WiFi-Bluetooth Ad Hoc File Transfer Application for Android Devices

Shuoyuan Chen
University of Waterloo
Waterloo, Ontario
s487chen@uwaterloo.ca

ABSTRACT

Wireless ad hoc network is an infrastructure-less local network connecting mobile devices. It frees its users from the cellular network, which may be essential or favored in certain scenarios. In this report, a novel Android application is designed for non-technician to be able to initiate and join an ad hoc network to perform quick file sharing. The application uses WiFi Direct and Bluetooth to facilitate routing and file propagation, which are available in most Android smart phones on the market.

1 INTRODUCTION

1.1 Wireless Technology on Mobile Devices

Mobile devices fabricated recent years, such as cellphone or tablets, all come with standard wireless technologies enabling local wireless connection among devices from other manufacturers. Bluetooth and WiFi Direct are the most ubiquitous two and are adopted by most Android manufacturers.

Bluetooth standard enables one device connecting to at most seven other devices at the same time over 2.4/5GHz radio frequency. It is capable of carrying out around 1Mbit/s data transfer in a 5-30m radius on typical mobile devices[12]. Starting from Bluetooth 4.0, Bluetooth Low Energy (LE) is introduced to reduce battery consumption with the sacrifice of link reliability and message size. The Bluetooth adopts an master-and-slave architecture, where one device operates as a master, maintaining parallel data exchange with its slaves (up to seven) and forms a picocell.

WiFi Direct is support on Android starting from version 4 at 2012. According to IEEE 802.11n standard, WiFi Direct can achieve one-to-one connection with speed up to typical WiFi speed, which is at most 600 Mbit/s for typical mobile entity with one antenna. The link operates in 30-100m radius with consideration of blockages, which cover much bigger space than Bluetooth. However, it also consumes more power than Bluetooth. WiFi Direct consumes 20 Watt at operation, while Bluetooth LE takes as low as 0.01 Watts [10].

1.2 Ad Hoc Network

A wireless ad hoc network (WANET) is a decentralized type of wireless network, which has no reliance on extra hardware backbones or any centralized routing management [2]. It is comprised of multiple basic units with the ability to broadcast message to and receive from neighbors. These nodes operate together based on the same pre-defined protocol, and are able to communicate with each other locally.

For existing long-term evolution (LTE) and WiFi network, the speed and capacity is greatly limited in populated areas. Sometimes data transmission is expensive in terms of time consumption and bandwidth occupation, for massive file exchange in public networks. Ad hoc can offload existing wireless network, and increase speed, because the unnecessary detour through public network is skipped. Therefore, WANET is a favorable communication approach when public network is either crowded or weak or even non-existent. For example, the on-air traffic would peak immediately after natural disasters, resulting in packet delay or drop [2]. It is also not uncommon for small mobile companies to cover only populated areas. Moreover, when it comes to high-volume file sharing, WANET has the dominant advantage for being cost-free.

Apple devices support Bluetooth WANET by default, which has been widely used for application deployment [11]. On the hand, Android system lacks software support for WANET even though it has been discussed for years. In this article, we explore the specifications and performance of a novel WiFi-Bluetooth WANET among unrooted Android devices.

2 RELATED WORK

Previous Android WANETs include pure Bluetooth, pure WiFi Direct and hybrid architecture. Since neither Bluetooth (one-to-seven connection) or WiFi Direct (one-to-one connection) supports multi-hop, pure WANET suffer from poor routing and hence heavy delay [6, 7]. The fact that each device can only link to part of its neighbors significantly limit the size of Android WANET. Besides, since unrooted Android devices does not support frequency division multiplexing for concurrent use, most Android WANETs went for time division multiplexing, where synchronizing all nodes in big ad hoc networks remains big challenge [1, 3].

Bluetooth by design works in Master/Slave manner, introducing hierarchy to the ad hoc network. It poses great difficulty in forming an optimized/simplest ad hoc network, i.e. number of master minimized, in a short amount of time. The four basic topologies in Bluetooth ad hoc include tree, mesh, ring and shared slave topology [4]. In [5], mesh topology is fulfilled by allowing each node to alternate between slave and master to talk to all neighbors. This approach gives the most efficient graph in theory but the synchronization is difficult when switching roles takes tiny arbitrary time. Any method with synchronization is not suitable for mass deployment.

