2023 10th International Conference on Advanced Informatics: Concept, Theory and Application (ICAICTA) | 979-8-3503-2991-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICAICTA59291.2023.10390307

Data Transmission to Mobile Devices via Multiple QR Codes

Muhammad Rizky I. F.
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
rizkyismail@gmail.com

Rinaldi Munir
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
rinaldi@informatika.org

***Abstract*—Data transmission to mobile devices can be done easily with the use of Bluetooth and Wi-Fi, but these methods have weaknesses such as the significant amount of time needed for the device search and authentication phase when transmitting small amounts of data. Previous works proposes transmitting data via multiple high capacity two-dimensional barcodes, which works as a partial solution but still has certain aspects that can be improved. In this paper, we propose a system for transmitting data via multiple QR codes that aims to maximize the overall data transmission rate by maximizing read accuracy and code processing speed. We also test the use of RaptorQ fountain code to improve handling of missed frames in the system. Analysis and tests done on the system proves that the proposed system can be reasonably used as an alternative data transmission method when transmitting files up to 500KB in size or when other data transmission methods are unavailable.**

***Index Terms*—QR code, data transmission, mobile device, RaptorQ, fountain code**

I. Introduction

The process of transmitting data between mobile devices has been made very easy with the introduction of wireless technologies such as Bluetooth and Wi-Fi, but these technologies still has their downsides. One of the downsides is that when users want to transmit data via these methods, they have to go through the process of device search and authentication that takes a relatively significant amount of time when the data to be transmitted is small [1] [2] [3].

Another downside is a problem called the Wi-Fi Spectrum Crunch, which is when wireless communication that uses electromagnetic signals does not function well in places where there are many other users [4]. One solution to this problem is Visible Light Communication (VLC) with which data is transmitted in signals that are not in the electromagnetic spectrum but in the visible light spectrum so that the signals are less likely to interfere with each other [4].

There are already many data transmission systems that uses visible light. One of which is Li-Fi, where data is transmitted by the use of a single LED light that flickers at a very high frequency and a camera that functions at an equally high frequency [5]. The downside of this technology is that it requires specialized equipment, so it is only commonly found in industrial usage such as in the health industry where it is used in specialized medical equipments [5].

On a more common device such as a smartphone which has a relatively slower camera, it is more optimal when light from multiple sources are read simultaneously. An example of such data transmission system is data transmission by using a single QR code. The downside of this system is that the data capacity that can be stored in a single QR code is only 2,956 bytes at the lowest level of error correction [6].

There exists multiple works that aim to increase the capacity of a QR code by introducing color into the structure [7] [8] [9]. These colored QR codes has a higher data capacity than regular QR codes, but at the cost of longer processing time and lower read accuracy due to more reliance on multiple factors such as lighting, camera quality, and display quality.

Another solution to the data capacity problem is by using multiple two-dimensional barcodes to transmit data. There already exists related works in which they use multiple two-dimensional barcodes to transmit data, but all of these works are forced to use high capacity two-dimensional barcodes to increase the data capacity of each code due to limitations in hardware and software [10] [11] [12]. In this paper, we propose a data transmission system that uses multiple regular QR codes to maximize data transmission rate by maximizing read accuracy and code processing speed instead.

II. Related Works

Memeti et al. (2013) proposes a data transmission system in [10] that uses a custom made two-dimensional barcode structure that they designed. The structure is based on regular QR codes but with error correction removed and with the use of colors so that each code can store a larger amount of data. The introduction of color in the code structure resulted in lowered read accuracy and longer processing time of each code. The proposed system achieved a data transmission rate of 430 byte/s according to the tests done on the paper.

Boubezari et al. (2016) proposes another data transmission system in [11] that also uses a custom made two-dimensional barcode structure. The difference is that the structure used in this system does not use color, but instead uses a format that is made to fit the screen of a smartphone. The structure also does not use search patterns to locate each barcode on the decoding phase, but instead uses the screen flickering on the transmitter device to locate the code. Due to this code search method, the

screen cannot be used to display any other information while data transmission is ongoing. The proposed system achieved a data transmission rate of 12.5 kilobyte/s according to the tests done on the paper.

