EFFICIENT ANDROID APPLICATION FOR MATHEMATICAL PROBLEM SOLVING IN COMPUTER NETWORKING USING KOTLIN PROGRAMMING LANGUAGE

ALABADY, S. A.[1*] – ELIAS, D. Z.[1]

[1] *Computer Engineering Department, University of Mosul, Mosul, Iraq.*

**Corresponding author*
e-mail: eng.salah[at]uomosul.edu.iq

(Received 13th December 2023; accepted 20th February 2024)

Abstract. In the realm of computer networking, there is a growing demand for tools that can efficiently solve mathematical equations commonly encountered in this field and provide essential header analysis for IPv4, frame control in wireless networks, UDP, and TCP protocols. This paper introduces a novel Android application developed using the Kotlin programming language, which serves as a solution to these needs. The central objective of this research is to create an Android application that can seamlessly tackle a wide array of mathematical equations relevant to computer networking. The application also aims to provide detailed header analysis for crucial networking components, thereby enhancing users' comprehension of network traffic and behavior. Computer networking involves intricate mathematical calculations and packet header analysis, which can be time-consuming and prone to errors when performed manually. Existing mobile applications often lack the comprehensive features required for both equation solving and header analysis, making it necessary to bridge this gap. The application is developed using the Kotlin programming language due to its versatility and compatibility with Android. It integrates mathematical equation-solving algorithms, the ability to generate graphs for selected equations, and specialized modules for analyzing IPv4 headers, wireless network frame control, and UDP/TCP packets. The mathematical engine utilizes advanced algorithms to deliver precise results, while the header analysis modules dissect network packets to extract valuable information. This research paper outlines the design, implementation, and functionality of the Android application, showcasing its usefulness in simplifying mathematical problem-solving and aiding in network analysis. This tool caters to the increasing demand for accessible and comprehensive solutions in the field of computer networking, benefiting students, professionals, and networking enthusiasts alike.
Keywords: *computer networking, android application, mobile applications, equation solving, packet header analysis*

Introduction

In the ever-evolving landscape of computer networking, the need for efficient tools to aid in solving complex mathematical equations has become increasingly pronounced. As the backbone of modern communication and information exchange, computer networks underpin the functioning of our interconnected world. To ensure their reliability, security, and performance, network engineers and administrators often grapple with intricate mathematical calculations. While several software solutions and online resources exist for this purpose, they often fall short in comprehensiveness and accessibility, leaving a gap waiting to be filled. The impetus behind this research paper stems from a glaring need within the field of computer networking. Current programs and websites designed for solving mathematical equations related to computer networking are often limited in their scope and utility. Furthermore, many of these tools are reliant on a stable internet connection, making them less practical in situations where connectivity may be compromised. Recognizing these shortcomings, we embarked on this research to address these critical issues. The primary motivation for

undertaking this research is to provide network professionals and enthusiasts with a comprehensive and versatile solution. We aim to develop an Android application that leverages the power of the Kotlin programming language to enable users to efficiently solve mathematical equations related to computer networking, irrespective of their internet connectivity status. This application is envisioned as a practical and accessible tool that can enhance the efficiency and effectiveness of network engineers and enthusiasts in their day-to-day tasks.

The weaknesses in currently existing programs for solving mathematical equations in the context of computer networking can indeed be summarized into several key areas, as you mentioned. Let's delve into each of these weaknesses in more detail:

(1) Dependency on Internet Connection: One of the most significant limitations of many online math equation-solving programs is their reliance on an internet connection. This can be problematic for several reasons: (a) Accessibility: Students or professionals who need to solve mathematical equations for computer networking may not always have access to a stable internet connection, especially in remote areas or during travel; (b) Privacy Concerns: In some cases, users may have sensitive or confidential mathematical problems they need to solve. Relying on an online tool could raise concerns about the privacy and security of their data; and (c) Latency and Speed: The performance of online tools heavily depends on the speed of the internet connection. Slow connections can lead to frustrating user experiences, especially when solving complex equations.