Pure WiFi Direct ad hoc network supporting only one-to-one link, which makes message propagation extremely slow. In [1], a tree-like topology is adopted and each node alternately connects its parent and children to pass data from top to down, where synchronization is done by maintaining a shared time stamp. Time division is proposed in [7, 8], which allows for mesh-like topology. The exchange between nodes is much easier than tree structure but it requires system rooting.

The WANET routing mechanism is predominantly on-demand distance vector routing (AODV) in literature, due to its less occupation of bandwidth and less power consumption than Dijkstra [6, 11]. However, AODV reacts slowly to mobility-induced topology changes. Machine Learning is introduced in [9] to relieve the issue.

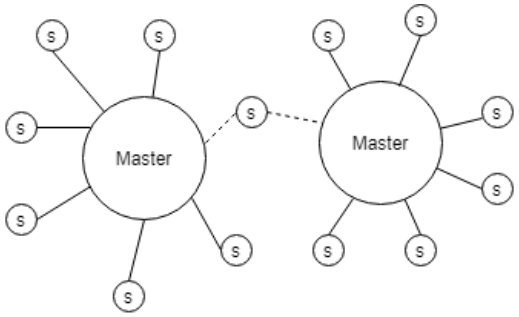

Figure 1: Bluetooth master-master communication via a shared slave. The solid line represents permanent link and the dash line represents impermanent link.

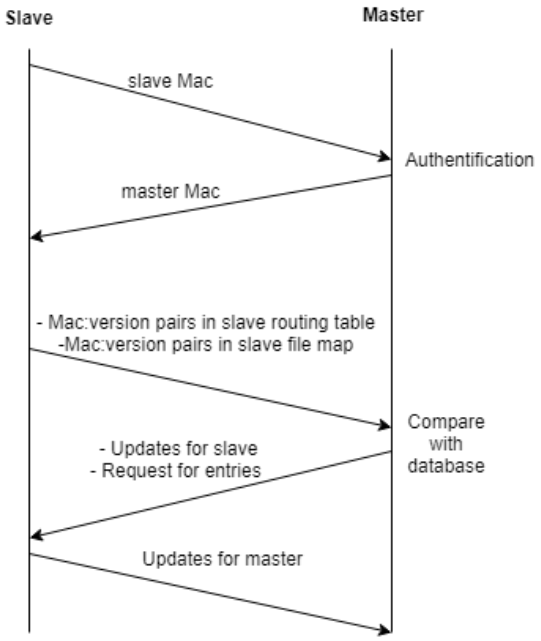

Figure 2: handshake between a shared slave and one of its master.

3 SYSTEM DESIGN

3.1 WiFi-Bluetooth-Adhoc

Since our application focuses on file sharing, the timeliness is not a priority. The multiplexing is not required because now the stability of transfer and size of network are more important than maximizing the usage of free bandwidth. Duo-band hybrid WANETs can combine Bluetooth and WiFi to increase speed, expand capacity and combat user mobility.

In "WiFi-Bluetooth-Adhoc", the data plane function and control plane function are split and assigned respectively to WiFi Direct and Bluetooth. The Bluetooth functions include continuous linking and unlinking between members and WiFi is turned on all the time, which in combination drains battery quicker than using mere WiFi or Bluetooth mechanisms [10]. It can be justified by fact that such file transfer application is never meant to run for long time, which can be turned of when the transfer is completed. While a resilient and stable network can result in more optimized routing, which increase the efficiency of file propagation and reduce the total running time.

3.2 Bluetooth Layer

The control plane function is realized by a foreground Android service named "Bluetooth service", controlling Bluetooth functionalities. Bluetooth is designed to work in master-slave mode, which means the master of a local picocell can talk with each of its slave in parallel. The master initiates as a discoverable node, where all slaves discovers the master and connects to it. However, for ad hoc network larger than eight members, master-master communication between multiple picocells needs to be considered.

In our architecture, master-master communication is carried by an intermediate slave as is illustrated in figure 1.

Unlike normal slave whose connection is permanent, the shared slave communicates with two masters by turns. Since only newly paired devices require authentication, the constant switching can be done automatically. The handshake between the master and shared slave is illustrated in figure 2, after which additional message may be transferred.

