Parameter Configuration in Bluetooth Mesh: Performance and Limitations

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Robotics and autonomous systems
May 2025
Teemu Mäkinen

UNIVERSITY OF TURKU
Department of Computing

Teemu Mäkinen: Parameter Configuration in Bluetooth Mesh: Performance and Limitations

Master of Science (Tech) Thesis, 68 p., 1 app. p.
Robotics and autonomous systems
May 2025

Development of Internet of Things (IoT) networks requires careful planning, not only to meet application specifications but also to maximize performance. Selecting a suitable communication technology can be a challenging task, but it is only one part of the process. Optimization of technology-specific parameters plays a significant role in achieving desirable results for different application use cases.
This thesis studies a networking technology known as Bluetooth Mesh, focusing on its configurability. The effects of various important parameters are assessed through an experimental evaluation, which involves different network scenarios aiming to consider variations in network density, coverage area, and traffic intensity. Communication reliability and latency are the primary performance statistics to be considered, although other important aspects, such as energy consumption, are examined as well. Additional attention is given to underlying limitations that largely define the overall dimensions of system configuration, especially in terms of memory usage.

Keywords: bluetooth, mesh, configuration, wireless, iot, performance

Contents

Appendices

1 Introduction

The Internet of Things is one of the most prominent concepts in today's technological landscape. Enabling whole new levels of automation and smart systems, the IoT has gained increasingly widespread popularity in recent years. Wide range of domains from industry and urban infrastructure to healthcare and housing are all affected by the emergence of new IoT applications.

As the number of connected devices continues to grow, and with constantly evolving technological needs for new applications, the development of robust, scalable and efficient communication technologies has become increasingly critical. Even the most extensively used communication technologies, such as Wi-Fi and Bluetooth, have to adapt to new requirements set by the IoT industry.

This thesis focuses on studying in greater detail a communication technology known as Bluetooth Mesh. As the name implies, Bluetooth Mesh is a networking protocol built on top of the Bluetooth Low Energy (BLE) standard, enabling decentral many-to-many communication with mesh networking principles. The Bluetooth Mesh standard was originally released in 2017, and it has since received its first revision update in 2023 with plenty of feature additions and enhancements.

As is the case with any other protocol, the question of whether Bluetooth Mesh is good or not depends greatly on the application. Since no single technology is always suitable for all use cases, it becomes crucially important to understand the restrictions, requirements and performance elements of different technologies. An-

other interesting aspect to consider comes from the configurability of the technology, as it often tends to affect these three key points.

1.1 Motivation and objectives

Perhaps partly due to the novelty of the standard, only a relatively few research papers study Bluetooth Mesh in more depth. Eg. the coverage of Bluetooth Mesh configurability is quite limited, especially considering the wide range of configuration options available even in the base standard. Additionally, most related studies have been published before the protocol revision 1.1 release in 2023.

With this in mind, the main research objectives of this thesis are to gain a better understanding of the impact of various Bluetooth Mesh network configuration parameters in order to determine the technology's main limitations from an application design perspective and to identify any other potential issues with a focus on mitigating them. To achieve this, an experimental study will be conducted with small network installations consisting of nRF52840 development kits. During the experimentation, some core network performance metrics will be evaluated against various network configurations. Importantly, resource usage aspects (eg. memory and power consumption) of the different configuration options will be considered as well.

1.2 Thesis structure

Chapter 2 provides some additional context of today's IoT application landscape and clarifies important concepts related to networking and device-to-device communication. Chapter 3 discusses the operation principles of Bluetooth Mesh technology and some of its core features. Chapter 4 describes the setup of the experimental phase of this thesis. The different configuration parameters involved in the experi-

mentation are also listed and explained. In chapter 5, the results of the experimental phase are analyzed and discussed in detail. Finally, the conclusions of this thesis are drawn in chapter 6.

2 Internet of Things

The term Internet of Things has gained quite a lot of popularity and it has been used relatively freely in different contexts. Due to this, the definition of IoT is not completely unambiguous, although the basic concept always relates to some sort of interconnectivity between multiple devices [1]. While the precise definition may vary, understanding the key concepts of IoT is instrumental when discussing communication technologies, such as Bluetooth Mesh, from an application perspective.

2.1 Applications

Typical applications of IoT systems often cover some sort of monitoring, control and/or automation functions. Good examples of application fields that are projected to benefit most from IoT include smart homes, smart cities, smart business management, environmental monitoring, healthcare and security [2]. Within these areas, new applications leveraging the power of IoT can help conserve energy (eg. smart lighting and thermostats), increase the efficiency of infrastructure and logistics (traffic monitoring) and even save lives (disaster alarm systems and healthcare applications).

The healthcare sector is a prime example of an area where the IoT can bring revolutionary improvements in service quality and efficiency, but it also poses significant challenges from an application development perspective. For instance, IoT enables real-time remote monitoring of essential patient data (eg. heart activity or

glucose levels) with the help of wearable or implantable sensors. Of course, reliable and fail-proof systems are required to ensure continuous, accurate data delivery. Moreover, privacy is a fundamental concern for all systems that handle sensitive data, meaning that robust security measures need to be implemented at every level of the system architecture. Efficient capitalization of the opportunities created by IoT also requires seamless interoperability between different devices, underlining the importance of standardized protocols and interfaces. [3]

The IoT landscape has seen rapid growth in recent years, both in terms of technological advancements and application prospects. Estimates on the number of connected devices already reach tens of billions, and with no sign of slowing down in the future [4]. In addition, considering the wide range of IoT application areas, it should come as no surprise that the requirements IoT systems need to face, and the benefits that they can bring, will have huge variations depending on the application scenario. For example, while data privacy is an utmost important component in healthcare applications, other sectors, such as industrial automation or smart cities, may prioritize factors like scalability, flexibility, energy efficiency, or responsiveness [5].

2.2 Communication technologies

The revolutionary impact of IoT applications can largely be attributed to the distributed system design made possible by the widespread availability of wireless communication. Today, devices no longer need to be connected by cables, allowing them to be installed virtually anywhere, carried, or mounted on vehicles — all while remaining connected to other devices. In practice, IoT systems typically consist of various devices with different roles and capabilities, necessitating multiple layers of communication.

Over the years, new communication technologies have been developed for vari-

ous kinds of uses, each technology with its own strengths and weaknesses. In the context of IoT, a handful have become prominent and widely supported standards. Many IoT applications leverage multiple communication standards in a complimentary manner. For example, a network of sensor devices might use Bluetooth Mesh for close proximity connectivity, with some of the devices acting as gateway devices also capable of utilizing LTE for long-distance communication. Understanding the key properties of different communication technologies is vital in IoT application development. Aspects such as range, power consumption, data rate, cost and security need to be considered based on application requirements.

Bluetooth was originally published in 2001 and has since become one of the most popular wireless communication standards. The standard has been actively developed to this date, and it has received a number of major updates to suit the needs of ever-growing user markets. A complete overhaul to the standard was introduced in Bluetooth 4.0 with the release of Bluetooth LE, a technology optimized for low-power, and low-data rate applications. Combined with its low cost and extensive hardware support, this makes Bluetooth LE an ideal choice for IoT devices. The most recent revisions of Bluetooth LE support data rates of up to 2 Mbps and ranges of up to 400 m. [1]

Zigbee, Z-wave and 6LoWPAN (and its derivative Thread) are popular low-power, low-data rate communication technologies. As opposed to traditional BLE, these technologies are designed to support larger networks with mesh networking capabilities. Their maximum data rates are however considerably lower than that of BLE, with Zigbee and 6LoWPAN supporting only up to 250 kbps at 2.4 GHz band. They all offer direct radio ranges of up to around 100 m, but network coverages can be extended far beyond that thanks to their networking features. [1]

To provide another alternative to these technologies, the Bluetooth Special Interest Group (SIG) released the Bluetooth Mesh standard in 2017. It is based on

Bluetooth 4.2, utilizing the Bluetooth LE physical and link layer specifications, but also offers extended compatibility with standard Bluetooth devices. Bluetooth Mesh is also designed to provide support for energy-constrained devices, although not all devices in a network can effectively utilize the low-power features. [5]

Bluetooth Mesh is not the only mesh networking protocol built on top of Bluetooth. Over the years, especially before the release of Bluetooth Mesh, multiple solutions have been developed for mostly proprietary and academic uses. But without proper standardization, their usage has not become widespread. Still, many offer interesting features that are not available in the official standard, such as the 6BLEMesh protocol, which uses an IPv6-based networking scheme for Bluetooth devices in a similar approach as 6LoWPAN. [6]

When higher data rates are required, Wi-Fi stands out with its latest generations supporting gigabit-level data rates. Wi-Fi is based on the IEEE 802.11 protocol family which had its first standard officially released already in 1999. Similarly to previously mentioned technologies, Wi-Fi also has a very limited direct communication range, but larger network installations can be achieved by mesh networking. However, Wi-Fi suffers from significantly higher power consumption than the aforementioned alternatives. To mitigate this issue, the less power demanding 802.11ah (Wi-Fi HaLow) amendment was released in 2017, although even it is yet unable to reach the low energy consumption and resource demands that, for example, BLE and Zigbee can offer. [1]

It goes without mention that countless number of communication technologies are used today. Technologies such as LTE, LoRa, Wi-SUN, NB-IoT, etc. have become popular for longer range wireless communication. Meanwhile wired communication protocols, such as Ethernet and PLC, too have their uses in different kinds of IoT applications. But as these technologies have essentially different properties and uses in comparison to Bluetooth Mesh, they are not covered here in more detail.

2.3 Mesh networking

Network formation is a crucial design aspect of all IoT systems as it establishes the properties and capabilities of the network. Typically, networks are built following either a centralized or distributed formation strategy. In centralized networks, communication relies on unique devices, which coordinate tasks and manage information. Centralized networks can be organized in various ways, such as hierarchically to form a kind of layered network. The advantages of centralized schemes come from their simple design, which allows efficient and highly predictable control over the entire network. Centralization, however, limits network scalability and can be inoptimal in terms of energy consumption. Distributed networks tackle these drawbacks by eliminating reliance on a central coordinator, allowing devices (a.k.a. nodes) to communicate and make decisions independently. [7]

Mesh networks operate on the principles of distributed communication. Mesh topology is based on the concept of having any node in the network able to relay messages forward without the need for any central coordinator devices. This gives mesh networks the capability to self-organize and self-configure, which also makes them inherently easy to scale [8]. Mesh topology is also great at enhancing network robustness, as messages can take multiple paths to reach their destination, improving fault tolerance and resilience against node failures. The lack of centralization further enhances network robustness, as a single network device won't bring down the whole network upon failure. However, the way how mesh networks function, also means that messages may often need to take multiple "hops" to reach their destination, which naturally increases the communication delay by some amount. Also, even though massive scalability is supported by the core mechanics of mesh topology, maintaining throughput becomes increasingly challenging as the network grows [9].

There are different schemes to handle the message delivery process in mesh networks. In the simplest method commonly known as flooding, messages are broad-

casted to all neighbouring nodes, which in turn broadcast to their neighbours, and so on. This works well if all nodes are indeed meant to receive the message, but otherwise, this method can be somewhat inefficient in terms of time and energy utilization on a broader scale. Still, at the same time, redundant transmissions can increase overall network robustness as more message delivery paths can be used. More sophisticated methods work by forwarding messages through network paths defined with some routing algorithm. These, however, are typically more resource-demanding and harder to implement.

Most mesh networking protocols use a routing approach to facilitate communication. Bluetooth Mesh has instead been designed to use a flooding method, which is referred to as *managed flooding* by the protocol. In this approach, all nodes implement a message cache to prevent redundant retransmissions. Additionally, the number of hops that messages can take is limited by a configurable time-to-live (TTL) value. But even with these built-in features, the flooding method can be problematic in some scenarios. Especially dense network installations with a high number of nodes are prone to so-called broadcast storms, which can be caused when multiple devices try to send messages to the network at the same time [10]. This quickly leads to communication channels becoming overly congested, degrading overall network performance.

3 Bluetooth Mesh

Having established the role and relevance of Bluetooth Mesh in the broader context of IoT networking, this chapter takes a closer look at the core concepts that define how the technology operates. Key topics include network structure, message handling, addressing schemes, and the mechanisms that support Bluetooth Mesh communication.

3.1 Bluetooth Mesh concepts

Nodes and elements

Devices that are part of a mesh network are referred to as nodes. All nodes have the ability to communicate in a mesh network by both transmitting and receiving messages. If a device has several functional components, they can be represented as separate elements. For example, a ceiling light consisting of three lamps could have one element representing each lamp. Each element in the network has a unique unicast address, which allows for very precise control over individual elements across all nodes. Unicast addresses in Bluetooth Mesh networks consist of 15 bits, which means that a total of 32767[1] unicast addresses can be assigned per network. This naturally sets the limit of elements in a network to the same amount. Additionally, the maximum number of elements within a single node is 255.

[1] Address 0x0000 is used to represent unassigned address and can not be assigned to any element.

Messages

The features that an element supports through the mesh network are defined by the element's models, which in turn control the device state and define what kinds of messages the element can send and receive. These are referred to as access messages, and they can be divided into three categories based on their purpose:

- GET messages are used to request the state of the recipient.
- SET messages are used to change the state of the recipient. They can be either acknowledged or unacknowledged type depending on whether the message sender wants an action confirmation from the recipient or not.
- STATUS messages hold a state value, and they are usually sent as a response to a GET message or acknowledged SET message.

Models

Models are generally classified as either server models or client models. Server models are responsible for managing and controlling specific data or resources. They can handle requests from client models and perform actions based on those requests. Server models may also publish STATUS messages without separate requests, usually either periodically or on state change. Client models on the contrary can request data from server models and often also control and configure them with SET messages. Client models can also be set to just receive STATUS messages sent from server models.

The Bluetooth Mesh Model specification [11] defines a number of models for various use cases. These predefined models are designed to cover the most typical application needs in a standardized way, maximizing interoperability between devices from different manufacturers and accelerating application development. Some examples of these predefined models are the Generic OnOff models, which are used

for binary state control (eg. a switch or a lamp), and the Sensor models, which are used to support features used in various sensor systems (eg. a thermometer). Device manufacturers also have the option to define their own models for more specific use cases if the predefined models are seen as insufficent. The Bluetooth Mesh Protocol specification [12] also defines some special models, which are known as foundation models. These models have specific roles in configuring and managing mesh networks.

Addressing

In addition to unicast addresses, two types of multicast addresses are specified. These are group addresses and virtual addresses. Group addresses and virtual addresses are similar to one another in that both are used to refer to a certain set of models across any combination of nodes and elements. This is achieved by each node having its own subscription list, which contains information about all the addresses that are mapped to local models. A model can subscribe to multiple addresses, so for example a Generic OnOff server model controlling the state of a lamp could be configured to handle messages pointed to group addresses "all-lamps-in-building" and "all-lamps-in-floor-3". Most group addresses are defined dynamically during network installation, with the exception of a few fixed group addresses (eg. "all-nodes"). Group addresses are 14 bits long, meaning that up to 16128 dynamic group addresses can be assigned per network (256 addresses are reserved for fixed group addresses).

Virtual addresses differ from group addresses in that they are formed from 128-bit Label UUIDs. This has the effect of significantly enlarging the available address space. Another advantage of virtual addresses is that manufacturers are able to assign specific Label UUIDs to devices in production, reducing the network configuration workload of the user. To speed up the address-matching process, 14-bit

hashes are used in messaging with virtual addresses.