Toh et al. (2016) proposes a data transmission system in [12] that uses regular QR codes that are multiplexed into colored QR codes to increase the data capacity in each frame. The multiplexing phase and demultiplexing phase causes the processing time of each code to be slower than regular QR codes and the introduction of color causes read accuracy to be lower. The proposed system achieved a maximum encoding speed of 357 byte/s and maximum decoding speed of 5.92 byte/s according to the tests done on the paper.

III. Design

A. Considerations

There are two main points that were considered in the making of the data transmission system proposed in this paper: data splitting and merging system and code structure.

First, data splitting and merging in the previous works are done by naively splitting the input data into parts, sequentially transmitting these parts to the receiver device via QR code in a loop, then merging these parts back into the input data on the receiver device when all of the parts have been received.

This method is slow when QR code read failure occurs because the receiver would then have to wait until the transmitter device transmits the missing part QR code again. When there is a large amount of data parts, and therefore a large amount of QR codes that needs to be displayed in an order, this results in a significant amount of waiting time.

In the proposed system, the use of RaptorQ fountain code [13] will be implemented to let the receiver receive part QR codes in any order and thus be able to handle read failures better. This comes at the cost of a slightly longer processing time for each data part, so a configuration of the system that does not use RaptorQ will also be tested for comparison.

Second, the code structures used in the previous works has a lower read accuracy and longer processing time compared to regular QR codes. The proposed system will use the regular QR code structure to benefit from its integrated error correction capabilities, high read accuracy, and fast processing speed.

B. Architecture

The overall flow of the proposed system can be seen in Fig. 1. The proposed system consists of two mobile devices, each with an application that implements the functionalities of the proposed system.

The application on the transmitter device converts input data into a set of displayable QR codes in four phases:

1) Input Data Selection (A1)
2) Input Data Splitting (A2)
3) QR Code Encoding (A3)
4) QR Code Display (A4)

The application on the receiver device reads these multiple QR codes via the camera module and converts them back into the original input data in four phases:

1) Image Capturing (B1)
2) QR Code Decoding (B2)
3) Output Data Merging (B3)
4) Output Data Storing (B4)

When the naive data splitting and merging method is used, the data part structure that is used is the one shown in Fig. 2. The input data is split into data parts sized as large as possible that can fit into a single QR code along with the data part index and total data part count. The first data part will additionally contain the file name information.

When the RaptorQ data splitting and merging method is used, the data part structure that is used is the one shown in Fig. 3. There will be two types of parts: metadata parts and data parts. Metadata parts will contain the file name information along with RaptorQ parameters that are needed on the receiver side and needs to be fully received by receiver devices before it can receive data parts. Data parts will contain the encoded data generated by the RaptorQ encoder and will need to be received in a certain amount by receiver devices in any order to be able to fully reconstruct the original data.

IV. Implementation

The implementation of each phase of the system will be explained separately in the following subsections.

A. Input Data Selection (A1)

In this phase, the user selects the input data from two sources: the local storage or the clipboard. If the user chooses to transmit data from the clipboard, a file name will be generated containing the word ”clipboard” and the date and time of this transmission. Otherwise, the original file name will be sent as metadata.

B. Input Data Splitting (A2)

In this phase, the input data is put through a data splitting function. The data splitting function will either be a naive data splitting into the structure defined in Fig. 2 when RaptorQ is not used, or structure defined in Fig. 3 when RaptorQ is used. The RaptorQ encoder that is used in the implementation is from the publicly available and open-source *OpenRQ* library [14].

C. QR Code Encoding (A3)

In this phase, each data part (and metadata part, if RaptorQ is used) is encoded into a QR code. There are two variables that can be configured for this step: the QR code version and the error correction level. The QR code version determines the amount of modules (black and white squares) in a QR code, and therefore also determines the amount of data that can be stored in each QR code. The error correction level represents the amount of redundant data stored in each QR code for error correction. The QR code encoder that is used in the implementation is from the publicly available and open-source *fastqrcodegen* library [15].

