Received Date:29th November, 2025
Revision Date: 7th December, 2025
Accepted Date: 3rd January, 2026

A Telecommunication System Based on Wireless Relay Network

Dishan Shakya[1*], Diwas Dahal[2], Nabin Shrestha[3], Sajen Maharjan[4], Praches Acharya[5]
[1]Dept. of Electronics and Computer Engineering, Thapathali Campus, Nepal. Email: dishanshakya@gmail.com
[2]Dept. of Electronics and Computer Engineering, Thapathali Campus, Nepal. Email: dahaldiwaas@gmail.com
[3]Dept. of Electronics and Computer Engineering, Thapathali Campus, Nepal. Email: nabin.078bei023@tcioe.edu.np
[4]Dept. of Electronics and Computer Engineering, Thapathali Campus, Nepal. Email: sajenmaharjan927@gmail.com
[5]Asst. Professor, Dept. of Electronics and Computer Engineering, Thapathali Campus, Nepal. Email: pacharya@tcioe.edu.np

Abstract — A wireless relay network is made up of multiple low-power nodes that pass messages between two endpoints by forwarding them through intermediate helper nodes. This relay process greatly increases the communication range of small wireless devices but requires a custom communication protocol rather than traditional centralized routing methods. In this work, we present the FMTG (Find Me That Guy) protocol, a multi-hop system that uses simple broadcast discovery and chained acknowledgements to build communication paths. It removes the need for periodic control messages entirely, cutting idle overhead by 100% compared to protocols like OLSR and BATMAN and uses a compact 28-byte packet with an 11-byte header to keep communication efficient. Practical experiments with microcontroller-based nodes show that this approach extends communication distance from about 15 m to 38 m across four hops. These results demonstrate that it is a low-overhead and power-efficient option for building distributed communication networks.

Keywords — Discovery, Intermediate Node, Node, Packet, Relay

Introduction

Communication is a fundamental pillar of modern society, enabling the exchange of information, coordination of activities, and connection between individuals and communities. However, the telecommunication systems that support this communication are traditionally built around centralized infrastructure such as towers, base stations, and control centers. These centralized elements make networks expensive to deploy and maintain, and their failure can disrupt communication on a massive scale. During disasters like earthquakes, centralized hubs are particularly vulnerable; damage to a Mobile Switching Centers (MSCs) [1] or control facility can disable entire regions, leaving people without access to critical communication channels. Centralization also concentrates security risks, as a single successful attack can compromise data integrity or shut down essential services. These vulnerabilities highlight the limitations of relying solely on centralized communication models, especially in countries such as Nepal where only a few telecom providers exist and resilience is crucial.

To address these challenges, this project explores a decentralized alternative based on wireless relay networks. At the core of this system is a custom communication protocol called FMTG (Find Me That Guy). FMTG enables nodes to discover one another, establish multi-hop paths through intermediate relays, and transfer information once a chained-acknowledgement route has been formed between the sender and recipient. Its design allows distributed nodes to dynamically locate users and extend communication far beyond the physical range of any individual device. This makes the protocol useful for emergency communication, for extending the reach of existing networks, and for situations where centralized infrastructure is unavailable or unreliable.

Along with its applications and benefits, it faces some limitations. One limitation is that an intermediate node is a must if the end communicating nodes are not in each other's range. If there is no intermediate node, then no path can be ever formed. Also, if there has been a connection between the end nodes and during the duration of the data transmission, if the only intermediate node fails or breaks, there is no other way to reconnect the two end nodes.

The remaining content of this paper focuses on the related works, methodology used, results, future enhancements, conclusions and acknowledgements.

I. Related Works

A. OLSR

The Optimized Link State Routing (OLSR) protocol is a proactive, table-driven routing solution developed for mobile ad hoc networks (MANETs). It maintains routing information by periodically exchanging topology updates

* Corresponding Author

with other nodes. A key feature of OLSR is its use of Multipoint Relays (MPRs), a selected subset of one-hop neighbors with symmetric links, which are responsible for forwarding control messages. This mechanism minimizes the number of transmissions required to flood the network with routing information, enhancing overall efficiency. MPRs also play a crucial role in link-state dissemination. For OLSR to compute shortest path routes, it is sufficient for MPRs to advertise the link-state information of the nodes that selected them. Additional link-state data may be used for redundancy. Nodes that have been selected as MPRs periodically announce this status in their control messages, thereby informing the network of their connectivity to specific nodes. Since MPRs are chosen based on bidirectional links, the protocol avoids issues related to unidirectional transmission, such as the lack of link-layer acknowledgments in certain unicast scenarios [2].

