applied sciences

MDPI

Article

Performance Evaluation of an IEEE 802.15.4-Based Thread Network for Efficient Internet of Things Communications in Smart Cities

Sohaib Bin Altaf Khattak [1], Moustafa M. Nasralla [1,*], Haleem Farman [1] and Nikumani Choudhury [2]

[1] Smart Systems Engineering Laboratory, Department of Communications and Networks Engineering, Prince Sultan University, Riyadh 11586, Saudi Arabia; skhattak@psu.edu.sa (S.B.A.K.); hfarman@psu.edu.sa (H.F.)
[2] Department of Computer Science and Information Systems, Birla Institute of Technology and Science Pilani, Hyderabad 333031, India; nikumani@hyderabad.bits-pilani.ac.in
* Correspondence: mnasralla@psu.edu.sa

Abstract: The increasing demand for Internet of Things (IoT) applications has resulted in vast amounts of data, requiring the utilization of big data analytics. The integration of big data analytics in IoT-based smart cities can greatly benefit from the development of wireless communication protocols, among which the Thread protocol has emerged as a promising option. Thread is IEEE 802.15.4 based and has advanced capabilities like mesh networking, IPv6 support, and multiple gateways providing no single point of failure. This paper presents the design and evaluation of a low-cost mesh network using Raspberry Pi, nRF52840 dongle, and OpenThread 1.2 (i.e., an open-source software implementation of the Thread protocol stack). The research elaborates on the hardware and software solutions used, as well as the network topologies adopted. To evaluate the performance of the developed system, extensive real-time tests are performed, considering parameters, such as jitter, packet loss, and round trip time. These tests effectively demonstrate the effectiveness of the Thread network. Furthermore, the impact of varying payload size and bitrate on the network is analyzed to understand its influence. The behavior of the multi-hop network is also examined under link failure scenarios, providing insights into the network's robustness. Our findings provide valuable insights for researchers interested in designing low-cost and efficient mesh networks for various IoT applications, including home automation, building/campus monitoring systems, distributed industrial IoT applications, and smart city infrastructure.

Keywords: ad hoc networks; IEEE 802.15.4; IPv6; internet of things; mesh network; smart cities; thread network; wireless sensor networks

Citation: Khattak, S.B.A.; Nasralla, M.M.; Farman, H.; Choudhury, N. Performance Evaluation of an IEEE 802.15.4-Based Thread Network for Efficient Internet of Things Communications in Smart Cities. *Appl. Sci.* **2023**, *13*, 7745. https://doi.org/10.3390/app13137745

Academic Editors: Muhammad Babar, Saleem Iqbal and Aftab Khan

Received: 7 June 2023
Revised: 24 June 2023
Accepted: 27 June 2023
Published: 30 June 2023

1. Introduction

Advancements in computing technology and networking have given rise to the Internet of Things (IoT), which aims to enable objects to gather and share data over the Internet using smart devices [1,2]. The IoT was envisioned as a large-scale network of Internet-connected objects that are uniquely addressable and reachable using standard networking protocols [3]. It has played a pivotal role in the development of smart cities, where interconnected devices and systems work together to enhance the efficiency, sustainability, and quality of urban life [4]. Some prominent applications include smart energy management systems [5], building controls [6], and smart transportation [7,8], which have gained significant attention from both academia and industry. Many networking protocol stacks have been designed to meet the specific needs of diverse smart city applications and IoT-based systems [9]. Zigbee and Bluetooth are widely adopted protocols in smart city applications. Zigbee is used in various monitoring systems and smart infrastructure [10], while Bluetooth is employed for occupancy detection, asset tracking, and seamless connectivity in smart buildings and transportation systems [11].

The majority of IoT applications in the real world make use of single-hop wireless connectivity to the gateway based on a star topology. Alternatively, the mesh topology has also been considered that enables each node to communicate with every other node and gateway, enhancing reliability [12]. While wireless standards such as Bluetooth and ZigBee are continuously evolving to support mesh networks. To extend the support towards mesh networks, a new protocol named Thread has been developed using open and proven standards, such as IEEE 802.15.4 and 6LoWPAN [13].

The Thread is a communication protocol that is maintained by Thread Group [14]. It has been designed to meet the requirements of applications that demand low power consumption, low data throughput, and a short communication range. Thread protocol is designed to meet the specific needs of diverse smart city applications, providing secure and scalable connectivity to a large number of devices in an energy-efficient and reliable manner. The Thread-based IoT mesh network can be connected to the global Internet using a gateway called a border router that provides connectivity through Ethernet or Wireless Local Area Network (WLAN). Thread provides end-to-end encryption, no single-point failure, and consumes low power. Thread is a relatively new protocol compared to ZigBee and Bluetooth, so its performance has not been extensively studied, especially in real-world applications with varying scenarios.

To fill the aforementioned gap, this work is carried out with the objective to develop a Thread network from off-the-shelf components by analyzing the performance with varying topologies. In order to do so, a test bed was built to analyze the performance by considering one hop, two hops, and a multi-hop environment. In the literature, the performance of the Thread protocol is analyzed by using the Round Trip Time (RTT) parameter, and few have considered throughput, which is insufficient to assess its functionality. In this work, we have performed extended experiments by considering various parameters, such as RTT, packet loss ratio, and jitter. Further, the impact of varying the payload size and bit rate was also considered to analyze the performance in these varying situations. Moreover, in order to check the robustness of a network, the test bed was subjected to link failures. The experiments were performed using Raspberry Pis, manufactured by The Raspberry Pi Foundation based in Cambridge, UK, as nodes and nRF52840 dongles by Nordic Semiconductor, a company based in Oslo, Norway, as a transceiver.

The contributions of this paper are the following:

- Comprehensive performance evaluation of the Thread mesh network protocol using a test bed. While RTT and packet loss performance parameters are commonly examined in the literature on Thread, we have also specifically investigated the impact of jitter, thereby providing a more thorough analysis of the network's behavior.
- A standardized IoT architecture based on Thread mesh technology provides a robust and scalable network for designing IoT solutions.
- Based on the outcomes of the analysis, this paper provides recommendations for selecting the most appropriate topology for different situations.

This paper proceeds as follows. Section 2 discusses the related work available in the literature, and Section 3, provides a comprehensive overview of the Thread protocol and its key features. Section 4 presents a comparison between Thread and other wireless personal area networks. Section 5 discusses the experimental setup and implementation of a mesh network using OpenThread. In Section 6, we evaluate the performance of the implemented model and in Section 7 we provide recommendations as per the findings. Finally, concluding remarks are provided in Section 8.

2. Related Work

In the rapidly evolving landscape of smart cities, the seamless integration of IoT devices has become imperative to enable efficient and reliable communication. Among the various wireless protocols available, the IEEE 802.15.4-based Thread network has emerged as a promising solution for IoT deployments in smart cities.

Recently, several studies have been conducted on various aspects of ad hoc networks [15], including the performance evaluation of IEEE 802.15.4 networks [16]. Silicon Labs, Austin, TX, USA, has conducted a performance analysis of Thread networking standards through an experimental study in their R&D facility [17]. The analysis evaluated the performance aspects of multi-hop RTT latency for unicast, and multicast end-to-end latency on different ranges of networks. Similarly, performance analysis of a large Thread network was conducted by NXP, which focused on unicast and multicast latency utilizing a precision time protocol, considering latency and RTT using Internet Control Message Protocol (ICMP) packets [18]. Silicon Labs also conducted experimental tests on the comparison of Thread, Zigbee, and Bluetooth mesh networking standards, and found that Thread outperformed its peers [19]. All these experimental evaluations mainly relied only on RTT measurements, and the payload size considered was also on a limited scale. Moreover, the performance of the Thread network did not include the varying bitrate.

In [20], authors present the performance evaluation of Thread-based commercial lighting systems. They use key performance indicators such as time for complete coverage, end-to-end latency, packet delivery ratio, and synchronization to analyze the impact on the user experience. The authors use unicast and multicast measurement results in typical lighting applications to analyze their impact on the user experience. In [21], a comparison study of Thread mesh with other widely used wireless protocols for IoT devices, including Bluetooth Mesh, ZigBee, NB-IoT, Sigfox, and LoRa, is provided. The comparisons made are without any empirical performance evaluations, instead based on general characteristics. The study analyzes the requirements for wireless connectivity in smart homes, smart cities, and rural areas. Moreover, it considers the usability of these protocols in the Internet of Things, Services, and People (IoTSP) applications. In [22], authors investigated the potential of Thread protocol in industrial applications, with a focus on the smart factory domain of Industry 4.0. A pack predictive scheduling system for smart factories was proposed using Thread protocol. The performance evaluation was performed using the MCU FRDM-KW41Z board, developed by NXP Semiconductors, Eindhoven, The Netherlands. These papers also describe the hardware and software for implementation and use case scenarios; however, they do not provide any performance evaluation statistics.

Authors in [23] proposed a theoretical system-level model to investigate the Thread network protocol for Proximity Services in building automation and smart homes. The TA testbed was implemented in a real environment to analyze the latency, and to compare the experimental findings with analytical results. They also did not use any other parameters for performance evaluation and limited their analysis to RTT. In [24], authors demonstrate a new approach of using Thread as a wireless communication protocol, instead of Wi-Fi or Bluetooth, in vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) platforms and test it in existing vehicles. This approach greatly reduces the power consumption during the wireless transmission of information. This work does not provide any information on the performance evaluation of the Thread network. In [25], a non-NCP (Network Co-Processor) Thread border router was designed and prototyped based on the OpenThread implementation using Linux system. The experimental results in terms of performance evaluation of this work are limited to RTT. Moreover, ref. [26] also provides an analysis of OpenThread using RTT, regarding several input parameters such as various packet sizes, node distance, interference situations, and test duration. However, no other performance metrics were considered in their evaluations.

Keeping in view the performance of the Thread protocol analyzed in the previous studies, this work extends the experimentation by varying the payload size and bitrate and its impact on jitter, and packet loss ratio. Moreover, to check the robustness of the network, link failures were introduced in the network. A standardized IoT architecture based on Thread mesh technology provides a robust and scalable network for designing IoT solutions. Based on the outcomes of the analysis, this paper provides suggestions for selecting the most appropriate topology for different situations. The details of the related works on Thread performance evaluation are summarized in Table 1.

Table 1. Comparison between performance evaluation in our work and existing literature.

Reference	Parameters	Bitrate Range (kbps)	Payload Size Range (Bytes)	Link Failure
[17]	RTT, packet loss	No	10–300	No
[19]	RTT, packet loss	No	10–300	No
[18]	RTT	No	10–50	No
[20]	RTT, packet loss	No	10–200	No
[23]	RTT	No	10–70	No
[25]	RTT	No	8–1232	No
[26]	RTT	No	64 (fixed)	No
Our Work	RTT, packet loss, Jitter	5 to 250	500 to 5000	Yes

3. Thread Protocol Overview

This section provides an overview of the Thread networking protocol. The Thread is maintained by the Thread Group, which is a working group founded by some of the most prominent names in the technology industry, including Google, Apple, and Qualcomm [13]. The Thread was created to offer connectivity for home automation systems, but now it has received attention from other sectors like remote monitoring and industrial applications [22,23]. We divide this section into two parts; the first part explains the Thread network protocol stack, and the second talks about the topology and types of roles played by the devices based on their functionalities.