Each ad hoc member keeps a link-state table and a file map containing each member's shared files with a version number. The link-state table contains link-state information of all the nodes in the network with a version number. The file map contains the names and hash code of the files ready to be shared (users' choice) on each member. It also keeps a version number for the file list in each node. Once a node makes changes on its own link-state or file list, it will increase the corresponding version number. During each exchange between master and shared slave, each entry is updated to its latest version and that is how message is circulated within the network.

Mac	Role	AdjMaster	Slaves	Ver.
00:05:5D:E8:0F:A3	M	[mac1,...]	[mac1,...]	1
00:5e:5D:00:67:12	S	[mac]	[]	1
00:02:19:17:21:03	SS	[mac1,mac2]	[]	2
00:16:5D:23:0F:83	M	[mac1,...]	[mac1,...]	3

Table 1: Example of link-state table on each node

Mac	FileNames:Hash	Version
00:05:5D:E8:0F:A3	{name1:hash1,name2:hash2}	1
00:5e:5D:00:67:12	{name1:hash1}	1
00:02:19:17:21:03	{}	2
00:16:5D:23:0F:83	{name1:hash1,...}	3

Table 2: Example of file map on each node

As shown in table 1, roles of node include master (M), normal slaves (S) and shared slaves (SS). The AdjMaster field indicates the surrounding masters. For slaves, it means the masters it directly connects; For masters, it means the other master it indirectly connects through shared slaves. The shortest path is calculated using Dijkstra's algorithm. The concern on bandwidth occupation does not apply here because Bluetooth does not carry data plane function. The limiting factor in our architecture is the dis-/re-connect time when propagating data, which is inevitable.

As demonstrated in table 2, all files to be shared in the node are visible to the whole network. The files include those picked by the user and those in the cache. Each time a node forwards one file, it has a temporary file in cache. The cached file may be deleted when the allocated cache storage is full, but it shall not interrupt file transfer for reasons that will be discussed in section **file transfer request**.

3.3 Periodical Release of Slaves

The master periodically sends command to each slave, asking them to disconnect, discover nearby masters, and report unknown masters. If an unseen master is found, the slave first finding the new master is promoted to shared slave. There is only one shared slave between any two masters. This mechanism enables detection of new entries after the network is already formed.

3.4 Initiation of Ad Hoc Network

A necessary requirement for successful initiation is that none of the members are out of Bluetooth reach. It is assumed that all potential ad hoc members remain relatively stationary during initiation. This application adopts a tree expanding strategy where new masters are defined one-after-one, which forms the backbones of the network. The starting device defines the ID of the network and starts as a master, whose name is "WifiAdhoc_ $\langle ID \rangle$". Other users need to enter the ID to join a network. Those joining devices change their Bluetooth name to "WifiAdhoc_ $\langle ID \rangle$" and starts as a slave looking for a master with the same name.

The devices near the origin device would soon form the first picocell. After the picocell is stable (no new connections for 200ms), the master sends "expand" command to one slave. The slave receiving the "expand" command disconnects from master, temporarily promotes itself to master for 5s and accepts only one pairing. If a pairing succeeds, the slave sends a command to promote the new node to master. Finally, it changes itself to a shared slave between the two master and reports back to origin master. After receiving the report, the master sends "expand network" to next slave. All new masters go through the above initializing procedures and the network therefore expands.

Once all slaves finishes "expand" command, the initialization period is over. Since the application works on unrooted devices, the pairing of new devices require authentication. The whole setup process requires constant users' attention to handle the popup windows asking for connection permissions, which is unfortunately inevitable.

3.5 Ad Hoc Network Fluctuation

The members of ad hoc network can quit and re-connect due to poor connectivity. New devices may want to enter a network that is already running. Moreover, the mobility of users may cause the member to switch from one piconet to another.