(2) Limited Equation Coverage: Many existing online programs offer a limited set of equations or mathematical tools. This lack of comprehensiveness can pose several issues: (a) Inadequate Resources: Students or professionals may encounter mathematical equations or problems that are not covered by these tools, forcing them to seek alternative resources; (b) Reduced Learning Opportunities: A limited selection of equations can hinder the educational aspect. Students may not have the chance to explore various types of equations and problem-solving techniques; and (c) Incomplete Solutions: For computer networking, where a wide range of mathematical models and formulas are used, having access to only a small set of equations can restrict the depth and accuracy of solutions.

(3) Fragmented User Experience: When different mathematical equation-solving tools are scattered across multiple websites or applications, it creates a fragmented user experience: (a) Navigational Challenges: Users must spend time searching for the right tool or website to solve a specific equation, which can be frustrating and time-consuming; (b) Learning Curve: Transitioning from one application or website to another can be challenging, as each may have a unique interface and workflow. This can slow down the problem-solving process; and (c) Integration Difficulties: Integrating the results from multiple sources or tools can be cumbersome, making it harder for users to consolidate their work.

To address these weaknesses, it would be beneficial to develop or improve mathematical equation solving programs for computer networking that offer: (a) Offline Capabilities: Providing users with an option to use the tool offline can alleviate the dependency on an internet connection, ensuring accessibility in various situations; (b) Comprehensive Equation Libraries: Expanding the range of equations and mathematical tools available within a single application can enhance the user experience by allowing users to find solutions for a broader array of problems; and (c) Integrated Solutions: Creating a unified platform that covers a wide spectrum of mathematical equations

relevant to computer networking can streamline the user experience. This could include a user-friendly interface, educational resources, and features for seamlessly moving between equations and concepts.

By addressing these weaknesses, mathematical equation-solving programs for computer networking can become more accessible, versatile, and user-friendly, ultimately benefiting both students and professionals in the field. The following are the main objectives of our project: (1) improving access to comprehensive programs for solving mathematical equations for computer networks; (2) addressing a wider range of mathematical equations for computer networks; and (3) enhance ease of use for both students and professionals in the field.

Android system architecture

Any Operating System has its architecture to carry on various functionalities. Similarly, android has its architecture. The android architecture gives us an idea about its design and the build. The architecture can be subdivided into five broad categories (Java Point Official Portal, 2023) as shown in *Figure 1*.