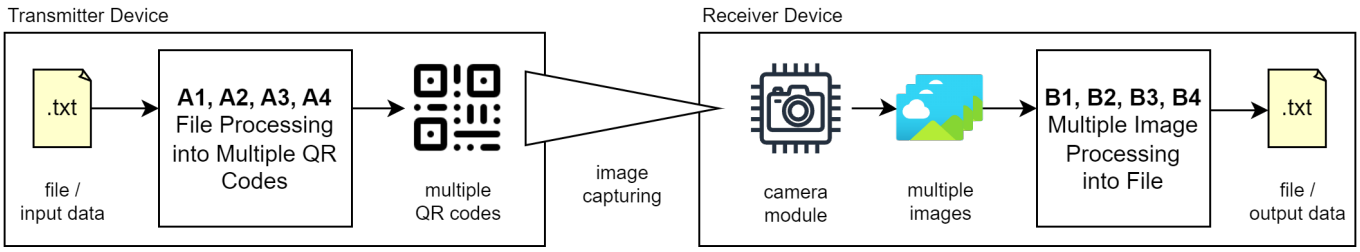

Fig. 1. The overall flow of the proposed system.

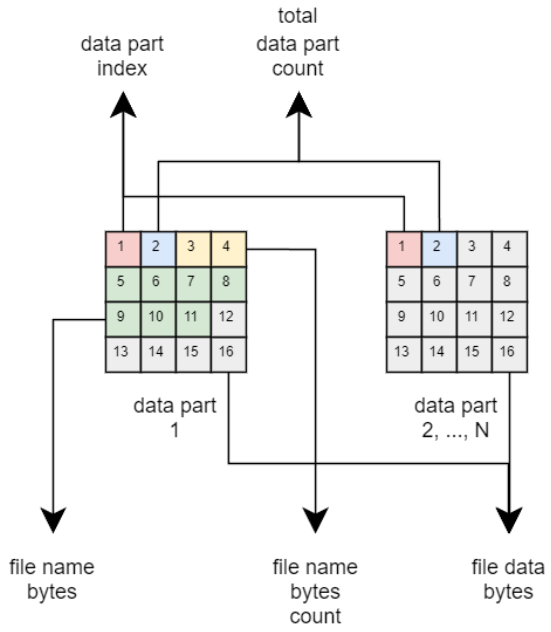

Fig. 2. The structure of data parts used by the system with the naive data splitting and merging method (16 bytes for each data part, with 7 bytes for the file name)

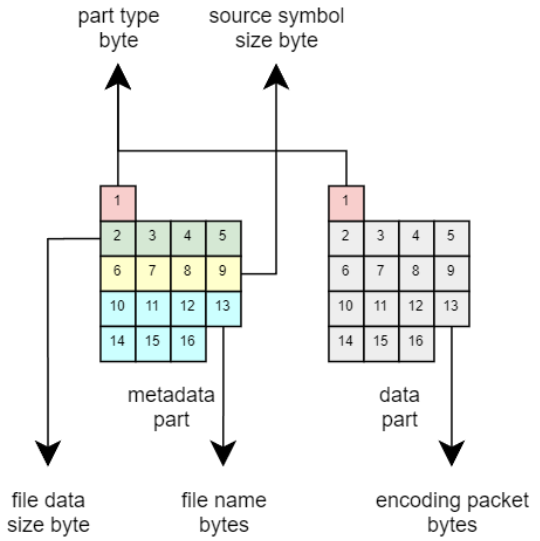

Fig. 3. The structure of data parts used by the system with the RaptorQ data splitting and merging method (16 bytes for each data part, with 7 bytes for the file name)

D. QR Code Display (A4)

In this phase, each generated QR code is shown in an loop at the transmitter device. When RaptorQ is used, the application will initially display the metadata QR code, only displaying data QR codes after the user has confirmed the successful transmission of the metadata QR code by tapping the transmitter device's screen. The rate at which the QR codes are displayed are configurable in the mobile application.

E. Image Capturing (B1)

In this phase, the receiver mobile application starts capturing frames from the camera module to process in the next phase. Which camera module to use, the rate at which frames are captured, and the zoom level are configurable in the mobile application.

F. QR Code Decoding (B2)

In this phase, each frame received from the previous phase is run through the QR code decoding function. The QR code decoder used in the implementation is from the publicly available and open-source *ZXing* [16] and *BoofCV* [17] library . Which library is used to decode is configurable in the mobile application. This choice of QR code decoding library is due to prior research showing that *ZXing* is superior in processing speed while *BoofCV* is superior in read accuracy, so both are tested for comparison.