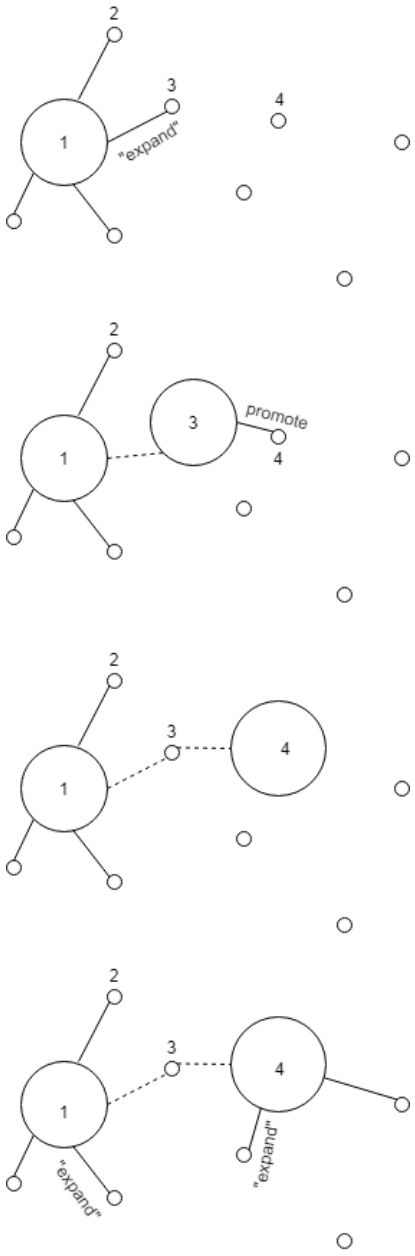

Figure 3: Initialization of ad hoc network. Big circle indicates master mode, small circle indicates slave mode. Solid line represents permanent connection, dash line represents impermanent connection.

A slave is considered lost once its master loses its connection for 5 seconds. The master is considered lost when a shared slave cannot discover it for 5 seconds. Once an adjacent master or slave is detected to be lost, the corresponding master will update its own link-state in its link-state table by removing its link to the lost node. The update has a higher version number and therefore circulates the whole network. Each receiving node will remove the lost link, and the lost node will thus be isolated eventually. Duplicate message is possible from two ends when a master-master link breaks, but it causes no harm.

New connections and Re-connections start as discovering nearby masters for 10 seconds. If the master once linked to the device, it can silently re-connects to the master. After the discovering time expires, the device change its role to master, which becomes visible to the slaves temporarily released by their masters to find unknown nodes. The new node then joins the existing network as a master with shared slaves only. When an shared slave is lost, the connection between two masters is disabled. A regular slave between the two masters will be promoted to shared slave when it recognizes the disconnected neighboring master as new node during its temporary freedom.

The resilience towards lost node has boundary. Assuming two parts of the network is only connected by one node. All other nodes in one half are not reachable from any node in the other half. The loss of that bridging node will break the network.

3.6 File Transfer Request

The user who wishes to download a certain file needs to enter its name. If the file map on the device does not have the wanted file name, the application shall wait until the file map updates. Once the closest node with file is located in the network, a file request is sent to the next-hop on the shortest path towards to that node. At each node before destination, the closest node with file is recalculated, which makes it possible to forward the request to a closer destination that was previously unknown to origin node due to slow transmission of updates on file map.

File Name	Node Sending Request

Table 3: Format of file request

3.7 WiFi Direct Layer

The data plane function is achieved using WiFi Direct, which is regulated by a foreground Android service named "WiFi service". When idle, the WiFi Direction function is turned on on each member device, able to be discovered by any device in proximity. When a file upload request is pushed from the Bluetooth service, the WiFi service discovers other WiFi Direct devices in its neighborhood.

The list of discovered devices are then compared with the Bluetooth routing path from current node to the node requesting the file. The device closest to destination is linked

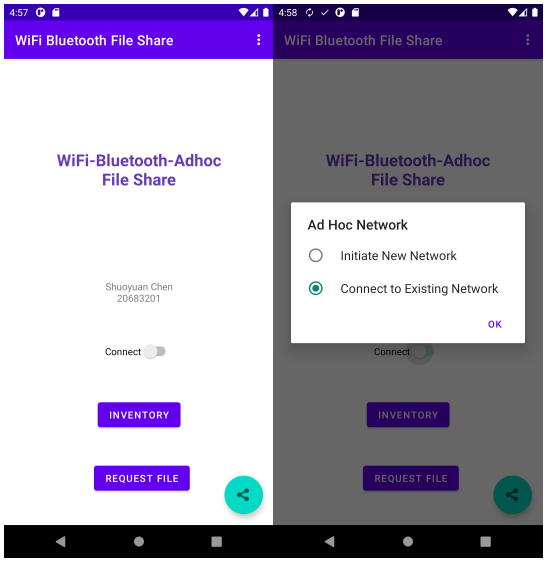

Figure 4: Screenshots of the Application

via WiFi Direct. Then the destination name and actual file is forwarded. The new node checks its own link-state table and WiFi neighbors again and forwards the file to the next hop. The same process repeats until the file arrives at destination. Since each relaying node picks its own optimal next-hop based on local conditions, it is resilient to network fluctuations.