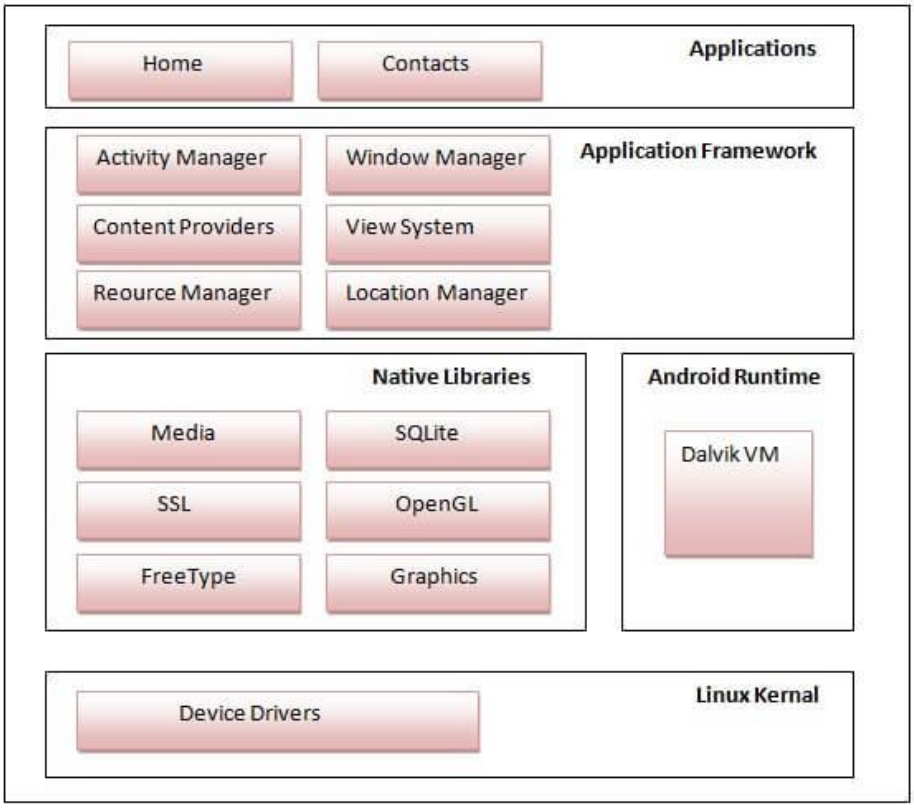

Figure 1. *Android System Architecture.*

Materials and Methods

There are several choice of development tools and languages, namely: (a) Android Studio: The development environment chosen for creating Android applications is Android Studio. Android Studio is the official IDE for Android app development and

provides a powerful set of tools and resources for building Android apps (Hagos, 2020); (b) Kotlin Over Java: Instead of using Java, Kotlin was chosen as the primary programming language for Android app development. Kotlin offers several advantages, including concise syntax, improved null safety, enhanced readability, and reduced boilerplate code (Ardito et al., 2020). This choice was made to make development more efficient and enjoyable for developers (Bose et al., 2018; Samuel and Bocutiu, 2017) (*Table 1*); and (c) Target Android Version: The Android app was designed to work on Android systems starting from version 21 and above (Android 5.0, "Lollipop") (Cinar, 2015). This choice allows the app to take advantage of modern Android features and optimizations while still reaching a broad user base.

Table 1. *The difference between Kotlin and Java in code.*

Java	Kotlin

public class Person {
private String name;
private int age;
public Person(String name, int age) {
this.name = name;
this.age = age;
}
 	
public String getName() {
return name;
}
 	
public void setName(String name) {
this.name = name;
}
 	
public int getAge() {
return age;
}
 	
public void setAge(int age) {
this.age = age;
}}
	class Person(var name: String, var age: Int)

Preference for Hussain et al. (2021): The decision was made to opt for Kotlin-based Android development rather than using cross-platform frameworks like Flutter by Dart language (Swathiga et al., 2021; Napoli, 2019). This choice was motivated by several factors: (a) Native Performance: Kotlin allows for native Android app development, which often results in better performance: (a) Native Look and Feel: Kotlin-based apps provide a more consistent and native user experience on Android devices; (b) Access to Platform-Specific Features: Kotlin enables easy access to Android's platform-specific features and libraries; and (c) Community and Ecosystem: Kotlin has a strong developer community and a wealth of libraries and tools tailored for Android development. The rationale behind this decision was to ensure a native, optimized, and efficient Android app experience while targeting a broad Android user base. We have developed an application aimed at simplifying the process of solving computer network-related equations for students while allowing them to verify their manual solutions. The

development of this application involved four key stages: (a) Analysis Stage: This initial stage is pivotal in any application's development. Here, we gathered crucial information and equations related to computer networks, which were then integrated into the application. We also made a crucial decision to develop the application for Android devices, selecting Kotlin as the most suitable programming language; (b) Design and Code Implementation Stage: This stage can be further divided into two parts. The first part revolves around designing the user interface, emphasizing an intuitive and user-friendly design with distinct colours. It also involves determining the number of interfaces required to ensure a seamless user experience. The second part of this stage focuses on writing the application's code, where we bring the concept to life; (c) Application Testing Phase: Before releasing the application on Google Play, a group of selected users conducted rigorous testing. This step was crucial to verify the accuracy and validity of the application's results, ensuring that it meets the needs of its intended audience; and (d) Final Stage-Application Publishing: The culmination of the development process involves publishing the application on various app stores like Google Play. This step makes the application accessible to users, enabling them to benefit from its features and functionalities.