3.2 Node features

Each node in a Bluetooth Mesh network can be assigned with a set of special *node features* which determine the node's additional networking capabilities. The Bluetooth Mesh Protocol specifies the following four node features: Relay, Low-power, Friend and Proxy. It is possible that some nodes may have multiple features enabled while some nodes have none.

Relay feature

Nodes that have enabled the relay features are capable of relaying network messages from adjacent nodes to other adjacent nodes. This makes it possible for nodes to communicate with other nodes, even if they are not within direct range of radio communication. Since message relaying is a fundamental operation principle of mesh networks, it is highly usual that some (if not all) nodes in a Bluetooth Mesh network have the relay feature enabled.

Low-power feature

In many cases, it is necessary to employ devices that don't have access to an external power source (eg. battery-powered sensory devices). With these devices, keeping the energy consumption at a minimum level is highly critical in order to maximize their lifetime. This is usually achieved by keeping the device in sleep mode and shutting down its peripherals, waking up only when necessary. For solely sending messages, this is not an issue, since the transmitter can be turned on and off as normal. But to be able to receive messages as well, the device would have to keep the receiver on at all times. This is needed when for example, some sensor parameters might have to be configured later without physical access. Additionally, the security protocols

built into Bluetooth Mesh require that all nodes in the network can receive messages. The Low-power feature is designed to solve this problem by introducing a mechanism where the Low-power nodes (LPNs) do not keep their receivers active continuously. Instead, these nodes rely on dedicated Friend nodes to store incoming messages temporarily. The selection of these Friend nodes follows a set of rules described in the Bluetooth Mesh Protocol specification [12].

Friend feature

As mentioned, Friend nodes are required to support Low-power nodes in the same network. In addition to other functions, Friend nodes also receive and store messages on behalf of any Low-power nodes that are maintained by the Friend node. Periodically, when an LPN wakes up, it requests its assigned Friend node to send it all the messages that were received during its sleep period. This naturally adds some latency to the communication, depending on the LPN's wakeup interval.

Proxy feature

By default, all communication in a Bluetooth Mesh network uses a so-called *Advertising bearer*, which is discussed in more detail in section 3.4. However, not all Bluetooth LE devices support this. Thus, for enhanced compatibility, Proxy feature can be used. Devices within the radio range of a Proxy node can, aside from communicating over the advertising bearer, establish a GATT (Generic Attribute Profile) connection with the Proxy node. This allows devices that do not natively support Bluetooth Mesh networking stack, but are compatible with Bluetooth LE, to interact with the mesh network indirectly. A good example of such a device would be a smartphone, which can allow users to access and control other devices in the mesh network.

3.3 Networking stack

The networking functionality of Bluetooth Mesh is divided into multiple layers, which together form the mesh networking stack. Each layer has specific responsibilities related to message processing and interacts with adjacent layers to facilitate seamless data exchange across the mesh network. This layered architecture allows for modular design, making it easier to implement, maintain and upgrade the system. The Bluetooth Mesh Protocol defines the following seven layers (from top to bottom):

- Model layer
- Foundation model layer
- Access layer
- Upper transport layer
- Lower transport layer
- Network layer
- Bearer layer

The whole stack is built upon the LE Physical Transport defined in the Bluetooth Core Specification.

When sending messages, the message data is passed down the stack through the different layers, so that often some layer-specific header data is added alongside the message, resulting in more informative data structures known as PDUs (Protocol Data Unit). At the receiving end, this process is inverted: the PDUs are interpreted in their corresponding layers and the contained data is then extracted and passed up in the stack when necessary.

Model layer and foundation model layer

On top of the mesh networking stack lie the model layer and the foundation model layer, which are responsible for implementing all the models and foundation models supported by the node as per their respective specifications. Consequently, these layers also define the states and messages used by the models.

Access layer

The access layer acts as an intermediary between the upper transport layer and the layers higher in the stack. It defines the message data format and controls how the data should be encrypted and decrypted in the upper transport layer. It is also responsible for checking the validity of received data before passing it up to higher layers.

Upper transport layer

The upper transport layer is responsible for encryption, decryption and authentication of application data sent to or received from the access layer. The upper transport layer also handles sending and receiving transport control messages between nodes. Control messages are used in certain network management tasks, such as those related to friendship and directed forwarding. The upper transport layer introduces its own PDU, which adds a TransMIC (Transport Message Integrity Check) value consisting of 32 or 64 bits to all access messages. The TransMIC field is used to ensure that the message contents have not been altered.

Lower transport layer

The lower transport layer takes care of a process called segmentation and reassembly. The maximum allowed message size is 380 octets for access messages and 256 octets for transport control messages. However, transmitting packets this large is

not possible in Bluetooth Mesh networks. Instead, large messages are split into segments, which are transmitted separately. The receiving end then reconstructs the original messages from their segments.

Lower transport PDUs have a size limit of 16 octets, meaning that 15-octet long or smaller access messages can be sent without segmentation (as one octet is reserved for header data). When segmentation is used, additional header information needs to be included in a lower transport PDU, which reduces the size of the message payload. The lower transport layer also introduces a number of ways to increase communication reliability with segmented messages, such as retransmissions and segment acknowledge messages. These are further discussed in 4.3.3. Segmentation can optionally be enabled for even those access messages, that could also be sent without segmentation, for these additional reliability-increasing benefits.

Network layer

The network layer is responsible for generating and handling network PDUs, which are used in communication on the bearer layer. These PDUs include source and destination addresses, security and relaying related data fields and the message form. The network layer also manages network security by encrypting, decrypting and authenticating all incoming and outgoing messages using a network key. In addition, message relaying is done within the network layer.

Bearer layer

The bearer layer is the lowest layer in the mesh networking stack, defining how the underlying communication systems are used. Two bearers are available for Bluetooth Mesh networks: the advertising bearer and the GATT bearer. The advertising bearer is the main bearer designed to be used with Bluetooth Mesh, taking advantage of Bluetooth LE advertising and scanning features for network communication. The

GATT bearer is designed to offer support to those devices, that do not have the ability to use advertising bearer, allowing any Bluetooth LE device to participate in mesh network communication using so-called Proxy protocol. A node with the Proxy feature enabled is required to forward messages between these two different bearers.

3.4 Advertising bearer

Using the Advertising bearer is the default means of communication in Bluetooth Mesh networks. In the context of Bluetooth, advertising refers to publishing data over the air, allowing any other devices to pick up that data by performing scanning operations. In order for data transfers to succeed, the advertising device and the scanning device have to operate on the same channel at the same time. The Bluetooth core standard has various built-in mechanisms which should help ensure that this type of communication can occur reliably and efficiently.

The 2.4 GHz band used by Bluetooth Low Energy is divided into 40 channels, out of which three primary channels (37, 38 and 39) are used in advertising. When data is to be transmitted over the advertising bearer, an advertising event occurs. During this event, advertisement PDUs are sent once on each of the primary advertising channels with intervals of 10 ms or less. For Bluetooth Mesh, the order of the channels as well as the intervals are randomized, as recommended in the bearer layer specification.

It is possible to process multiple advertising events simultaneously as long as the advertising data for each event is stored together in *advertisement sets*. This can be utilized in Bluetooth Mesh especially when relaying data in a congested part of a network.

In Bluetooth 5.0, a feature known as *extended advertising* was introduced. With extended advertising, it is possible to utilize the remaining 37 channels for data

transmission, and the primary channels are only used for transmitting the necessary information about scanning the data. This helps in freeing bandwidth on the primary channels and ultimately reduces congestion. However, this feature is not supported on the current version of Bluetooth Mesh, as it is based on the Bluetooth 4.2 standard.

3.5 Security

Security in Bluetooth Mesh is a fundamental aspect of the protocol. Due to this, all messages in Bluetooth Mesh networks are encrypted and authenticated. The security design draws on a principle called "separation of concerns", meaning that network, application and device level securities are separated. To achieve this, three different security keys are used: network key (NetKey), application key (AppKey) and device key (DevKey).

NetKeys are shared with all devices in the same network. They are used to decrypt and authenticate network PDUs, and the encryption keys are directly derived from them. This approach allows relay nodes to forward messages to nearby devices without being able to interpret the message contents thenselves. In turn, devices that do not possess the corresponding NetKey, cannot participate in any part of communication in that network. A network can also consist of multiple subnets, each of which uses its own NetKey. Some devices can possess multiple NetKeys, granting them access to multiple subnets. Additionally, NetKeys are used to derive privacy keys, with which network PDU headers are obfuscated. This makes it difficult for eavesdroppers to analyze network traffic and track devices.

It is entirely possible that distinguish devices connected to the same network carry out completely different tasks (eg. lighting and air conditioning). In many cases, messages are strictly linked to certain applications and there is no need for other applications to access the message contents. For this reason, AppKeys are used

to secure communication at the upper transport layer. These AppKeys are bound to NetKeys, so for a device to process a message sent from another device, that device must hold both the corresponding AppKey and Netkey. As with NetKeys, multiple AppKeys can be possessed by a single device.

The third key type, DevKey, is a special application key unique to each device and it is only used in configuring nodes.

Secure distribution of keys is an important function carried out by the Provisioner. The Provisioner is a special node, which takes care of adding new nodes to a network. This process is known as *provisioning*. The device to be added initiates the process by broadcasting an Unprovisioned Device beacon. The Provisioner then sends an invitation to this device, which is eventually followed by an exchange of public keys and authentication. This can be achieved in different ways, such as entering a code or using an X.509 certificate, depending on the devices' capabilities. Then, after the communication has been secured, the Provisioner distributes relevant network data such as the NetKey and assigns an unicast address to the new node, finishing the provisioning process.

Securely removing a node from a network is also important to prevent so-called *trashcan attacks*. Trashcan attack refers to an attack, in which the keys of a disused device are used to infiltrate into its previous network. In Bluetooth Mesh, this is handled by a Key Refresh Procedure, which replaces all network and application keys of all nodes that remain in the network, making the old keys obsolete.

Bluetooth Mesh also employs security features to combat *replay attacks*, in which an attacker captures messages that it will later retransmit in an attempt to bluff the recipient. Due to this, all network PDUs are associated with a sequence number, which is incremented on each transmission from the same element. Receiving a message with a sequence number that is less than or equal to the last received sequence number from the same source indicates a potential replay attack, and the

message is discarded. Each node maintains a Replay Protection List (RPL), which stores the most recent sequence numbers for all known source addresses. When a new message is received, the node checks the sequence number against the RPL to determine if the message is valid.

The sequence number is a 24-bit value, meaning that after 16 777 216 messages it can no longer be incremented. This issue is solved by using a secondary security number known as the IV Index. The IV Index value is shared between all nodes in the same network, and used during encryption. Once the sequence number reaches the limit value, an IV Update procedure is triggered, during which each device in the network must increment its IV Index and reset all sequence numbers to zero. The IV Index is a 32-bit number which should never reach its limit in practice considering that IV Update procedures don't occur very often.

3.6 Network traffic control

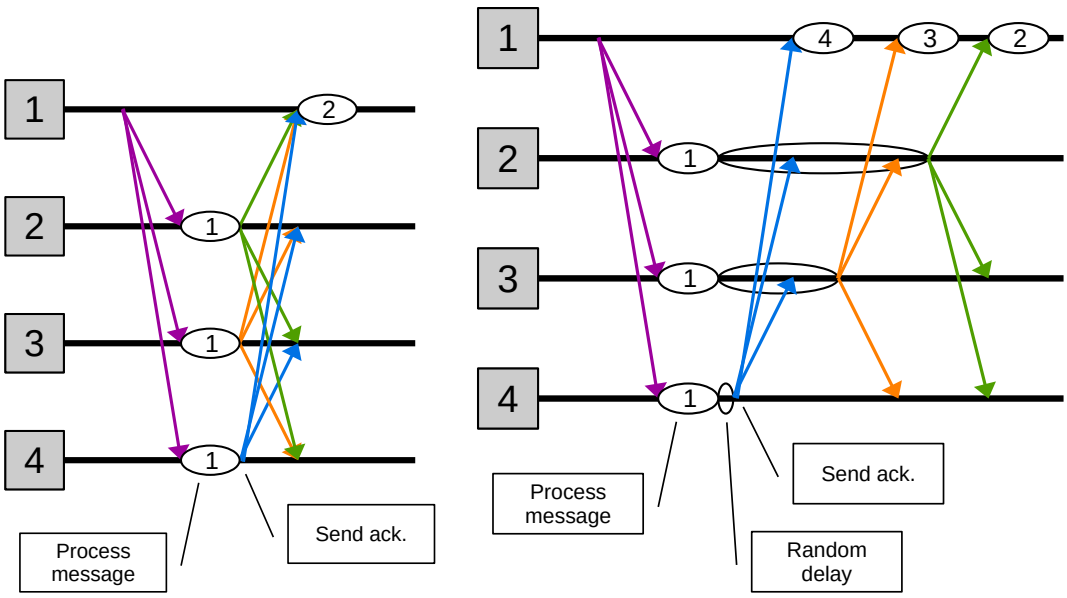
As already discussed in the previous chapter, the managed flooding technique used in Bluetooth Mesh is relatively simple to implement and generally provides a good level of reliability, thanks to multipath delivery. However, ineffective use of the communication channels can be problematic with regard to the consumption of energy and bandwidth. When multiple nearby devices attempt to advertise in the same channel simultaneously, a packet collision occurs risking successful message delivery. Due to message flooding, this can easily lead to broadcast storms, causing major instability in the network. To mitigate congestion, the managed flooding method of Bluetooth Mesh employs mechanisms such as message TTL values and message caches. When properly configured, these measures can help reduce network traffic.

Of course, configuring networks for optimal performance is not a trivial task. For example, gains in traffic reduction from minimizing message TTL values might actu-

ally end up outweighed by possible losses in path diversity, which would ultimately result in lowered communication reliability. This can also be seen in simulations run by Esposito et al. [13], which show that using a minimal TTL to reach a target node also results in notably worse overall packet delivery rate (PDR), while even a one-step increment can help bring the PDR from below 80% to easily above 95%. In this case, close to 100% could be obtained with even higher TTL values, although quickly bringing any network traffic reductions to a very marginal level. It's worth noting, however, that with alternate network topologies results could be drastically different due to variations in message paths.

Another way to mitigate the risk of broadcast storms is to implement random delays before sending messages, especially when multiple nodes are involved in the communication process. For example, sending an acknowledged SET message to a multicast address can potentially trigger hundreds or even thousands of simultaneous responses, which can easily result in temporary congestion of communication channels [14]. Figure 3.1 illustrates the effect of random delays in a multicast communication context. Introducing randomness reduces the likelihood of multiple devices attempting to transmit data simultaneously over the same communication channel. A higher number of relays, regular network nodes, and retransmissions significantly increase both the probability and severity of broadcast storms, whereas allowing wider ranges for random delays helps to reduce this risk [15]. The natural downside of introducing randomness is that it can also lead to increased latency, as some transmissions may be delayed longer than necessary. The access layer specification recommends a random delay of 20 to 50 ms for unicast message responses and 20 to 500 for multicast message responses [12].

