Iraqi Journal of Information and Communications Technology(IJICT)
Vol. 4, Issue 3, December 2021

ISSN:2222-758X
e-ISSN: 2789-7362

SINGLE HOP AUTHENTIC BROADCASTING IN SOCIAL MOBILE NETWORKS USING BLUETOOTH

Basma A. Qasim [1], Emad H. Al-Hemiary [2]
[1,2] College of Information Engineering, Al-Nahrain University, Baghdad, Iraq
{Basma.ahmed, emad}@coie-nahrain.edu.iq [1,2]
Received:10/3/2021, Accepted:31/3/2021

***Abstract*- This paper aims to create infrastructure-free networking capabilities for people in need such as when inadequate infrastructure including mobile cell towers and power supplies are damaged or when all communication means are restricted. In general, this means finding a way to communicate without mobile cell towers, repeaters, or Wi-Fi hotspots on mobile devices. This paper focuses on introducing an ad-hoc network based on Bluetooth where users can communicate without barriers. The proposed network implementation uses Google's API Nearby Connection which is one of Google Play Services that can provide high bandwidth, low latency, encrypted data transfers between nearby devices in a fully-offline P2P mode. Kotlin language was used to program an application called BlueEmergency installed on Android devices to broadcast data as bytes payloads in a form of text messages from one node to all nodes in proximity using Bluetooth only. The test and verification of the system are performed showed that successful communication is achieved with the latest versions of both the Android operating system and Bluetooth. Finally, the designed network can be extended to cover large geographical areas by multi-hopping the messages between devices to make the system more efficient to use in the time of need.**

keywords: **Ad-hoc network, Bluetooth, Broadcasting, Google API nearby connection, Kotlin.**

I. INTRODUCTION

Communication applications are very much widespread in today's world. WhatsApp, Telegram, Hangouts etc. are a trend in the application world, but all these communication applications utilize either cellular data which is a service you must pay for or Wi-Fi which is not always available and when available the connection strength varies wildly from place to place. Even when one wants to send a message to another person on the same floor or few feet away, they depend on the availability of Wi-Fi or cellular data. Our motive was to create a prototype application that facilitates communications in a small firm (Like schools, hospitals, or clinics) completely free of charge. To accomplish this Bluetooth is found to be the best technology to work with as it requires low power resulting in longer battery life [1]. It can be used within the range of 10 to 100 meters, which is reasonable within a small building. It is readily available as it's built-in in any Android device and the expense of setting up a Bluetooth network is much less in comparison to that of Wi-Fi. The introduced network design uses Google's API Nearby Connections [2] which is a broad framework developed by Google to support the development of pervasive, peer-to-peer mobile apps that uses Wi-Fi hotspots, Bluetooth LE & classic Bluetooth BR/EDR under the hood to discover and establish connections to nearby devices using the M to N Clustering topology to broadcast messages with Bluetooth only and without the need to pair the devices, and to ensure network security a hash function was implemented in the system and authentication tokens were exchanged in every connection occurring in the network [3].

II. RELATED WORK

Researches have been done on the subject of offline social mobile networks by utilizing Bluetooth enabled smartphones as it requires overcoming the limitation of the pairing process between the devices to broadcast information between