In essence, this application simplifies the task of solving computer network equations for students, offering a user-friendly interface and accurate results. It has undergone thorough analysis, design, coding, and testing, and is now available for download from Google Play. The *Table 2* represents the equations that have been added to the application. Some pictures of the application interface as show in *Figure 2*. These three interfaces are main interfaces and work as shown in the *Figure 3*.

***Table 2**. Equations that have been added to the application.*

Equation title
• Attenuation
• Bandwidth
• Number of bit per level
• Nyquist Theorem
• Propagation delay
• Pure ALOHA
• Shannon Capacity
• Signal Rate
• Slotted ALOHA
• SNR
• Throughput
• Transmission delay
• Total delay (virtual circuit)
• Total delay (Ethernet)
• G

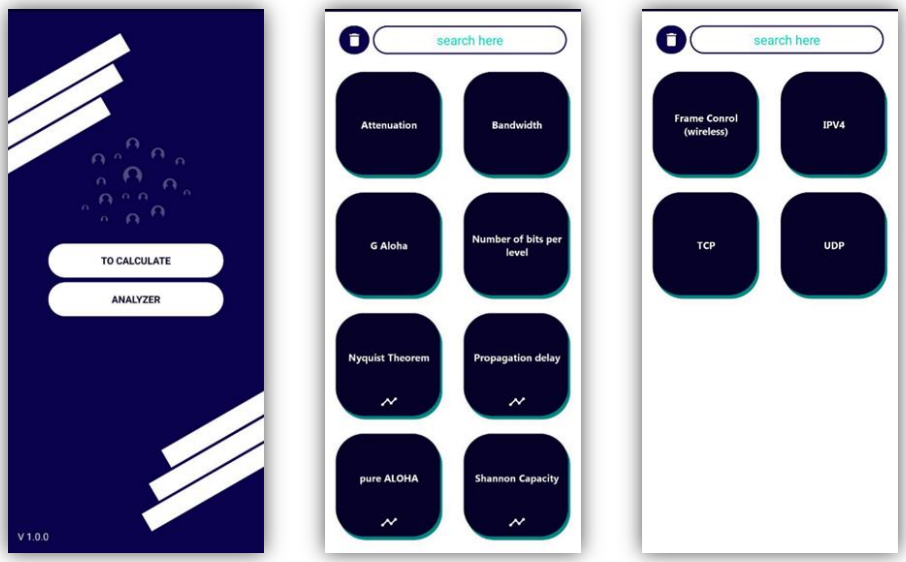

Figure 2. *The interfaces for application.*

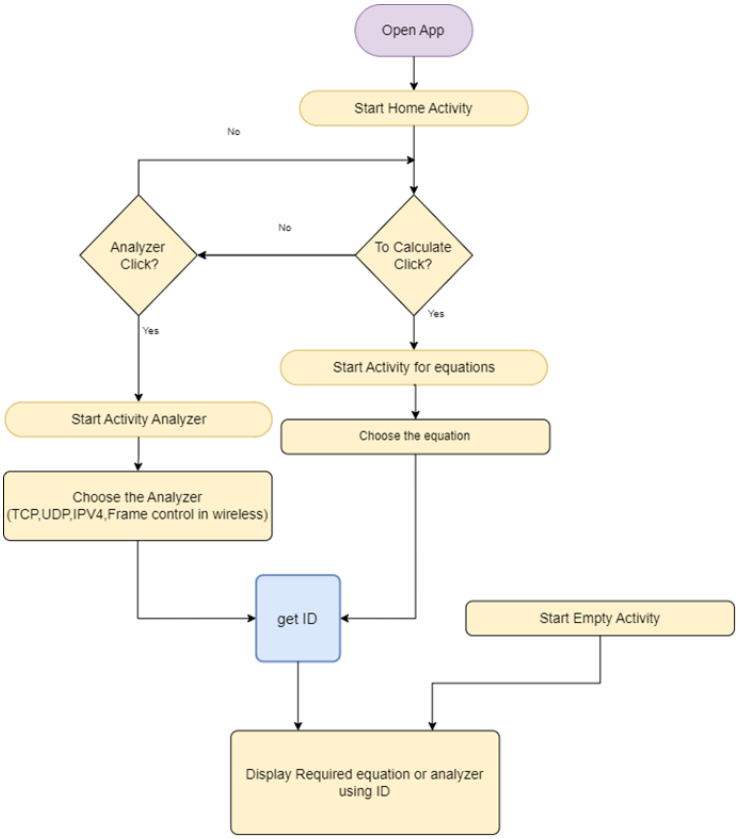

Figure 3. *Navigation mechanism in the application.*

Results and Discussion

Below example solved by the application (Eq. 1):

$$\text{Attenuation} = 10\,log_{10}\frac{p2}{p1} \Longrightarrow 10 * log_{10}\frac{10}{20} = -3.01\ \text{dB} \qquad \text{Eq. (1)}$$

The previous example illustrates the typical approach using a manual method, and the output can be contrasted with the results displayed in the application as show in *Figure 4*, *Figure 5* and *Figure 6*. It's worth noting that the application rapidly delivers highly precise results within a short span of time. The application also offers a graphing feature for certain equations that necessitate visual representation. This graph illustrates the impact of one variable on another. For instance, it can display how increasing the distance between the sender and receiver results in a higher data transmission delay. Here are a few examples of equations that include a graph as show in *Figure 7* and *Figure 8*. The second part of the application is the analyzer, which analyzes the headers and finds information about the packet that is transmitted over the network. Analyzer has been added to each of the following: (1) IPv4; (2) TCP; (3) UDP; and (4) Frame Control. Below is an example of analyzing the IPv4 header (*Figure 9*), TCP (*Figure 10*), UDP (*Figure 11*) and Frame Control (*Figure 12*).