G. Output Data Merging (B3)

In this phase, the data parts received from the previous phase is merged together once all of the required data parts have been received. When RaptorQ is not used, this is done by simply ordering the data part by their index then merging them together. When RaptorQ is used, the data parts are inputted into the RaptorQ decoder to merge back into the originally transmitted data. The RaptorQ decoder used in the

implementation is from the publicly available and open-source *OpenRQ* library [14].

H. Output Data Storing (B4)

In this phase, the fully merged data is then stored to either the local storage or into the clipboard according to the user's choice.

V. Testing and Analysis

As seen in the previous section, the implementation has a large amount of configurable parameters, each of which may or may not have an effect to the goal of this paper which is to make a data transmission system that maximize data transmission rate by maximizing read accuracy and code processing speed. These configurable parameters are tested in order to infer which configuration is most suited to the goal of this paper in the following subsections.

A. Read Accuracy

The read accuracy of the system was tested with a Samsung Galaxy Tab A8 as the transmitter device, a Xiaomi Poco F4 as the receiver device and QR code version 15. The results can be seen in Table I. The read accuracy was measured by getting the percentage of frames where a QR code was successfully decoded relative to the total amount of frames received from the camera module since the first QR code was detected.

From this test, it was concluded that to achieve maximum read accuracy, the camera framerate must be as high as possible while the display framerate must be as low as possible. The choice of QR code decoder does not seem to have any conclusive effect on read accuracy, as some tests showed one decoder having a higher read accuracy than the other library while some other tests showed the opposite.

B. Code Processing Speed

In order to measure the total code processing speed, we measured the average encoding time and decoding time of a single code across multiple configurations.

1) Encoding Time: The average encoding time of the system was tested with a Samsung Galaxy Tab A8 as the transmitter device and a Xiaomi Poco F4 as the receiver device. The results can be seen in Table II.

From this test, it can be seen that encoding time decreases consistently as the QR code version is decreased. It can also be seen that the use of fountain codes has a very small effect on the encoding time of each QR code.

2) Decoding Time: The average decoding time of the system was tested with a Samsung Galaxy Tab A8 as the transmitter device and a Xiaomi Poco F4 as the receiver device. The results can be seen in Table III.

From this test, it can be seen that decoding time also decreases consistently as the QR code version is decreased. It can also be seen that the use of fountain codes has a more significant effect on the decoding time compared to the encoding time. It is also shown that the ZXing QR code decoder is significantly faster than BoofCV at decoding.

TABLE I
Read Accuracy Test Results

Test Num.	Display Framerate	Camera Framerate	QR Code Decoder	Read Accuracy
1	15 FPS	30 FPS	ZXing	58.91%
2	24 FPS			21.28%
3	30 FPS			-
4	45 FPS			-
5	54 FPS			-
6	60 FPS			-
7	15 FPS	60 FPS		72.29%
8	24 FPS			59.54%
9	30 FPS			49.25%
10	45 FPS			32.82%
11	54 FPS			14.06%
12	60 FPS			-
13	15 FPS	30 FPS	BoofCV	51%
14	24 FPS			21.43%
15	30 FPS			-
16	45 FPS			-
17	54 FPS			-
18	60 FPS			-
19	15 FPS	60 FPS		74.64%
20	24 FPS			60.64%
21	30 FPS			49.18%
22	45 FPS			24.42%
23	54 FPS			12.64%
24	60 FPS			-

TABLE II
Encoding Time Test Results

Test Num.	QR Code Version	Fountain Code Usage	Encoding Time
1	15	Yes	6ms
2	20		8,5ms
3	25		11,5ms
4	30		15,5ms
5	35		20,5ms
6	40		26ms
7	15	No	6ms
8	20		7ms
9	25		11ms
10	30		15ms
11	35		19ms
12	40		25ms