All discoverable devices are idle and all occupied devices are temporarily invisible to other devices. The mechanism excludes the potential congestion on key conjunctions and makes sure that free nodes are utilized as much as possible. It improves the overall utilization of free bandwidth and balances the battery loss among members. The file propagates to the destination in the fewest hops, which saves bandwidth and cache space on other members.

4 EVALUATION

The final application can be seen in figure 4, which supports file import/export and starting/joining ad hoc network. Though not deployed on real devices, simulations are done to estimate the potentials of the application.

A simulation of initiation of ad hoc is performed using python, the result of which is shown in figure 1. It is also assumed that the pairing of two new devices takes 0.5s and lasts for at least 0.5s. Each device discovery takes 120s in slave node and the discoverable period for a master node is 120s too. The switch-over time between slave and master mode is set as 0.5s and the communication duration is neglected. The topology of user devices is simplified into an undirected graph, where nodes within Bluetooth range are connected. The graph is randomized each run and the average time is obtained for different numbers of nodes by repeating same procedure 100 times. Supposing that the application on all devices are started at the same time.

The simulation result is demonstrated in figure 5. It is obvious in the figure that the setup time can vary greatly depending on topology. The number of the bottleneck nodes has major impact on initialization performance, where the expanding branches are blocked. The lower-bound and average value shows that good performance can be expected for user numbers around 10. The optimal performances are not difficulty to achieve, grouping people together in crowded places such as classroom or office would give very good result.

Adopting the same presumptions, propagation through each link takes 1s. The average latency versus node number is simulated as shown in figure 6. At most five pairs of nodes are randomly picked in each topology. Unlike some instant message application, for file transfer purpose, the latency is acceptable for 10 people scenario. For both Bluetooth and WiFi Direct, the route is re-calculated again and again at each node. Such dynamical adjustments can provide good resilience to slow message travel.

Considering that the data exchange over Bluetooth is frequent but each packet is small in size, it would be beneficial to switch from conventional Bluetooth to Bluetooth LE. It has the advantage that it requires no manual authentication to pair with new devices, which allows for automation in connecting/disconnecting devices. It gives a certain degree of freedom to the ad hoc network to be able to proactively reconstruct itself with the movement of its members.

LE is similar to UDP connection, where delivery is best-effort. The package-drop-rate could be unacceptably high in a one-to-many architecture. Moreover, our proposed architecture requires success in each exchange of data, which is not guaranteed in LE and hence demands extra re-transmission. It requires more investigation to know whether Bluetooth LE is more feasible than conventional Bluetooth. Another future improvement may be implemented is security measurement. The application now spends no effort on preventing eavesdropping and virus packages. A password or encryption model may be employed to encode the transactions.

5 CONCLUSION

In conclusion, an Android application aiming at fast propagation of files when internet is not available or not favored is built and its protocol is investigated. It utilizes Bluetooth to construct a hierarchical Ad hoc network for routing, and WiFi Direct to facilitate high-volume file transfer. The application can perform well connecting nearly 10 people according to simulation. It is user-friendly and requires not root

