Modeling of Bluetooth discovery for Model-Based Testing

Ruud Rupert
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
r.i.rupert@student.utwente.nl

ABSTRACT

Bluetooth devices require bandwidth to allow data transmissions in a timely manner. When interference is present or the devices are not timed correctly, retransmissions can happen. This can be exacerbated during the discovery phase of Bluetooth when devices are not synchronized yet. This paper researches the behavior of the Bluetooth discovery phase by producing a simplified Bluetooth model that makes use of physical waves to communicate. By using Model-Based Testing as our framework, a simulation can be formed by which the correctness of this system can be further verified. Results can also be gathered surrounding its performance and to gain insight into its implementation. This will be done within the Matlab Simulink environment, where the protocol will be simulated.

KEYWORDS

Bluetooth, Matlab Simulink, Model-Based Testing, Probabilistic Testing, Transmission Protocol, Extended Finite State Machines

1 INTRODUCTION

As more Internet of Things devices become integrated with our daily lives, interference becomes commonplace due to the limited bandwidth available. Because these WLAN devices can belong to separate networks, agreements in frequency channel usage behavior are required to mitigate collisions and interference during packet transmissions. For short-range devices running Bluetooth, this implies running both FDMA (Frequency Division Multiple Access) and TDMA (Time Division Multiple Access) [27]. TDMA specifies the time interval in which devices are allowed to transmit, while FDMA dictates which channel frequency to use. This combination helps sustain multiple different Bluetooth networks on a given bandwidth by minimizing the chance of using the same channel. This is then enhanced with frequency hopping, whereby the packets are transmitted on alternating channels in a pseudo-random manner [1]. This helps prevent interference as the numerous networks that share the 2.4 GHz bandwidth range are less likely to broadcast on the same channel at a given moment. In addition to this, AFH (Adaptive Frequency Hopping) tries to further record the presence of noisy channels [17]. These channels are then excluded, which can reduce interference from potentially frequency-locked networks [26].

43rd Twente Student Conference on IT July 4th, 2025, Enschede, The Netherlands.

While Adaptive Frequency Hopping does assist with noise, its usage is only after a connection has already been formed between the master device and one or more slave devices. Before such a connection has been formed, the discovery phase has to take place so that both devices are aware of each other's presence. A study from 2006 [7] replicated a simplified version of the protocol (according to Bluetooth 1.2 specification) without noise present and quantified the time required for a device to inquire and receive a response from a listening device. They discovered that this time can vary with an expected reply duration going from 625 μs to a maximum of 2.5716 s [7]. As the current Bluetooth 6 specification has not seen any alterations related to this particular protocol [1], this range is still applicable.

Due to the long channel hopping sequences both the inquiring and listening devices have to make, failure to successfully transmit a packet at the right moment can delay the discovery phase. This, in turn, can be detrimental to time-critical applications, for example, mobile devices that frequently need to switch networks. As such, this paper researches the Bluetooth discovery phase by modeling it in the form of a simplified state machine and comparing it to a system running under an environment capable of simulating packets. To aid development, Model-Based Testing will be applied to increase certainty of the system's behavior by checking if the implementation abides by the Bluetooth specification. Since the model abstracts from the implemented system, it will be possible to verify desired behaviors without needing to pay attention to the low-level implementation [25]. The creation of a simple yet functional model also allows for the automatic generation of test cases to test the system and produce traces of its behavior, upon which can be reasoned, providing insight into its properties.

The novel contributions provided by this paper are thus:

(1) The modeling of the Bluetooth discovery protocol as an extended finite state machine.
(2) The subsequent application of Model-Based Testing using this state machine on an implemented system of Bluetooth discovery.
(3) The results gathered from experimental data provided by running the system under test, additionally compared to the initial model.

2 RESEARCH QUESTIONS

Bluetooth discovery modeling has been done in numerous ways, though in most papers, the research has been more analytical. These models are too abstract from the implemented variants of this protocol and lack lower-level details. As such, a system has to be formed that is suited for simulating the discovery phase in a realistic way. To do this, Model-Based Testing will be used as it is highly suited

for conformance checking. By using a simplified model and an implemented system, behavior during physical transmission can be checked and compared to the specification given by Bluetooth.

As such, the following research questions can be derived:

- **RQ1**: How can the Bluetooth discovery protocol be modeled within the MBT framework?
- **RQ2**: How do low-level implementations affect Bluetooth discovery behavior concerning inquiry time duration?

This paper begins with section 3, discussing the research already done on the topic of Bluetooth modeling and how it differs from what is desired for our research. Section 4 provides background information surrounding the Model-Based Testing framework. Section 5 continues from this and explains the process of developing a model suitable for Bluetooth discovery and transforming it to allow for MBT. Section 6 examines the results produced by the model and the system that have been implemented within Matlab, while section 7 discusses these results. In addition, section 8 provides possible room for improvements for both the produced models and the testing environment used, as section 9 gives the general conclusion of the paper. A complementary appendix has also been added to further showcase the system and the ways it interacts with its environment.

3 RELATED WORK - BLUETOOTH MODELING

Multiple studies have already been performed which analyze Bluetooth's transmission when interference is present from IEEE802.11 (WiFi) [5, 12, 13, 14, 16], different Bluetooth networks [5, 8, 9, 15], and other short-range protocols like Zigbee [5]. However, these do not focus on the discovery phase as desired and limit themselves to expected throughput and or general packet loss present within the network.

Duflot et al [7] have modeled the Bluetooth discovery phase while making use of a probabilistic model checker called Prism. Their model is capable of checking the many possible states in which both the inquiring and listening devices can be, though it lacks features such as interference or multiple listeners. This minimizes the probabilistic behavior present, such as exponential backoff, which helps mitigate multiple listeners replying to a received inquiry simultaneously [2]. Other non-deterministic actions can occur when a packet is dropped, requiring either an inquiry or reply packet to be retransmitted.

Chakraborty et al [4] have modeled the Bluetooth discovery phase similarly to Duflot et al, but did include the backoff behavior present in the original Bluetooth specification [1]. Their research has produced similar results, focusing primarily on computing the average time required for an inquiry to be performed. However, their results are more experimental in methodology as they run their simulation with given values for the offset between the inquirer and listener before reasoning on the probabilistic results gathered.