B. BATMAN

The Better Approach to Mobile Ad-hoc Networking (B.A.T.M.A.N.) is a proactive routing protocol designed for multi-hop wireless mesh networks, particularly mobile ad hoc networks (MANETs). It originated from the Ger- man "Freifunk" community as a successor to the Optimized Link State Routing (OLSR) protocol, which proved inadequate for large-scale mesh network performance. B.A.T.M.A.N.'s central innovation lies in its decentralized routing intelligence; no single node requires complete knowledge of the entire network topology. Instead, each node retains only information about the direction from which it receives messages and forwards packets accordingly. As a result, data is dynamically routed on a per-packet basis, creating a distributed and adaptive routing environment. Unlike protocols that rely on complete path discovery, B.A.T.M.A.N. focuses solely on determining the optimal single-hop neighbor for reaching a given destination. This strategy enables efficient routing by simplifying the routing table: each node maintains only the best local gateway for each destination. The protocol operates proactively, keeping track of all reachable nodes through both single-hop and multi-hop connections [3].

C. Ad hoc On-Demand Distance Vector (AODV) Routing

The Ad hoc On-Demand Distance Vector (AODV) routing protocol is designed for use in mobile ad hoc net- works, where nodes can move unpredictably. AODV enables dynamic, self-starting, and multihop communication between mobile devices. Unlike proactive protocols, AODV discovers routes on demand, establishing a path only when a node needs to send data. This approach minimizes memory and processing overhead, reduces unnecessary network traffic, and allows the protocol to react efficiently to topology changes. When a node needs a route to a new destination, it initiates a route discovery process. If the route becomes invalid due to a link failure, AODV promptly notifies the affected nodes so they can discard the broken path. Importantly, among multiple available paths to a destination, the node always selects the route with the highest sequence number, ensuring it uses the most recent and reliable route information [4].

D. The Dynamic Source Routing Protocol (DSR)

The Dynamic Source Routing (DSR) protocol is a lightweight and efficient routing solution tailored for multi- hop wireless ad hoc networks composed of mobile nodes. It enables fully self-configuring and infrastructure- free networking, allowing devices to autonomously establish routes without relying on centralized administration. DSR is built around two core mechanisms: Route Discovery and Route Maintenance. These components function together to allow nodes to discover paths to destinations and dynamically maintain those paths in response to topology changes.[5]

II. System Methodology

A. Working Principle

The system consists of nodes which is a custom designed hardware device each capable of communicating with other nodes and each having the same range. Now let us consider two nodes A and Z.

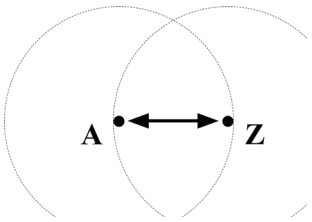

Fig. 1 Two-Node System

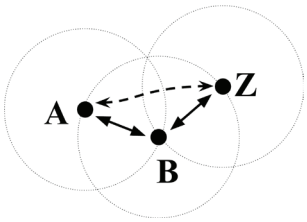

Fig. 2 Three-Node System

The dotted circles represent the range of each node. Node A wants to communicate to node Z. Since they both are in range of each other, they can directly communicate with each other as shown by the bold arrow. Suppose node Z is not in range of node A, but there is another node in between them, node B. Then since node B is in range of both node A and Z, there is a way for node A to communicate to node Z by using node B as the intermediate node which will relay the data sent by node A to node Z and vice versa. The dashed arrow represents the virtual path between node A and node Z.

Now let us extend further and suppose that node Z is not in range of neither node A nor node B.

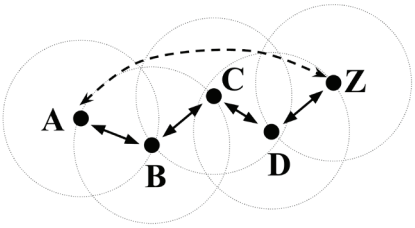

Fig. 3 Five-Node System

Node A cannot reach node Z with node B alone now. But as long as there are intermediate nodes in between like node C and node D, there is a way for node A to communicate to node Z by using the intermediate nodes in between to relay data. So if node A wants to send something to node Z, the message will have to be relayed through nodes B to C to D and finally to Z. This is the working of the system and the protocol that will be used to accomplish this system is the FMTG protocol. This will be described in a later section.