3.1. Protocol Stack

The Thread protocol stack is a combination of various standards like IEEE 802.15.4 and 6LoWPAN, etc. [21,25]. It has six different layers, as can be seen in Figure 1. Each of the layers is described below:

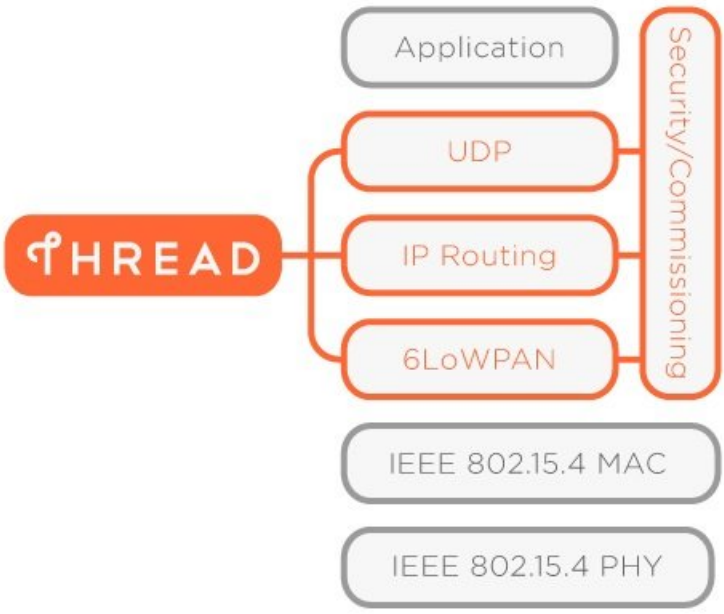

Figure 1. Thread protocol stack [14].

- **IEEE 802.15.4 Physical Layer**
  The physical layer is used for the transmission and reception of packets over a physical medium. The physical and MAC layers of the Thread are defined according to the IEEE 802.15.4 standard. In the physical layer, it operates in the 2.4 GHz free ISM band and is divided into 16 channels numbered from 11 to 26. It offers a data rate of 250 kbps [12].

- **IEEE 802.15.4 MAC Layer**
  This layer is responsible for message handling and management, congestion control, and error correction. The MAC layer includes CSMA-CA (Carrier Sense Multiple Access-Collision Avoidance), allowing Thread devices to utilize the bandwidth efficiently. It has features like beacon management, channel access, Guaranteed Time Slot (GTS) management, etc. [27].
- **6LoWPAN**
  It stands for IPv6 for low-power wireless personal area network (6LoWPAN). It facilitates the Thread network by providing adaptation between the IP layer and the 802.15.4 MAC layer. Packet fragmentation happens here as the packets are assembled into IPv6 format while passing from the MAC layer to the IP layer and vice versa. The layer adapts the bigger frames into smaller ones, making it more suitable for energy-constrained and low-bandwidth devices. 6LoWPAN supports both mesh and star topologies [9].
- **IP Routing**
  As Thread supports the IPv6 architecture, DHCPv6 is used for router address assignment. The IPv6 frames are 40 bytes long and contain an address field of 128 bits, allowing more addresses than the IPv4. In IP routing, the mesh local addresses of routers are maintained in a routing table, ensuring constant connectivity and the availability of paths. Thread uses the distance vector routing protocol, which aims to maximize the amount of routing information in a single message. The routing cost is estimated using the Mesh Link Establishment (MLE) mechanism [14,25].
- **User Datagram Protocol**
  Thread uses User Datagram Protocol (UDP) for messaging between devices. It also supports TCP-based services for application-layer communication. UDP enables messaging with minimum connection overhead, providing a faster and higher throughput of messages. Applications requiring a guarantee may use the TCP protocol, assuring reliability by detecting network congestion and packet acknowledgement and re-transmission [13,14].
- **Application Layer**
  The application layer provides an interface to the network. Thread is application-layer agnostic and provides the flexibility to choose from a variety of application layers to enable device connectivity like Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), etc. [22,24]. Thread, however, provides multicast and UDP messaging as essential application layer services to allow communication over IP.

3.2. Device Types and Topology

Thread network supports a connectivity model where the end devices are only connected to their parent nodes while all the routers are connected together, forming a mesh network. The devices in Thread play different roles based on their functionalities [14,24,25], as shown in Figure 2. In IEEE 802.15.4 networks, the commonly used terminologies of coordinator, router, and end device are prevalent. However, the Thread network architecture introduces specific roles and functionalities that bring added clarity and facilitate customized network design for IoT applications.

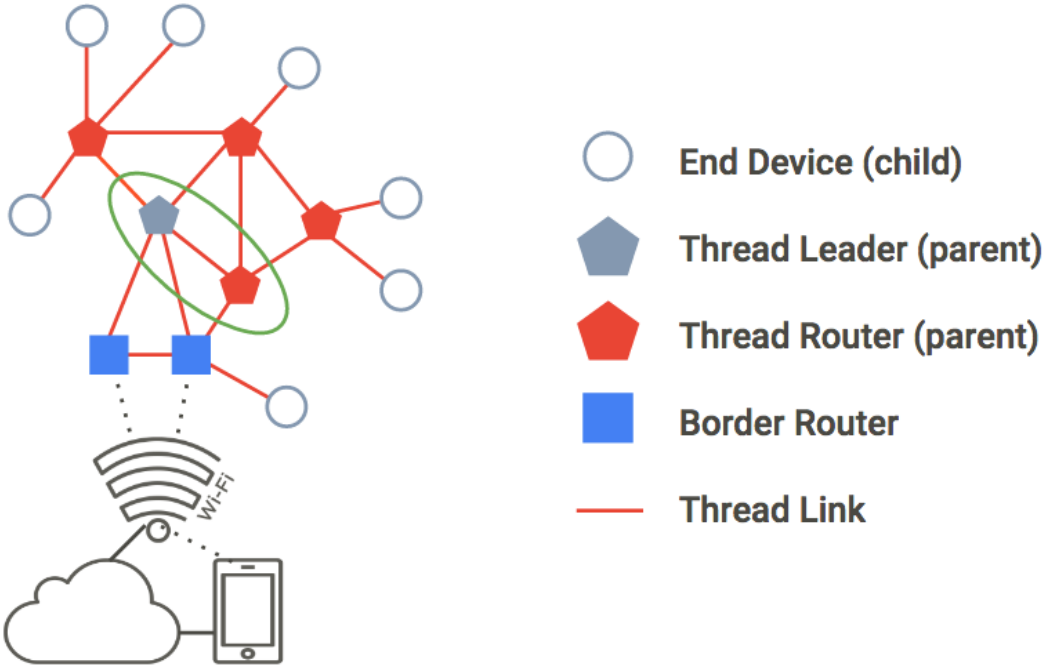

Figure 2. Basic topology and device types of Thread network [14].

- **End Device**
  These devices have reduced functionalities and are energy-constrained, similar to their counterparts in other wireless protocols like ZigBee and Bluetooth. However, Thread provides specific features to optimize energy consumption, such as allowing end devices to enter sleep mode to conserve energy. Unlike some other protocols, Thread end devices are always connected to their parent node, which has routing capabilities and remains continuously active. Thread also introduces the concept of Router Eligible End Devices (REEDs) that can be promoted to routers for improved performance. The dynamic up-gradation and down-gradation of devices is managed both explicitly by the network manager and automatically by the network based on prevailing conditions.
- **Router**
  Thread routers share similarities with routers in other protocols, such as ZigBee and Bluetooth, in terms of their routing capabilities. These devices require power throughout and are also called Full Thread Devices (FTD). All routers in the network are connected together, forming a mesh, and route the data packets from source to destination nodes. Additionally, Thread routers also provide joining and commissioning services to the end nodes. These nodes are also capable of becoming border routers or leader routers, and if a leader goes down due to some reason, these nodes can be automatically elected as leaders by the network.
- **Border Router**
  This router connects the Thread network with the outside world and enables data flow in and out of the network. It can also be termed the gateway of the Thread network to the global internet or cloud. It can support both WiFi and Ethernet for connection to an external network. Unlike other networks having a single gateway, in Thread multiple border routers can also exist in a network, thus avoiding a single point of failure.
- **Leader**
  Any router or border router that manages the Thread network is called a leader. It has all the functionalities of the router but also has additional responsibilities like assigning IP, upgrading or downgrading the router status of REEDs, and distributing the network configuration information to the routers in the network. Unlike other wireless protocols, such as Bluetooth and ZigBee, which may have central devices or

coordinators, if the Thread leader goes down, a router is elected as a leader by other thread routers to perform network management tasks.

4. Thread vs. Competitors

Several wireless communication technologies are used for IoT deployment, depending upon the application requirements [9,12]. These communication technologies can also be classified as long and short-range. The most commonly used short-range communication technologies are, Wireless Fidelity (Wi-Fi), ZigBee, and Bluetooth. The common features among most applications are low cost, low processing, low power, low storage, and a low bit rate.

Wi-Fi is a wireless protocol from the IEEE 802.11 family that operates over unlicensed bands of 2.4 GHz and 5 GHz. Its basic purpose was to replace wired Ethernet with wireless. Wi-Fi is a strong choice for IoT deployments, but due to its higher power consumption, is not suitable for power-constrained and battery-powered IoT networks [28]. On the other side, communication technologies like Bluetooth, Zigbee, and Thread require considerably low power and are more practical for applications like smart cities, health care, home automation, etc. [29,30]. These applications are not bandwidth-hungry and require a small number of sensors to provide some parametric data; hence, low-bandwidth technologies are suitable for such systems. These three short-range communication technologies have a common characteristic, they all use the 2.4 GHz ISM band. Zigbee is particularly developed to fulfill the needs of machine-to-machine and IoT networks, providing low-cost and low-power wireless connectivity, while Bluetooth is short-range wireless and very low-power but a high-bandwidth connectivity option [12]. As mentioned earlier, the Thread protocol connects different existing technologies like IEEE 802.15.4, 6LoWPAN, and IETF IPv6 while providing mesh networking and distinguishing itself from its peers. Some of its distinguishing characteristics are described below

- **IPv6 support (highly scalable in terms of connectivity, compatible with existing IP-based systems)**
  Resource-constrained devices can achieve device-device, device-mobile, and device-cloud communication via IPv6. It can employ several IP-based concurrent applications and ecosystems. End-to-end IP support facilitates customized applications and services. It also unifies convergence across PHY/MAC networks [31].
- **Highly resilient (auto-configuring and self-healing, no central hub)**
  The network automatically adapts to changing conditions. The number of routers in the network can be increased or decreased as REEDs can act both ways. Thread is robust and has a self-healing mesh network. Every Thread network has a leader node that is responsible for network management, in the event of failure of this leader node, another router is elected as a leader without compromising the network performance [14].
- **Reliable (mesh network, no single point of failure)**
  Thread is a mesh network where each router node is connected to all available router nodes. If a connection breaks down between any two nodes, these nodes can still communicate as the network reroutes the traffic through another route. Another distinguishing feature of the Thread network is that it can support multiple border routers. As described before, these border routers provide connectivity from local mesh to the global internet, so multiple border routers guarantee connectivity to the internet and make the network more reliable with no single point of failure [20,24].
- **Scalable (suitable for small and large networks)**
  Thread supports 32 routers, and each router can connect up to 511 devices, reaching over 16 thousand devices. Multiple border routers also make it more scalable [20]. The Thread has 15 channels for radio communication, which can help achieve smooth performance in situations with high node density.
- **Secure (AES encryption, and secure commissioning)**
  Thread employs link-layer security, encrypts all network communication, and only permits authorized nodes to connect to the network. Datagram Transport Layer Security (DTLS) encryption technology is used for authentication in order to prevent tampering

and message forging. Depending on the type of program being used and the type of end device, further application layer-level security may also be applied [14].