Cordeiro et al [6] have modeled Bluetooth transmission, computing the average packet success probability with the presence of interference from other nearby pico networks. Though not focusing on the discovery phase of Bluetooth, their paper does illustrate the importance of distance and decibels with regard to interference influence.

4 BACKGROUND - MODEL-BASED TESTING

To test the behavior of a system, testing would be necessary. Though a software engineer could hypothetically write individual unit tests for each component, additional integration testing and system tests as a whole, the process could potentially diverge in correctness as specification requirements are not taken into account properly. The use of formal methods, such as Model-Based Testing, can help derive tests systematically [3]. Model-based testing entails the generation of test cases and subsequent execution of test cases to evaluate a system's compliance with its requirements-abiding model [10]. The model acts as an oracle system; given any input to it, the model compares the expected output with the one given by the system. Using this setup, a wide range of test cases can be generated on-the-fly as the system is traversed for all possible behaviors.

The primary advantage of Model-Based Testing compared to other testing methods is the separation of internal workings from the instructed behavior. The model only provides inputs and outputs on which the system will be tested; as such, the implementation of the system can be inaccessible and could be a more elaborate and detailed variant of the model or only a slight modification [24]. The main focus lies on whether or not the behavior of this implementation abides by the requirements present within the model. Similarly, the formal model produced can also be simplistic. In most cases, the model only describes high-level processes. This allows for abstraction as the lower-level details can be largely omitted.

To summarize, the execution of Model-Based Testing would then consists of: creating a requirement complying model, classifying the input and output actions, mirroring those to derive the model to compose with the system, and finally parallel composing this mirrored model (with use of an adapter if need be) to the implemented system. When formulating a formal model, the use of state machines is not necessary, although examining its execution path could be easier compared to regular code. Additionally, as coverage is regularly a metric for the completeness of a test, with finite state machines in particular, full coverage can be achieved by continuing test case generation until all possible states and edges have been reached by the model. For extended finite state machines as used in this paper, this might be more difficult to achieve as data variables increase the state space and subsequently the duration of testing.

The composed system forms an implementation under test. This system is assumed to have an implementation relationship with the formal model it conforms to, though whether this is the case or not can only be made known through conformance testing [22]. As the system interacts with the composed model, it will produce a set of observable actions called a trace [24]. If the implementation under test produces a trace that ends in a failing state, then it will have failed that respective test case and can be assumed to not comply with the requirements specification [11]. After sufficient test cases have been run without failure, the possibility that the system does not conform becomes improbable.

5 METHODOLOGY

5.1 MBT approach

To verify that a system meets the requirements laid upon it, Model-Based Testing will be used. By producing a simplified Bluetooth model that integrates all the specified requirements, the model is

capable of showcasing the expected behavior of the system under test. As the model allows for the generation of test cases, the system will be driven by these inputs and outputs to derive as many different responses and interactions as possible. Given that there is an implementation relationship between the model and system, if these tests all pass, the system can be seen as conforming to the model.

Since only the visible behavior is checked, the inner workings of the system can remain a black-box for the tester; only the inputs and outputs will be connected. Within the Bluetooth discovery model context, these would consist of inquiry and reply packets, which can be represented as simple events happening at discrete points in time. This allows for abstraction when it comes to the processes that take place within the Bluetooth system, such as the timing mechanisms, the wave generation, and the frequency hopping scheme. The same applies to the actions of sending and receiving packets, as the model does not need to receive the actual packets. Only the fact that a packet has been sent or has been received is needed for it to function.

To produce a testing model, an initial formal model of the specified requirements will need to be created. As input-output conformance will be the main focus, the interactions the model makes with the outside world will be represented as specially indicated transitions to allow for such behaviors to be visible. For this paper's Model-Based Testing, inputs are marked ending with a question mark, while outputs end with an exclamation point. Other transitions could be seen as internal processes and should not be made accessible for checking the system's behavior. By subsequently mirroring the visible actions as visible in Figure 2, the model now checks whether an expected action has occurred or moves to a state, such as to get to such a point. During testing, both the test model and system will be traversed as actions are passed to and from one another. Depending on the strictness of the testing, actions which are not classified as possible transitions at a given state could either lead to failure or be ignored [23]. Further considerations have to be made when it comes to quiescence, δ, when no visible actions take place. Tretman formalizes that the event of not seeing any output could be considered a type of output itself [24], traces which contain these actions are suspension traces and can be modeled directly by providing a dedicated transition and respective state for quiescence, turning the model into a suspension automaton. Classifying this inactivity can cause issues as it forces the model to wait indefinitely to conclude the lack of an output action. A more practical approach mentioned by Tretman is the inclusion of time-outs, which need to be long enough as not to falsely return a quiescence transition. For our model, this method will be applied.

Though Model-Based Testing is not limited to state machines, their use helps showcase the expected input and outputs given by a model, as transitions can highlight when a model expects an outside event to take place, in this case, a packet to be sent or received. The events originating from the system can be logged and checked for their consistency with regard to the model. This would produce a trace of actions that can be reasoned on further using input-output conformance.

The resulting model and adapter combination (see Figure 1) will be capable of providing inputs and receiving outputs to and from the system to check its correctness. Additionally, the model can be

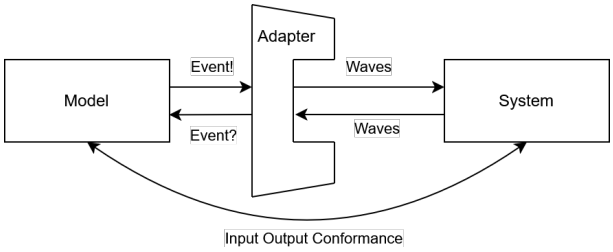

Figure 1: Setup of Model-Based Testing environment

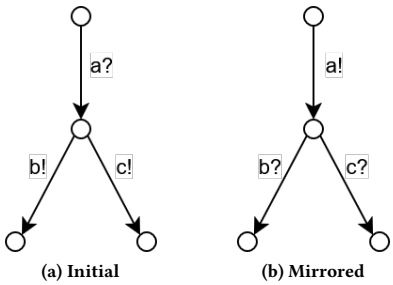

Figure 2: Example of mirroring

used to verify that the system meets the requirements given by the Bluetooth discovery protocol and can provide measurements that can be reasoned with.