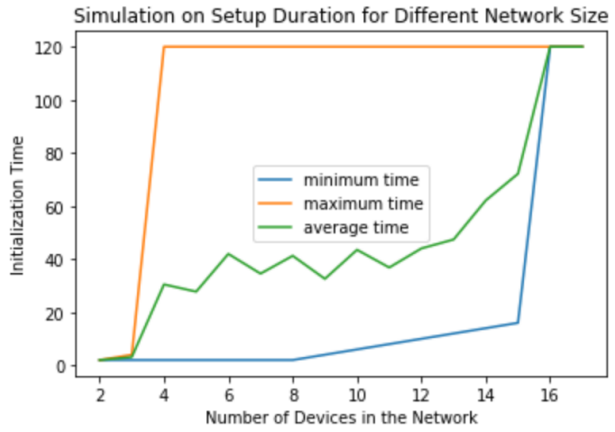

Figure 5: Simulation on Setup Duration for Different Network Size

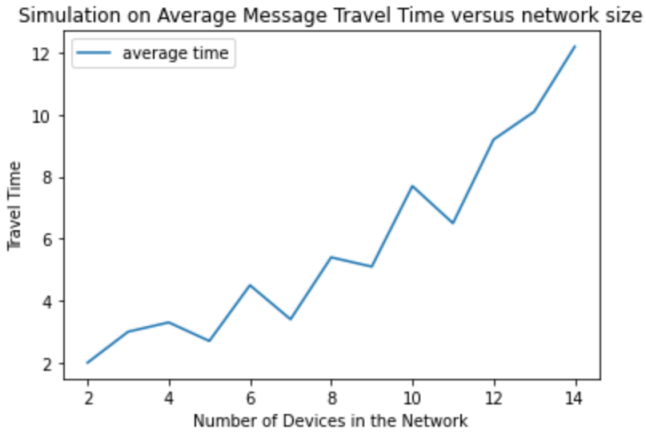

Figure 6: Simulation on Setup Duration for Different Network Size

permission like many other ad hoc implementations do. Future efforts can be laid upon mobility management of Ad hoc members, and changing roles of nodes to achieve optimal topology when addition and removal of nodes happen. One solution that may prove fruitful is introducing WiFi Direct to routing. On the other hand, low energy Bluetooth is worth exploiting, but its high packet loss and low message capacity requires advanced protocols. The security could be improved as well.

6 APPENDIX

The WiFi-Bluetooth-Adhoc application can be found on https://github.com/s487chen/wifi_bluetoothadhoc

REFERENCES

[1] Funai Colin, Tapparello Cristiano, and Heinzelman Wendi. 2017. Enabling multi-hop ad hoc networks through WiFi Direct multi-group networking. *International Conference on Computing, Networking and Communications (ICNC)* (2017).

[2] Onwuka E. 2011. MANET: A RELIABLE NETWORK IN DISASTER AREAS. *JORIND* 9, 2 (2011), 105–113.

[3] Vergetis Evangelos, Guérin Roch, A., Sarkar Saswati, and Rank Jacob. 2005. Can Bluetooth Succeed as a Large-Scale Ad Hoc Networking Technology? *IEEE Journal on Selected Areas in Communications* 23, 3 (2005), 644–656.

[4] Vergetis Evangelos, Guérin Roch, A., Sarkar Saswati, and Rank Jacob. 2019. A survey on Bluetooth multi-hop networks. *Ad Hoc Networks* 93, 101922 (2019).

[5] Hinojos G., Tade C., Park S., Shires D., and Bruno D. 2013. BlueHoc: Bluetooth Ad-Hoc Network Android Distributed Computing. *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)* (2013), 445.

[6] BACARREZA NOGALES Ivris, Marcelo. 2007. Routing Protocol for Mobile Ad-hoc Wireless Networks. *RADIOENGINEERING* 16, 3 (2007), 86.

[7] Liu Kecheng, Shen Wenlong, Yin Bo, and Cao Xianghui. 2016. Development of mobile ad-hoc networks over wi-fi direct with off-the-shelf android phones. *IEEE international conference on communications (ICC)* (2016).

[8] Aneja Nagender and Gambhir Sapna. 2018. Profile-Based Ad Hoc Social Networking Using Wi-Fi Direct on the Top of Android. *Mobile Information Systems* 2018 (2018).

[9] Deepika P. and Indurani P. 2017. HYBRID INTELLIGENT TECHNIQUE FOR IMPROVING QUALITY OF SERVICE IN MANET. *One Day National Conference on Green Computing (CRGC)* (2017).

[10] Friedman R., Kogan A., and Krivolapov Y. 2013. On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones. *Transactions on Mobile Computing* 12, 7 (2013), 1363–1376.

[11] Shrivastava Sanjeev and Pargania Vivek. 2020. Survey of Routing Protocols, Simulation, Testing Tools and Mobility Models in MANET. *High Technology Letters* 26, 1 (2020), 23–30.

[12] Bluetooth SIG. 2019. Bluetooth Core Specification Version 5.2 Feature Overview. (2019). https://www.bluetooth.com/bluetooth-resources/bluetooth-core-specification-version-5-2-feature-overview/