- **Low latency**
  The Thread connects devices directly in a mesh, which means lower latency. Nodes that are not routers and are accessed through parent nodes can also be reached efficiently, as the network is always looking for the best route, which translates into latency. Tests conducted by Silicon Labs have shown that Thread outperforms Zigbee and Bluetooth in terms of latency [19,23].
- **Application-layer agnostic**
  The Thread protocol is application layer agnostic, which means there is no defined application layer. This enables Thread products to choose freely between available application layers and can support many IP-enabled application protocols [22,24].
- **Low power consumption and low cost**
  The 6LoWPAN reduces transmission overhead using header compression and link layer packet forwarding for multi-hop packets [31]. The Thread also supports sleepy end devices, which keep their radios off and only work when an event occurs [22]. In routing, Thread uses the best route possible, resulting in low power consumption.
- **Open standard and Easy setup**
  There is also an open-source Thread stack implementation called OpenThread. It simplifies application development, enabling the quick deployment of Thread-based products to the market [21,31]. Thread stack is an open standard that is based on existing IEEE and IETF standards and is thus easily understandable by industry and academia.

Thread's IPv6 support, self-healing mesh network, scalability, low latency, low power consumption, open standard, and easy setup make it an ideal choice for smart city applications. Its unique combination of features ensures reliable, efficient, and compatible communication across a large number of devices and systems.

A comparative analysis of the features provided by various existing wireless communication protocols is presented in Table 2. The information about these standards is derived from [12].

Table 2. Comparison between various communication protocols used for short range.

Specification	Wi-Fi	Zigbee	Bluetooth	Thread
IEEE standard	802.11	802.15.4	802.15.1	802.15.4
Power consumption	High	Low	Low	Low
Frequency band	2.4 GHz/5 GHz	2.4 GHz, 868 MHz, 915 MHz,	2.4 GHz	2.4 GHz
Coverage	Normally 10–40 m	Normally 10–100 m	Normally 10–100 m	Normally 10–30 m
Bandwidth	1Gbps	250 kbps	1 Mbps	250 kbps
IP support	Yes	No	No	yes
Cloud integration	Router	Gateway	Gateway	Border router
Network topology	Star	Mesh	Star/Mesh	Mesh
Modulation	OFDM	O-QPSK	GFSK	O-QPSK

5. System Model and Experimental Setup

This section presents the technical details of the test bed design, such as the hardware specifications, the environment, and the architecture of the test bed. The test bed comprises Thread nodes forming a wireless mesh network that can be remotely controlled and monitored over an Ethernet. The Thread sensor nodes are developed using custom-built hardware devices that consist of a Raspberry Pi (RPi) 4 model B as the main processing unit, and a Nordic semiconductor nRF52840 dongle as an IEEE 802.15.4 radio transceiver. The RPi functions as a router or a Thread device while the nRF52840 dongle is responsible for data transmission in the network. In addition to Thread, this dongle has support

for Bluetooth and Zigbee [32]. A Windows-based application named "nRF Connect for Desktop" can be used to configure the dongle by flashing a binary hex file.

The Thread stack is fully implemented on our testbed. The dongle provides the PHY and MAC layers, while the host device is a RPi having Raspbian OS, providing the rest of the layers. The 6LoWPAN runs over the RPi, and it handles IPv6 packet fragmentation between the layers. Power banks are used to power the Thread devices, as depicted in Figure 3. Google Nest's OpenThread is used for implementation. OpenThread is an open-source implementation of the Thread protocol that offers all Thread network functionalities and has Git repository access [33]. OpenThread provides a command line interface (CLI) to configure the individual nodes and the network.

The RPi was installed with Raspbian Operating System while OpenThread repositories and OpenThread border router (OTBR) settings were installed from Github [34]. To make the Thread device functional, we need to flash the Nordic dongle, for which relevant OpenThread binaries are required to be generated through the OpenThread repositories. After installing OTBR and flashing the dongle, we verify the services and configure the RPi through CLI commands to build the Thread network. When the first node is configured, it will initialize itself as the leader and start creating the Thread network. The subsequent device will follow the same steps and join the existing Thread network. Once the network is formed, these nodes can communicate with each other using ping and UDP.

Figure 3. Thread node based on Raspberry Pi 4B, Nordic nRF52840 dongle, powered by a power bank.

The test bed was set up in an office environment with various physical obstacles such as concrete walls, furniture, and human occupancy. The office area also had other wireless communication sources, such as WLAN access points and other Bluetooth devices, in the background which makes the testbed more realistic. The Thread network had a maximum capacity of 32 active routers, and router-eligible end devices (REEDs) could be promoted to routers if the network had fewer than 32 routers. The role of REEDs can be adjusted in the network according to the requirements. They can become child nodes and delegate the routing tasks to their parent nodes. We evaluated the network performance by considering three different topologies.

In Topology 1, we considered two thread devices, a leader and a router, as depicted in Figure 4. The network was extended in Topology 2, where a child node was added to the

network as shown in Figure 5. The child node could only communicate with the leader through the router and has no direct link between them. This scenario was designed to test the Thread network in situations where a child node had to relay data to the leader node via an intermediate device. To further evaluate the network performance we considered a network having complex topology in terms of mesh network. In Topology 3 we have three child nodes, three routers, and one leader, as shown in Figure 6. These typologies and roles of individual nodes are verified on the OpenThread web interface, as shown in Figures 7–9. It provides a graphical user interface that enables users to configure various parameters.

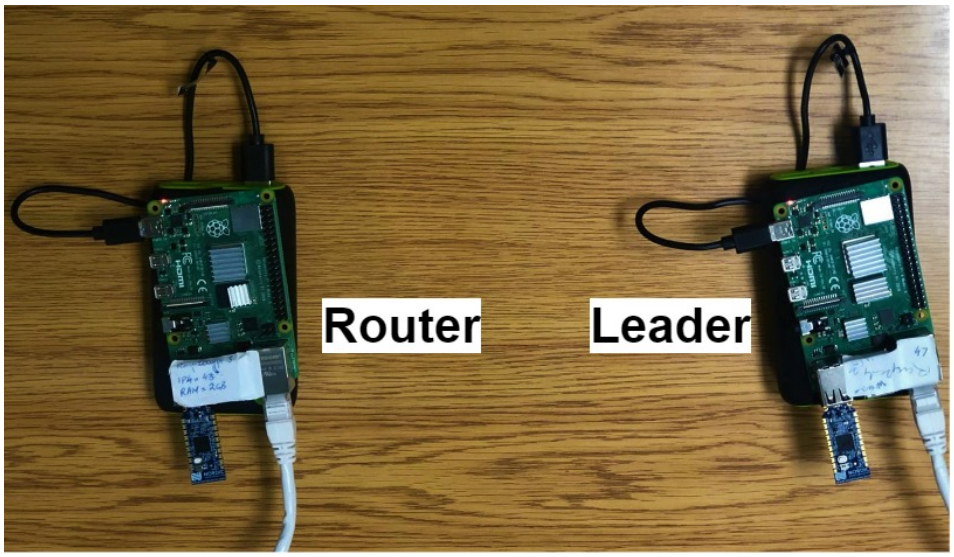

Figure 4. Topology 1 setup.

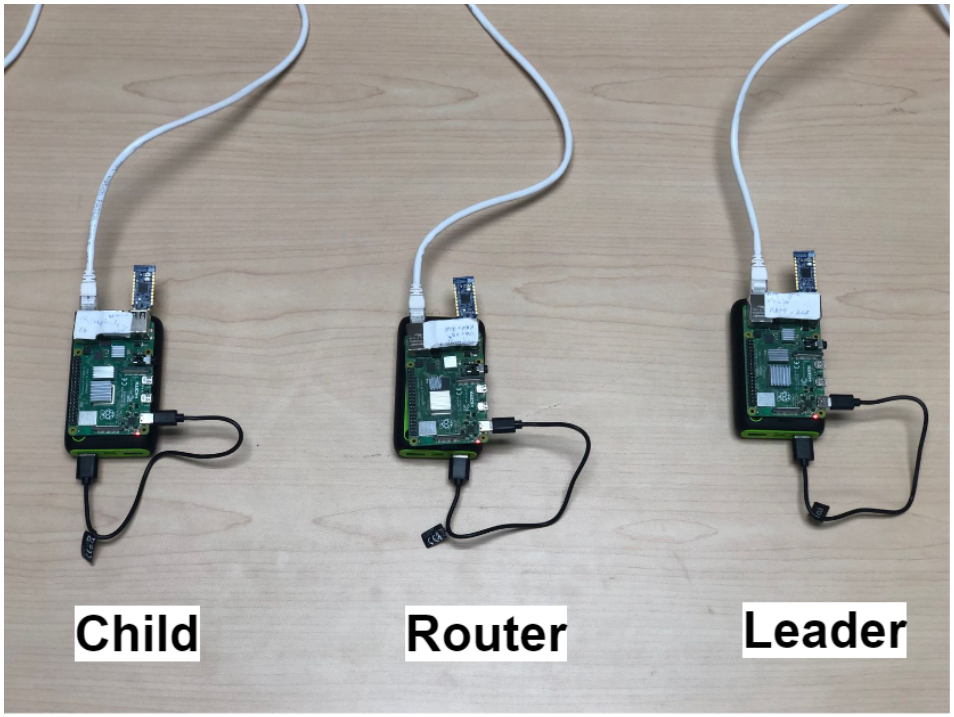

Figure 5. Topology 2 setup.