5.2 Environment and Tools

The Model-Based Testing framework is implemented within Matlab, which possesses both Simulink, a package capable of modeling extended finite state machines, and the Bluetooth Toolbox, which can be used to generate Bluetooth packets and waves. These components allow for the simulation of lower-level physical signals that would otherwise overcomplicate the adapter's and system's code. Because of this, the system and its respective model will be simulated within this toolbox. Furthermore, Matlab possesses numerous functionalities and tools such as signal multiplexers, sample time generators to set execution interval of the model, and apps to display results in the form of diagrams. Since the model and system need to follow strict time intervals, these will be vital for checking correctness.

5.3 Modeling the Bluetooth discovery protocol

The formal model produced follows a similar formation to the labeled transition system given by Duflot et al [7]. Although the researchers end up implementing their model using Prism. This environment is well-suited for modeling systems and protocols that possess probabilism [21], though it lacks features to simulate less abstract and more physical concepts. Our paper intends to make use of their formal model by refactoring it to an extended finite state machine. This is required due to the need to maintain the values of variables such as the current and subsequent channel frequencies, but also to calculate the random number of time slots the listener needs to sleep for the backoff state. As specified by the

Bluetooth specification, the discovery phase consists of two specific components, an inquiring device and a listening device [2]. The purpose of the protocol is for the inquiring device to discover all listening devices in range for a given amount of time. This is done by transmitting inquiry packets to dedicated channels within the 2.4 GHz band range. Each inquiry packet can be transmitted to one of 32 specified frequency channels, which are separated from one another by 2 MHz. The transmission of these packets happens on a controlled basis using time slots lasting 0.3125 milliseconds each. The frequency channel to which these packets are transmitted relies on the frequency hopping scheme described by Bluetooth. The pattern consists of two channel frequency sets, which are iterated from 1 to 16, defined as track A, and 17 to 32, called track B. Track A is repeated 256 times, at which point track B is followed for the same number of iterations. As soon as a given track has iterated 128 times, its first non-swapped value will be swapped with its equally indexed value on the opposite track. This way, both tracks will have fully swapped after 4096 iterations and returned to their original values after 8192 iterations. After a packet has been transmitted on a given time slot, the inquirer listens to the same frequency it has transmitted to after 2 time units. This means that an inquirer (see Figure 3) can transmit to two different channels sequentially and afterwards listen to those two respective channels for reply packets from any nearby listeners. As soon as a reply has been received, the central inquirer can finish the discovery phase and start paging the peripheral listener to initiate a connection using its newly received address and clock offset [1], or it can continue scanning for other listening devices in the vicinity.

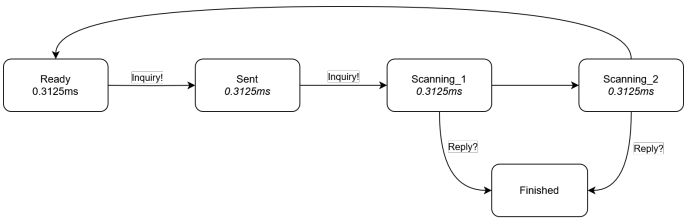

Figure 3: Labeled Transition System inquiring device

The listener (see Figure 4) follows a similar process to search for inquiring devices. However, instead of actively searching for inquiring devices, it spends the majority of the time sleeping. The device wakes up periodically and listens to transmitting devices using the same frequency hopping scheme as inquiring devices. As the listener is not continuously hopping sequences, the inquiring device will eventually catch up to the listening device and be capable of transmitting a packet to the same channel the listener is scanning for. At this point, the listener will wait 2 time slots such that it can return a reply packet as the inquirer is scanning the frequency channel. As multiple Bluetooth listeners can be present within a given area, listeners will need to showcase probabilism to mitigate interference from simultaneous inquiry replies. Within Bluetooth discovery, this entails having a random backoff period after a reply has been sent to an inquiring device. The waiting time can be simplified to a random number between [0, 128] multiplied by twice the average time slot.

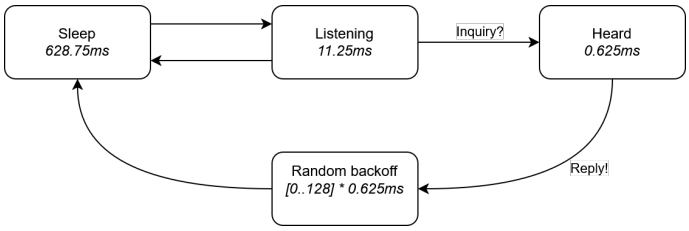

Figure 4: Labeled Transition System listening device

The extended finite state machine of the Bluetooth discovery protocol showcases these requirements and allows inquiring devices to establish inquiries towards listening devices. The listener model is similar to the one retrieved from Duflot et al [7], though the implementation of this model within Matlab will need minor alterations as transitions will need to be marked for when a state runs out of time. The same applies to the inquirer who will only remain in a given state for a limited amount of time, listening to incoming actions before moving on to the next state.

5.4 Transformation Model-Based Testing

Although this model would be capable of running stand-alone and deriving results, the main interest is to compare it to a system under test while simulating low-level physical interactions in the form of waves and packets. To accommodate providing inputs and outputs this way, an adapter is needed between the model and the system. The Bluetooth Toolbox of Matlab contains functions that allow for the generation of Bluetooth packets in the form of waves, and subsequent receiving of these waves to transform them back into packets. Decoding capabilities are also present to indicate what type of packet has been received. This signal can be fed into the model to indicate that a given action has taken place.

For Bluetooth discovery, Model-Based Testing can be applied in two distinct ways: focusing on the discovery process as a holistic whole or testing the individual components (the inquirer and the listener) for their behavior. Testing the system as a whole increases complexity as synchronization becomes difficult to maintain. Additional issues can occur when probabilistic actions happen when the model becomes unsure how long a backoff period should last. The presence of quiescence implies the lack of visible output actions; in this case, the mirrored model would have to send an inquiry packet at an unknown amount of time, as the listener system is outputting no actions for a randomized amount of time. Without breaking encapsulation and finding out directly if the listener is done with its backoff, the model could wait indefinitely or fail the test due to prematurely detecting quiescence incorrectly.

