., Farhan, K., & Irvanizam, I. (2018). Analysis on the comparison of retrofit and volley libraries on Android-based Mosque Application. *2018 International Conference on Electrical Engineering and Informatics (ICELTICs)(44501)*, 117–121. https://doi.org/10.1109/iceltics.2018.8548881

Sawyer, S. F. (2009). Analysis of variance: The Fundamental Concepts. *Journal of Manual & Manipulative Therapy*, *17*(2). https://doi.org/10.1179/jmt.2009.17.2.27e

StatCounter Global Stats. (2024). *Mobile Operating System Market Share Indonesia*. https://gs.statcounter.com/os-market-share/mobile/indonesia

Square, Inc. (2019). *OKHTTP*. Overview - OkHttp. https://square.github.io/okhttp/

Zhou, R., Shao, S., Li, W., & Zhou, L. (2016). How to define the user's tolerance of response time in using mobile applications. *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 281–285. https://doi.org/10.1109/ieem.2016.7797881