It is important to note that packet collisions are not a challenge unique to Bluetooth Mesh networks. The 2.4 GHz frequency band is heavily utilized by numerous other wireless communication standards. The presence of other actively transmit-



(a) Without random delay (b) With random delay

Figure 3.1: Simplified example of network behaviour when a multicast message (from node 1) is acknowledged by three non-relay nodes (2, 3 and 4), both with and without additional random delay preceding the response message.

ting wireless devices can therefore cause interference with Bluetooth Mesh devices. Especially devices that use the same channels, such as other Bluetooth LE advertisers or devices from other Bluetooth Mesh networks, can be difficult to deal with, as external interference can't often be controlled. This can lead to reduced reliability, throughput or energy efficiency, and should thus be addressed in network design and deployment [16].

Directed forwarding

In version 1.1, the Bluetooth Mesh Protocol introduced *directed forwarding* as an alternative relaying scheme to managed flooding. Designed to offer increased network efficiency by avoiding excessive transmissions, directed forwarding is a prominent feature, particularly for larger network deployments. In comparison to managed flooding, which bases its operation on blindly spreading all messages to neighbouring nodes hoping that the messages would eventually reach their destination, directed

forwarding takes advantage of previously discovered paths. Only relay nodes that are part of such paths are allowed to relay messages further. These paths can be either created manually using a configuration application, or they can be formed dynamically as a result of automated path establishment procedures. [17]

4 Experimentation

The goal of this part is to provide a comprehensive understanding of how exactly Bluetooth Mesh devices and network performance can be tuned with software parameters to best fit varying application requirements. This experimentation mostly focuses on evaluating reliability and latency of node-to-node communication, as well as device memory usage. Energy consumption will be also considered with some applicable parameters.

4.1 Hardware description

The experiments are done using nRF52840 development kits manufactured by Nordic Semiconductor [18]. The nRF52840 SoC contains 1024 kB of internal flash memory and 256 kB of RAM and features a 32-bit Arm Cortex-M4 processor with a floating point unit, capable of running at 64 MHz. It also comes packed with numerous security features and peripherals, most notably a 2.4 GHz radio transceiver providing support for multiple wireless communication protocols, such as Bluetooth 5.4, Thread, and Zigbee. This makes it a fairly popular choice among researchers[1] and industry, especially for wireless connectivity applications.

The 2.4 GHz radio transceiver in nRF52840 is compatible with 1 Mbps, 2 Mbps, 500 kbps, and 125 kbps Bluetooth 5 data rates. Radio transmit power is adjustable

[1]In the articles studied for this thesis, nearly all with an experimental setup used nRF52 or nRF51 -series SoCs.

to up to +8 dBm. According to the official radio electrical specification, the receiver draws around 4.6 mA in 1 Mbps mode and 5.2 mA in 2 Mbps mode, while the transmitter typically draws around 4.8 mA (with 0 dBm TX power). The radio peripheral is also capable of measuring the strength of the received signal in 1 dB resolution.

4.2 Software description

The software is built with nRF Connect SDK (v2.7.0), a software development kit provided by Nordic Semiconductor and specifically designed for their nRF series of microcontrollers. It offers a comprehensive set of tools, libraries, and examples to facilitate the development of applications that utilize various wireless communication protocols, including Bluetooth Low Energy and Mesh. The SDK is built on top of the Zephyr real-time operating system (RTOS), which is an open-source real-time operating system designed for resource-constrained devices, particularly in the context of IoT. Zephyr provides a flexible and modular framework that supports a wide range of hardware architectures.

Zephyr uses a configuration system called KConfig, which is also used in eg. the Linux kernel. With KConfig options it is possible to choose which modules and features are included in the software and configure their functionalities. Some of these options merely set default values to various parameters also modifiable in runtime, and this is also the case with most Bluetooth Mesh configuration parameters that are discussed in the next section. Then again, there are also many options that affect the compilation process and can not be changed later without a rebuild. But whatever the case, these options often require careful assessment and understanding of the application context. Several KConfig options related to Bluetooth Mesh were analyzed for the memory evaluation part of this study.

The performance experiments are carried out with multiple devices, each run-

ning the same application software. Network creation, provisioning, and binding of application keys are handled using the nRF Mesh mobile application [19] also developed by Nordic Semiconductor. Rest of the configurations are dynamically managed by one of the nodes which acts as the master (later labeled as node 1). In addition to controlling the configuration of all nodes in the network, the master node is responsible of measuring the network performance with different parameter values. The results of each phase are saved to device memory, from which they are finally extracted in CSV format for further analysis.

4.3 Examined parameters

This experimentation focuses on examining only parameters that affect the general performance of Bluetooth Mesh networks and devices. As such, any parameters that are related to specific models or special features (eg. provisioning and proxies) are left out. Notably, the left-out features also include directed forwarding, which is still unsupported on the nRF Connect SDK platform at the time of writing this thesis. Additionally, some other configuration aspects (such as those related to LPNs and friendship) have already been well assessed in previous works, and thus they are excluded from this experimentation. The results of these works are still discussed later.

4.3.1 Publication parameters

The Bluetooth Mesh Protocol [12] defines a Model Publication state, which determines how and when a model sends messages to other nodes in the mesh network. Each model has its own Model Publication state, which can be configured through a common Configuration Server model of a node. The Model Publication state holds values for publication address, some security options, and the parameters listed in

PARAMETER	DEFAULT	RANGE
Publish Retransmit Count (PRC)	0	0 - 7
Publish Retransmit Interval (PRI)	50 ms	50 - 1600 ms
Time-to-live (TTL)	7	0 - 127
Publish Period	Disabled	100 ms - 10.5 h

Table 4.1: Publication parameters

table 4.3.1.

The *Publish Retransmit Count* and *Publish Retransmit Interval*[2] parameters control how many times and how often messages published by a model are retransmitted. The *Publish Period* parameter on the other hand controls the rate at which new messages are published, but this affects only status messages. The *TTL* (time-to-live) parameter defines how many hops published messages can take before being discarded from the network. Models can also be set to use a default TTL value of the node (also configured through the Configuration Server model) when there is no need to configure the TTL for every model separately. In these experiments, the TTL parameter is omitted as larger test setups would be required to properly analyze its practical effects.

4.3.2 Network layer parameters

Retransmissions also occur at the network layer, both in the source node and when relaying PDUs sent from other nodes. This behaviour is also controlled by specific parameters, which can be configured through a Configuration Server model. These parameters are presented in table 4.3.2.

The *Network Transmit Count* and *Network Transmit Interval* as well as their relaying variants describe the number and rate of retransmissions of network PDUs. The relay parameters only affect those nodes that can act as relays. The Bluetooth Mesh Protocol specification states that for each retransmission a random delay from

[2]Interval parameters in Bluetooth Mesh are often defined in interval steps format instead of milliseconds, but in this text, milliseconds format is used for clarity.

PARAMETER	DEFAULT	RANGE
Network Transmit Count (NTC)	3	1 - 8
Network Transmit Interval (NTI)	20 ms	10 - 320 ms
Relay Retransmit Count (RRC)	2	0 - 7
Relay Retransmit Interval (RRI)	20 ms	10 - 320 ms

Table 4.2: Network layer parameters

0 to 10 ms should be added along with the delay defined in the *Network Transmit Interval*. Additionally, the bearer may apply its own restrictions on minimum intervals between successive transmissions: eg. the advertisement bearer has a minimum interval of 20 ms.

4.3.3 Segmentation related parameters

As discussed in 3.3, message segmentation is a key feature in Bluetooth Mesh networking. Its precise functionality is controlled by a quite large number of different parameters. These parameters can be divided into two distinct groups:

- SAR transmitter parameters (listed in table 4.3.3)
- SAR receiver parameters (listed in table 4.3.3)

From Bluetooth Mesh version 1.1 onwards, the transmitter and receiver parameters can be configured through a SAR server foundation model, if it is supported by a node. These parameters and their intended behaviour are defined in the Bluetooth Mesh Protocol specification.

It should be noted that performing large number of SAR processes at the same time would require large memory buffers, and thus some implementation specific limitations may need to be put in place. This is also the case with Zephyr implementation, which offers some configuration options that affect SAR behaviour and its associated memory cost. These are studied in a further part of this section.

PARAMETER	DEFAULT	RANGE
SAR Segment Interval (SSI)	60 ms	10 - 160 ms
SAR Unicast Retransmissions Count (SRC)	2	0 - 15
SAR Unicast Retransmissions Without Progress Count (SRWPC)	2	0 - 15
SAR Unicast Retransmissions Interval (SRI)	200 ms	25 - 400 ms
SAR Unicast Retransmissions Interval Increment	50 ms	25 - 400 ms
SAR Multicast Retransmissions Count (SRC)	2	0 - 15
SAR Multicast Retransmissions Interval (SRI)	250 ms	25 - 400 ms

Table 4.3: SAR transmitter parameters

PARAMETER	DEFAULT	RANGE
SAR Segments Threshold (SST)	3	0 - 31
SAR Acknowledgment Delay Increment (SADI)	2.5	1.5 - 8.5
SAR Acknowledgment Retransmissions Count (SARC)	0	0 - 3
SAR Discard Timeout	10 s	5 - 80 s
SAR Receiver Segment Interval (SRSI)	60 ms	10 - 160 ms

Table 4.4: SAR receiver parameters

For the transmitter side, the *SAR Segment Interval* parameter defines the interval between successive segment transmissions. The *SAR Multicast Retransmissions Count* and *Interval* parameters directly define how many times and how frequently the entire message is retransmitted by the lower transport layer when the destination is a multicast address. This behaviour is also illustrated in figure 4.1 with 60 ms segment interval, one retransmission and 250 ms retransmission interval.

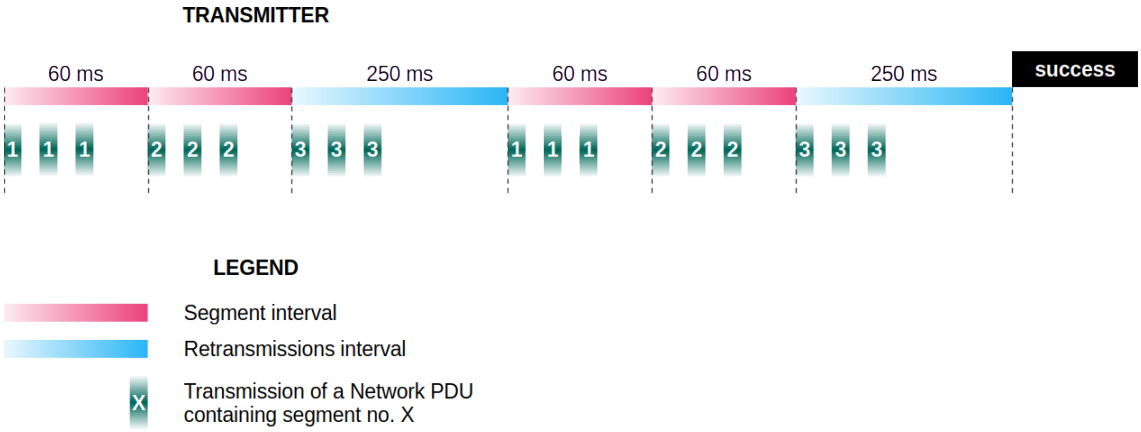


Figure 4.1: Example of segmentation behaviour whenthe destination is a multicast address and the message is divided into three segments. In this case, the receiving nodes do not send Segment Acknowledgment messages.

When the destination address is a unicast address, performance improvements can be achieved with Segment Acknowledgement messages. Segment Acknowledgement messages are sent by the receiver to indicate to the transmitter which segments have been successfully received. This also has to be accounted for on the transmitter side. When sending messages to unicast addresses, the number, interval and contents of retransmissions are dynamically adjusted based on the received Segment Acknowledgement messages. This is demonstrated in figure 4.2. The maximum interval between retransmissions is also affected by message TTL with a weight defined in parameter *SAR Unicast Retransmissions Interval Increment* (not examined separately in the experiments). Additionally, the parameter *SAR Unicast Retransmissions Without Progress Count* is introduced, which allows the transmission to end quicker if no acknowledges are received. Figure 4.3 demonstrates this with the aforementioned parameter set to zero.

The majority of the SAR receiver parameters are also related to sending Segment Acknowledgment messages. The parameters *SAR Receiver Segment Interval*

and *SAR Acknowledgment Delay Increment* control the delay between the last received segment and the transmission of a new Segment Acknowledgment message. The Segment Acknowledgment messages may also be retransmitted if the message length exceeds the value of the parameter *SAR Segments Threshold*. In this case, the retransmission count and interval are defined by the parameters *SAR Acknowledgment Retransmissions Count* and *SAR Receiver Segment Interval*. The parameter *SAR Discard Timeout* defines the time window for receiving new segments of a segmented message. If no new segments are received within this period, the message reception is considered as failed and the already stored part of the message is discarded.

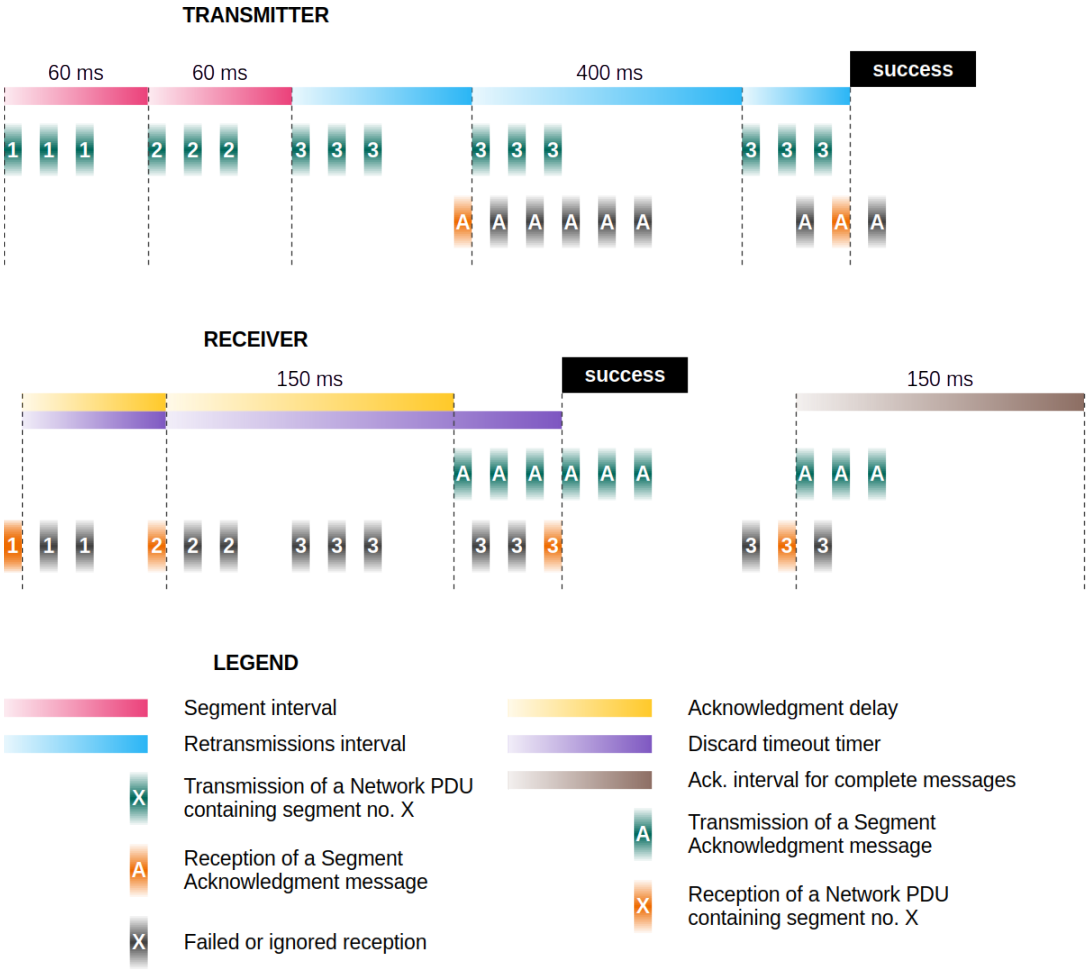


Figure 4.2: Example of segmentation and reassembly when the destination is a unicast address. The receiver successfully receives two out of three transmitted segments, and the transmitter receives the first Segment Acknowledgment message indicating this. Retransmissions occur only with the failed segment. Once the final segment is received, the receiver replies with a Segment Acknowledgment message, first instantly and then again later when receiving that same segment (but only once within a certain time period).