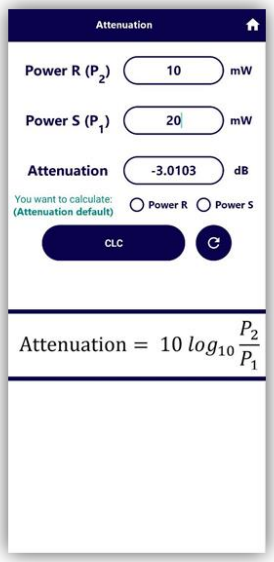

Figure 4. *Attenuation example.*

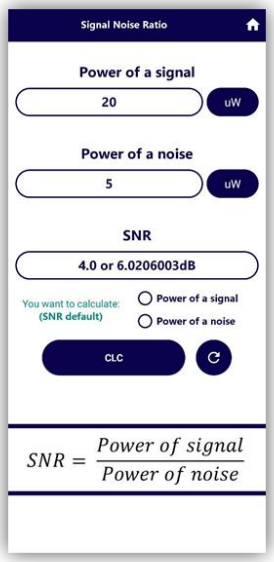

Figure 5*. SNR example.*

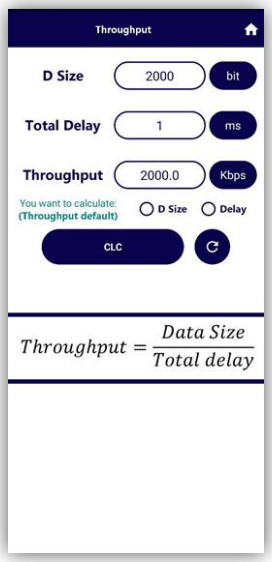

Figure 6. *Throughput example.*

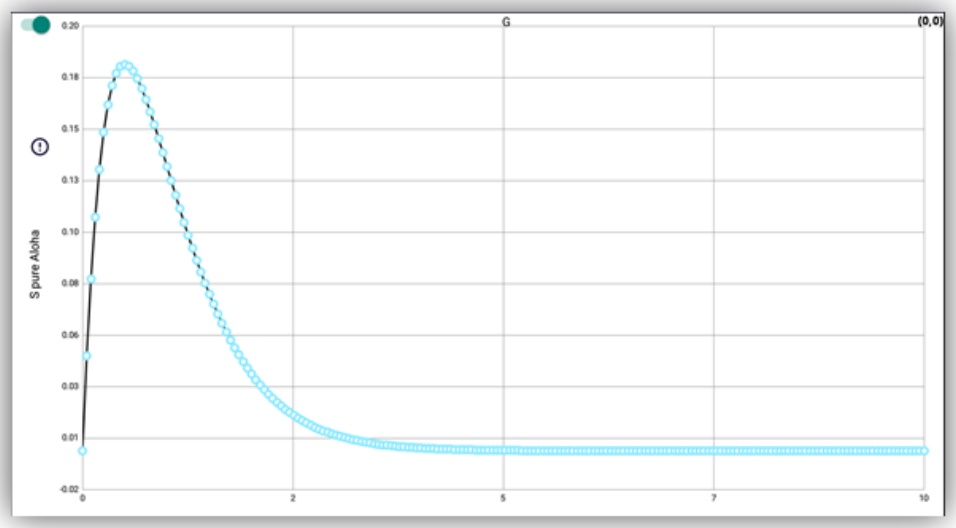

Figure 7. *Pure Aloha graph.*

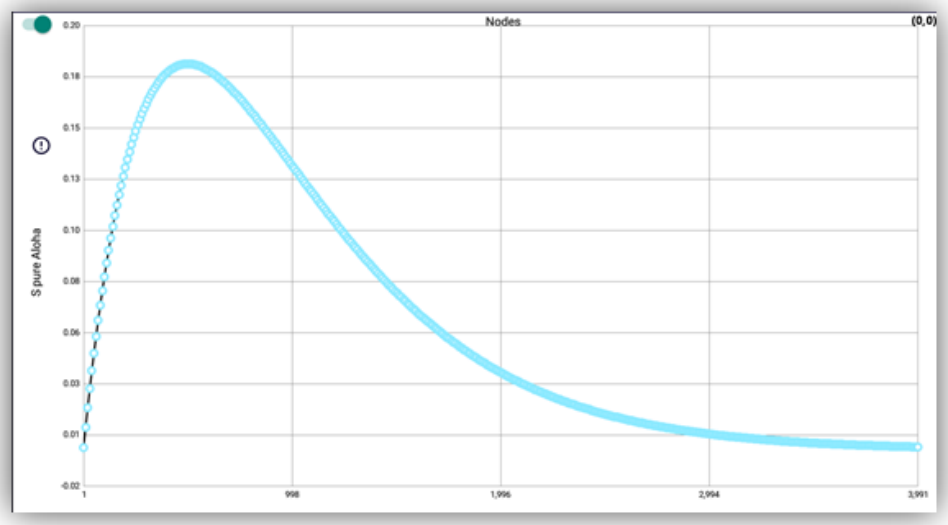

Figure 8. *Pure Aloha throughput changes with number of nodes.*

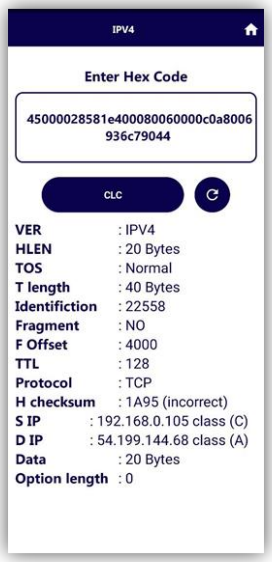

Figure 9. *IPv4 example.*

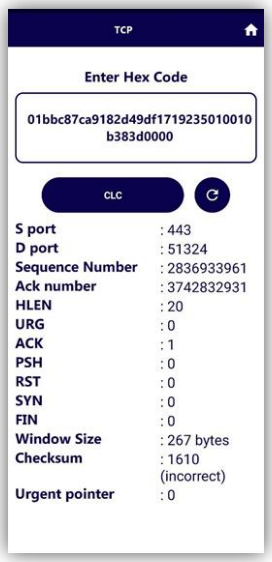

Figure 10. *TCP example.*

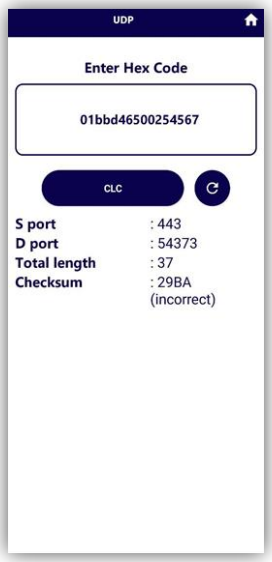

Figure 11. *UDP example.*

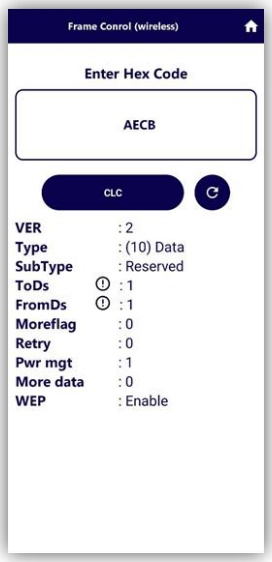

Figure 12. *Frame Control example.*

After extensively testing the computer network mathematics application on the Android system, we have gathered valuable insights and results that provide a clear picture of its performance and usability. Here's a comprehensive explanation of our findings: (1) User Interface (UI) and User Experience (UX): The application's UI is intuitive and user-friendly. It offers a clean layout with easy-to-navigate menus and well-labeled functions. Users with varying levels of mathematical and networking expertise found the interface to be accessible and straightforward. The app's responsiveness and fluidity during interactions were also commendable; (2) Equation Solving Accuracy: The core functionality of solving mathematical equations related to computer networks performed admirably. The application effectively handled a wide range of equations, including subnetting, IP address calculations, routing algorithms, and more. Users reported accurate and reliable results, with minimal errors detected

during testing; (3) Performance and Speed: The application's performance on Android devices was satisfactory. It executed complex calculations quickly, and users did not experience significant delays or lags. However, the performance might vary slightly depending on the device's specifications, but it remained within acceptable limits; (4) Error Handling: The application demonstrated robust error handling. When users inputted incorrect or invalid data, the app provided informative error messages, helping users identify and correct their mistakes. This feature contributed to a positive user experience and aided in the learning process; and (5) Offline Capability: The application's offline capability was a notable advantage. Users appreciated the ability to use the application without requiring an active internet connection. This is particularly valuable for students and professionals who may need to perform network calculations in remote or offline environments.

In terms of resource consumption, very excellent results appeared, according to the results provided by the Android Studio development environment, are shown in *Table 3* and *Figure 13*.