TABLE III
DECODING TIME TEST RESULTS

Test Num.	QR Code Version	Fountain Code Usage	QR Code Decoder	Decoding Time
1	15	Yes	ZXing	7ms
2	20			8ms
3	25			9ms
4	30			10ms
5	35			20ms
6	40			-
7	15	No		4ms
8	20			4ms
9	25			7ms
10	30			7ms
11	35			8ms
12	40			9ms
13	15	Yes	BoofCV	17ms
14	20			23ms
15	25			27ms
16	30			34ms
17	35			46ms
18	40			-
19	15	No		10ms
20	20			19ms
21	25			24ms
22	30			25ms
23	35			27ms
24	40			30ms

C. Data Transmission Rate

Finally, we tested the configurations that had the highest read accuracy and code processing speed to determine which configuration will have the highest overall data transmission rate. The device used is the same as with the previous tests. The results can be seen in Table IV. The data transmission rate is calculated by calculating the time needed to transmit a 50 KB file.

From this test, it is shown that the configuration with the highest data transmission rate is with 30 FPS display framerate, 60 FPS camera framerate, ZXing QR code decoder, and with the use of RaptorQ fountain code. It is also shown that the use of RaptorQ in general increased the data transmission rate, because the time needed for transmission increases significantly when there is a missing data part and fountain code is not used.

To make this test more complete, we then tested the optimal configuration that was found in the previous test with input data of varying sizes. The result of this test can be seen in Table V.

It is shown that the data transmission rate is stable for transmitting data up to 500KB, with the exception for the test with 500 byte data size: the data was likely too small for the test speed to be properly measured. In the test with 1 MB data size, the amount of read failure was large enough that it took a while to receive all the required data parts with the tested configuration. Further testing is likely needed to determine a better configuration for larger data sizes.

TABLE IV
DATA TRANSMISSION RATE TEST RESULTS

Test Num.	Display Framerate	Camera Framerate	Decoding Library	Fountain Code Usage	Data Transmission Rate
1	15 FPS	30 FPS	ZXing	Yes	6.37 KB/s
2	15 FPS	60 FPS			6.71 KB/s
3	24 FPS				9.83 KB/s
4	30 FPS				13.21 KB/s
5	45 FPS				6.63 KB/s
6	15 FPS	30 FPS	BoofCV		6.81 KB/s
7	15 FPS	60 FPS			7.29 KB/s
8	24 FPS				9.55 KB/s
9	30 FPS				11.68 KB/s
10	45 FPS				3.59 KB/s
11	15 FPS	30 FPS	ZXing	No	2.66 KB/s
12	15 FPS	60 FPS			7.81 KB/s
13	24 FPS				7.71 KB/s
14	30 FPS				8.58 KB/s
15	45 FPS				2.11 KB/s
16	15 FPS	30 FPS	BoofCV		2.23 KB/s
17	15 FPS	60 FPS			6.75 KB/s
18	24 FPS				6.36 KB/s
19	30 FPS				3.88 KB/s
20	45 FPS				1.54 KB/s

TABLE V
DATA SIZE TEST RESULTS

Test Num.	Data Size	Data Transmission Rate
1	500 byte	1,42 KB/s
2	50 KiB	12,02 KB/s
3	100 KiB	12,15 KB/s
4	150 KiB	12,28 KB/s
5	200 KiB	11,71 KB/s
6	500 KiB	11,33 KB/s
7	1 MiB	6,12 KB/s

VI. Conclusion

From the testing and analysis performed, the optimal configuration for the system to reach maximum data transmission rate for transmitting files 50KB in size was found. With it, the system achieved a maximum data transmission rate of 13.21 KB/s on the test devices. It was also found that the use of RaptorQ fountain code increased the overall data transmission rate of the system.

When compared to existing methods such as data transmission via Bluetooth and Wi-Fi, the proposed system has some advantages but also some disadvantages.

One advantage is that users would only need to open an application and immediately start transmitting or receiving data, without the need to connect to wireless networks or directly to other devices. Another advantage is that it is more scalable: transmitting data from from one giant screen in a stadium to hundreds or even thousands of receiver mobile phones would theoretically be as fast as transmitting data from a single mobile phone to another, whereas the same is not true when transmitting data via Bluetooth or Wi-Fi.