Instead, by simulating the system piecewise as independent components, the results gathered can be more accurate and less prone to failure due to these factors. Furthermore, the structure itself becomes more comprehensible, which assists with the goal of requirements checking. To test the inquirer, the model will need its transitions to be mirrored. That way, when the behaviors match,

an output from the system will feed directly into the respective input of the transition in the model. The modeling of the listener follows a similar pattern. Since the component under testing needs to eventually finish execution, it will be limited to a certain number of iterations before finishing automatically. The listener can exhibit quiescence at some points of execution for unknown amounts of time, which makes accurately testing it in the same manner difficult. Because of this, its Model-Based Testing will be left out, though it would not be impossible to achieve in future adaptations (possibly breaking encapsulation to speed up testing runtime).

6 RESULTS

6.1 Results RQ1: Bluetooth Discovery Modeling

The setup as described in Figure 1 within the methodology has been implemented inside the Matlab Simulink environment. There, the inquirer component (see appendix - Figure 7) runs using the Model-Based Testing framework as events are given and read from it while being indirectly compared to its respective inquirer model (see appendix - Figure 10). These actions are used to call functions, which correlate to the adapter, where their respective packets are made and transformed into waves using a Bluetooth wave generator function. The Bluetooth Toolbox is limited in how the waves can be generated. Bluetooth requirements state that Bluetooth packets are transmitted using Gaussian Frequency Shift Keying, which modulates the Bluetooth packet to be sent to a specified carrier frequency [1]. As the given functions only allow for general packet wave generation, this requirement cannot be met using the wave's frequency properties. Instead, an indirect method was used whereby the receiver checks the frequency at which the packet was supposed to be sent and compares it to the frequency of the channel it is currently scanning. Possible adaptations could be made to manually create, transmit, receive, and decode Bluetooth packets; however, this would naturally increase complexity and processing time as signal processing would modify the transmit time of a packet and require modifications to the time intervals for each state. Another option would be to decode the packets generated from the Bluetooth toolbox back to bits before remodulating them to a specified frequency. This would naturally produce excess noise, consequently requiring filtering stages, which can be computationally heavy. Although the current method of generating general Bluetooth waves lacks the capabilities to set the carrier frequency directly, it is still capable of producing semi-randomized packet waves. In the case of a reply packet, these waves will differ the most as they usually carry payload describing the listener device's properties, such as address and access codes. Further differences between waves are present due to noise generated during their transmission. Noise levels can be tweaked, although for Model-Based Testing purposes, this would most likely prevent actions from being accurately decoded back into signals and would be detrimental to generating and feeding test cases. The signals are then fed back into the extended finite state machines to allow for conditional transitions to take place, such as changing the current state after an inquiry packet has arrived.

6.2 Results RQ2: Inquiry time duration

The original model, as described in the methodology, has also been implemented within the Matlab Simulink environment (see appendix - Figure 7). On each iteration, the inquirer and the listener attempt to perform the inquiry protocol before resetting and randomizing their initial variables. This mimics the randomness of the algorithm as its real-world implementation relies on the current time for initialization. After continuously running the model, a set of sample times was collected with differing amounts of elements. These results are displayed in Figure 5. During the best-case scenario, the listener scans while the inquirer is transmitting, allowing a minimal time of 2 time slots (0.625ms), while the worst-case scenario gives a time range of 5.75s. Duflot et al [7] provide a maximum of 2.57s, which is way lower than this supposed worst-case scenario.

On the topic of average inquiry time duration, the samples collected produce a mean value of around 1.45s. Chakraborty et al [4] results differ depending on the number of devices present and the backoff period present within the listener, for our current setup of 2 devices (1 inquirer and 1 listener) and a backoff limit of 255, their estimate ranges around the 2.1s mark.

The possibility of synchronization issues within the Matlab environment or even glitched states that prevent correct transitioning of the model should not be excluded. Although Matlab attempts to discretely model its simulation, the possibility exists of concurrent writes or reads overlapping in non-deterministic manners. This is especially the case when working with multiple individual models that run simultaneously.

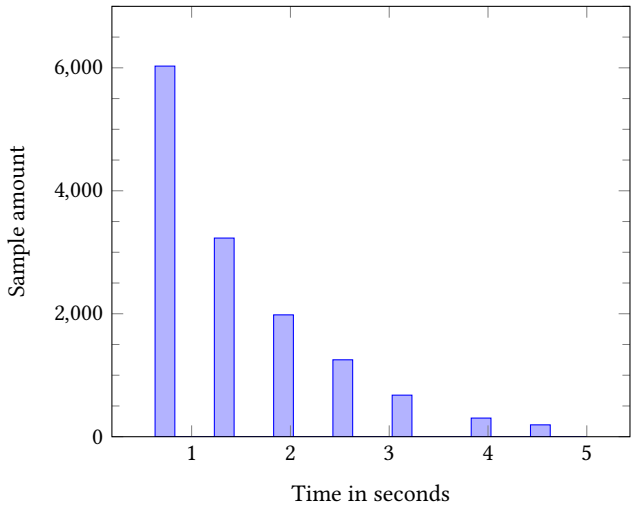

Figure 5: Inquiry success model (n=13820)

For the physical-simulating system (see appendix - Figure 8), tests are run similarly to the original model by using an adapter that forces it to transform inquiry and reply actions into their respective wave packets. These are subsequently decoded before being passed to any listening components. This necessitated minor

modifications to the runtime configurations due to the strict loop-avoidance measures Matlab takes to prevent recursive event calls [18]. In the case of event broadcasts, which can be seen as the inputs and outputs being passed to and from the models, these can feed into themselves or accidentally update themselves within the same tick, causing a recursion loop to occur [19, 20]. To return the system to a time-based system instead of an event-based one, the inclusion of a periodic function call generator was needed. This function regularly passes events to the system, causing it to wake up every 0.3125 milliseconds to check its current processes and transition to a new state if allowed. A major side effect of using a more detailed system is the increase in the runtime duration. Although the model can execute multiple seconds of simulation time within a single runtime second, the system requires multiple seconds of real time for a single simulation second. This considerably limits the number of samples that can be gathered within a reasonable time span. The results for the system are displayed in Figure 6.