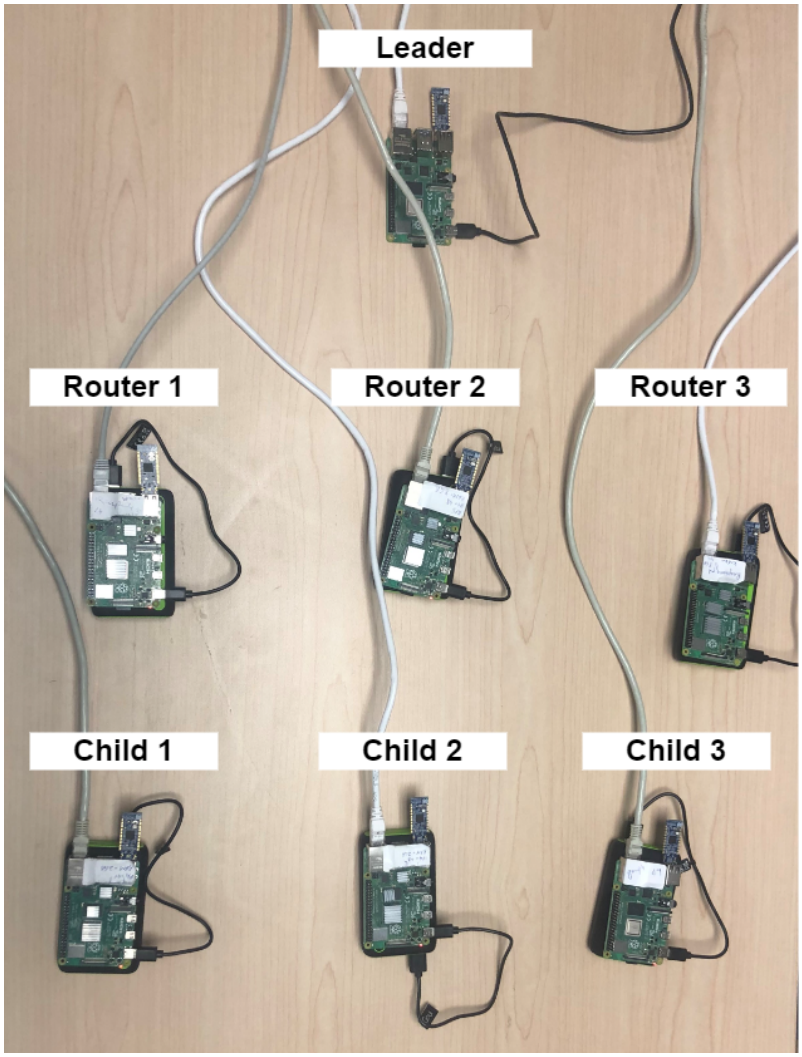

Figure 6. Topology 3 setup.

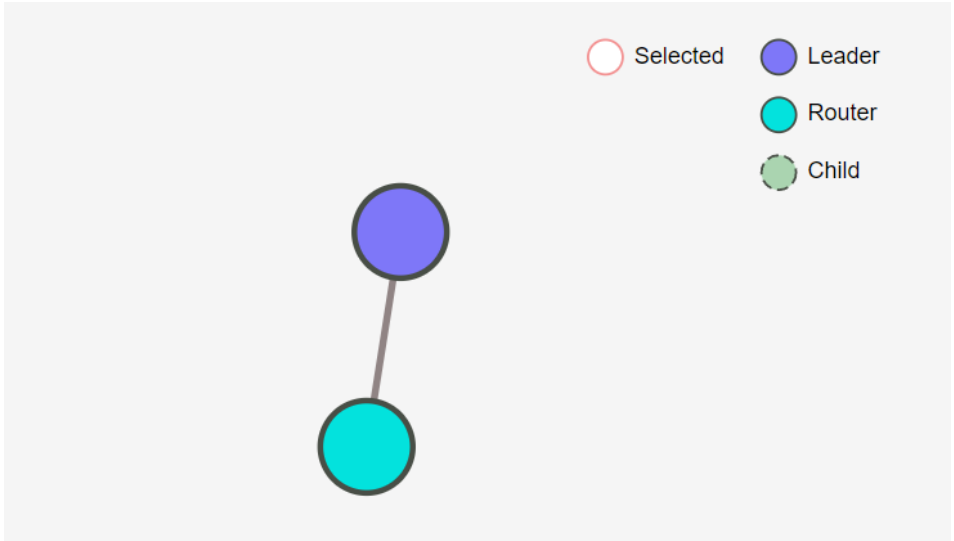

Figure 7. Topology 1: OpenThread web interface.

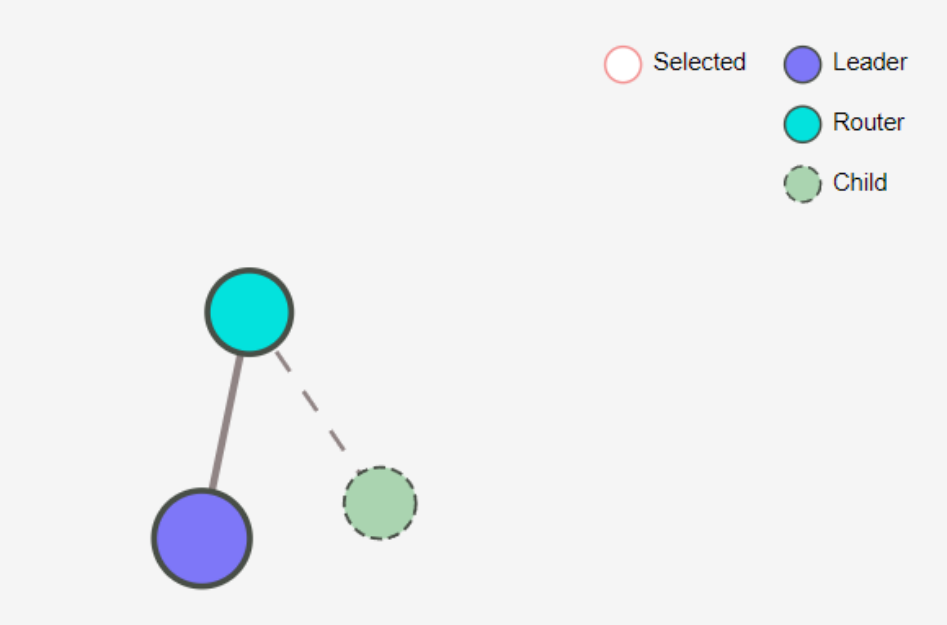

Figure 8. Topology 2 : OpenThread web interface.

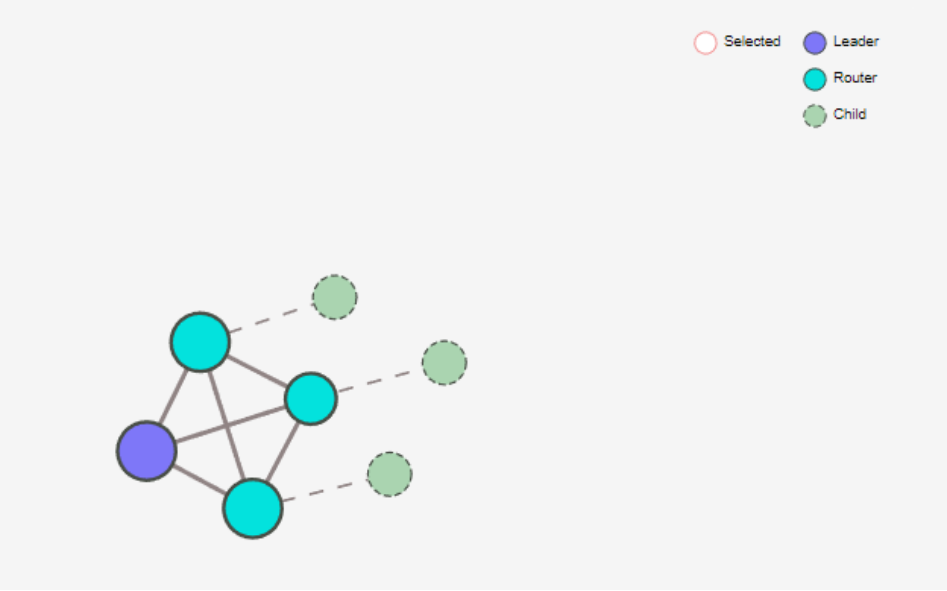

Figure 9. Topology 3: OpenThread web interface.

6. Performance Evaluation

The performance of the Thread network was analyzed by conducting different experiments using the iperf testing tool. To measure the network's performance, iperf uses a server and client structure. This tool can provide detailed information about the network's behavior. The metrics used for performance evaluation are as follows:

- **Jitter**: It is the variation in time for a packet to travel across a network and is usually measured in milliseconds (ms). A high jitter negatively impacts network performance and causes irregularities in data transfers. This degraded quality of service is highly undesirable, especially for real-time applications, such as voice and video communications, where a consistent and predictable delay is crucial.
- **Packet loss ratio**: Furthermore, known as packet drop ratio, represents the number of lost packets to the total number of packets sent. It represents the percentage of packets that did not reach their intended destination due to various network issues. It is an

important metric to evaluate the reliability and performance of a network. Its higher percentage means loss of data and reflects degraded network performance.

- **Round Trip Time**: It refers to the time taken for a packet to travel from a source to a destination and then back to the source, measured in milliseconds (ms). It is considered an important metric in networking to measure the quality of a network and is commonly used to diagnose the reliability of the network. A long RTT in a network is undesirable and can negatively impact the performance of real-time applications.

While jitter is often associated with RTT and packet loss, in this paper, we treat it as an independent parameter to analyze its impact on the network performance. It is important to note that their behavior is interrelated. When network delays are stable and consistent, the RTT tends to be low, indicating efficient packet delivery and low jitter. Conversely, if the network delays exhibit variability and fluctuations, the RTT will be high, leading to increased jitter. High jitter, in turn, can elevate the probability of packet loss. Consequently, understanding and analyzing jitter as a separate parameter enables us to gain insights into the network's stability, consistency, and overall quality of service.

We conducted different experiments for UDP traffic, where each experiment had a different scenario for varying bitrate and payload size. The details of the experimental scenarios can be found below:

- **Varying Network Topology**
  To assess the scalability of a network and gauge its performance as it expands, we generate diverse network topologies by modifying the number of devices and their assigned roles. This enables us to examine the network's capacity for growth and measure the effects on the network's performance after changing the roles of individual devices.
- **Varying Payload Size**
  We conduct a network performance analysis by varying the payload size. This enables us to measure the jitter and packet loss ratio against each communication.
- **Varying Bitrate**
  We conduct a performance analysis of the network by varying the bitrate, enabling us to observe and evaluate the network's performance under these modified conditions.
- **Link Failure**
  To check the robustness of the network, link failure was introduced between a router and a leader. Traffic was generated in the topology to check the network response to link failure.

In our experiments, we have configured the network by varying the bitrate and payload size. The UDP packet size is 1208 bytes by default and 250 kbps is the theoretical maximum data rate provided by Thread, however, we conducted experiments by systematically varying the metrics, as presented in Table 3. This section is divided into several subsections, each discussing specific experiments that aim to analyze and evaluate different aspects of network performance under various scenarios. The first experiment identifies the network bottleneck for each topology. The next two experiments examine the impact of varying bitrate and payload size on jitter and packet loss for all three topologies. The fourth experiment focuses on the network's response to link failure and finally the fifth experiment investigates the RTT for all topologies.