ISSN:2222-758X
e-ISSN: 2789-7362

devices. These researches can be summarized as follows: Joseph Paul Cohen, in 2013 [4] presented a method of wireless message dissemination without the need to trust other users. This technique utilizes the ability of modern wireless adapters to exchange knowledge about device name and identification. Using the scanning features built into Bluetooth and Wi-Fi, messages can be exchanged using their device names. Interchange of multiple messages using a user-defined scanning interval method along with a response-based framework to discoverable and non-discoverable devices. By selectively relaying messages, each user is responsible for their involvement in the ad hoc network. Martha and Hanh, in 2013 [5] presented a proposal for an ad hoc mobile phone network and a prototype system that developed routing algorithms to decide which routing protocols are well suited to reduce costs for such a network and build spontaneous ad hoc networks that enable users of mobile phones to directly link and exchange information using technologies such as Bluetooth. Amirta and Swarnabha, in 2014 [1] The python programming language has been used to build an Android messenger application that connects through using Bluetooth. A few ideas were discussed here, such as Bluetooth link between two or more Android smartphones, whereby users can speak to each other, as well as the Bluetooth connection between a server and an Android smartphone, however, users can update and synchronize chat records with the server at any time. Data structures are used against the respective usernames to store and update the data (messages). S finite expressions and state machines are used to achieve robustness, delivering error-free messages in short-distance communication. Tanweer and Mohammed, in 2016 [6] An Ad-Hoc network for communication between Android-based Wi-Fi devices were developed and introduced, allowing Android devices to connect with other devices using point-to-point communication without an access point, giving Android-based smart devices the ability to transmit data over the network of all active devices without access to the network. In contrast with Bluetooth networks, Wi-Fi systems are used for faster speeds over longer distances. Samir and Islam, in 2017 [7] A Bluetooth chat software, As an interactive and collaborative learning aid, researched, designed, built and tested, it has been created to enable students with disabilities to connect students without access to Wi-Fi or the Internet with their peers within the Bluetooth range. Users can give their friends free messages sitting in the classroom, school playgrounds and parties, while users of Bluetooth messaging can connect with anyone with a Bluetooth-enabled phone nearby. No internet connection is needed for the application and it only works with Bluetooth communication. Utku et al, in 2017 [8] proposed and evaluated three realistic schemes for building self-organizing and self-healing Wi-Fi Direct networks using Android devices. In terms of previous knowledge of the network and communication between the systems, experimental results indicate that the proposed methods are feasible with different overheads. The first known approaches to the automated production and maintenance of MANETs using everyday mobile devices were provided by these techniques. This paper aims to provide a method that is different to most established work in the field as the proposed work establishes a network of Bluetooth enabled devices to communicate fully-offline in a P2P manner without the need to pair between the devices while avoiding the challenging privacy issues by using a simple hash function to authenticate the users by exchanging keys to establish a secure channel in the network.

III. Google API Nearby Connection

The nearby Google Connections API [3] enables developers to offer services in their Android applications based on proximity. A nearby service allows users of the same application to only share data in the "nearby" area, for example within

ISSN:2222-758X
e-ISSN: 2789-7362

the radio range of Bluetooth. The API uses Bluetooth BR/EDR, Bluetooth LE and Wi-Fi, claiming to use the best features of each of them automatically according to the type of communication required. For example, Bluetooth is used in short, low-latency communications and wireless Internet for high-bandwidth medium-range communications. The API provides two different topologies (star and cluster) for connecting discoverers and advertisers using different strategies. As part of Google Play Services, the nearby connections API is implemented. Google Play Services is a proprietary, closed-source and vague library that enables Google to provide any Android device with the same services regardless of the operating system version. The API can be updated without the user interaction with any Android device, version 4.0 or later, and is updated by Google. The Nearby Connections API design and implementation details are not publicly available. The only public information source about the library is its documentation and a few blog posts from Google developers. The API uses default encryption but does not require user authentication according to these sources. The API manages several physical layers and a device can be a discoverer and an advertiser simultaneously and connect to several devices simultaneously. The application for a nearby connection is uniquely recognized by a service ID [9] string and clients and servers which are not able to connect with different services ID or connection strategies.

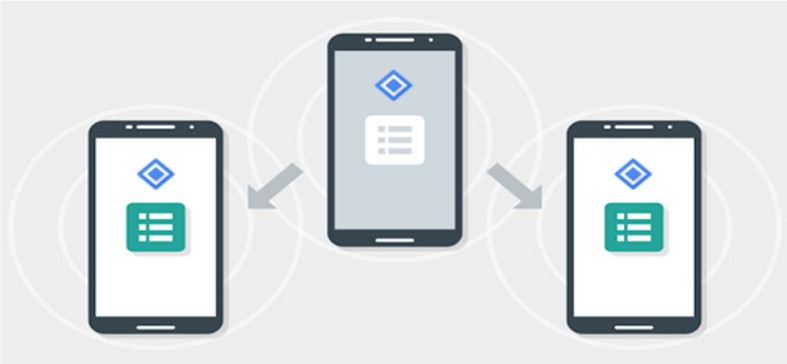

Figure 1: P2P nearby connected devices

IV. The Proposed Model

The presented network model consists of a network with mobile nodes that establishes a communication link with each node that is nearby within the Bluetooth radio range. Advertising, discovering, and sending a request for a connection is part of a phase called a pre-connection phase. If the link is successfully created, the next step of data sharing (often referred to as the post-connection phase) is symmetrical, i.e. , there is no difference between the advertiser and the discoverer as the device can advertise themselves and discover nearby devices simultaneously, the entire operation is depicted in Fig. 2. The initial step of the project setup was importing the Google Play Services library into the application and permitting AndroidManifest.xml file to access the location of the nearby devices, then the cluster strategy was chosen for the connection, this enabled the connection of unorganized clusters of devices within the Bluetooth radio range where each device can both initiate outgoing connections to (M) other devices and accept incoming connections from (N) other devices, then the pre-connection phase will begin when the devices have to start advertising to advertise themselves or start discovery