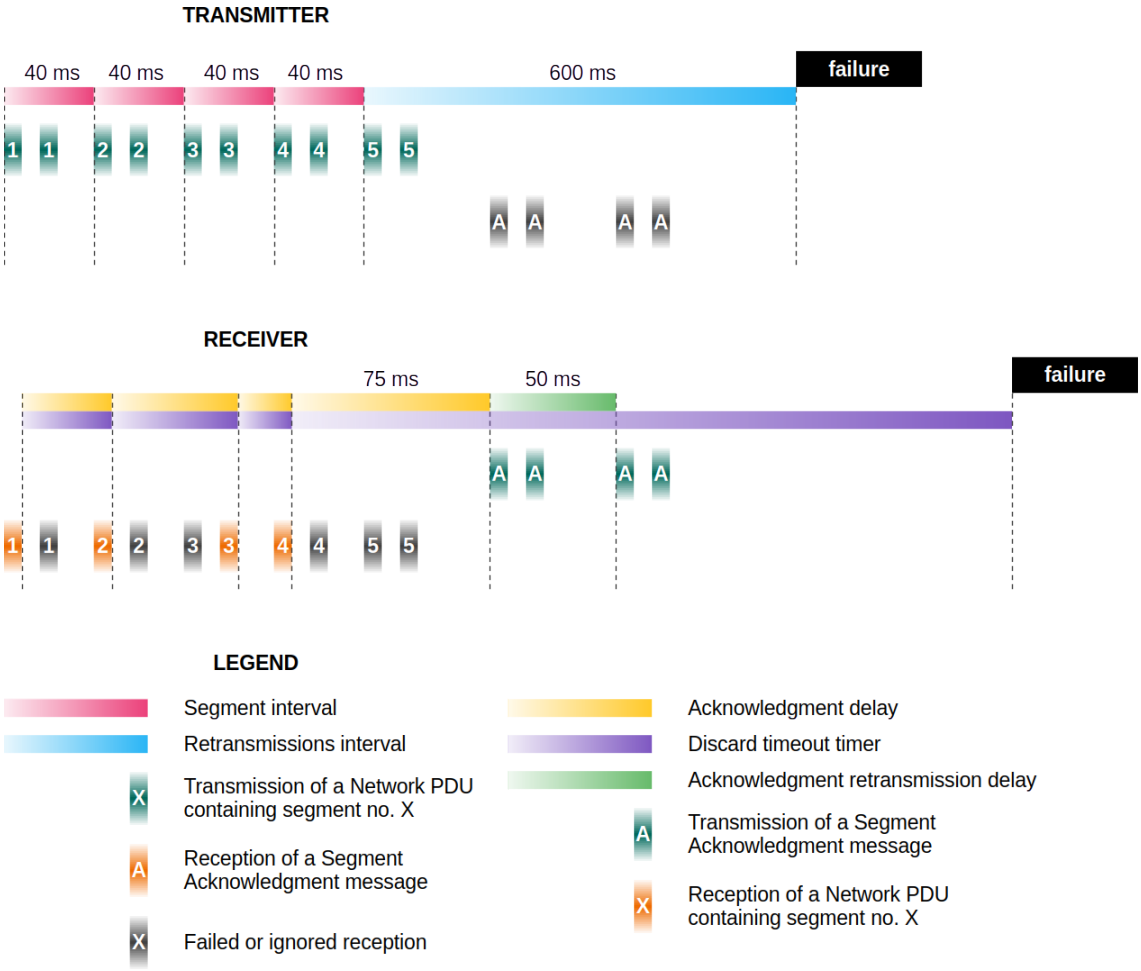


Figure 4.3: Another example of segmentation and reassembly when destination is a unicast address and the message is divided into five segments. Here only the first four segments are successfully received, causing the receiver to send a Segment Acknowledgment message. This is retransmitted since the total number of segments is above the threshold. The transmitter, however, fails to receive any of the Segment Acknowledgment messages. It also has the *SAR Unicast Retransmissions Without Progress Count* parameter set to 0, causing the transmission to be considered failed. Eventually, the receiver discards the message.

4.3.4 Memory control options

The amount of available memory can be a significant constraint, especially for low-cost devices commonly used in IoT applications. For this reason, efficient memory management is highly critical in many practical implementations. This of course also applies to Bluetooth Mesh. Zephyr, for example, has numerous KConfig options which allow control over the memory requirements of Bluetooth Mesh. Some of these also have an impact on communication performance. Table 4.3.4 lists a few such options.

With network layer retransmissions and possibly multiple communication paths, it is likely that a node receives the same network PDU more than once. In order to avoid acting unnecessarily upon receiving an already-seen PDU, a network message cache is used. The *network message cache size* option defines how many distinct network PDUs can be stored in the network message cache at the same time. Along with the memory impact, this option can also affect PDU processing time, as noted in the Zephyr documentation [20].

Advertising buffer count, as the name indicates, is an option defining the number of available advertisement buffers. Advertisement buffers are used by the advertising bearer to temporarily store advertising data during and between transmissions. In other words, this option effectively sets a limit to the number of simultaneous transmissions over the advertising bearer.

As mentioned earlier, segmentation and reassembly features also require proper limits to prevent excessive memory usage. The *maximum segmented messages* options define how many segmented messages can be processed simultaneously, both incoming and outgoing. In Zephyr it's allowed to set these options to value zero to completely disable the SAR feature. The *maximum segments per message* options define the maximum size of segmented messages that the device is able to process. The *number of segment buffers* option sets the size of the memory pool, from which

PARAMETER	DEFAULT	RANGE
Advertising buffer count	6	1 - 256
Network message cache size	32	2 - 65535
Maximum segmented messages (in)	1	0 - 255
Maximum segmented messages (out)	1	0 - 255
Maximum segments per message (in)	3	1 - 32
Maximum segments per message (out)	3	1 - 32
Number of segment buffers	64	$\leq$ 16384

Table 4.5: Various memory control options

memory is allocated for processing segmented messages.

4.3.5 Physical layer configuration

In addition to parameters defining the functional details of Bluetooth Mesh, the networking performance can also be tuned with options related to the underlying BLE core stack. A few of these are presented in table 4.3.5. The adjustment options for these parameters might vary from device by device.

Perhaps the simplest way to drastically improve radio communication reliability and range is to increase the power of the transmitter (often known as TX power). This however comes with a cost in power consumption. In contrast, tuning down TX power can potentially offer significant energy savings, so looking for other ways to increase range and reliability might be preferred.

When BLE was originally introduced in Bluetooth revision 4.2, the specification for its physical layer implementations, or PHYs, only included one called LE 1M with a 1 Mbps data rate. In Bluetooth revision 5.0, additional PHYs were introduced: LE 2M and LE Coded with two different coding schemes (S=2 and S=8). The LE 2M PHY doubles the data rate of the LE 1M, whereas the LE Coded PHY uses error correcting codes, offering increased range and reliability at the cost of a lowered data rate. However, LE 2M PHY is only available for extended advertising packets. Also, as Bluetooth Mesh is based on Bluetooth revision 4.2, these additional PHYs

PARAMETER	DEFAULT	RANGE
TX power	0 dBm	-40 to +8 dBm
LE PHY	1M	1M / 2M / Coded (S=2) / Coded (S=8)

Table 4.6: Investigated physical layer settings. TX power value range showed here is the one supported by nRF52840

are not supported by the official protocol specification and may cause potential compatibility issues with some devices.

4.4 Evaluation methods

As mentioned earlier, this study focuses on analyzing communication reliability, latency and memory usage. Analysis of memory usage is the most straightforward since it can be done with built-in memory analysis tools based on compiled software images, and different parameter values affect memory usage linearly. Reliability and latency analyses, in contrast, require a practical experimental approach.

In this study, the metric used for latency is round-trip time (RTT). It describes the time that passes between sending a message to another element and receiving a response from that element. This eliminates the need to implement an accurate time synchronization method between multiple nodes.

Communication reliability in this experiment is measured with packet loss rate (PLR). PLR is a metric used to quantify the percentage of packets that are lost during transmission over the network. In the context of this study, "packets" refer to access messages rather than individual network PDUs, as the focus is on the application level communication reliability. For simplicity, communication reliability in this study is measured in the same round-trip manner as latency, meaning that a communication event is considered successful only when both the original message and its response are received successfully.

Additionally, the RSSI value of each received message is also taken into consid-

eration, as it allows for monitoring that uncontrollable external conditions do not deviate significantly during the experimentation.

4.5 Experimental setup

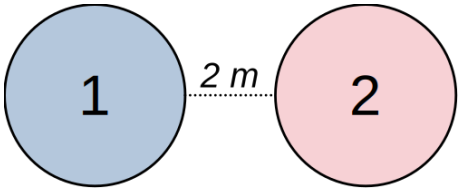
All experiments consist of at least two nodes, named node 1 and node 2. Measurements are carried out so that node 1 sends a total of 100 requests to node 2 with an interval of at least 2500 ms between messages. The interval is adjusted phase by phase to account for variable transmission durations with different parameter combinations. This is important for avoiding any uncontrollable message collision or buffer capacity issues (which are discussed later). Each request message contains an indexing number, and node 2 will send a response with a matching indexing number back to node 1, allowing accurate latency measurements. After all 100 requests are sent, the PLR and averages of RTT and RSSI are calculated and saved for further analysis. This process is repeated multiple times with different parameter values. The same parameter adjustments are always made on both nodes 1 and 2 throughout the experimentation.

All message responses are sent following the recommended delay settings discussed in section 3.6. This of course introduces some amount of randomness into the experiments. However, since 100 messages are sent in each phase, the results should average out any irregularities and reflect real-world network behaviour quite well.

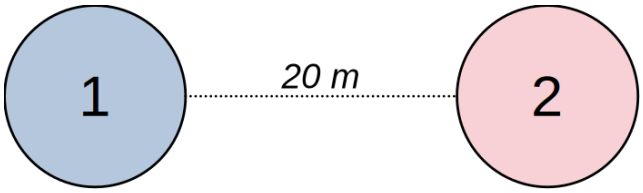
In order to examine the effects of SAR parameter adjustments with messages of different lengths, extra bytes are added to the messages. Thus all SAR-related experiments are performed with segment counts of 2, 6 and 18. Message intervals are also adjusted accordingly to ensure that consequent message transmissions won't overlap. Additionally, publishing to both unicast and multicast addresses is tested, although node B will still always use a unicast address for its response.

The performance of Bluetooth Mesh may vary greatly depending on external conditions, such as distance between nodes, relay count, network scale and network density [21]. To study how these conditions alter the impact of different parameters, several experiments with varying network setups are conducted. The following scenarios are considered:

Setup A: Two nodes only This network setup consists of only two nodes that are in direct radio range of each other. In order to determine the effect of signal quality vs. parameter values, experiments with this setup are conducted twice using different distances between the nodes: first with 2 m distance and then with 20 m distance. An illustration of this is provided in figure 4.4.



(a) 2 m distance between nodes



(b) 20 m distance between nodes

Figure 4.4: Test setup A with two distance variations.

Setup B: Two nodes with an additional relay In this network setup, a relay node is added between the nodes 1 and 2. This is illustrated in figure 4.5. The distance between nodes 1 and 2 is also increased to make communication more reliant on the relay node.

Setup C: Two nodes in a larger network This network setup demonstrates a possible real-world scenario, where the two nodes 1 and 2 are part of a large

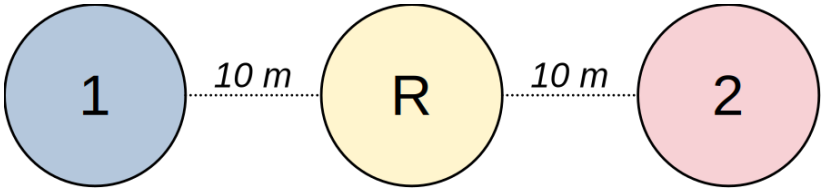
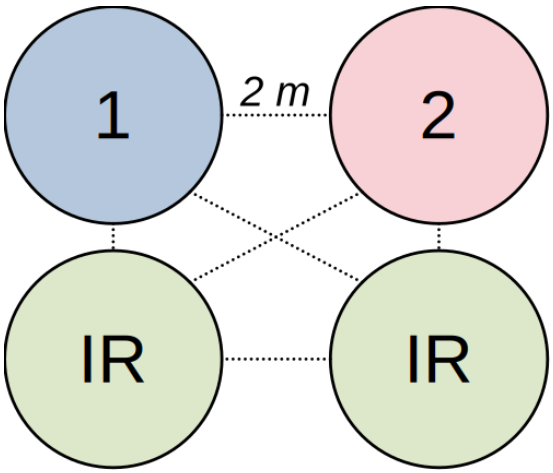


Figure 4.5: Test setup B with 10 m distance between adjacent nodes.

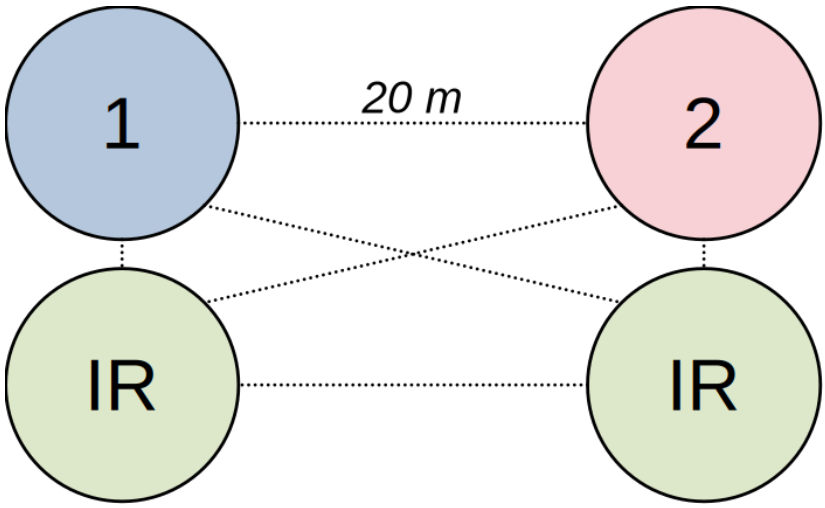
mesh network where also many other nodes are trying to communicate with each other. This is achieved by adding two interfering relay nodes between the nodes 1 and 2. Optimally, these nodes would be set to relay actual network messages from other nodes. But as this would require a very large number of devices and more configuration, these interfering nodes only represent relay nodes connected to a vast network of other nodes, transmitting new messages very frequently at intervals of 125 ms. As with any other relay nodes, they also forward messages between nodes 1 and 2. This setup also comes with two variations representing networks of different densities. These are illustrated in figure 4.6.