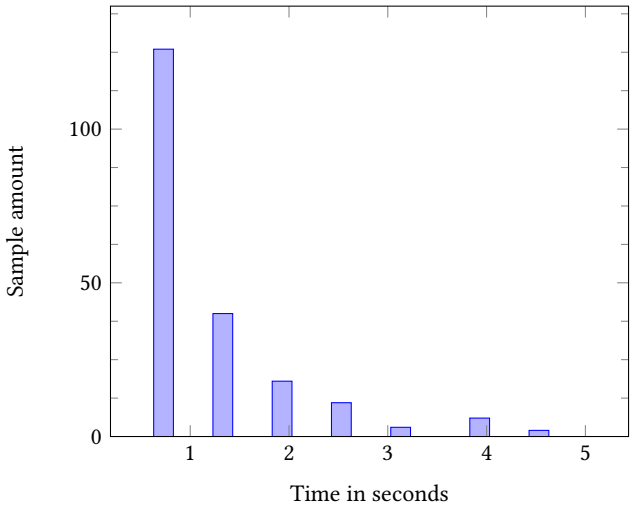

Figure 6: Inquiry success system (n=210)

7 DISCUSSION

Using the results gathered from the previous sections, the previously declared research questions can be answered.

***RQ1**: How can the Bluetooth discovery protocol be modeled within the MBT framework?*

Using the Matlab Simulink environment, the Bluetooth discovery protocol can be partially modeled by way of extended finite state machines. As stated before within the related works section, this way of modeling assists with the translation of the requirements needed and further helps with classifying the inputs and outputs as they match with the transitions between states. This model can then be implemented in more detail as a system before being mirrored to use as a test case generator. By linking both parts using an adapter, the model maintains high abstraction as the details are taken care of by the coupling between it and the system. Additional inclusions, such as unexpected inputs that result in a fail state or long periods of inaction, may also be added to detect incorrect behavior. At this point, the model and system are composed, and as long as the environment has been configured correctly by way of sampling time and execution order, they will continue testing until an error occurs.

The use of a divide and conquer approach is recommended for larger and complex systems or those that rely heavily on synchronization. As Bluetooth discovery requires both systems to be in sync with one another during execution, splitting up the individual components and testing them individually becomes a necessity. Within Bluetooth discovery, the presence of probabilistic behavior in internal process durations heavily restricted the modeling process. The only way to efficiently model would then be by breaking the encapsulation of the system, though in non-transparent systems, this could be difficult to achieve.

***RQ2**: How do low-level implementations affect Bluetooth discovery behavior concerning inquiry time duration?*

As shown by the graphs, the results differ slightly, though this could be due to the smaller sample size produced by the implemented system. Changes within the implementation to account for lower-level interactions have led to differences in results as the system suffers from delays in packet arrival and starts missing key events. Although both implementations suffer from deviations compared to the other Bluetooth discovery papers, the order of magnitude is the same and would indicate the presence of exceptions taking place within the code or Matlab environment.

8 LIMITATIONS

Numerous modifications were made during the modeling and implementation of the system due to limitations within the Matlab environment or circumstantial constraints. Limitations more specific to the methodology of this paper would be the reuse of the model for the system's implementation. Although both the model and system are alike in functionalities and behavior, the system is expected to be less abstract and more fine-grained in its processes. Within this paper's context, using the initial model as the system would not be incorrect, as the requirements are still being met. Though one major concern is the performance of the produced system, as the underlying code consists primarily of a state machine, which should be translated to a more efficient code base. Fortunately, the environment depends on sample time and not real time, as Matlab can spend as much time as needed to execute the system while the simulation time is kept to a frame of microsecond-long time units.

Another missed factor would be the lack of drift inside the environment, as both model and system follow the same sample rate with the same phase. As a result, there is no direct concern for out-of-time packet transmissions or interruption during transmission, though this does reduce the accuracy of the findings. Other lapses in realism are the presence of only one listener and inquirer. It is not unusual for multiple Bluetooth devices to be present within range and ready to reply to incoming inquiry packets. As such, any given inquiry could lead to multiple wave packets being returned

and resulting in interference. Although the backoff protocol to minimize the effect of this has been implemented within the listener component, simulating multiple listeners has not been done due to time constraints, and further increases in complexity it would cause.

Finally, the state machines provided by Matlab lack granularity to adjust the waiting times of transitions. Adding a transition to every possible state to leave under the usual time slot duration length of 628.75ms or 11.25ms in the case of a listener would potentially fix this, although performance-wise and clarity-wise, this is not ideal. As a result, this limits the randomness of the initial states, as it is not possible to be in a continuous state where it is already partially done waiting. This skews the collected measurements to the right, as the inquiry process always starts with the listener waiting patiently in the waiting state. This means the minimum inquiry time is at least 628.75ms larger than expected when compared with the results from Duflot et al [7].

9 CONCLUSION

In summary, this paper has investigated the Bluetooth discovery protocol with regard to simulating it and using Model-Based Testing to verify its behavior. By following the requirements given by the Bluetooth specification and subsequently translating them into a model, we produced an extended finite state machine capable of generating test cases for a Bluetooth system to follow. The model itself has been compared with other similar Bluetooth discovery papers' results and has shown differences, due to the lower precision of the model and more likely differences in the implementation and environment. The Matlab environment has showcased the possibility of modeling lower-level physical Bluetooth waves for use between a system and its respective formal model. The inclusion, however, has led to increased complexity and a worsening runtime as a result. This trade-off might not be worth it for analytical research, though, for more accurate and realistic testing, it could be of use. The further inclusion of Model-Based Testing has assisted with requirements checking and system verification.

10 FUTURE WORK

The research performed in this paper has provided insight into the modeling of the Bluetooth discovery and the subsequent testing thereof. Future works that intend to follow this paper's methodology should keep in mind the limitations mentioned as possible points of improvement. More general alterations could be made to the wave generation itself, as given in the results section of research question 1. Transmitting packets according to Gaussian Frequency Shift Keying would produce more true-to-reality waves and could help with system realism. Another point to focus on would be the timing between the model and the system. As this paper worked within a discrete Simulink environment, changes in phase and timing were nonexistent. As such, more research should be done while variations in time are present. Finally, during the development of the model and system, extended finite state machines were heavily used in combination with the Simulink environment. These helped with producing an interface in synchronization with both the model and the system. If the system implemented were not a state machine nor accessible during development, connecting it this way would have been difficult to achieve. Further research on the topic of synchronizing and adapter modeling with regard to non-timed systems could be of use.