ISSN:2222-758X
e-ISSN: 2789-7362

if they want to be discovered this process is controlled by SDP (service discovery protocol) which provides a means for applications to discover which services are available and to determine the characteristics of those available services. As soon as the advertiser is discovered, its unique endpoint ID is passed as a parameter to the discovering node. Finally, for a discoverer to send an SDP connection request to the advertiser, it needs to request a connection and the connection key is being passed between the devices. In order to receive broadcast messages, the connection was made to be automatically accepted from any node that requests a connection in the network an SDP response was passed in the accept connection method. These nodes were distinguished by the uniquely identified string named service ID that is different from endpoint ID as it is implemented in the API package so that clients and servers with different service ID (or connection strategies) will not be able to connect to this network, this provides a layer of security to the application in addition to the SHA512 hash function that is used to generate a 128 random string key (message digest) to be exchanged by the two connected nodes to secure the connection. After the successful connection between the advertiser and discoverer, the post-connection phase starts and both devices can send data to each other using RFCOMM protocol which is a simple transport protocol used in Bluetooth communications. Sharing of data happens in the form of Payload objects and the bytes payload type was chosen for this type of communications, as soon as any one of the two (advertiser and discoverer) receives the very first byte of the incoming payload along with the generated message digest the connection is authenticated and a callback is triggered in a payload Received method. Whenever the payload is transferred completely, a payload transfer update method is called. The last step in this phase is calling the disconnect from the endpoint method whenever the connection is terminated between the devices and the endpoint lost method is triggered whenever a device disappears from the device's Bluetooth range. The workflow of the model is shown in Fig. 3.