With this kind of relay emulation approach, a few issues should be addressed to properly understand the results. Firstly, separate buffers are used for actual relayed messages and those that only represent relayed messages. Since the fake relayed messages follow a publication cycle of 125 ms, adjusting NTC and NTI to result in longer transmissions than 125 ms would quickly cause buffer exhaustion (described in more detail in the following chapter). This in turn causes a selective bottleneck scenario, in which actual relayed messages are received and sent normally, but fake relayed messages can't be sent due to advertising buffer incapacity. Another limitation of this emulation approach is that only one advertisement set is used for all source-originated messages, whereas multiple separate advertisement sets are used for relayed messages, only amplifying the aforementioned issue. To counteract this, these nodes have been configured with constant default parameters, which are not updated during the experiment phases. This results in a roughly 24 Hz

original message transmission rate, as each message is always transmitted thrice in the network layer. However, relay parameters are still updated for these nodes too as normal.



(a) Four nodes packed together



(b) Scarcer network

Figure 4.6: Test setup C with two density variations. Due to spacing constraints, the two interfering nodes are both positioned next to nodes 1 and 2 respectively.

The experiments are divided into separate steps. During each step, one or two parameters are adjusted, going through a set of predefined sample values. Since most of the parameters are related to other parameters in some way, they are examined in pairs. All parameters that are not examined in a certain step are set to their default values. A more detailed description of the experiment phase is presented in

appendix A, including the sample values that were tested. The test environment is a quite typical office space with some desks, chairs and divider elements, creating a moderately obstructed indoor setting, which can slightly lower signal reception quality, but still within acceptable levels.

5 Findings and discussion

Based on the experimental study described in chapter 4, this chapter continues by analyzing the results obtained and discussing their implications both in terms of memory and network performance. To evaluate the various parameters and configuration options as comprehensively as possible, analyses of results obtained from other related studies are also used to overcome some of the limitations experienced in this study.

5.1 Memory analysis

The memory impacts of various configuration options were determined using the memory report tool available in nRF Connect SDK.

For example, the whole application used to perform the experiments required roughly 290 kB flash memory and 70 kB RAM, as reported by the memory report tool. With purely application specific parts excluded, Bluetooth Mesh parameters set to defaults and most debugging and logging options turned off, the foundational part of the Bluetooth Mesh application was left with requirements of 265 kB flash and 50 kB RAM.

In this example, the underlying Bluetooth host stack accounted directly for over 20 kB of the RAM requirement, while the mesh part on top of it required just a litte over 6 kB RAM. The remaining part of the RAM allocation was shared between process stacks and buffers related to eg. cryptography functions and the Zephyr

kernel.

In the case of flash memory, the relative memory usage statistics were quite similar, although the mesh part accounted for a slightly larger part of the total flash requirement with respect to the host stack. Of course, this is greatly affected by application requirements, since all used models and optional networking features increase the Bluetooth Mesh memory usage to some extent. Additionally, Nordic proprietary libraries required a significant portion (over 100 kB) of the flash memory.

While these numbers serve as a good reference for further analysis, it should be noted that no big efforts were made to optimize the base memory usage. For instance, many stack and buffer sizes in the Zephyr kernel or Bluetooth host level could be adjusted based on application needs, but these are not studied in this thesis.

The implementation of Bluetooth Mesh in Zephyr allows a good level of customization with options related to enabling or disabling certain features and defining certain numeric values. In terms of memory, most often the options which enable or disable features tend to affect application size and therefore flash memory usage, whereas application RAM usage can be better controlled with some numeric definitions. The choice of features that get enabled usually comes from application-specific requirements, so there is only very little adjustment range concerning these types of definitions in practical applications. Many of the options affecting RAM usage instead apply to core functionalities of Bluetooth Mesh while also having a much wider configuration range.

In Zephyr, most of these configuration options are set to minimum or close to mimimum values by default, leaving again very little room for memory optimization. However, to suit the needs of real-life applications, some adjustments to these values are often required. But in many cases, the required adjustment would still likely be very slight and largely insignificant in terms of RAM usage.

5.1.1 Options defining device capabilities

Device configurability related options such as the maximum numbers of application keys per network, group addresses per model or label UUIDs didn't show to have much of an impact unless very large values for these options were used or the total number of models were very high. An increase in the maximum number of subnets showed a slightly larger impact, as around 200–300 bytes get reserved for each subnet, depending on a few other options such as enabling the GATT proxy feature.

5.1.2 Message cache and RPL

The impacts of message cache and replay protection list sizes were also shown to be quite small, both taking only eight bytes per entry. In Zephyr, the default values for these are only 32 and 10 respectively. It should be noted, however, that depending on network size, layout and traffic intensity, these values might need to be adjusted significantly in real-life applications. A large network consisting of 500 devices, for example, might have nodes (eg. gateway nodes or subnet bridges) that need to process messages from all other nodes, each requiring its own slot in RPL. Another example could be a network with potentially long relay loops. In this case, it might be possible for some nodes to receive the same message twice, even after a long wait period. With enough network traffic, the message cache would need to include hundreds or even thousands of entries to prevent acting on previously received messages.

5.1.3 Advertisement buffers

The amount of network traffic that a node can handle using the default advertisement bearer is significantly affected by the number of advertisement buffers. Each advertised network packet consumes an advertisement buffer which is later freed when the advertisement finishes. The duration that the buffer is reserved is affected

by the number of retransmissions and retransmission interval. If no advertisement buffers are available, any new transmissions will automatically fail.

By default, 6 advertisement buffers are available for local messages and 32 for relayed messages. The need to increase the number of these buffers quickly becomes critical when messages are sent in bursts with additional retransmissions or increased transmission intervals. This especially applies when sending segmented messages. Since each advertisement buffer takes up 60 bytes of RAM, this configuration option can have a significant impact on memory usage (32 buffers already take up almost 2 kB of RAM). This should especially be considered in the case of relays, where a high number can easily be needed if network density and traffic intensity are high.

5.1.4 Segmentation memory options

Segmented messages also require additional memory in the transport layer for the SAR process. The experimental analysis showed that one segmentation buffer takes 104 to 232 bytes, scaling with the maximum configured segment count. Reassembly buffers require slightly more memory, again scaling with the maximum segment count from 126 to 264. While these are still very moderate numbers, it's worth noting that if multiple segmented messages need to be processed in parallel, more buffers are needed and the memory impact grows substantially. Again, the duration that these buffers are reserved depends on various parameters, but also on communication reliability and latency in the case of unicast communication, as described in section 4.3.3. Failed transmissions can cause additional delays on the receiving end, as discard timeout timer triggering is required to end the process.

In Zephyr, the contents of the messages are not stored within these buffers, but instead in separate segment buffers, each 12 bytes in size. These segment buffers are shared for segmentation and reassembly processes, and their count can be adjusted in Zephyr as well.

5.2 Performance analysis

Two key network performance metrics, communication reliability and latency, were evaluated in the experimental phase with varying network configuration parameters. Additionally, multiple different device setups were used to better understand how changes in network layout and external conditions affect the performance impact of the examined parameters. With many parameters, effects became clearly visible only in certain conditions: eg. retransmission counts only had a notable impact when the distance between nodes was high enough. As the total amount of data generated for this analysis is too huge to be covered in its entirety, the discussion will try to focus on these more specific cases that show the most interesting results.

Some diagrams are utilized to better present the experimental results of selected phases. These plots show the obtained latency statistics and PER of successful round-trip communication events (starting at the beginning of message transmission and ending at the reception of a corresponding acknowledge message). With the exception of phases involving segmented multicast messages or publication parameters, communication from node 1 to 2 and back to node 1 was designed to be symmetrical in terms of message length, contents and configuration parameters, and as such, one-way communication statistics could be roughly estimated in these cases with:

$$Latency_{one-way} = \frac{RTT}{2}$$

and

$$PLR_{one-way} = 1 - \sqrt{1 - PLR_{two-way}}$$

Latencies in the plots are expressed with solid-colored lines showing average latency, and lighter dashed lines showing the min-max range. This range represents the consistency of latency, which can also be an important consideration in some

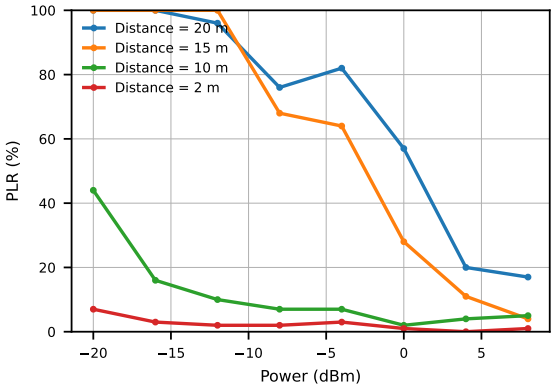


Figure 5.1: TX power impact on PLR at varying node distances

applications [15]. An example of such an application could be a lighting system involving multiple lamps, where it is easy to see if all lamps don't turn on or off within a very limited period.

5.2.1 Effect of transmit power adjustment

The effect of TX power was seen clearly in the experiments as expected. Especially when the distance between nodes was higher, even small changes in TX power made a big difference in reception reliability, as can be seen in figure 5.1.

As already discussed in the previous chapter, TX power is a setting that can significantly influence device power consumption. Table 5.1 shows how current consumption increases with TX power on nRF52840. Below 0 dBm, the differences in current consumption are relatively small in comparison to those in the 0 to +8 dBm range. But as transmission periods are typically very short (measured times were around 2 ms), and considering that regular nodes will be the rest of the time in scanning mode drawing constantly around 4.6 mA [18], the power impact may be negligible unless the device is regularly transmitting. For example, a device transmitting at an average rate of 10 Hz, would consume around 5.3% more power if TX power is set to +8 instead of -20 dBm. However, with a 0.1 Hz rate, the respective increase would be only around 0.053%. In addition to the publication rate of the

TX Power (dBm)	Current (mA)
-40	2.3
-20	2.7
-16	2.8
-12	3.0
-8	3.3
-4	3.1
0	4.8
+4	9.6
+8	14.8

Table 5.1: Typical nRF52840 Radio transmitter current consumption (DC/DC, 3V) [18].

node, this transmit rate is affected by all retransmissions as well as possible relaying. Thus increasing TX power can still reduce total power consumption if the number of retransmissions can be lowered in return.

For low-power nodes, the power impact of TX power adjustment is significantly more drastic when a much larger portion of the radio-on period is accounted for transmissions. This is further discussed in section 5.2.6.

5.2.2 Transmission count impact

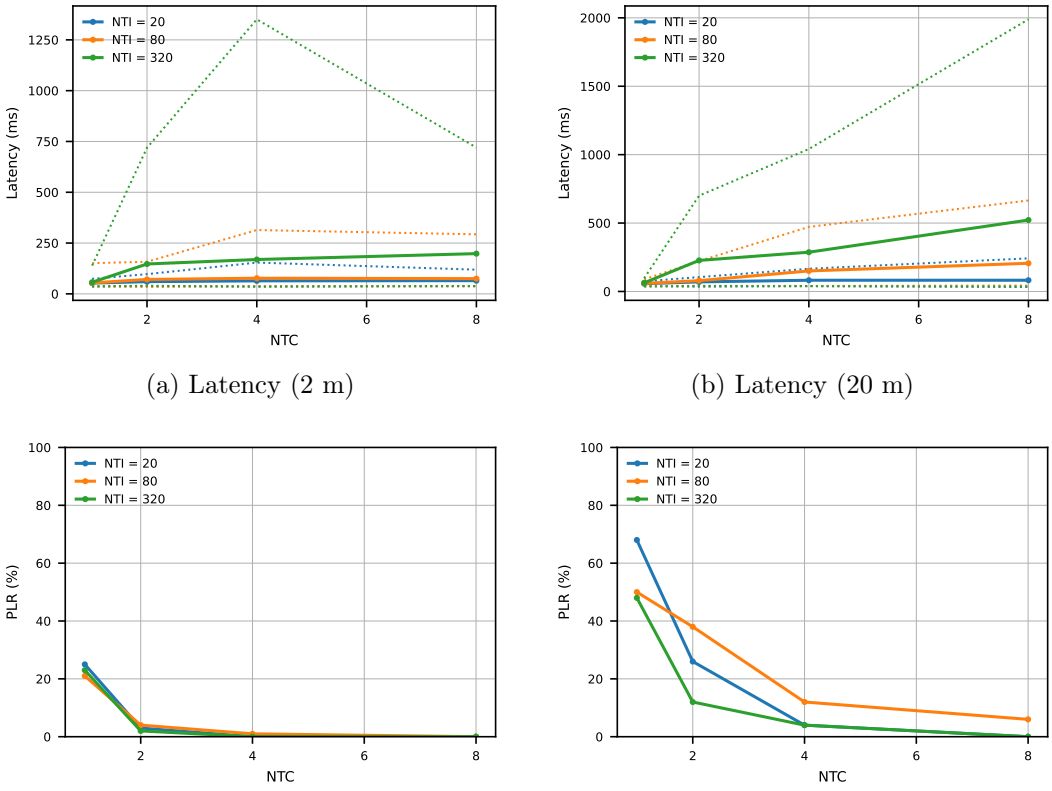
Most of the examined parameters controlled message retransmissions in some way, with distinct retransmission parameter sets existing for publication, segmentation and reassembly, relaying and network transmission. While these parameters appear mostly very similar by their description, they have a varying impact on performance depending on context.

In theory, the communication reliability effect of retransmissions can be calculated with

$$PLR_n = (PLR_1)^n$$

in which PLR_1 is the PLR of a single transmission, and n is the total number of transmissions. Figure 5.2 shows clearly how increasing the number of retransmissions

enhances communication reliability exponentially in a simple two-node scenario. The effect is more noticeable when the distance between nodes is longer, as the PLR of a single transmission is much greater due to a weaker signal.



(a) Latency (2 m)

(b) Latency (20 m)

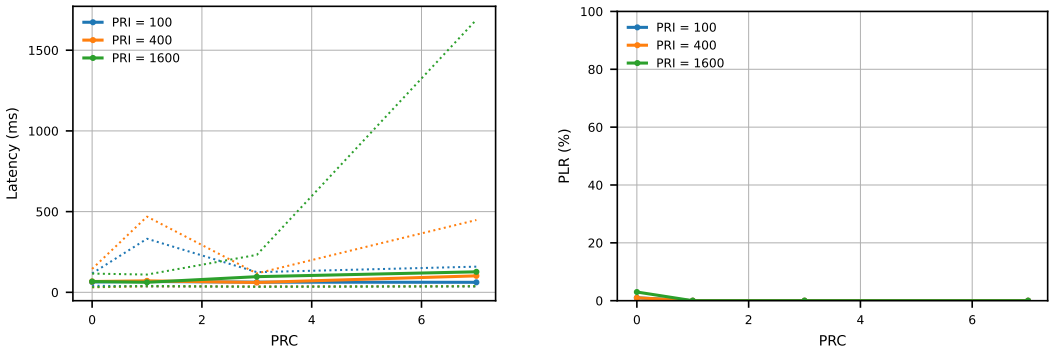
(c) PLR (2 m)

(d) PLR (20 m)

Figure 5.2: Effect of Network Transmit Count (NTC) and Network Transmit Interval (NTI) on latency and PLR. (Setup A).