REFERENCES

[1] Bluetooth. 2024. Bluetooth core specification. Volume 2, Part B.2. (Aug. 2024). Retrieved June 17, 2025 from https://www.bluetooth.com/specifications/specs/core-specification-6-0/.

[2] Bluetooth. 2024. Bluetooth core specification. Volume 2, Part B.8. (Aug. 2024). Retrieved June 17, 2025 from https://www.bluetooth.com/specifications/specs/core-specification-6-0/.

[3] Ed Brinksma. 1999. Formal methods for conformance testing: theory can be practical. In *Computer Aided Verification*. Nicolas Halbwachs and Doron Peled, (Eds.) Springer Berlin Heidelberg, Berlin, Heidelberg, 44–46. ISBN: 978-3-540-48683-1.

[4] Goutam Chakraborty, Kshirasagar Naik, Debasish Chakraborty, Norio Shiratori, and David Wei. 2010. Analysis of the bluetooth device discovery protocol. *Wireless Networks*, 16, (Feb. 2010), 421–436. DOI: 10.1007/s11276-008-0142-1.

[5] R. Challoo, A. Oladeinde, N. Yilmazer, S. Ozcelik, and L. Challoo. 2012. An overview and assessment of wireless technologies and co- existence of zigbee, bluetooth and wi-fi devices. *Procedia Computer Science*, 12, 386–391. Complex Adaptive Systems 2012. DOI: https://doi.org/10.1016/j.procs.2012.09.091.

[6] C. Cordeiro, Dharma Agrawal, and Djamel Sadok. 2003. Interference modeling and performance of bluetooth mac protocol. *Wireless Communications, IEEE Transactions on*, 2, (Dec. 2003), 1240–1246. DOI: 10.1109/TWC.2003.819026.

[7] Marie Duflot, Marta Kwiatkowska, Gethin Norman, and David Parker. 2006. A formal analysis of bluetooth device discovery. en. *Int. J. Softw. Tools Technol. Transf.*, 8, 6, (Oct. 2006), 621–632. DOI: 10.1007/s10009-006-0014-x.

[8] Wang Feng, N. Arumugam, and G. Hari Krishna. 2002. Impact of interference on a bluetooth network in the 2.4 ghz ism band. In *The 8th International Conference on Communication Systems, 2002. ICCS 2002.* Vol. 2, 820–823 vol.2. DOI: 10.1109/ICCS.2002.1183244.

[9] F. Floren, A. Stranne, O. Edfors, and B.-A. Molin. 2001. Throughput analysis of strongly interfering slow frequency-hopping wireless networks. In *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No.01CH37202).* Vol. 1, 496–500 vol.1. DOI: 10.1109/VETECS.2001.944892.

[10] Marcus Gerhold. 2018. *Choice and chance: model-based testing of stochastic behaviour*. English. PhD Thesis - Research UT, graduation UT. University of Twente, Netherlands, (Dec. 2018). ISBN: 978-90-365-4695-9. DOI: 10.3990/1.9789036546959.

[11] Marcus Gerhold. 2018. *Choice and chance: model-based testing of stochastic behaviour*. English. PhD Thesis - Research UT, graduation UT. University of Twente, Netherlands, (Dec. 2018). ISBN: 978-90-365-4695-9. DOI: 10.3990/1.9789036546959.

[12] N. Golmie. 2004. Bluetooth dynamic scheduling and interference mitigation. *Mobile Networks and Applications*, 9, 1, (Feb. 2004), 21–31. DOI: 10.1023/A:1027313621955.

[13] N. Golmie and F. Mouveaux. 2001. Interference in the 2.4 ghz ism band: impact on the bluetooth access control performance. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240).* Vol. 8, 2540–2545 vol.8. DOI: 10.1109/ICC.2001.936608.

[14] N. Golmie, R. E. Van Dyck, A. Soltanian, A. Tonnerre, and O. Rébala. 2003. Interference evaluation of bluetooth and ieee 802.11b systems. *Wireless Networks*, 9, 3, (May 2003), 201–211. DOI: 10.1023/A:1022821110023.

[15] A. Karnik and A. Kumar. 2000. Performance analysis of the bluetooth physical layer. In *2000 IEEE International Conference on Personal Wireless Communications. Conference Proceedings (Cat. No.00TH8488)*, 70–74. DOI: 10.1109/ICPWC.2000.905775.

[16] J. Lansford, A. Stephens, and R. Nevo. 2000. Wi-fi (802.11b) and bluetooth: enabling coexistence. *IEEE Network*, 15, 5, 20–27. DOI: 10.1109/65.953230.

[17] Jason Marcel. [n. d.] How bluetooth technology uses adaptive frequency hopping to overcome packet interference. Retrieved June 18, 2025 from https://www.bluetooth.com/blog/how-bluetooth-technology-uses-adaptive-frequency-hopping-to-overcome-packet-interference/.

[18] MathWorks. 2025. Algebraic loop concepts. (2025). Retrieved June 24, 2025 from https://nl.mathworks.com/help/simulink/ug/algebraic-loops.html.

[19] MathWorks. 2025. Broadcast local events to synchronize parallel states. (2025). Retrieved June 24, 2025 from https://nl.mathworks.com/help/stateflow/ug/broadcasting-events-to-synchronize-states.html.

[20] MathWorks. 2025. Resolve unintended recursive behavior. (2025). Retrieved June 24, 2025 from https://nl.mathworks.com/help/stateflow/ug/guidelines-for-avoiding-unwanted-recursion-in-a-chart.html.

[21] PRISM. 2025. Prism - probabilistic symbolic model checker. (2025). Retrieved June 25, 2025 from https://www.prismmodelchecker.org/.

[22] G.J. Tretmans. 1996. Test generation with inputs, outputs and repetitive quiescence. Undefined. *Software : concepts tools*, 17, 3, 103–120. An update of this work has been published as http://dx.doi.org/10.1007/978-3-540-78917-8$_1$.