B. The FMTG Protocol

The heart of the project lies in the protocol. FMTG stands for Find Me That Guy, where the caller asks other intermediate nodes to help it "Find the Recipient" and hence the name. The protocol is packet based. Each communicating node is mandatory to contribute in relaying packets when needed and helping for formation of communicating paths. Each node will have a unique identifier or address.

Each FMTG protocol packet has a fixed size of 28 bytes. The packet consists of four 2-byte address fields, a 1-byte packet type field, 2 bytes for the selected duplex communication channels, and a 17-byte payload. The source address is the address of one of the end nodes which is attached on every packet the end node sends. The end node may either be the caller or the recipient. The destination address is the address of the other end node.

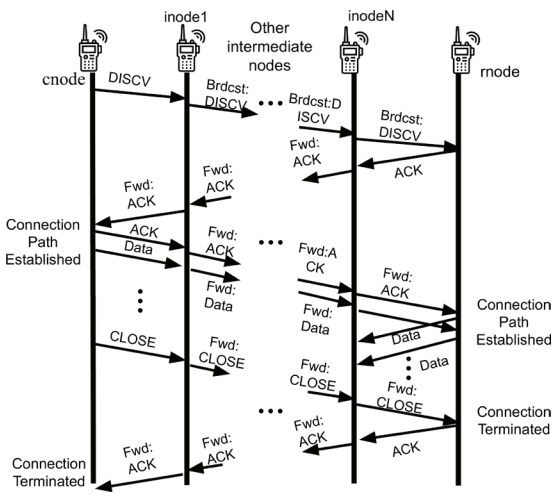

Fig. 4 Packet Flow Diagram

Intermediate sender and receiver address are the addresses of the nodes that relay the packets. A relaying node receives packets from the intermediate sender node and sends them to the intermediate receiver node.

The packet type can be one of discovery, acknowledgement, reconnect or data type. The intermediate receiver and sender channels are the frequency channels which the respective nodes are listening to. Payload is the field that contains the data. The first byte of the payload denotes the type of payload and can be one of ring, voice or termination type.

The operation of the FMTG protocol can be understood as a sequence of key processes that each node in the network performs to enable reliable communication between a caller and a recipient. These processes involve the discovery of recipient nodes, the formation of routing paths through intermediate nodes, acknowledgment of received messages, data transmission, and dynamic handling of node failures. The main steps are summarized as follows:

Table I

Protocol Packet Structure

Field	Size	Description
Source Address	2 bytes	Unique ID of original sender
Destination Address	2 bytes	Final intended receiver
Intermediate Sender Address	2 bytes	Node currently forwarding the packet
Intermediate Receiver Address	2 bytes	Next hop node
Packet Type	1 byte	DISC / DATA / ACK / REC
Intermediate Receiver Channel	1 byte	Channel used by next hop
Intermediate Sender Channel	1 byte	Channel used by sender
Payload	17 bytes	Application-level data

1) Discovery: The caller node initiates communication by broadcasting a discovery packet containing its own address as the source, the recipient's address as the destination, and the broadcast address as the intermediate receiver. The intermediate sender is set to the caller's address, and its channel field corresponds to the caller's receiving module. The caller adds a corresponding entry to its routing table, initializing other fields to zero. Intermediate nodes relay the packet, updating the intermediate sender and channel fields in the packet and maintaining a routing table entry with the source, destination, and received intermediate sender. After several hops, the recipient receives the packet and

acknowledges the caller through a recursive chain.

2) Routing Table: Each node keeps a record of the caller, recipient and other intermediate node's address in a routing table which will be used to forward packets and maintain the connection path. The routing table consists of entries containing the source, destination, intermediate sender and receiver's address and each entry is indexed using the source and destination address combined, as primary keys. The format of the routing table is given below:

Table II

The Routing Table Format

No Sender	Src	Dst. Inter. Receiver	Inter.	IRPC	ISPC	IR Channel	IS Channel
1	0x1111	0x2222	0x3333	2	0	20	30
		0x4444					
2	…	…	…	…	…	…	…

IRPC and ISPC stands for Intermediate Receiver/Sender Packet Count respectively and are used in the broken node detection algorithm which is described in a later section. Basically these fields record the number of packets received from the intermediate sender and receiver nodes. IR and IS channels denote the Intermediate Receiver and Intermediate Sender Channels respectively and record the channel on which the intermediate sender and receiver nodes are listening to.