Table 3. Details of the performance evaluation parameters.

Performance Metric	Details
Topology	3 topologies (2 nodes, 3 nodes, 7 nodes)
Bitrate range	5–250 kbps
Payload range	500–5000 Bytes

6.1. Experiment 1: Bottleneck Scenario

In the context of computer networks, a bottleneck refers to a situation where the data flow experiences degradation, either in terms of delays or packet loss [35]. It occurs when the system delivers more data than the existing capacity of the network. Identifying network bottlenecks can prevent traffic congestion and significantly improve network performance.

The network's performance and potential bottlenecks were analyzed through comprehensive testing on all three topologies. We performed the tests for various payload sizes by varying the bitrates, and the results showed that in Topology 1, at all payload sizes (i.e., 500–5000 bytes), without packet loss communication is observed with bitrates lower than 100 kbps, which can be seen in Figure 10. While for Topology 2, where the child node communicates with the leader node through a router, this level drops to 40 kbps, as shown in Figure 11. It indicates that the network's performance is highly dependent on the network topology; direct communication between two routers experiences less packet loss and can be more suitable for certain scenarios. This experiment helps to understand the network's performance and highlights the need for proper network design according to the requirements of the scenario.

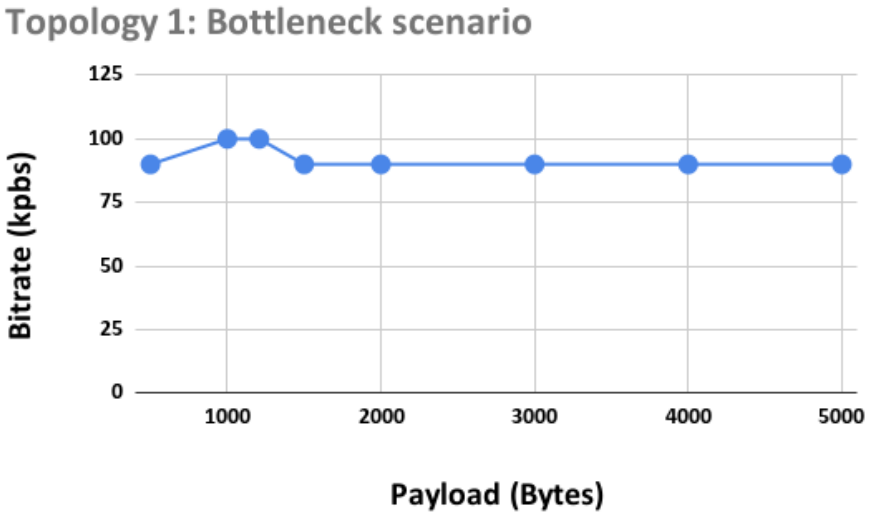

Figure 10. Bottleneck for Topology 1.

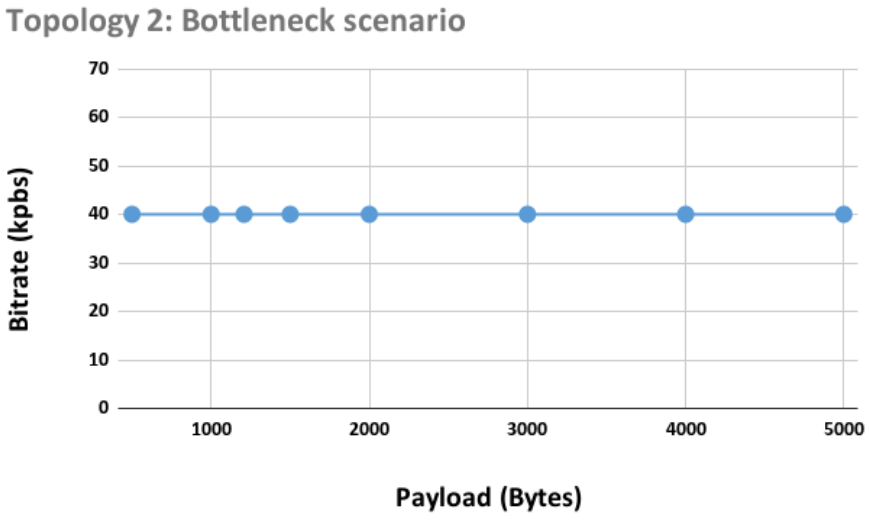

Figure 11. Bottleneck for Topology 2.

Similarly, this experiment was repeated for Topology 3 for two different scenarios. In the first scenario, child-to-leader communication is observed, where all three nodes are sending data to the leader node, making the network congested. Figure 12 shows that in such a congested environment the bottleneck drops to 10 kbps. This drop occurs because due to high congestion and high bitrate, the packet loss will be more frequent. Similarly, on the same topology, another communication is also observed where child-to-child communication is observed. In this scenario, the communication takes place over three hops, as shown in Figure 13. It is worth noting that the bottleneck occurred at 30 kbps. In this particular scenario, the network environment was not congested as other nodes did not participate in any communication. It is evident that the bottleneck condition occurs beyond that of the congested communication scenario, despite the two-hop communication. However, it still exhibited lower performance compared to the two-hop communication scenario employed in Topology 2.

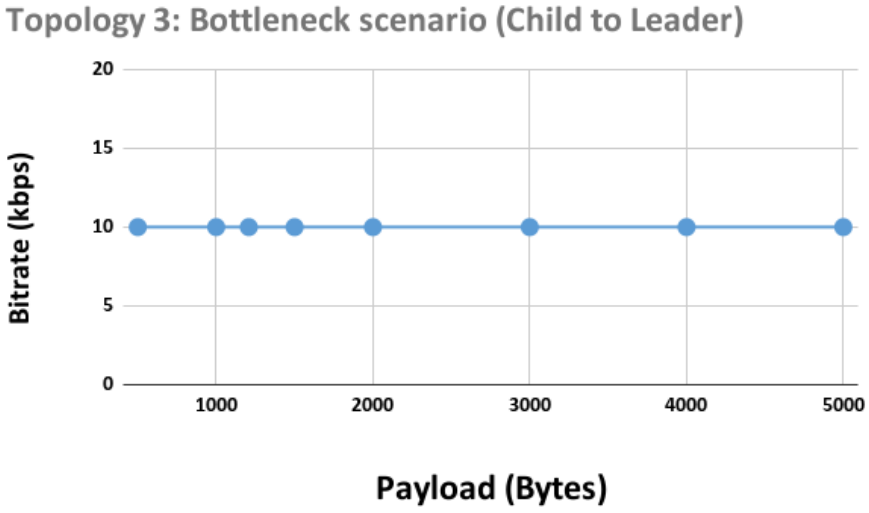

Figure 12. Bottleneck for Topology 3 (Child to Leader).

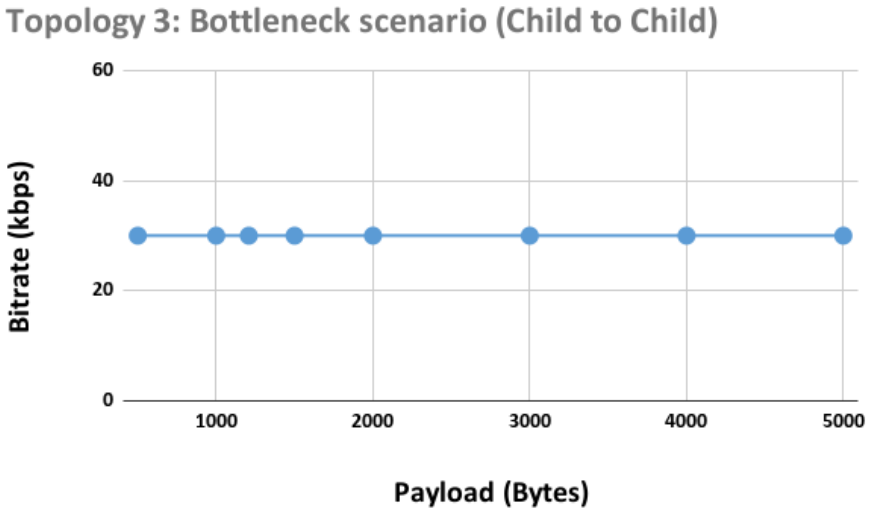

Figure 13. Bottleneck for Topology 3 (Child to Child).

6.2. Experiment 2: Jitter with Increasing Bitrate and Payload Size

In this experiment, we varied the bitrate and payload size in the network and evaluated its impact on the jitter. Increasing the bitrate and payload size caused an increase in jitter, causing a significant impact on the network's performance. This can be seen in

Figures 14 and 15 in detail for Topology 1 and Topology 2, respectively. This experiment highlights the importance of carefully selecting the bitrate and selecting an appropriate payload size to avoid excessive jitter in the network. The jitter increases with the increase in bitrate as the number of datagrams being sent also increases, making the network congested. After a certain level of increase in the bitrate, the jitter starts to decrease. This decrease in jitter is due to high packet loss, as the number of datagrams exceeds the capacity of the network.

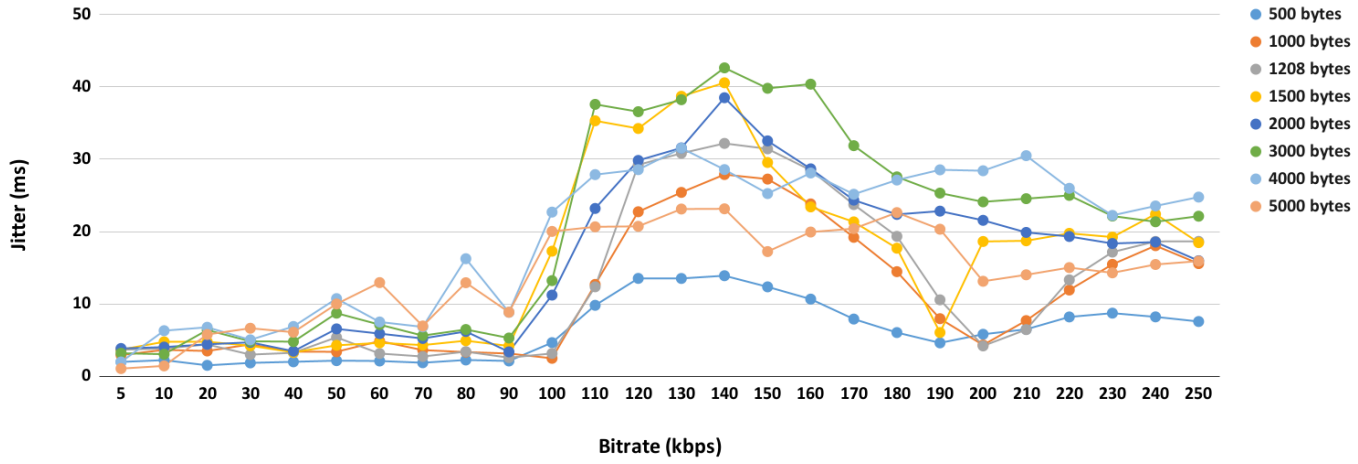

Figure 14. Toplogy 1: Jitter with increasing bitrate and payload size.