[23] Jan Tretmans. 1996. Conformance testing with labelled transition systems: implementation relations and test generation. *Computer Networks and ISDN Systems*, 29, 1, 49–79. Protocol Testing. DOI: https://doi.org/10.1016/S0169-7552(96)00017-7.

[24] Jan Tretmans. 2008. Model based testing with labelled transition systems. In *Formal Methods and Testing*. Lecture notes in computer science. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–38. https://link.springer.com/chapter/10.1007/978-3-540-78917-8_1.

[25] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A taxonomy of model-based testing. *Software Testing, Verification and Reliability*, 22, (Aug. 2012). DOI: 10.1002/stvr.456.

[26] Yin Wenlong, Cheng Yunpeng, and Shen Liang. 2011. Adaptive frequency-hopping in hf communications. In *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, 427–430. DOI: 10.1109/TMEE.2011.6199233.

[27] Martin Woolley. 2025. Bluetooth 5.2 feature overview. (Jan. 2025). Retrieved May 24, 2025 from https://www.bluetooth.com/wp-content/uploads/2020/01/Bluetooth_5.2_Feature_Overview.pdf.

11 APPENDIX

This section contains supplementary images that showcase the system and the environment where it is used.

Figure 7 displays the extended finite state machine as modeled within the Matlab Simulink environment. The model consists of the two Bluetooth discovery protocol components and an additional component used to set up the testing sequence. All three portions of the finite state machine run in an and-relationship (shown by the dotted line surrounding them), indicating that they run concurrently. Due to the timed nature of the protocol, each phase has only a limited amount to run before being transitioned out by "[after()]" conditions. These are checked every tick (0.3125 milliseconds) and record the amount of time that has elapsed within a given state.

Further features which are important to understand the model are the presence of events within some of the transitions (these are indicated in yellow). These events match with actions that are input/output concerning the model. The send() function implies that the event is outgoing.

Figure 8 displays the physical modeling of the system. Within the yellow block called "System Under Test" sits the extended finite state machine as described in Figure 7. The outgoing event calls are used as inputs, as they are hooked up to code blocks capable of generating Bluetooth packet waves. These are subsequently decoded and fed back into the system, forming a loop. In this Simulink configuration, loop avoidance is not seen as an issue due to there being only a single model to reference.