3) Recursive Chain Acknowledgement: After receiving a packet, the recipient sends an acknowledgement back to the caller via intermediate nodes. If multiple copies arrive, the recipient chooses the first one. The acknowledge- ment packet includes the recipient as the Source Address, the caller as the Destination Address, and the recipient as the Intermediate Sender. Unlike the discovery packet, it is unicasted, with the Intermediate Receiver set to the chosen intermediate node. The recipient updates its routing table with the packet fields. Each intermediate receiver forwards the acknowledgement to the node listed as the Intermediate Sender in its routing table, updating the In- termediate Receiver field accordingly. This process continues recursively until the acknowledgement reaches the caller, establishing the communication path.

4) Data Transmission: Once the acknowledgement packet arrives at the caller, the caller node can then start the transmission of data using the established path. The caller node now sends the data to the intermediate receiver from which it received the acknowledgement packet and the receiver node simply forwards it using the routing table. Similar process happens at the recipient where the recipient node sends data to its intermediate sender.

5) Node Break Detection and Management: During data transmission, if a node in the path fails, the path breaks. Instead of re-establishing a new path from scratch, this protocol replaces the broken node. Neighboring nodes detect the failure by monitoring packet counts: the IRPC and ISPC fields track packets from intermediate receivers and senders, resetting when packets arrive. If a node stops receiving packets from its neighbor, the respective count exceeds a threshold, signaling a failure.

Here we have four nodes A, B, C, and D with routing tables showing only Intermediate Receiver/Sender Packet Counts (IRPC/ISPC). Nodes A, B, and C form an active communication path, while node D is idle, near B, and in range of both A and C. IRPC and ISPC track packets received from intermediate receivers and senders. For the caller node A, the intermediate sender is itself, storing its own packet count, and similarly for the recipient C. When a packet arrives from an intermediate sender, ISPC increments and IRPC resets; when from an intermediate receiver, IRPC increments and ISPC resets. If connections with neighbors are healthy, both counts stay below a threshold (ideally below 2), as each packet from one neighbor resets the other's count. In practice, some packet loss may cause counts to exceed 2.

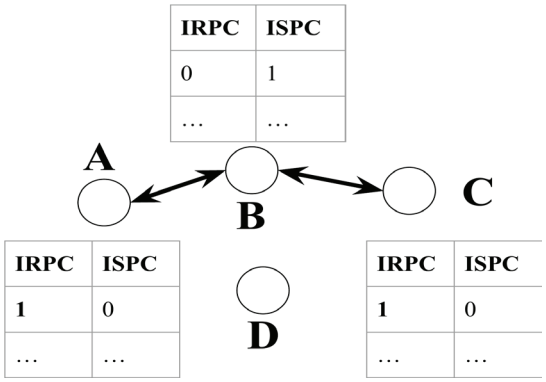

Fig. 5 Non-broken Path

If node B breaks, the A–B–C connection fails. Node A stops receiving packets from C through B, causing its ISPC to increase over time; without resets from B, the count eventually reaches a threshold, confirming B's failure. Similarly, C's IRPC rises to the threshold, notifying it of B's failure. Both neighbors can then prepare for reconnection.

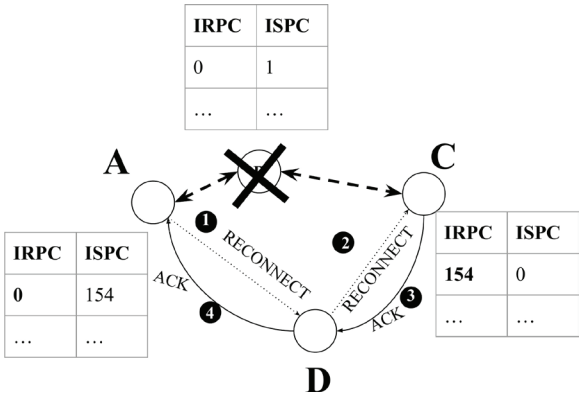

Fig. 6 Broken Node and Reconnection

Reconnection occurs if a nearby node can replace the broken one. The neighbor with the broken intermediate receiver (A) broadcasts a reconnect packet with the same source and destination and itself as the intermediate sender. Nearby nodes check their routing tables: nodes without an entry rebroadcast, nodes with active neighbors or IRPC below threshold ignore, and nodes previously connected to the broken node with IRPC above threshold accept and acknowledge recursively. Once the initiating node receives the acknowledgment, the new path is formed and all involved nodes update their routing tables with new intermediate senders and receivers.