The main disadvantage is that the transmission speed of 13.21 KB/s or 105.68 Kb/s is significantly slower when compared to data transmission via Bluetooth or Wi-Fi. Data transmission rates for Bluetooth 5.0 can reach 2 Mb/s [18] and for Wi-Fi 802.11ac can reach 55.5 Mb/s [19]. Another disadvantage is that receiver devices has to be in direct line of sight and positioned in a certain angle in relation to the transmitter device for it to be able to properly read the displayed QR codes.

In conclusion, the system proposed in this paper can be used as an alternative data transmission method when transmitting data up to 500KB in size, when transmitting data from one transmitter to a large amount of receivers, or when other data transmission methods are unavailable.

References

[1] G. Castignani, A. Arcia-Moret, and N. Montavont, "A study of the discovery process in 802.11 networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 15, pp. 25–36, 03 2011.

[2] D. Yadav and A. Sardana, "Authentication process in ieee 802.11: Current issues and challenges," in *Communications in Computer and Information Science*, vol. 196, 07 2011, pp. 100–112.

[3] K. V. S. S. S. S. Sairam, N. Gunasekaran, and S. R. Redd, "Bluetooth in wireless communication," *IEEE Communications Magazine*, vol. 40, no. 6, pp. 90–96, 2002.

[4] L. E. M. Matheus, A. B. Vieira, L. F. M. Vieira, M. A. M. Vieira, and O. Gnawali, "Visible light communication: Concepts, applications and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3204–3237, 2019.

[5] D. Khandal and S. Jain, "Li-fi (light fidelity): The future technology in wireless communication," *International Journal of Information & Computation Technology*, vol. 4, no. 16, pp. 1687–1694, 2014.

[6] S. Tiwari, "An introduction to qr code technology," *2016 international conference on information technology*, pp. 39–44, 12 2016.

[7] M. E. V. Melgar, A. Zaghetto, B. Macchiavello, and A. C. Nascimento, "Cqr codes: Colored quick-response codes," *2012 IEEE Second International Conference on Consumer Electronics-Berlin*, pp. 321–325, 09 2012.

[8] N. Taveerad and S. Vongpradhip, "Development of color qr code for increasing capacity," *2015 11th International Conference on Signal-Image Technology & Internet-Based Systems*, pp. 645–648, 11 2015.

[9] A. Abas, Y. Yusof, and F. K. Ahmad, "Improving data capacity of qr code version 40 using multiplexing and multilayered techniques: Embedding text based short story in qr code," *Advanced Science Letters*, vol. 22, pp. 2841–2846, 2016.

[10] J. Memeti, F. Santos, M. Waldburger, and B. Stiller, "Data transfer using a camera and a three-dimensional code," *PIK - Praxis der Informationsverarbeitung und Kommunikation*, vol. 36, no. 1, pp. 31–37, 2013. [Online]. Available: https://doi.org/10.1515/pik-2012-0140

[11] R. Boubezari, H. Le Minh, Z. Ghassemlooy, and A. Bouridane, "Novel detection technique for smartphone to smartphone visible light communications," *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2016.

[12] S. R. Toh, W. Goh, and C. K. Yeo, "Data exchange via multiplexed color qr codes on mobile devices," in *2016 Wireless Telecommunications Symposium (WTS)*, 2016, pp. 1–6.

[13] L. Minder, A. Shokrollahi, M. Watson, M. Luby, and T. Stockhammer, "RaptorQ Forward Error Correction Scheme for Object Delivery," RFC 6330, Aug. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6330

[14] J. Lopes and R. Fonseca, "Openrq," 2022, accessed: Jan. 22, 2023. [Online]. Available: https://openrq-team.github.io/openrq/

[15] Nayuki, "Fast qr code generator library," 2019, accessed: Jan. 22, 2023. [Online]. Available: https://www.nayuki.io/page/fast-qr-code-generator-library

[16] ZXing, "Zxing," 2022, accessed: Jan. 22, 2023. [Online]. Available: https://github.com/zxing/zxing

[17] BoofCV, "Boofcv," 2022, accessed: Jan. 22, 2023. [Online]. Available: http://boofcv.org/index.php?title=Main_Page

[18] E. Au, "Bluetooth 5.0 and beyond [standards]," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 119–120, 2019.

[19] T. Kaewkiriya, "Performance comparison of wi-fi ieee 802.11 ac and wi-fi ieee 802.11 n," *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, pp. 235–240, 2017.