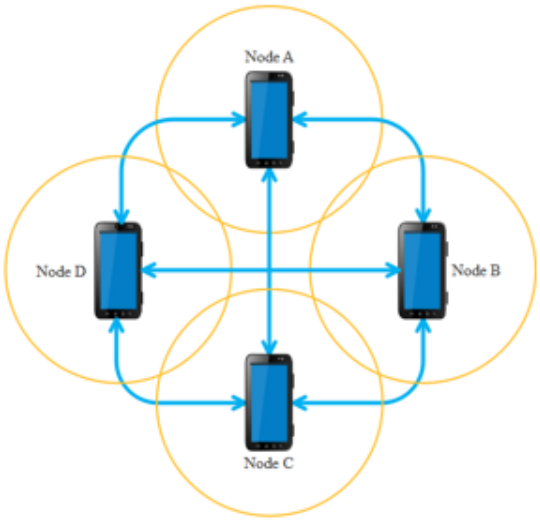

Figure 2: Network diagram of 4 mobile nodes

ISSN:2222-758X
e-ISSN: 2789-7362

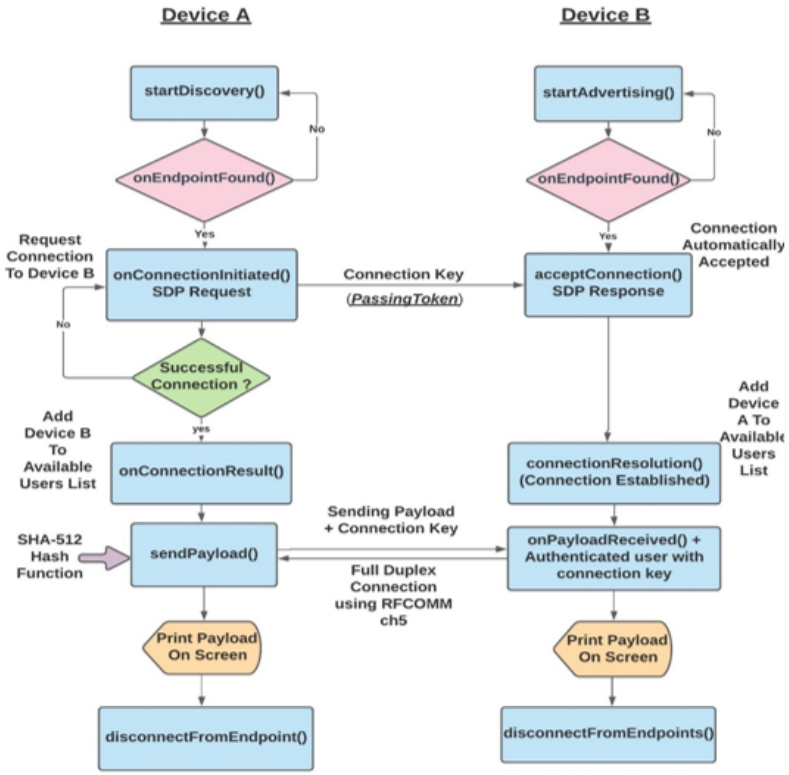

Figure 3: Workflow diagram of the application

V. Practical Implementation and Testing of The Network

Four smartphones (Samsung Galaxy Note 9, HTC U11, Samsung Galaxy S6 Edge and HTC One) with Devices named Device A, Device B, Device C, and Device D respectively as shown in Fig. 4, and Table I that shows the specification of these devices, these devices were utilized to implement this scenario to broadcast a message to all the nearby devices. For achieving this goal all those devices were connected via Bluetooth using the BlueEmergency software program its code was written in Kotlin programming language, the user interface was written using XML and was designed to be user friendly and easy to be used by anyone as shown in Fig. 5. The Nearby Connection API package which supports Bluetooth broadcasting was imported by Android Studio. These 4 devices were placed in a small area at the beginning of the test as the connection operation was done by initializing the application in the 4 nodes simultaneously as they were performing advertising and discovery in all of them at the same time and all of them were sending connection requests (SDP requests) to all the other nodes and accepting the connection (SDP response) from the other nodes, this resulted in slowing the connection process and in some nodes the connection never happened. Then the scenario was repeated by initializing the app in one device and then the others one by one and waiting for the connection to complete in two nodes and then the next one was introduced to the network until 4 connected nodes were capable of sending and receiving broadcast messages simultaneously. The software was initialized in the first device which is the Samsung galaxy note 9 named (Device A), then

ISSN:2222-758X
e-ISSN: 2789-7362

it starts searching for the available users nearby and after they were found an automatic connection was done successfully, after the connection initiation is completed, sending, and receiving of messages begins. All the messages have arrived at Device A successfully and the timestamps of each message were displayed and it was noticed that all of them were less than one second except for the message received from Device D (HTC One) that has the older specification of the whole group as it took 35.613 seconds for the message to arrive from Device D to Device A as shown in the message log screen in Fig. 6.