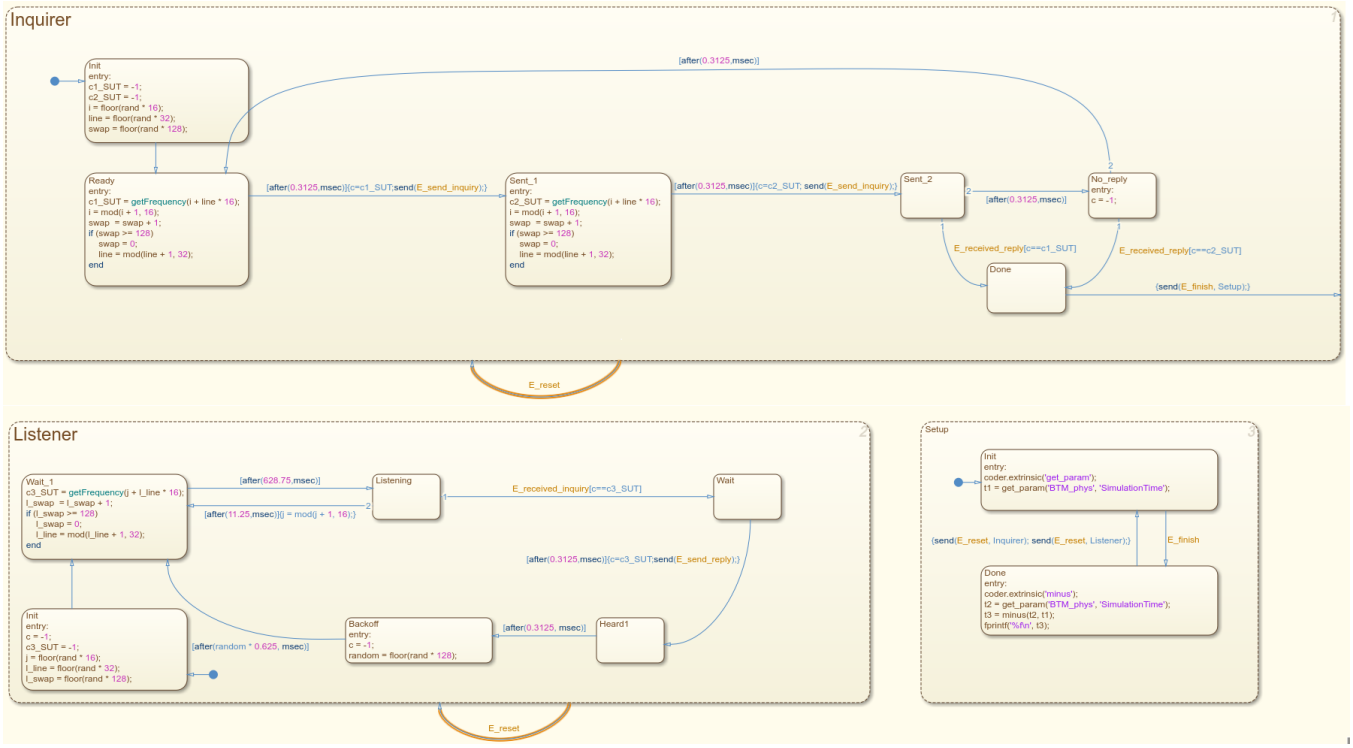

Figure 7: Extended finite state machine used by both system and model

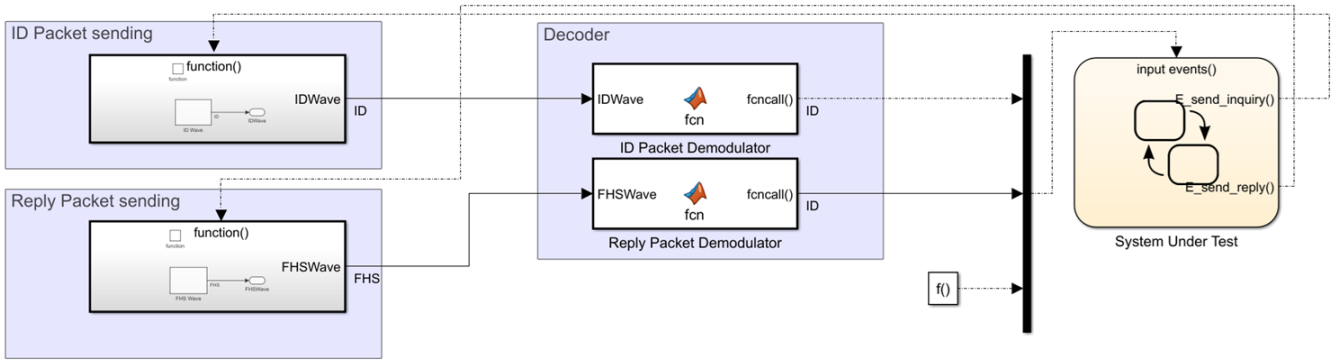

Figure 8: Matlab setup system (System Under Test block contains Figure 7)

Figure 9 displays the Simulink environment where Model-Based Testing takes place. Its appearance is similar to the setup provided in Figure 8, though the results of transmitting a packet of the system are fed to the model instead of looping back into the system. The model also transmits to the system, which is used to both test if the inquirer is capable of accepting reply packets and to indicate the end of testing. Some decoding signals are not fed back into the model or system, as they do not make use of them.

Figure 10 displays the mirrored implemented version of the inquirer model as prescribed in the methodology (see Figure 3). Its input and output actions are swapped such that it expects an inquiry twice before progressing with either transmitting a reply packet or doing nothing. Since the system finishes executing when it receives a reply packet, this branch happens only after a given number of iterations has occurred, which can be set using a variable.

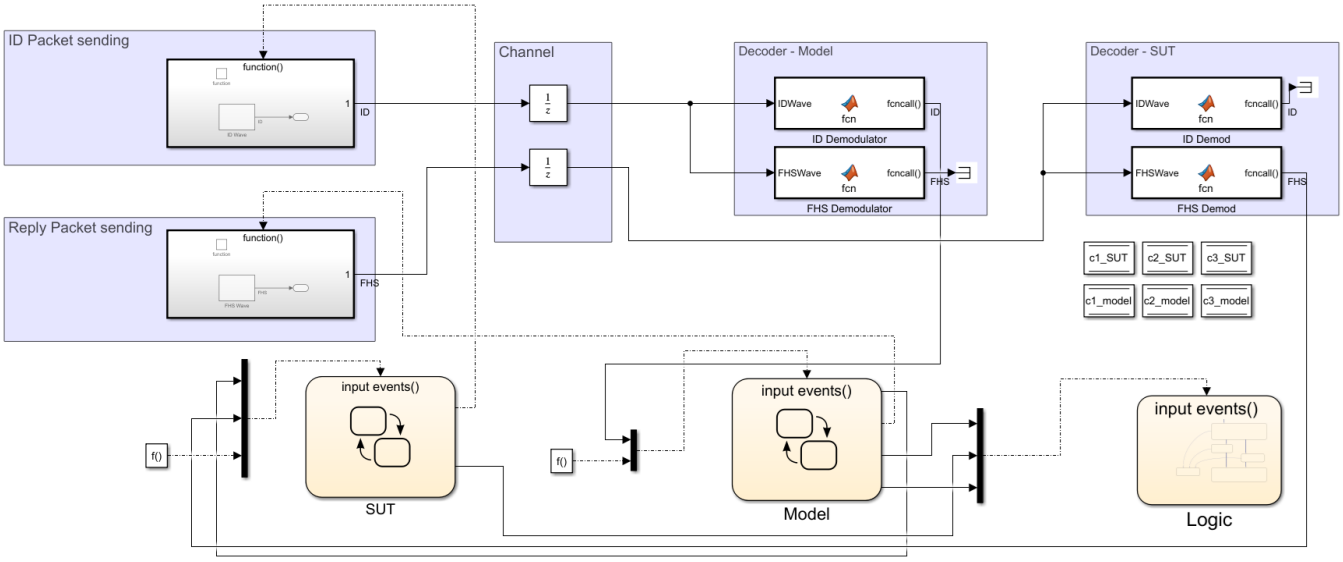

Figure 9: Model-Based Testing setup inquirer

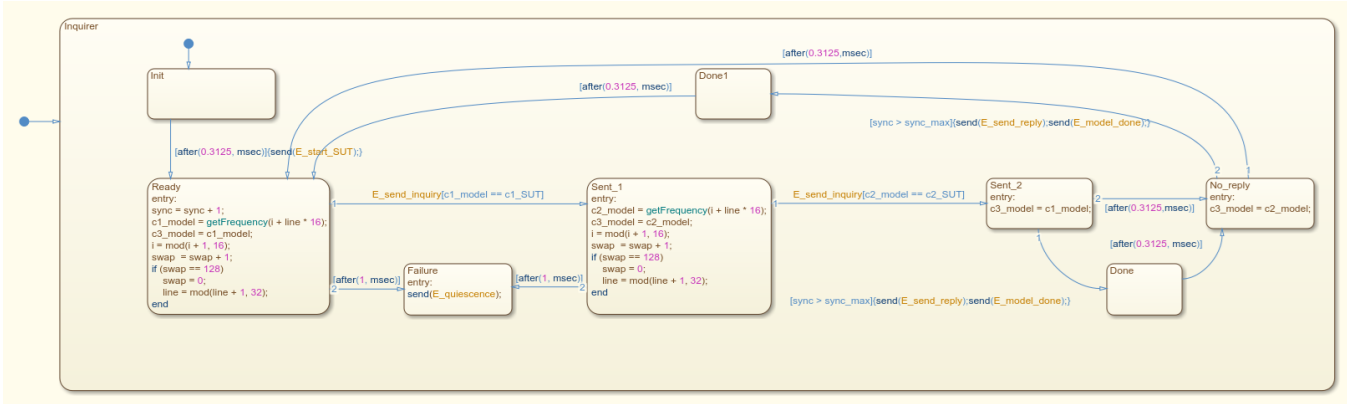

Figure 10: Mirrored model of inquirer