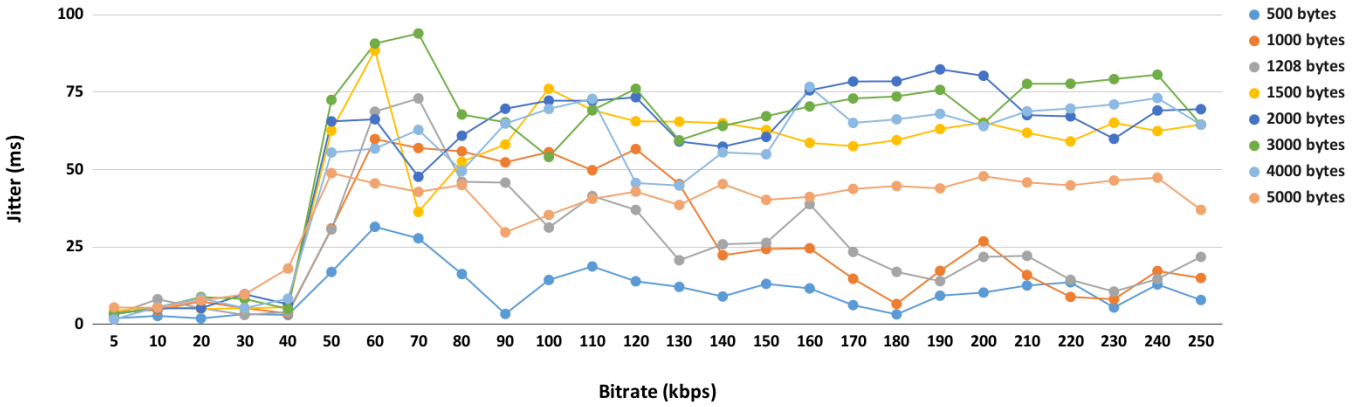

Figure 15. Toplogy 2: Jitter with increasing bitrate and payload size.

As can be seen in Figure 14, overall in Topology 1, the optimal performance in terms of jitter is achieved with a packet size of 1000 bytes with a 100 kbps bitrate and 2.507 ms jitter. It could transfer the data with 0% packet loss as 100 kbps is before the bottleneck occurs. It can be observed from the results as shown in Figure 15 when the number of hops increases, there can be a significant increase in jitter. Both the topologies represent a similar trend, as the bitrate increases, the number of datagrams also increases, which leads to an increase in jitter. The optimal performance in Topology 2 is also achieved by a 1000-byte packet size with a 40 kbps bitrate. It could transfer data with 0% packet loss as the bitrate is not beyond the bottleneck scenario and 3.527 ms jitter.

The detailed representation of the experiments performed for evaluation of the Thread network on varying the bitrate can be seen in Table 4. The results in Table 4 are generated from Topology 1 where a router node communicates with the leader node, for the default packet size of 1208 bytes in UDP transmission. It can be seen in the table that there are no packet losses till the provided bitrate reaches the bottleneck of 100 kbps. We can also see that with the increasing bitrate, the number of datagrams increases leading to congestion

and ultimately packet losses. The same experiment is repeated for Topologies 2 and 3 to check the network performance in terms of jitter and packet loss with different packet sizes.

Table 4. Detailed performance analysis of the network with 1208 byte packet size and varying bitrate for Topology 1.

Bitrate (kbps)	Bitrate at Receiver (kbps)	Data Transferred (KBytes)	Data Received (KBytes)	Datagrams/s	Lost/Total Datagrams in 30 s (%)	Jitter (ms)
5	5	18.9	18.9	0–1	0/16 (0%)	3.694
10	10	37.8	37.8	1	0/32 (0%)	3.861
20	20	74.3	74.3	2–3	0/63 (0%)	4.424
30	30	111	111	3–4	0/94 (0%)	3.02
40	40	147	147	4–5	0/125 (0%)	3.269
50	50	184	184	5–6	0/156 (0%)	5.405
60	60	221	221	6–7	0/187 (0%)	3.167
70	70	257	257	7–8	0/218 (0%)	2.758
80	80	294	294	8–9	0/249 (0%)	3.427
90	90	330	330	9–10	0/280 (0%)	2.597
100	100	367	367	10–11	0/311 (0%)	3.180
110	101	403	380	11–12	20/342 (5.8%)	12.383
120	99.1	440	373	12–13	57/373 (15%)	29.222
130	101	477	379	13–14	83/404 (21%)	30.814
140	100	513	379	14–15	114/435 (26%)	32.180
150	101	550	379	15–16	144/465 (31%)	31.43
160	101	586	379	16–17	175/496 (35%)	28.481
170	99.9	623	376	17–18	209/528 (40%)	23.737
180	100	659	376	18–19	240/559 (43%)	19.349
190	99.9	696	376	20–19	271/590 (46%)	10.582
200	101	733	380	20–21	299/621 (48%)	4.219
210	100	769	378	21–22	332/652 (51%)	6.471
220	100	806	376	23–22	364/683 (53%)	13.338
230	97.3	842	366	24–23	402/712 (56%)	17.172
240	100	880	378	25	425/745 (57%)	18.644
250	101	917	379	26–25	455/776 (59%)	18.653

The impact of varying bitrate and payload on jitter has also been observed on Topology 3, in both child-to-leader with network congestion and child-to-child communication scenarios. The network performance of these scenarios can be seen in Tables 5 and 6, respectively. It can be clearly seen that the jitter is much higher in the network congestion scenario of two hops communication, as compared to three hops communication. The "0 ms" jitter values in Table 5 indicate that no contact was made. This happens due to high bitrate, as the network becomes even more congested and communication becomes impossible.

Table 5. Impact of increasing bitrate and payload size on Jitter for Topology 3, child to leader communication scenario.

Bitrate (kbps)	Jitter (ms)							
	500 Bytes	1000 Bytes	1208 Bytes	1500 Bytes	2000 Bytes	3000 Bytes	4000 Bytes	5000 Bytes
10	75.195	61.541	120.082	84.122	70.342	72.946	66.989	69.525

Table 5. *Cont.*

Bitrate (kbps)	Jitter (ms)							
	500 Bytes	1000 Bytes	1208 Bytes	1500 Bytes	2000 Bytes	3000 Bytes	4000 Bytes	5000 Bytes
40	95.888	134.741	117.504	208.674	223.109	214.725	142.883	0
50	90.146	143.145	151.893	249.356	219.242	228.315	191.704	0
100	78.488	115.648	133.618	241.004	206.343	160.182	173.23	0
150	80.819	104.642	125.031	249.36	216.368	203.016	138.337	0
200	86.399	138.047	178.823	270.122	228.178	234.024	138.879	0
250	88.512	113.075	141.16	250.633	215.769	0	0	0

Table 6. Impact of increasing bitrate and payload size on Jitter for Topology 3, child-to-child communication scenario.

Bitrate (kbps)	Jitter (ms)							
	500 Bytes	1000 Bytes	1208 Bytes	1500 Bytes	2000 Bytes	3000 Bytes	4000 Bytes	5000 Bytes
10	13.556	23.152	20.476	15.984	15.452	22.931	20.516	16.034
30	16.063	29.477	29.517	18.706	22.888	21.814	43.364	37.447
50	58.153	92.737	95.823	57.955	89.803	111.824	88.274	61.679
100	21.978	102.889	88.78	92.665	99.548	96.881	44.413	58.958
150	27.559	114.324	47.554	100.601	113.415	109.067	53.171	63.605
200	25.147	49.905	28.365	78.821	103.424	84.324	57.556	73.589
250	22.653	56.107	59.016	55.98	65.889	46.945	59.008	73.88

6.3. Experiment 3: Packet Loss with Increasing Bitrate and Payload Size

In this experiment, we observed packet loss in the network across different bitrates and payload sizes. The results showed that increasing the bitrate and payload caused an increase in packet loss, significantly impacting the network's performance. The results for Topologies 1 and 2 can be seen in Figures 16 and 17, while the results of two scenarios of Topology 3 are reflected in Tables 7 and 8. This is because of the same reason, when the bitrate is increased, more datagrams are transmitted, which leads to congestion and, eventually a high packet loss. This experiment highlights the importance of carefully selecting the bitrate and payload size keeping in view the use case scenario, to avoid excessive packet loss in the network. It has been further discussed in Section 7 in detail.

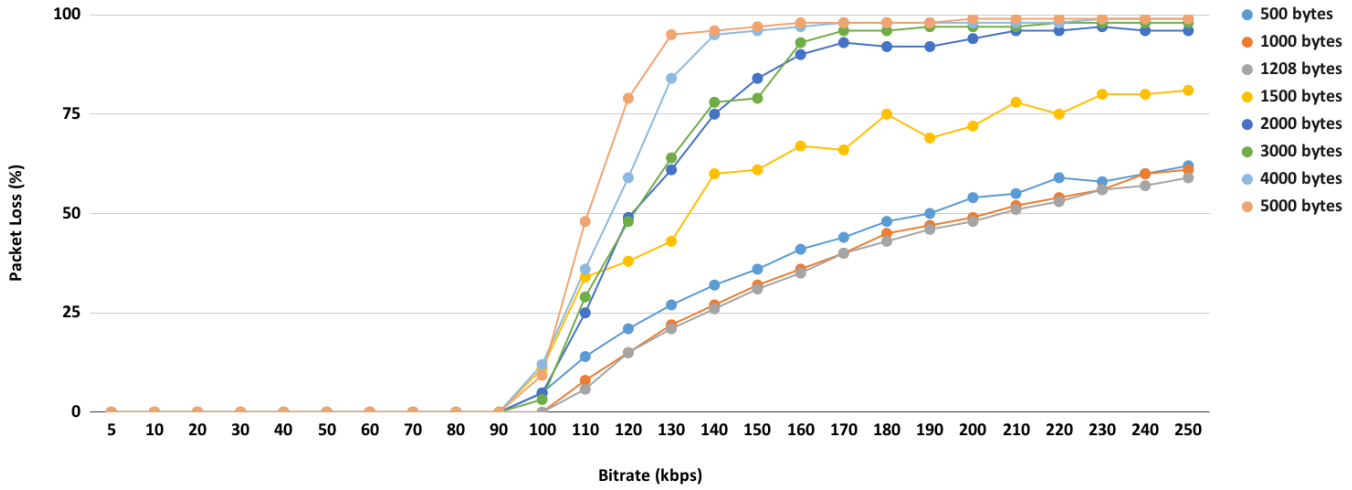

Figure 16. Topology 1: Packet loss with increasing bitrate and payload size.