A "side-effect" of extra retransmissions is that latency average and variance also seem to increase in proportion to transmission count, as can be seen in figure 5.2 with 20 m node distance. However, it can be somewhat misleading, since any increases in latency are solely the result of receiving additional transmissions a bit late as opposed to not receiving them at all. So in a sense, the number of transmissions doesn't really increase the latency by itself, as it's still just as likely to successfully receive a message on the first transmission undepending on the number of total transmissions. Figure 5.3 demonstrates this behaviour well, as it showcases a scenario in which close to 0%

PLR has already been achieved with network layer retransmissions alone, leaving publish retransmissions mostly redundant. The same effect is also visible in figure 5.2 with a 2 m node distance, and with NTC of 4 or more. This also explains why the minimum achieved latency always remains constant even with a 20 m node distance.



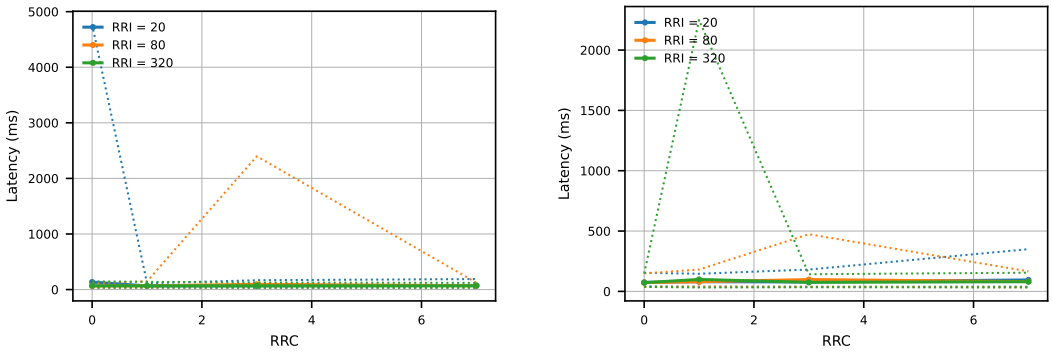
(a) Latency (b) PLR

Figure 5.3: Effect of Publish Retransmit Count (PRC) and Publish Retransmit Interval (PRI) on latency and PLR. (Setup A, 2m).

When different kinds of retransmissions are considered, it should be kept in mind that their effects are most often cohesive. For example, when a segmented message is published to a multicast address with a PRC of 2, while leaving all other parameters to their default (SRC=2, NTC=3), the total number of transmissions for each segment would be $(2 + 1) \cdot (2 + 1) \cdot 3 = 27$. This means that parameters affecting the retransmission of lower-level PDUs are very important, as they have an impact on all higher-level communication as well.

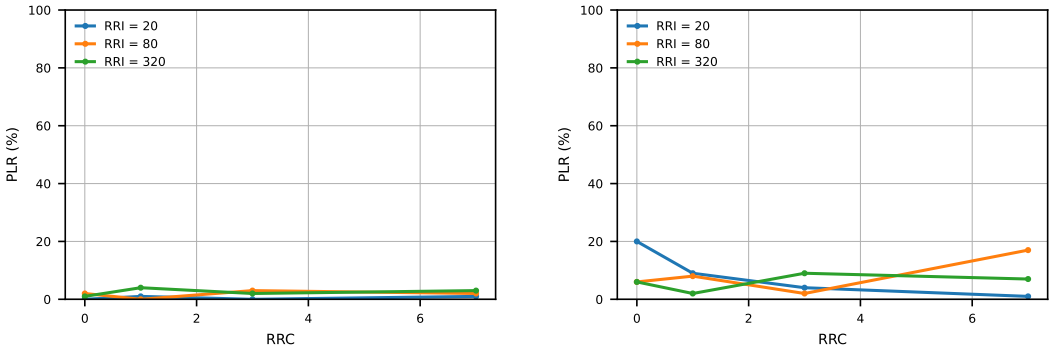
In this study, default values were used for all parameters not under focus in each of the phases, most notably including NTC=3. This was an intentional design choice to allow independent parameter analysis, but in some cases, it resulted in such a good baseline networking performance, that adjustments of other parameters were left with very little room for improvement, eg. with PRC values in setup A with 2 m distance, as already shown in figure 5.3. In fact, the only test scenario, where PRC

value adjustment had a more noticeable effect, was setup A with a 20 m distance, in which retransmissions based solely on NTC parameter were not numerous enough to maintain 0% PLR on their own.



(a) Latency (denser network)

(b) Latency (scarcer network)

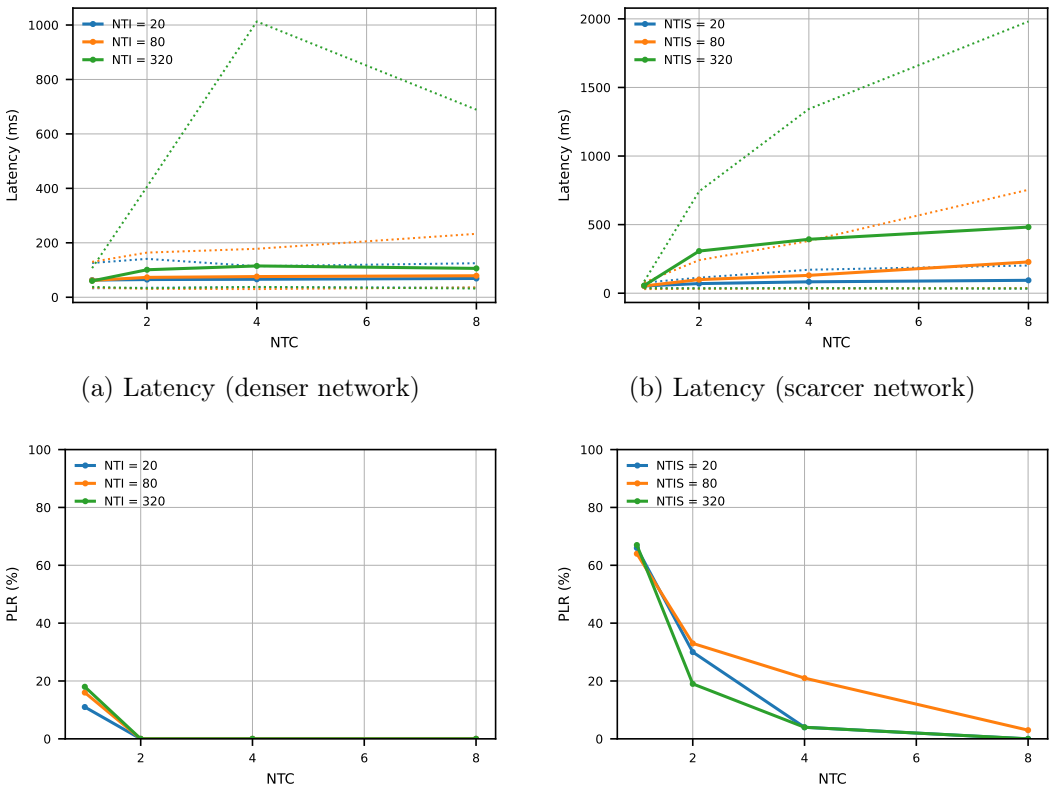


(c) PLR (denser network)

(d) PLR (scarcer network)

Figure 5.4: Effect of Relay Retransmit Count (RRC) and Relay Retransmit Interval (RRI) on latency and PLR. (Setup C).

This is also reflected in relay retransmit parameters, as the mere presence of a relay node in setup B was already enough to achieve a constant 0% PLR. In setup C, however, even though two relay nodes were present, the number of relay retransmissions had nearly zero impact on PLR, as shown in figure 5.4. This highlights how different challenges faced in real-life networking scenarios (low signal quality vs. congestion) can require different solutions.



(a) Latency (denser network)

(b) Latency (scarcer network)

(c) PLR (denser network)

(d) PLR (scarcer network)

Figure 5.5: Effect of Network Transmit Count (NTC) and Network Transmit Interval (NTI) on latency and PLR. (Setup C).

5.2.3 Transmission interval impact

While the number of transmissions clearly has a significant impact on communication reliability in some cases, the corresponding transmission intervals appear to affect normal node-to-node communication reliability relatively little, as can be seen in figure 5.2. In that, NTI of 320 ms yielded the best results, although only by a relatively small margin. Very similar results were seen with (scarcer) setup C, as figure 5.5 shows. Quite interestingly, out of the tested NTI values, 80 seemed to offer the worst PLR in all setups on average. The reason for this is not explicitly clear, but it proves that PLR and transmission interval are not always linearly correlated.

However, it appears that longer intervals primarily amplify the maximum latency (and thereby also its average and variance) caused by additional transmis-

sions. Longer transmission periods typically won't reduce the total amount of data transmitted nor the time that it takes to transmit the data. Yet, memory for that data still needs to be reserved for longer which reduces the average availabilty rate of buffers (eg. for advertising or segmentation, depending on the case) in the long run.

It is important to notice that if the interval between successive transmissions in some layer is lower than the time it takes to complete one transmission in a lower layer, buffers will start to exhaust faster than they are freed. For example, with NTC of 6 and NTI of 20 ms, it would always take more than 100 ms to complete a transmission in the network layer. In this case, if a segmented message would be sent with asegment interval of 40 ms, the network layer would not have time to completely process the first segment before the second one should be processed. Thus it is important to ensure that an adequate number of buffers remain available for transmissions, as otherwise messages will not be sent. In these experiments, buffer availability was ensured to be high enough for this problem not to occur.

Too low transmission interval could also become an issue on the receiving end if the receiver ends up in a state where it cannot temporarily handle incoming packets due to inadequate buffering capacity. This behaviour was uncaptured in the experiments, but it is still identified in related works. In a paper by Hernández-Solana et al. [22], the authors highlight this issue, especially in the context of relay nodes, which may need to process PDUs from multiple nodes at the same time. The likelihood of having to discard packets is of course affected by the time that the relay node needs to store them, as defined by relay retransmit parameters.

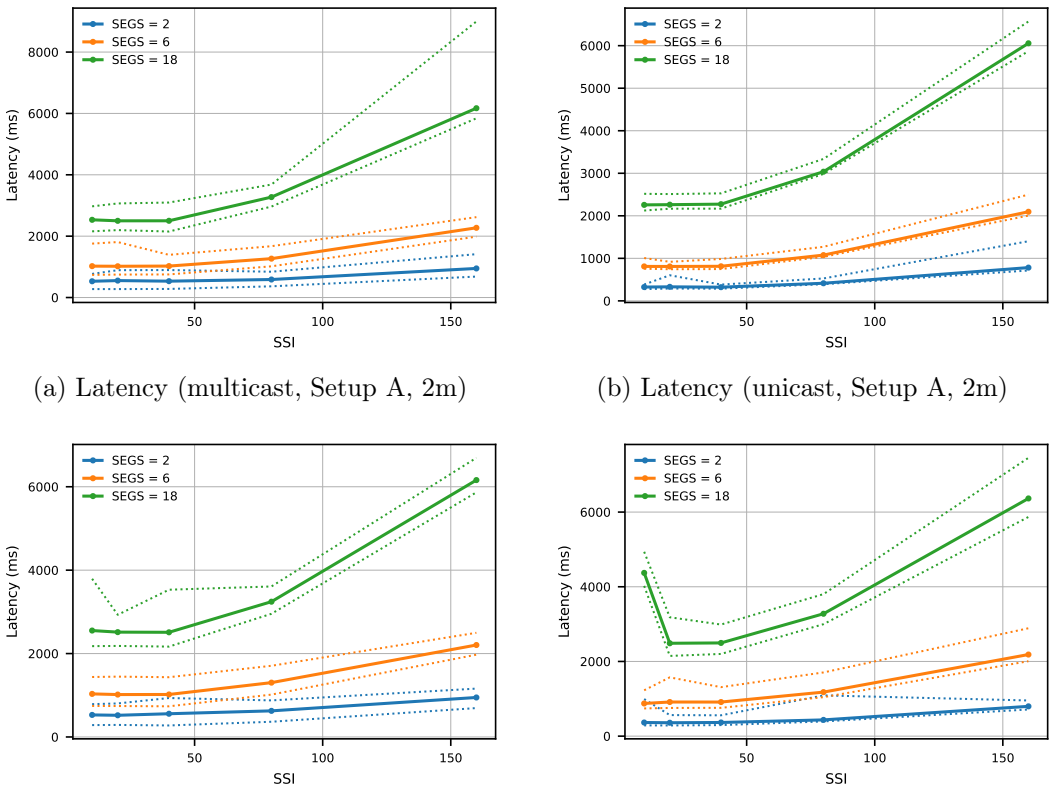
The authors of the paper also give an estimation formula for the collision probability of advertising PDUs from different nodes based on the retransmission interval parameter value. The formula estimates around 3% collision probability for two nodes with a 20 ms interval whereas a 50 ms interval would reduce the collision prob-

ability to around 1%. A higher number of closeby nodes would of course increase the collision probability. The calculation assumes that maximum-sized advertisement frames are used, which would result in 376 µs transmission time per channel. Still, it would have likely taken a much larger network of test nodes to clearly see this kind of behaviour in the experiment results. The addition of interfering relay nodes instead actually improved packet delivery in some cases, especially in short-range communication, because of extra delivery paths and repetitions.

In cases where communication reliability can't be further improved with additional retransmissions, such as the one shown in figure 5.3, larger retransmission intervals don't tend to increase typical latency nearly at all. But even then, it might be possible for occasional communication failures to occur due to eg. the aforementioned buffer incapacity problem. In these types of scenarios, it would probably be more beneficial to have greater delays between retransmissions to allow temporary communication issues to resolve before attempting another transmission. This approach helps prevent unnecessary congestion and reduces the likelihood of repeated failures due to transient network conditions, and can also probably explain why the lowest PLR was achieved with the highest NTI value in those experiments where the node distance was 20 m.

5.2.4 Effect of SAR transmitter parameters

The most impactful SAR parameter in terms of latency was SAR Segment Interval as illustrated in figure 5.6. Since it modifies the interval between each successive segment, it is heavily affected by the segment count of the message, both in unicast and multicast scenarios. The lower end of the adjustment range (10–50 ms) seemed to make very little difference in latency when used with a low number of segments, Even with 18 segments, the differences in latency were still relatively small below 50 ms segment interval, although in Setup B, a latency spike can be seen with the



(a) Latency (multicast, Setup A, 2m)

(b) Latency (unicast, Setup A, 2m)

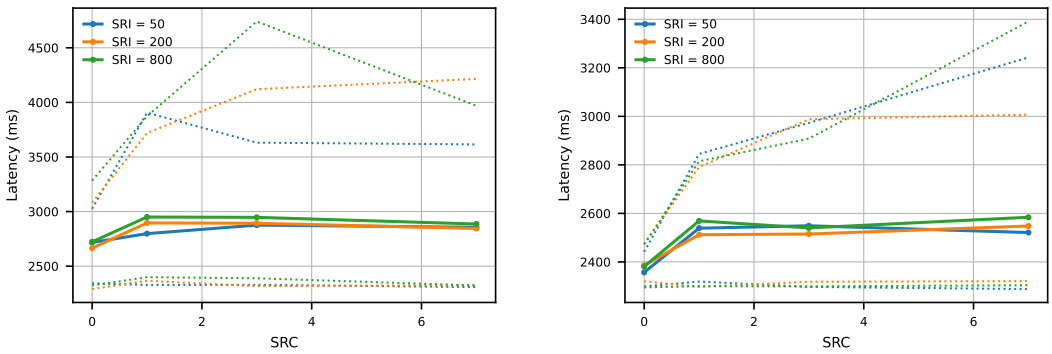
(c) Latency (multicast, Setup B)

(d) Latency (unicast, Setup B)

Figure 5.6: Effect of SAR Segment Interval (SSI) and PLR with variable message segment counts.