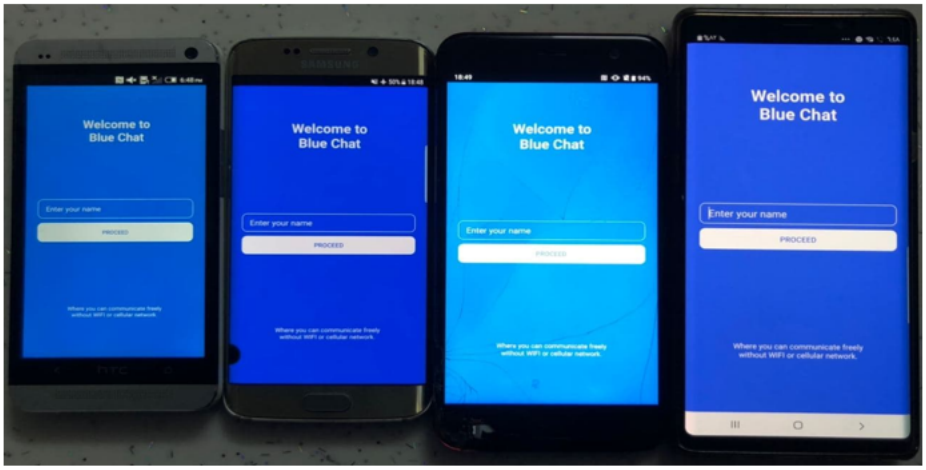

Figure 4: Real photo of 4 mobile device models

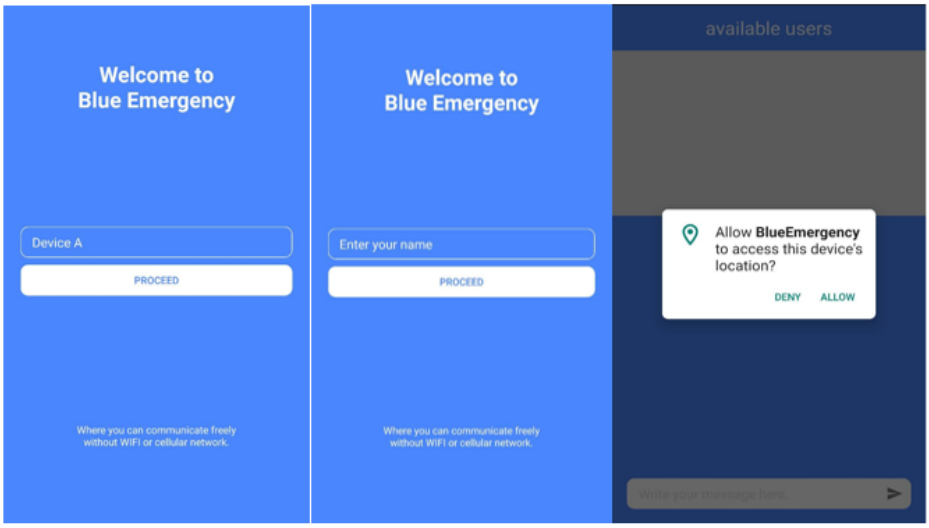

Figure 5: Shows the GUI of the application

ISSN:2222-758X
e-ISSN: 2789-7362

Figure 6: Device A's message log screen

TABLE I
The Software Specification of The Used Smartphones

Device Model	Android Version	API Level	Bluetooth Version
Samsung Galaxy Note 9	9 (Pie)	API Level 28	v4.2
HTC U11	9 (Pie)	API Level 28	v4.2
Samsung Galaxy S6 Edge	7.0 (Nougat)	API Level 24	v4.1
HTC One	5.0.2 (Lollipop)	API Level 21	v4.0

VI. Results and Discussions

These results of the proposed system were shown in form of timestamps by calculating the delay of each message from the moment it was sent to the moment it was received or arrived at its destination. These timestamps were used to prove that the delivery of a message depends upon the strength of the connection in that instant of time and the parameters related to the strength of the connection was mentioned previously and the size of the payload to be delivered as well as the distance from the transmitter and the receiver as if any obstacles between them could weaken the connection. The desired results

ISSN:2222-758X
e-ISSN: 2789-7362

have been achieved with the proposed network model and the layers of security were implemented as mentioned above in section IV using the message digest sent along with the transferred text to authenticate the communicating devices as shown in Fig. 7 and the test was performed with 4 different types of android devices as mentioned above in Table I. and next is Table II which will show the timestamps of each received message of the communicating devices in the network.

TABLE II
The Testing of The System With 4 Different Models

Text Message	Source & Destination	Device A Delay Time	Device B Delay Time	Device C Delay Time	Device D Delay Time
Hello From A	Message from A To B, C & D	X	0.608 sec	0.291 sec	0.291 sec
Hello From B	Message from B To A, C & D	0.242 sec	X	0.819 sec	0.591 sec
Hello From C	Message from C To A, B & D	0.712 sec	1.192 sec	X	0.381 sec
Hello From D	Message from D To A, B & C	35.613 sec	35.998 sec	35.044 sec	X
Long Random Text From A	Message from A To B, C & D	X	0.600 sec	0.355 sec	0.173 sec
Long Random Text From B	Message from B To A, C & D	2.938 sec	X	2.535 sec	3.304 sec
Long Random Text From C	Message from C To A, B & D	0.293 sec	2.974 sec	X	0.621 sec
Long Random Text From D	Message from D To A, B & C	0.575 sec	3.882 sec	1.059 sec	X