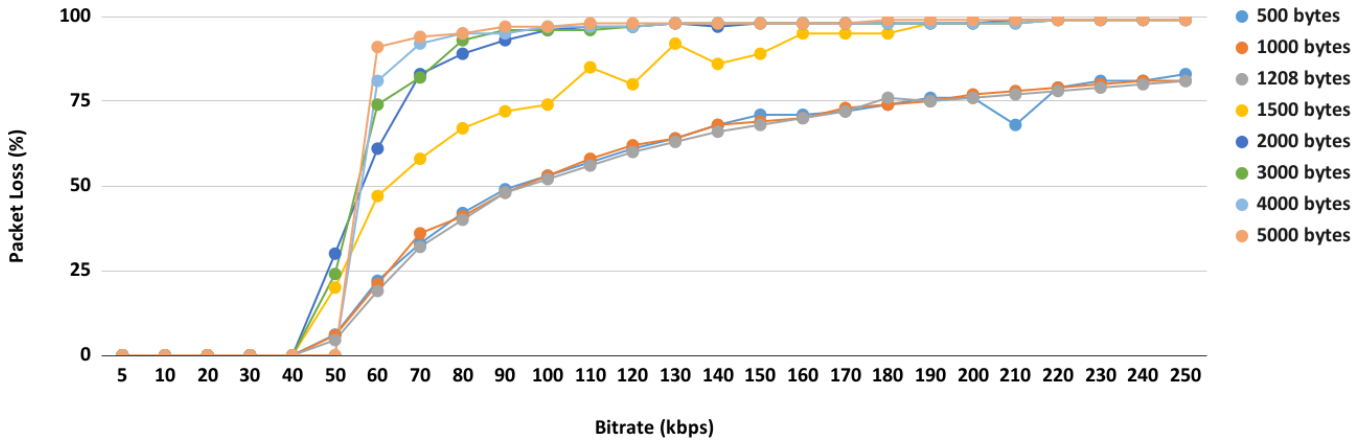

Figure 17. Toplogy 2: Packet loss with increasing bitrate and payload size.

Table 7 shows that due to high congestion, there is an increase in packet loss which can even cause a complete 100% packet loss. The results in this table can also be matched with that of Table 5, where we observed a "0 ms" jitter due to no communication. For the child-to-child communication scenario, it can be seen in Table 8 that till 30 kbps bitrate, we can achieve communication with zero packet loss, but as the bitrate and payload is increased, we observed high packet losses.

Table 7. Impact of increasing bitrate and payload size on packet loss ratio for Topology 3, child-to-leader communication scenario.

Bitrate (kbps)	Packet Loss Ratio (%)							
	500 Bytes	1000 Bytes	1208 Bytes	1500 Bytes	2000 Bytes	3000 Bytes	4000 Bytes	5000 Bytes
10	0	0	0	0	0	0	0	0
40	56	54	56	78	88	90	92	100
50	65	63	60	86	94	94	96	100
100	83	82	82	96	97	98	97	100
150	88	88	87	98	98	98	98.5	100
200	90	91	92	99	99	99	99	100
250	93	93	92	99	99	100	100	100

Table 8. Impact of increasing bitrate and payload size on packet loss ratio for Topology 3, child-to-child communication scenario.

Bitrate (kbps)	Packet Loss Ratio (%)							
	500 Bytes	1000 Bytes	1208 Bytes	1500 Bytes	2000 Bytes	3000 Bytes	4000 Bytes	5000 Bytes
10	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0
50	42	36	33	58	78	86	89	92
100	67	71	68	84	97	97	98	97
150	78	79	78	96	98	98	98	98
200	84	84	83	99	99	99	98	99
250	88	87	87	99	99	99	99	99

6.4. Experiment 4: Link Failure Scenario for Topology 3

In this experiment, we also analyzed the response of the Thread network to link failure. We performed this experiment over Topology 3, where the connection between Router 1 and Leader was broken. In a network that is not based on mesh, this link failure would leave Child 1 and Router 1 isolated. As Thread is a mesh-based topology, it routes the traffic from Child 1 through Router 1 to another router as per Thread's routing algorithm. This makes Thread more reliable and resilient, providing a better choice for sensitive applications which can not tolerate link breakage. Figure 18 shows the network performance when the communication is set up between Child 1 and the Leader under both normal and link failure conditions. We can see that the jitter is increased in the link failure scenario as the traffic has to be routed through the network taking a longer route than its original path. We observed that the jitter decreased as the payload size increased, this happens because, with a small packet size, more datagrams are transmitted and the network becomes congested, as discussed in Section 6.2 in detail. We also analyzed the performance of the network in terms of packet loss, as shown in Figure 19. Even with a link failure, it can be observed that the network is still able to transfer data; however, it will experience more packet losses due to longer routes.

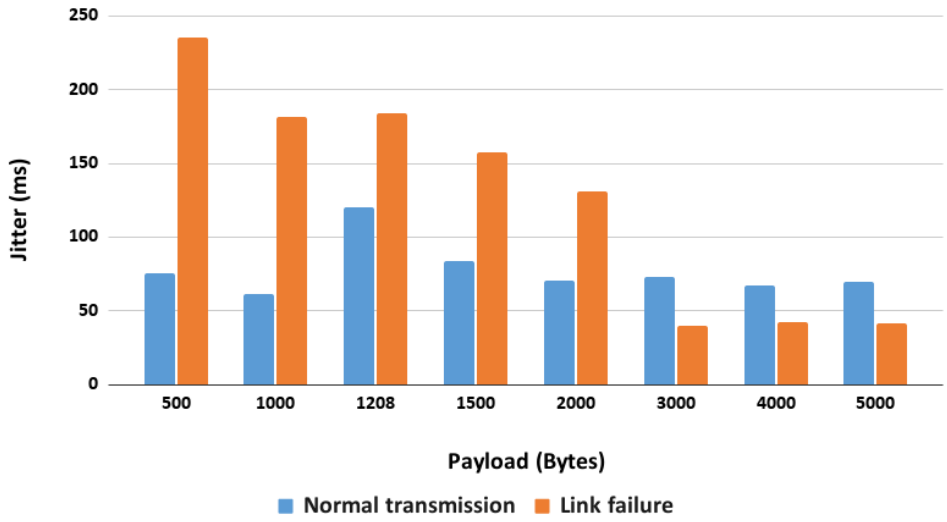

Figure 18. Jitter in link failure scenario for Topology 3.

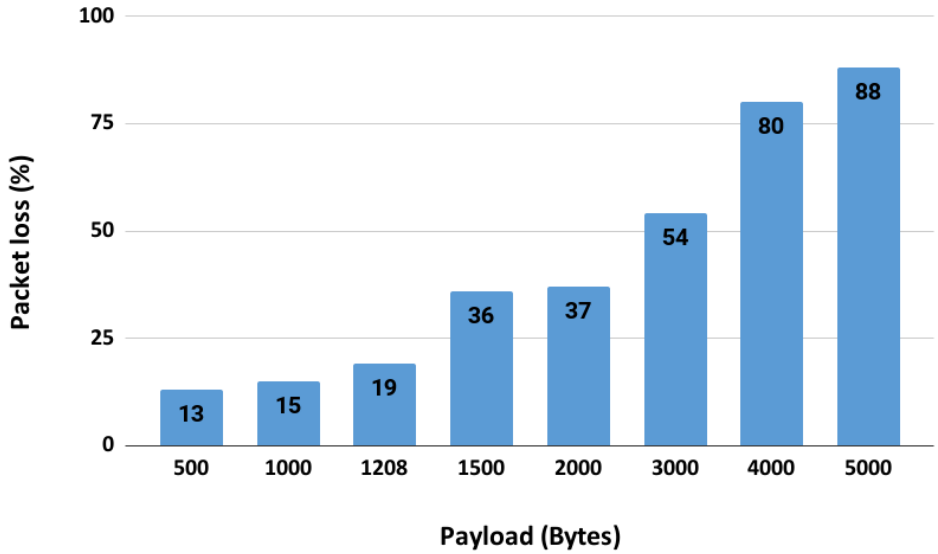

Figure 19. Packet loss in link failure scenario for Topology 3.

6.5. Experiment 5: Round Trip Time

As discussed in Section 2 and shown in Table 1, RTT is the most common performance evaluation metric for networks and also have been used in Thread-related literature. We also evaluate our network based on this metric for all three topologies as previously discussed. We analyzed the network response on these topologies using standard ICMP ping. Figure 20 shows that the RTT increases with an increase in the number of hops and the lowest RTT is observed with one-hop communication, i.e., less than 20 ms.

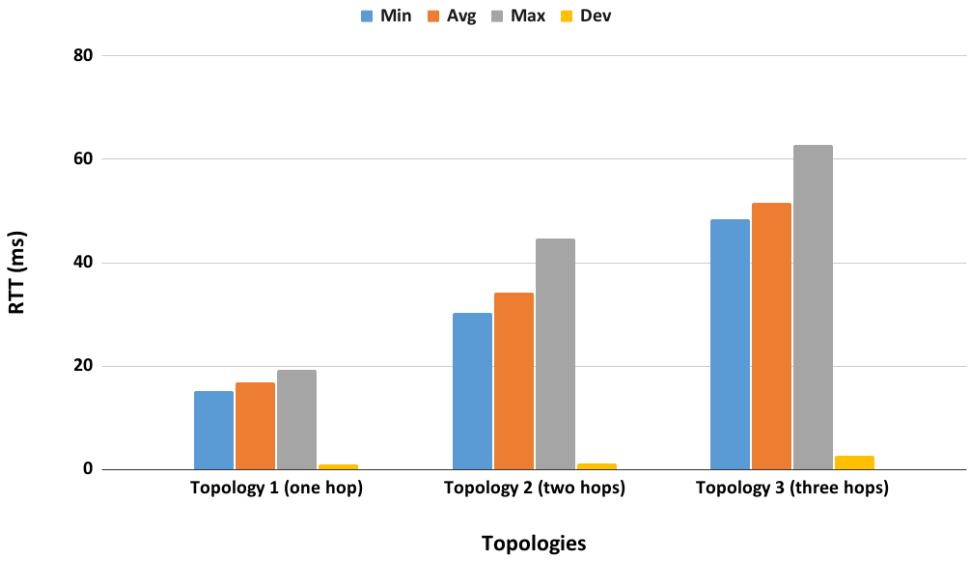

Figure 20. RTT for the three topologies.

7. Recommendations

In networks using the Thread protocol, the choice of the communication system (one hop, two hops, or multi-hop) depends on the specific application requirements. On the basis of the results, we have identified a few recommendations regarding the appropriate topology utilization. The one-hop topology is suitable for applications that need low latency and direct communication between a leader node and a router, such as remote control and sensors to the base station. The two-hop topology is beneficial for applications that need extended coverage or communication between the leader node and child nodes located further away, such as home automation, building/campus monitoring systems, distributed industrial IoT applications, and neighborhood area networks. While the multi-hop topology which is based on mesh topology, is ideal for applications that need robustness, scalability, and redundancy, such as industrial monitoring and control, wireless sensor networks, smart city infrastructure, and large-scale deployments. The choice of a communication system will consider factors, such as jitter, packet loss ratio, and round trip time by considering varying bitrate, payload, and link failure to match the specific requirements of each application.