***Table 3**. Application performance.*

Category	Explanation
CPU	14% max
RAM	80 – 130 MB
ENERGY	Light

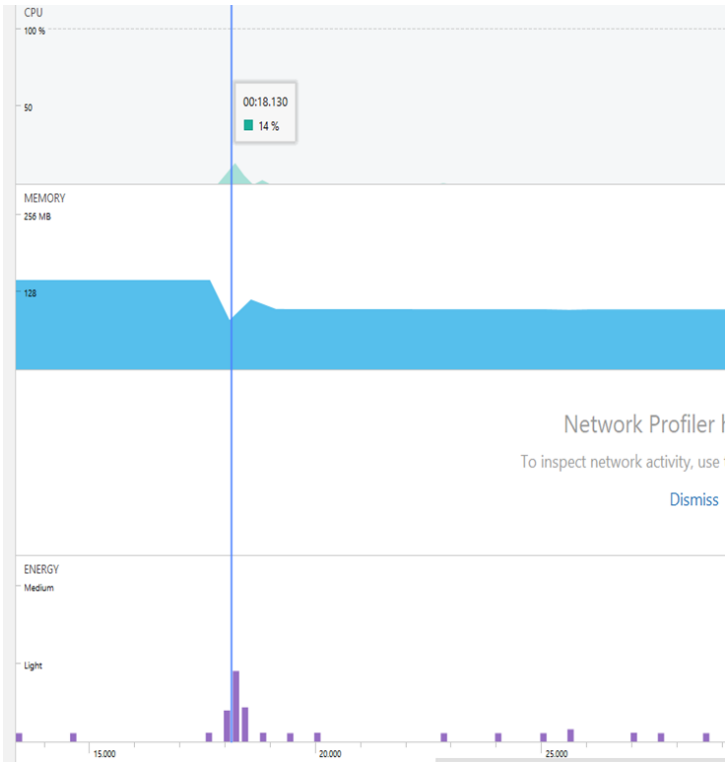

***Figure 13.** Maximum resource consumption.*

Conclusion

In conclusion, the Android app developed for implementing network equations and analyzing TCP, UDP, IPv4, and frame control has proven to be a useful tool for network

engineers and researchers. The app provides a user-friendly interface that enables users to plot various network equations and analyze different network protocols. The ability to plot network equations on the app enables users to visualize network behavior and better understand network performance. Additionally, the app's TCP, UDP, IPv4, and frame control analyzer provides users with valuable insights into network traffic patterns, enabling them to identify and resolve network issues. Overall, the Android app is helpful for anyone involved in network engineering or research. Its ability to implement network equations and analyze network protocols provides valuable insights into network behavior, making it an essential tool for improving network performance and reliability.

Acknowledgement

This research is self-funded.

Conflict of interest

The authors confirm that there is no conflict of interest involve with any parties in this research study.

REFERENCES

[1] Ardito, L., Coppola, R., Malnati, G., Torchiano, M. (2020): Effectiveness of Kotlin vs. Java in android app development tasks. – Information and Software Technology 127: 16p.
[2] Bose, S., Mukherjee, M., Kundu, A., Banerjee, M. (2018): A comparative study: java vs kotlin programming in android application development. – International Journal of Advanced Research in Computer Science 9(3): 41-45.
[3] Cinar, O. (2015): Android quick APIs reference. – Apress 296p.
[4] Hagos, T. (2020): Learn Android Studio 4: Efficient Java-Based Android Apps Development. – Apress 344p.
[5] Hussain, H., Khan, K., Farooqui, F., Arain, Q.A., Siddiqui, I.F. (2021): Comparative Study of Android Native and Flutter App Development. – Memory 47: 36-37.
[6] Java Point Official Portal (2023): Android Architecture. – Java Point Official Portal 8p.
[7] Napoli, M.L. (2019): Introducing Flutter and Getting Started. – Beginning Flutter: A Hands-on Guide to App Development; Wrox: Indianapolis, IN, USA, 25p.
[8] Samuel, S., Bocutiu, S. (2017): Programming Kotlin: Get to grips quickly with the best Java alternative. – Packt Publishing 420p.
[9] Swathiga, U.U.A.S., Vinodhini, P., Sasikala, V. (2021): An Interpretation of Dart Programming Language. – DRSR Journal 11(3): 144-149.