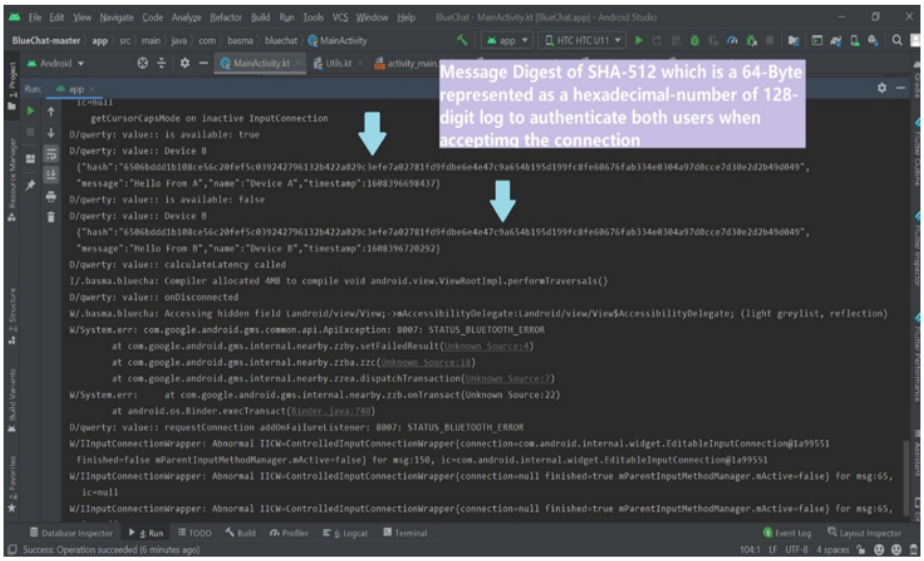

Figure 7: Shows the message digest of SHA-512 hashing algorithm at both sides of the authenticating devices

VII. CONCLUSION

A complete design of the Bluetooth based Ad-Hoc network is introduced in this paper and it showed how self-connection was possible by building an application to connect multiple users in the network at the same time, also sending and receiving broadcast messages by using only Bluetooth to help people at disasters to communicate and exchange warnings,

ISSN:2222-758X
e-ISSN: 2789-7362

locations, emergency needs, etc. The application was built using Android Studio (IDE) and Google Nearby Connections API package and programmed by Kotlin language. The network established in a small building by using 4 different models of Android smartphones. The application performance depended on a diversity of parameters, ranging from the strength of the connection to hardware capabilities of the tested devices to code optimization.

VIII. FUTURE WORK

The network can be extended to cover large buildings over large geographical areas by multi-hopping the messages between devices using a message forwarding mechanism to deliver messages to the farthest possible destination in a network to make the system more efficient to use in the time of need.

REFERENCES

[1] A. Deb and S. Sinha, "Bluetooth Messenger: An Android Messenger App Based on Bluetooth Connectivity" , IOSR J. Comput. Eng. , Vol. 16, No. 3, pp. 61-66, 2014.
[2] "Overview | Nearby Connections API | Google Developers" , [Online]. Available: https://developers.google.com/nearby/connections/overview. [Accessed: 22-Dec-2020] .
[3] L. Meftah et al. , "Testing Nearby Peer-to-Peer Mobile Apps at Large To cite This Version: HAL Id: Hal-02059088 Testing Nearby Peer-to-Peer Mobile Apps at Large" , 2019.
[4] J. P. Cohen, "Wireless Message Dissemination via Selective Relay over Bluetooth (MDSRoB)" , arXiv Prepr. arXiv1307.7814, pp. 1-5, 2013.
[5] M. Kamkuemah and H. Le, "Routing in Mobile Phone ad Hoc Networks" , Proc. - 5th Int. Conf. Comput. Intell. Commun. Syst. Networks, CICSyN 2013, pp. 316-321, 2013.
[6] T. Alam and M. Aljohani, "Design and Implementation of An Ad Hoc Network Among Android Smart Devices" , Proc. 2015 Int. Conf. Green Comput. Internet Things, ICGCIoT, pp. 1322-1327, 2015.
[7] M. S. A. El-Seoud and I. A. T. F. Taj-Eddin, "Developing An Android Mobile Bluetooth Chat Messenger As An Interactive and Collaborative Learning Aid" , Adv. Intell. Syst. Comput. , Vol. 545, No. December, pp. 3-15, 2017.
[8] U. Demir, C. Tapparello, and W. Heinzelman, "Maintaining Connectivity in Ad Hoc Networks Through WiFi Direct" , in Proceedings - 14th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS, pp. 308-312, 2017.
[9] D. Antonioli, N. O. Tippenhauer, and K. B. Rasmussen, "Nearby Threats: Reversing, Analyzing, and Attacking Google's Nearby Connections on Android" , No. February, pp. 24-27, 2019.