Overall, the performance evaluation results indicate that the Thread network based on OpenThread and the Raspberry Pi is capable of providing a low packet loss ratio and jitter. The Thread protocol has been tested in various varying conditions to check its performance. One-hop and two-hop topologies perform better than the multi-hop topology having less jitter and packet loss ratio. However, all topologies provide acceptable performance in terms of packet loss rates. Based on these observations, we believe the Thread system can be used for a wide range of applications, as mentioned previously.

8. Conclusions and Future Work

This work presented a detailed implementation of a Thread-based network using OpenThread, the Raspberry Pi, and the nRF52840 dongle. It provides a clear description of the Thread protocol stack layers and network topology. The assessment of system perfor-

mance, along with configuration parameters, such as jitter and packet loss, demonstrates the effectiveness and reliability of the implemented Thread network in one-hop, two-hop, and multi-hop communication. Furthermore, the investigation examines the network's performance by considering link failure situations. The performance assessment results not only highlight the system's capabilities but also emphasize the potential of Thread for efficient and dependable communication in IoT networks. The findings from our experiments can serve as useful guidance for optimizing the network's performance under various traffic conditions.

Our future work will focus on incorporating mobility into the network and exploring its performance in various real-world scenarios, such as smart homes, smart buildings, agriculture, and industrial automation.

Author Contributions: Conceptualization, S.B.A.K. and M.M.N.; Methodology, S.B.A.K., M.M.N. and H.F.; Validation, S.B.A.K., M.M.N. and H.F.; Formal analysis, M.M.N.; Investigation, S.B.A.K. and N.C.; Data curation, M.M.N.; Writing—original draft, S.B.A.K.; Writing—review & editing, H.F.; Visualization, H.F.; Supervision, M.M.N. and N.C.; Project administration, M.M.N.; Funding acquisition, M.M.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the research grants [SEED-2022-CE-106]; Prince Sultan University; Saudi Arabia [grant number SEED-2022-CE-106].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to acknowledge Prince Sultan University and Smart Systems Engineering lab for their valuable support. Furthermore, the authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khattak, S.B.A.; Jia, M.; Marey, M.; Nasralla, M.M.; Guo, Q.; Gu, X. A Novel Single Anchor Localization Method for Wireless Sensors in 5G Satellite-Terrestrial Network. *Alex. Eng. J.* **2022**, *61*, 5595–5606. [CrossRef]
2. Ilyas, A.; Mahfooz, S.; Mehmood, Z.; Ali, G.; ElAffendi, M. Two-way approach for improved real-time transmission in fog-iot-based health monitoring system for critical patients. *Comput. Syst. Sci. Eng.* **2023**, *46*, 3815–3829. [CrossRef]
3. Elfouly, F.H.; Ramadan, R.A.; Khedr, A.Y.; Yadav, K.; Azar, A.T.; Abdelhamed, M.A. Efficient Node Deployment of Large-Scale Heterogeneous Wireless Sensor Networks. *Appl. Sci.* **2021**, *11*, 10924. [CrossRef]
4. Jan, H.; Yar, H.; Iqbal, J.; Farman, H.; Khan, Z.; Koubaa, A. Raspberry Pi Assisted Safety System for Elderly People: An Application of Smart Home. In Proceedings of the 2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 3–5 November 2020; pp. 155–160. [CrossRef]
5. Tekler, Z.D.; Low, R.; Yuen, C.; Blessing, L. Plug-Mate: An IoT-based occupancy-driven plug load management system in smart buildings. *Build. Environ.* **2022**, *223*, 109472. [CrossRef]
6. Zhuang, D.; Gan, V.J.; Tekler, Z.D.; Chong, A.; Tian, S.; Shi, X. Data-driven predictive control for smart HVAC system in IoT-integrated buildings with time-series forecasting and reinforcement learning. *Appl. Energy* **2023**, *338*, 120936. [CrossRef]
7. Low, R.; Tekler, Z.D.; Cheah, L. Predicting commercial vehicle parking duration using generative adversarial multiple imputation networks. *Transp. Res. Rec.* **2020**, *2674*, 820–831. [CrossRef]
8. Jan, B.; Farman, H.; Khan, M.; Talha, M.; Din, I.U. Designing a Smart Transportation System: An Internet of Things and Big Data Approach. *IEEE Wirel. Commun.* **2019**, *26*, 73–79. [CrossRef]
9. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutorials* **2015**, *17*, 2347–2376. [CrossRef]
10. Allahham, A.A.; Rahman, M.A. A smart monitoring system for campus using Zigbee wireless sensor networks. *Int. J. Softw. Eng. Comput. Syst.* **2018**, *4*, 1–4. [CrossRef]
11. Tekler, Z.D.; Low, R.; Gunay, B.; Andersen, R.K.; Blessing, L. A scalable Bluetooth Low Energy approach to identify occupancy patterns and profiles in office spaces. *Build. Environ.* **2020**, *171*, 106681. [CrossRef]
12. Orfanos, V.A.; Kaminaris, S.D.; Papageorgas, P.; Piromalis, D.; Kandris, D. A Comprehensive Review of IoT Networking Technologies for Smart Home Automation Applications. *J. Sens. Actuator Netw.* **2023**, *12*, 30. [CrossRef]
13. Unwala, I.; Taqvi, Z.; Lu, J. Thread: An IoT Protocol. In Proceedings of the 2018 IEEE Green Technologies Conference (GreenTech), Austin, TX, USA, 4–6 April 2018; pp. 161–167. [CrossRef]

14. Thread Network Fundamentals, White Paper. September 2022. Available online: https://www.threadgroup.org/support (accessed on 27 April 2023).
15. Al-Shareeda, M.A.; Manickam, S. A Systematic Literature Review on Security of Vehicular Ad-Hoc Network (VANET) Based on VEINS Framework. *IEEE Access* **2023**, *11*, 46218–46228. [CrossRef]
16. Nakura, K.; Ishibashi, N.; Masaki, H.; Mizutani, K.; Harada, H. Experimental Evaluation of IEEE 802.15.4 OFDM for Wireless IoT Communication Systems. In Proceedings of the 2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Kyoto, Japan, 12–15 September 2022; pp. 1159–1164. [CrossRef]
17. Silicon Labs. Thread Mesh Network Performance. Available online: https://www.silabs.com/documents/login/application-notes/an1141-thread-mesh-network-performance.pdf (accessed on 27 April 2023).
18. NXP Semiconductors. Thread Large Network. AN12099. 2017. Available online: https://www.nxp.com/docs/en/application-note/AN12099.pdf (accessed on 27 April 2023).
19. Silicon Labs. Benchmarking Bluetooth Mesh, Thread, and Zigbee Network Performance. Available online: https://www.silabs.com/wireless/multiprotocol/mesh-performance (accessed on 27 April 2023).
20. Sistu, S.; Liu, Q.; Ozcelebi, T.; Dijk, E.; Zotti, T. Performance Evaluation of Thread Protocol based Wireless Mesh Networks for Lighting Systems. In Proceedings of the 2019 International Symposium on Networks, Computers and Communications (ISNCC), Istanbul, Turkey, 18–20 June 2019; pp. 1–8. [CrossRef]
21. Rzepecki, W.; Ryba, P. IoTSP: Thread Mesh vs. Other Widely used Wireless Protocols—Comparison and use Cases Study. In Proceedings of the 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), Istanbul, Turkey, 26–28 August 2019; pp. 291–295. [CrossRef]
22. Tupas Castro, C.M.; Sharma, A.; Kumar, D.S.; Abidi, K.; Kim, N. The implementation of Thread Network for a Smart Factory. In Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence (SSCI), Singapore, 4–7 December 2022; pp. 253–260. [CrossRef]
23. Lan, D.; Pang, Z.; Fischione, C.; Liu, Y.; Taherkordi, A.; Eliassen, F. Latency Analysis of Wireless Networks for Proximity Services in Smart Home and Building Automation: The Case of Thread. *IEEE Access* **2019**, *7*, 4856–4867. [CrossRef]
24. Chitanvis, R.; Ravi, N.; Zantye, T.; El-Sharkawy, M. Collision avoidance and Drone surveillance using Thread protocol in V2V and V2I communications. In Proceedings of the 2019 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 July 2019; pp. 406–411. [CrossRef]
25. Herrera, T.; Núñez, F. Design and Prototyping of a Thread Border Router Based on a Non Network-Co-Processor Architecture. *IEEE Access* **2020**, *8*, 60613–60625. [CrossRef]
26. Grohmann, A.I.; Nophut, D.; Sobe, M.; Perez, A.B.; Fitzek, F.H.P. Interference resilience of Thread: A practical performance evaluation. In Proceedings of the 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2021; pp. 1–4. [CrossRef]
27. Kurunathan, H.; Severino, R.; Koubaa, A.; Tovar, E. IEEE 802.15.4e in a Nutshell: Survey and Performance Evaluation. *IEEE Commun. Surv. Tutorials* **2018**, *20*, 1989–2010. [CrossRef]
28. Khattak, S.B.A.; Fawad, M.M.; Nasralla, M.A.; Mostafa, H.E.; Jia, M. WLAN RSS-Based Fingerprinting for Indoor Localization: A Machine Learning Inspired Bag-of-Features Approach. *Sensors* **2022**, *22*, 5236. [CrossRef] [PubMed]
29. Latif, S.; Driss, M.; Boulila, W.; Jamal, S.S.; Idrees, Z.; Ahmad, J. Deep Learning for the Industrial Internet of Things (IIoT): A Comprehensive Survey of Techniques, Implementation Frameworks, Potential Applications, and Future Directions. *Sensors* **2021**, *21*, 7518. [CrossRef] [PubMed]
30. Choudhury, N.; Nasralla, M.M.; Shrivastav, A.; Hazarika, A. DDAS: Distributed Delay Aware Scheduling for DSME based IoT Network Applications in Smart Cities. In Proceedings of the 2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Belfast, UK, 14–17 June 2022; pp. 535–540. [CrossRef]
31. Rzepecki, W.; Iwanecki, Ł.; Ryba, P. IEEE 802.15.4 Thread Mesh Network—Data Transmission in Harsh Environment. In Proceedings of the 2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), Barcelona, Spain, 6–8 August 2018; pp. 42–47. [CrossRef]
32. nRF52840 Dongle. Available online: https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dongle (accessed on 27 April 2023).
33. OpenThread. Available online: https://openthread.io/ (accessed on 27 April 2023).
34. OpenThread Git Repository. Available online: https://github.com/openthread (accessed on 27 April 2023).
35. Singh, N.; Vardhan, M. Multi-objective optimization of block size based on CPU power and network bandwidth for blockchain applications. In *Proceedings of the Fourth International Conference on Microelectronics, Computing and Communication Systems: MCCS 2019–2021*; Springer: Singapore, 2021; pp. 69–78.