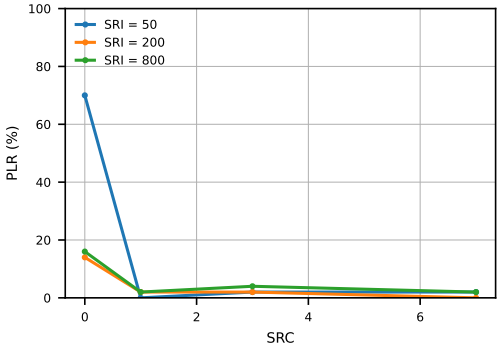
lowest SSI value. The flat area is likely caused by the network layer, which always needs at least 60 ms to transmit a PDU with retransmissions, meanwhile, other segments need to wait in buffers. The spike in latency with setup B shows well, how the one relay node can't keep up with this rate. It's likely that at some point the relay would have run out of buffers if the test message had consisted of more segments or the relay retransmit interval had been high enough.

In the experiments, SAR retransmissions had a notable difference between zero and one retransmissions, especially in setups A and C with 20 m node distance. Mostly in other cases, the PLR was always close to 0% and no significant changes were seen in latency either. This resembles the situation faced with publish retransmissions. However, a particularly interesting observation was that increasing the

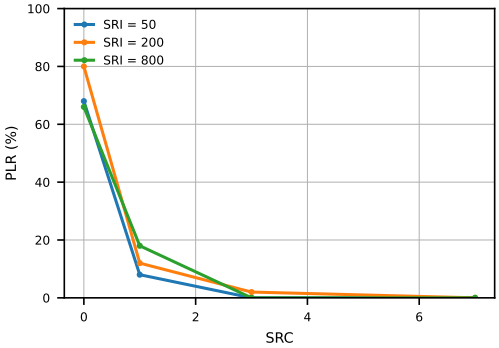


(a) Latency (multicast)

(b) Latency (unicast)



(c) PLR (multicast)

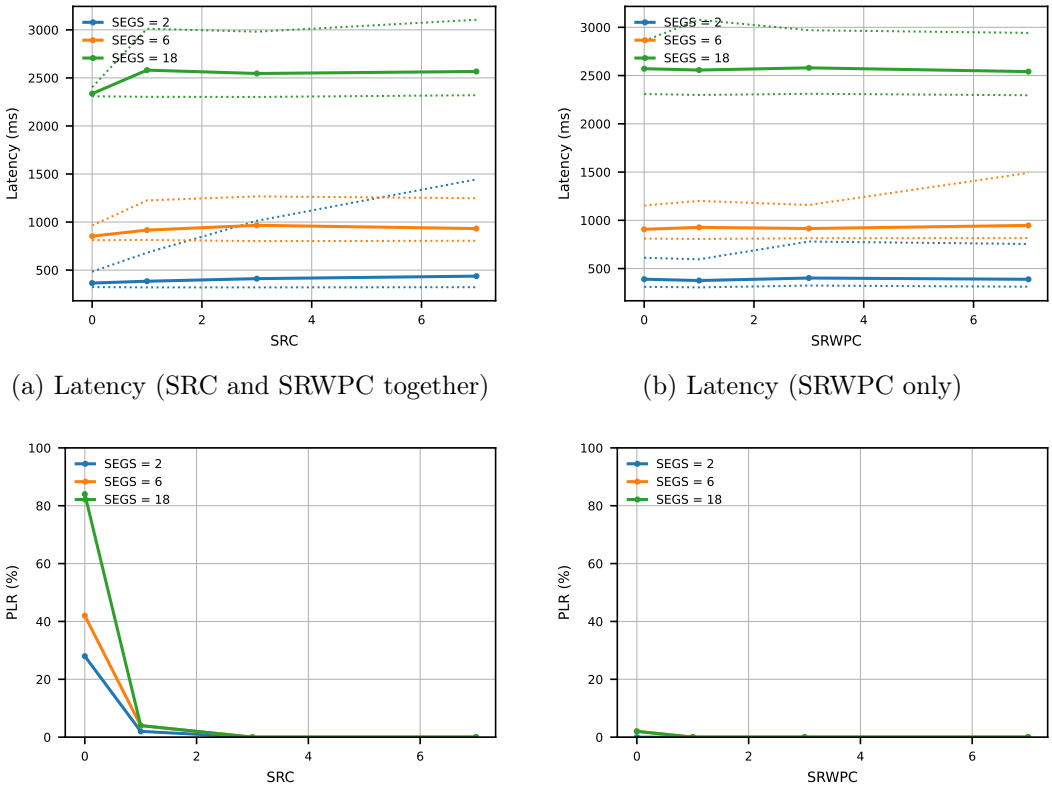


(d) PLR (unicast)

Figure 5.7: Effect of SAR Retransmit Interval (SRI) with different SRC values. (Setup A, 20 m, 18 segments).

SAR retransmission count helped reduce PLR much more in unicast mode, where only unreceived segments are retransmitted instead of all segments. In multicast mode, the PLR was already much better even with only one transmission, probably thanks to the much greater random back-off time used in multicast responses. The results shown in figure 5.7 support this statement well, as with just 50 ms SRI the PLR was bad even in multicast mode (although this was the case only in the 20 m, two-node setup). Otherwise, the SRI value seemed to make no significant difference to neither latency nor PLR with constant segment count. In unicast mode, however, slightly lower latencies could be achieved due to lower random back-off times and fewer redundant retransmissions. Additionally, around 5–15% higher average latencies were observed in multicast mode in setup C, where there was more network

traffic present, but in unicast mode, this did not have any noticeable effect.



(a) Latency (SRC and SRWPC together)

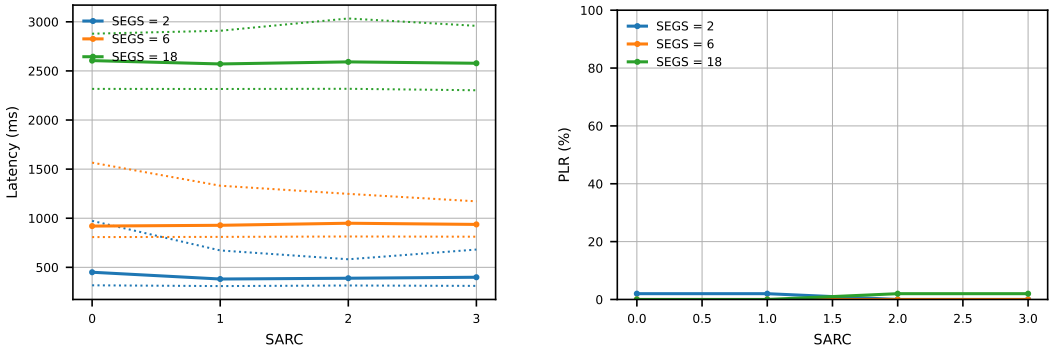
(b) Latency (SRWPC only)

(c) PLR (SRC and SRWPC together)

(d) PLR (SRWPC only)

Figure 5.8: Effect of SAR Retransmit Count, both total and Without Progress (SRC/SRWPC), on latency and PLR with variable message segment counts in unicast mode. (Setup A, 20 m). Here the maximum SRC value of 7 is used when only the SRWPC is adjusted.

Two segment retransmit count parameters are used in unicast mode, as discussed in chapter 4.3.3. Figure 5.8 shows how these parameters affect the communication performance in the 20 m, Setup A example scenario. The SRWPC parameter adjustment made only very little difference in the experiments. In even worse communication conditions and with larger messages, SRWPC adjustment would perhaps show results similar to those of the SRC, although with less impact.

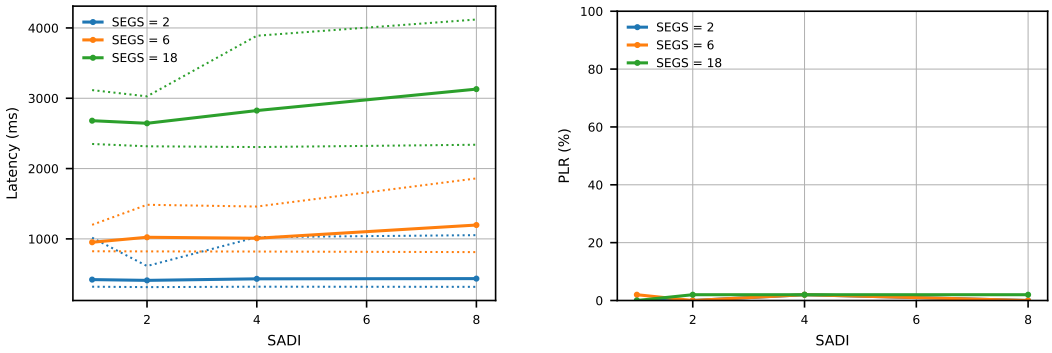


(a) Latency (b) PLR

Figure 5.9: Effect of SAR Acknowledge Retransmit Count (SARC). (Setup A, 20 m).

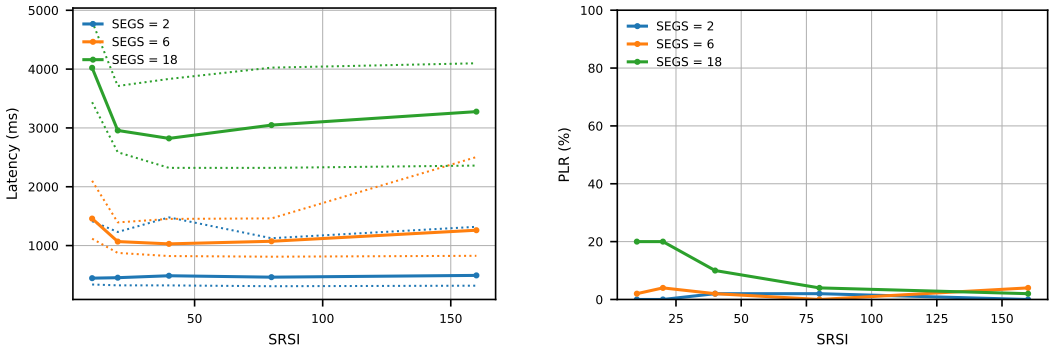
5.2.5 Effect of SAR receiver parameters

SAR receiver parameters, only used during segmented unicast communication, had minimal impact on networking performance. With setup A and 20 m node distance — where average signal quality was the lowest among all experiments — the most distinct differences in performance were observed. Increasing the transmit count of SAR acknowledgement messages was seen to reduce maximum latency a bit (shown in figure 5.9, but only with smaller messages. With the 18-segment message, the maximum latency actually increased a bit, although not enough to make a difference in average latency.



(a) Latency (b) PLR

Figure 5.10: Effect of SAR Acknowledge Delay Increment (SADI). (Setup A, 20 m).



(a) Latency (b) PLR

Figure 5.11: Effect of SAR Receiver Segment Interval (SRSI). (Setup A, 20 m).

The parameters SAR Acknowledgement Delay Increment and SAR Receiver Segment Interval, which control the timing of SAR acknowledgement messages, had a slightly more noticeable impact, as can be seen in figures 5.10 and 5.11. Larger SADI values mainly increased latency, especially for larger messages. Similarly, SRSI also increased latency with higher values, although it exhibited the same kind of latency spike at the 10 ms interval as seen with SSI (figure 5.6). This again suggests that such low intervals easily lead to buffering difficulties. But in contrast to other SAR receiver parameters, SRSI also had a slight impact on PLR, particularly with the largest message.

5.2.6 LPN and friendship parameters

Designed to be used with power-constrained devices in particular, the low-power feature of Bluetooth Mesh offers a trade-off between communication latency and energy consumption. The scale of this trade-off is of course configurable based on application needs. The protocol defines multiple parameters which affect the behaviour of low-power and friend nodes, although many of these are related to the friendship establishment process which is a more advanced topic and not covered in this thesis in more detail. Two parameters, however, become especially interesting

in general performance and energy consumption perspective; these are *PollTimeout*, which defines the maximum time that a friend node can wait for until a new poll request from LPN, and *ReceiveWindow*, which defines the maximum time for a LPN to scan responses from its friend node.

Darroudi et al. [23] have studied the impact of these parameters on average current consumption and expected battery lifetime. They also consider how the transmission frequency of a LPN affects these statistics. The calculations in the study are based on the energy consumption of the nRF51422 development kit powered by a 245 mAh button cell battery.

The study shows that for low PollTimeout values (from around 1 to even 100 s) the ReceiveWindow parameter has a significant effect on energy consumption as scanning is performed more frequently. For example, assuming without transmissions, a ReceiveWindow of 255 ms and PollTimeout of 10 s would mean an average current consumption of around 370 µA, whereas lowering the ReceiveWindow down to 1 ms would put the consumption below 20 µA. This would be seen as a massive improvement in battery lifetime. However, raising the value of PollTimeout past 1000 s quickly makes the ReceiveWindow parameter less significant in regards to current consumption, and at values of around 10000 s and above the ReceiveWindow becomes practically meaningless.

When data transmissions from the LPN are considered, the study shows that high data transmission rates (eg. 1 Hz) have a major impact on current consumption and battery lifetime, when the PtollTimeout value is high or the ReceiveWindow value is low. With low PollTimeout and high ReceiveWindow values, however, data transmission rate seems to have little to no effect. Also when the transmission rate is lowered to even around 0.1 Hz, the average consumption quickly drops relatively close to the case without any transmissions.

Although Darroudi et al. only focus on the energy consumption aspect of these

parameter configurations, it is also important to consider any other implications that these parameters might have on overall network performance. First and foremost is the latency trade-off that is tightly related to the PollTimeout parameter, as its value also defines the maximum communication latency when messaging to the LPN. Higher PollTimeout values also mean that the associated friend node might need to reserve more memory for awaiting messages in cases when the LPN receives larger amounts of data during its sleep period, otherwise, data losses are likely to occur. Lower ReceiveWindow values on the other hand tighten the timing constraints of the friend node and the LPN while also requiring more reliable communication to be effective.

5.2.7 Alternative PHYs

Alternative PHYs introduced in Bluetooth 5.0 offer interesting capabilities to tune Bluetooth Mesh performance. However, it is good to remember that they are not directly compliant with official Bluetooth Mesh specifications due to potential compatibility issues. Furthermore implementing the necessary integrations to existing software drivers often requires some modifications. Due to these reasons, only the default 1M PHY was used in the experimental phase of this thesis. Fortunately, some studies have already been conducted, in which the potential performance improvements of alternative PHYs are analyzed in the context of Bluetooth Mesh.

In a work by Ortiz et al. [24], an experimental comparison between 1M PHY and Coded (S=8) PHY shows that with the usage of Coded PHY, a very significant boost in reliable communication range can be achieved. The authors performed the comparison in an unobstructed outdoor environment with linear node placements, default transmit count parameters and -20 dBm TX power. With this setup, around 90% packet delivery rate (PDR) was achieved using the 1M PHY at a 15 m distance, The Coded PHY showed a comparable PDR at a 45 m distance, outperforming the

1M PHY even when it was used with a relay node installed in the middle.

The authors also tested the Coded PHY with +8 dBm TX power to find out its limits. At 350 m, a PDR of 95% was obtained without any relays. Using one relay in the middle, a relatively good PDR of 90% was obtained at 600 m distance, and even at 800 m, a PDR of 73% could still be obtained.

These results show that the Coded PHY does not only increase network coverage substantially, but also allows mesh networks to practically function with fewer relays. This can offer reductions to system costs, configuration work and energy consumption. An increased range of Coded PHY should also ensure better multipath delivery, which is important for error-resistant networks. Of course, the use of Coded PHY also comes with the cost of reduced data rate, to either 1/8 or 1/2 (depending on the used coding scheme S) than that of uncoded 1M PHY.

5.2.8 Throughput

Bluetooth Mesh has been designed to be used in applications with typically quite low throughput requirements. Implementations strictly sticking to Bluetooth Mesh protocol specification can offer very meagre maximum data throughput of less than 4 kbps in a single hop scenario, which is not even close to the maximum throughput of some rivalling technologies (eg. Zigbee: 21 kbps, Thread: 47 kbps) [25]. However, by fully utilizing the extended advertising feature and 2M PHY introduced in Bluetooth 5.0, Bluetooth Mesh can be modified to increase the maximum throughput massively. For example, Cerio et al. [25] have proposed a slightly modified implementation capable of achieving a theoretical throughput of nearly 500 kbps. It is still important to note that as more hops are introduced, the maximum achievable throughput starts to rapidly decline. In real-world applications, conditions are also often unideal, and communication errors have to be addressed with eg. retransmissions, further reducing the maximum achievable throughput.

5.3 Dynamic configuration

In many real-world applications, external conditions may fluctuate, some devices might move or get turned off, and new devices might be brought into existing networks. These types of shifts can significantly impact network performance, influencing factors such as connectivity, message propagation, and overall reliability. As a result, it is crucial to design Bluetooth Mesh networks with adaptability in mind, ensuring that they can maintain stable communication despite dynamic conditions. Strategies such as self-healing mechanisms, adaptive retransmission intervals, and efficient routing can help mitigate disruptions caused by node mobility, power cycling, or network expansion.

By understanding the impacts of different parameters, corrective measures can be taken to optimize network performance in response to deployment changes. The main method for runtime configuration in Bluetooth Mesh networks is based on tweaking node networking parameters through configuration models. Depending on the application scenario, this can be done either manually using eg. a smartphone app, or automatically with the help of a dynamic parameter tuning algorithm. Identifying the needs and means for future configuration is also an important step during application development, as new features and models can't be easily added to existing nodes later on without updating application firmware.

Even in the most basic scenarios, which only involve initial network configuration as part of system installation, it can be quite beneficial to customize at least some networking features to better align with the specific requirements of the deployment. In particular, a well-planned selection of relay-enabled nodes can have a significant impact on overall network performance. As an example, a study by Aijaz et al. [26] found that configuring only half of the nodes as relays in a test network improved the average latency by approximately 25% compared to using every node as a relay. Of course, aspects such as network density and structure can vary greatly between

deployments, making it challenging to establish optimal configurations that work well in all scenarios.

As already mentioned, it is also possible to implement more advanced network adjustment methods by using specific parameter-tuning algorithms. These algorithms could, for instance, allow nodes to increase or decrease retransmission counts or transmit power based on packet delivery rate, potentially even optimizing energy consumption as well. One such algorithm proposed by Silvestre-Blanes et al. [27] was shown to drastically improve relay networking reliability in simulated static and dynamic environments, especially under poor signal conditions.

6 Conclusions

In this thesis, the performance characteristics of Bluetooth Mesh networking were evaluated through both theoretical analysis and experimental study. The primary focus was placed on understanding how various network configuration choices influence key performance aspects such as communication reliability, latency, and network efficiency under different deployment scenarios, and furthermore, what are the limiting factors that should be considered when designing Bluetooth Mesh applications.

Throughout the work, several important findings emerged regarding the behavior of Bluetooth Mesh in both controlled and dynamic environments. In addition to assessing the impact of core configuration options, attention was also given to identifying potential challenges and limitations inherent to the technology, along with practical considerations for mitigating them in real-world applications. Here, the key results of the study are summarized, their implications are discussed, and possible directions for future research and development are proposed.

In the early chapters, the context of Bluetooth Mesh was examined from a broader perspective, considering both the current state of the IoT application field and related existing technologies. This led to the observation that Bluetooth Mesh and similar technologies encounter highly varied requirements across different applications, underlining the importance of versatile configuration options. Also, due to the rapid and constant development of the IoT landscape, forward adaptability remains a critical characteristic for any communication protocol that is expected to

stay relevant in the future.

Upon closer look into the technicalities of Bluetooth Mesh, many of the protocol's inherent strengths could be highlighted, including scalability, network robustness, multilayer security, extensive hardware support and configurability. At the same time, this analysis revealed several potential issues, especially in regards to communication reliability in certain scenarios, as well as limitations imposed by the protocol.

In terms of packet reception reliability, three main challenges were identified in this study: poor radio conditions, channel congestion, and inadequate buffering capacity. However, it was shown that some of these issues can be mitigated through appropriate configuration of the network nodes. This can be a complex task though, as different network deployments often vary in topology, device density, and scale, making it difficult to define universally optimal parameter settings. When networks grow in size and complexity, self-optimizing network mechanisms, such as directed forwarding, offer a promising approach. However, further research is needed to fully understand their potential and limitations.

Even though Bluetooth Mesh is based on Bluetooth Low Energy, the official protocol provides only limited support for energy-constrained devices, necessitating the presence of non-low-power nodes to maintain the core network infrastructure. Still, significant energy savings can be achieved for low-power nodes through careful parameter selection. It should also be noted that several adapted implementations have been proposed to overcome the protocol's dependence on always-on, non-low-power nodes to maintain message relaying and network structure [28], [29]. This highlights that energy-efficient operation remains an important and noteworthy area for future development of the Bluetooth Mesh protocol.

There is also an ongoing interest in integrating extended advertising and alternative PHYs from Bluetooth 5 into future versions of the Bluetooth Mesh specification.

Research papers and experimental implementations have explored the possible benefits of leveraging these features in mesh networking, indicating promising results in terms of both reliability and throughput. But of course, integrating new features while maintaining compatibility with the existing specification can present a considerable challenge. More easily implementable enhancements have also been suggested, including network traffic prioritization and situation-based relaying [30].

In conclusion, Bluetooth Mesh offers a powerful framework for building scalable and reliable IoT networks. However, as this study shows, the technology still poses many limitations and potential challenges that can make it unsuitable for some IoT network applications. In any case, through careful adjustment of system parameters, it is possible to make some changes to the balance between network performance aspects and resource demands to best suit application-specific requirements. Adaptive parameter tuning strategies have the potential to further enhance network performance and continue to be an interesting direction for future research.

References

[1] M. Mansour, A. Gamal, A. I. Ahmed, *et al.*, "Internet of things: A comprehensive overview on protocols, architectures, technologies, simulation tools, and future directions", *Energies*, vol. 16, no. 8, 2023. DOI: 10.3390/en16083465.

[2] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges", *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012. DOI: https://doi.org/10.1016/j.adhoc.2012.02.016.

[3] C. Li, J. Wang, S. Wang, and Y. Zhang, "A review of iot applications in healthcare", *Neurocomputing*, vol. 565, p. 127 017, 2024, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.127017.

[4] R. Chataut, A. Phoummalayvane, and R. Akl, "Unleashing the power of iot: A comprehensive review of iot applications and future prospects in healthcare, agriculture, smart homes, smart cities, and industry 4.0", *Sensors*, vol. 23, no. 16, 2023, ISSN: 1424-8220. DOI: 10.3390/s23167194.

[5] I. Natgunanathan, N. Fernando, S. W. Loke, and C. Weerasuriya, "Bluetooth low energy mesh: Applications, considerations and current state-of-the-art", *Sensors*, vol. 23, no. 4, 2023. DOI: 10.3390/s23041826.

[6] S. M. Darroudi, C. Gomez, and J. Crowcroft, "Bluetooth low energy mesh networks: A standards perspective", *IEEE Communications Magazine*, vol. 58, no. 4, pp. 95–101, 2020. DOI: 10.1109/MCOM.001.1900523.

[7] M. Carlos-Mancilla, E. López-Mellado, and M. Siller, "Wireless sensor networks formation: Approaches and techniques", *Journal of Sensors*, vol. 2016, no. 1, p. 2 081 902, 2016. DOI: https://doi.org/10.1155/2016/2081902.

[8] Y. Liu, K.-F. Tong, X. Qiu, Y. Liu, and X. Ding, "Wireless mesh networks in iot networks", in *2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition*, 2017, pp. 183–185. DOI: 10.1109/iWEM.2017.7968828.

[9] L. Lei, A. Tang, and X. Wang, "Achieving scalable capacity in wireless mesh networks", *Computer Networks*, vol. 253, p. 110 696, 2024, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2024.110696.

[10] R. Rondón, A. Mahmood, S. Grimaldi, and M. Gidlund, "Understanding the performance of bluetooth mesh: Reliability, delay, and scalability analysis", *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2089–2101, 2020. DOI: 10.1109/JIOT.2019.2960248.

[11] *Mesh model*, Rev. 1.1, Bluetooth SIG, Sep. 2023.

[12] *Mesh protocol*, Rev. 1.1, Bluetooth SIG, Sep. 2023.

[13] M. Esposito, A. Belli, L. Palma, S. Raggiunto, M. Mercuri, and P. Pierleoni, "Packet delivery ratio and latency analysis in ble mesh networks: The effect of protocol parameters configuration", in *2024 IEEE Sensors Applications Symposium (SAS)*, 2024, pp. 1–6. DOI: 10.1109/SAS60918.2024.10636398.

[14] P. Pierleoni, A. Gentili, M. Mercuri, A. Belli, R. Garello, and L. Palma, "Performance improvement on reception confirmation messages in bluetooth mesh networks", *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 2056–2070, 2022. DOI: 10.1109/JIOT.2021.3090656.

[15] B. Lin and H. Su, "Experimental evaluation and performance improvement of bluetooth mesh network with broadcast storm", *IEEE Access*, vol. 11, pp. 137 810–137 820, 2023. DOI: 10.1109/ACCESS.2023.3339795.

[16] M. Ghamari, E. Villeneuve, C. Soltanpur, *et al.*, "Detailed examination of a packet collision model for bluetooth low energy advertising mode", *IEEE Access*, vol. 6, pp. 46 066–46 073, 2018. DOI: 10.1109/ACCESS.2018.2866323.

[17] *Bluetooth mesh directed forwarding technical overview*. [Online]. Available: https://www.bluetooth.com/mesh-directed-forwarding/.

[18] *Nrf52840 product specification*, Rev. 1.11, Nordic Semiconductor, Oct. 2024.

[19] *Nrf mesh development tool*. [Online]. Available: https://www.nordicsemi.com/Products/Development-tools/nRF-Mesh.

[20] *Zephyr project documentation*. [Online]. Available: https://docs.zephyrproject.org/latest/index.html.

[21] M. Baert, J. Rossey, A. Shahid, and J. Hoebeke, "The bluetooth mesh standard: An overview and experimental evaluation", *Sensors*, vol. 18, no. 8, 2018. DOI: 10.3390/s18082409.

[22] Á. Hernández-Solana, D. Pérez-Díaz-De-Cerio, M. García-Lozano, A. V. Bardají, and J.-L. Valenzuela, "Bluetooth mesh analysis, issues, and challenges", *IEEE Access*, vol. 8, pp. 53 784–53 800, 2020. DOI: 10.1109/ACCESS.2020.2980795.

[23] S. M. Darroudi, R. Caldera-Sànchez, and C. Gomez, "Bluetooth mesh energy consumption: A model", *Sensors*, vol. 19, no. 5, 2019. DOI: 10.3390/s19051238.

[24] J. Ortiz, J. Silvestre-Blanes, V. Sempere-Paya, and D. Frau, "Evaluation of improvements in ble mesh through coded phy", in *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2021, pp. 1–4. DOI: 10.1109/ETFA45728.2021.9613315.

[25] D. Pérez-Díaz-De-Cerio, Á. Hernández-Solana, M. García-Lozano, A. V. Bardají, and J.-L. Valenzuela, "Speeding up bluetooth mesh", *IEEE Access*, vol. 9, pp. 93 267–93 284, 2021. DOI: 10.1109/ACCESS.2021.3093102.

[26] A. Aijaz, A. Stanoev, D. London, and V. Marot, "Demystifying the performance of bluetooth mesh: Experimental evaluation and optimization", in *2021 Wireless Days (WD)*, Jun. 2021, pp. 1–6. DOI: 10.1109/WD52248.2021.9508308.

[27] J. Silvestre-Blanes, J. Ortiz, and V. Sempere-Payá, "Dynamic autonomous set-up of relays in bluetooth mesh", in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2023, pp. 1–7. DOI: 10.1109/ETFA54631.2023.10275432.

[28] D. Pérez Díaz de Cerio, J. Valenzuela, M. Garcia-Lozano, A. Hernandez-Solana, and A. Valdovinos, "Bmads: Ble mesh asynchronous dynamic scanning", *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, Aug. 2020. DOI: 10.1109/JIOT.2020.3018022.

[29] S. Gautam and S. Kumar, "Low-power ble relay node operation in mesh-like architectures for precision agriculture", *IEEE Sensors Journal*, vol. 24, no. 20, pp. 33 347–33 360, 2024. DOI: 10.1109/JSEN.2024.3451987.

[30] S. S. Basu, M. Baert, and J. Hoebeke, "Qos enabled heterogeneous ble mesh networks", *Journal of Sensor and Actuator Networks*, vol. 10, no. 2, 2021. DOI: 10.3390/jsan10020024.

Appendix A Details on experiment phases

Mode	Param 1	Param 2	Notes
*	TX Power = -20 to +8 dBm		in steps of 4
*	NTC = [1, 2, 4, 8]	NTI = [20, 80, 320]	
*	RRC = [0, 1, 3, 7]	RRI = [20, 80, 320]	
*	PRC = [0, 1, 3, 7]	PRI = [100, 400, 1600]	
M	SRC = [0, 1, 3, 7]	SRI = [50, 200, 800]	SEGS = 18
M	SRC = [0, 1, 3, 7]	SEGS = [2, 6, 18]	
M	SSI = [10, 20, 40, 80, 160]	SEGS = [2, 6, 18]	
U	SRC + SRWPC = [0, 1, 3, 7]	SRI = [50, 200, 800]	SEGS = 18
U	SRC + SRWPC = [0, 1, 3, 7]	SEGS = [2, 6, 18]	
U	SSI = [10, 20, 40, 80, 160]	SEGS = [2, 6, 18]	
U	SRWPC = [0, 1, 3, 7]	SEGS = [2, 6, 18]	SRC = 7
U	SARC = [0, 1, 2, 3]	SEGS = [2, 6, 18]	SST = 0
U	SADI = [1.5, 2.5, 4.5, 8.5]	SEGS = [2, 6, 18]	
U	SRSI = [10, 20, 40, 80, 160]	SEGS = [2, 6, 18]	

Table A.1: Detailed description of phases and used parameter values. Messaging mode: * for unsegmented, M for segmented multicast, and U for segmented unicast. The parameter SEGS refers to the message segment count.