III.Proof-of-Concept Implementation

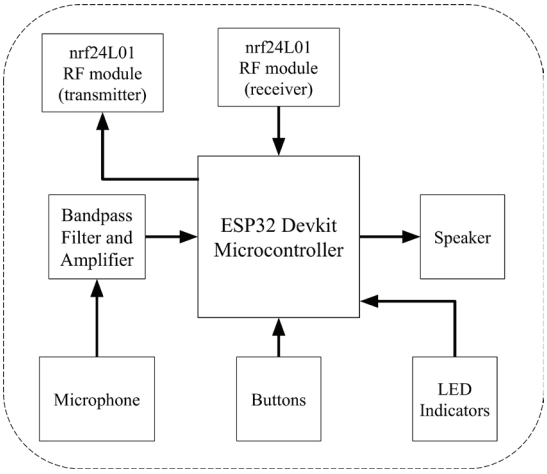

Fig. 7 Block Diagram of the Hardware Model

In order to demonstrate the functionality and flow of the designed protocol, a proof-of-concept (PoC) model was developed. Since current smartphones are hard to modify and adapt to new protocols, a low-cost device us- ing microcontrollers (ESP32 Devkit) and RF modules (NRF24L01) was developed to demonstrate the viability of this system. These devices include essential components such as microphones, speakers, and circuits for sig- nal processing, enabling voice and data communication. The integration of these components allows full-duplex communication through the use of separate RF channels for transmitting and receiving.

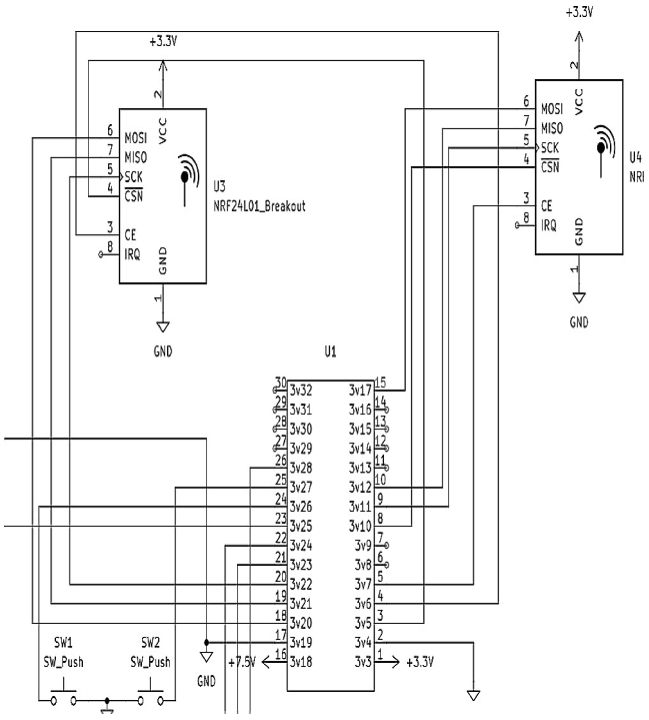

Fig. 8 Interfacing NRF Module

The NRF24L01 module communicates with the microcontroller via SPI, a widely used full-duplex serial proto- col using four pins: Chip Select (CS), Serial Clock (SCK), Master-Out-Slave-In (MOSI), and Master-In-Slave-Out (MISO). Communication begins when the master pulls the CS low, and during each clock cycle, the master sends a bit on MOSI while the slave simultaneously sends a bit on MISO. This sequence occurs even for one-way data transfer. Two NRF24L01 modules were used, connecting them to the ESP32 DevKit's two available SPI interfaces, as shown in Figure 8.

B. Speech Processing and Sampling

An electret microphone captures the sound, which is first passed through a bandpass filter circuit [6][7] to remove unwanted noise and then through an amplifier circuit to adjust the signal strength. The processed signal is subsequently sampled by an ESP32 microcontroller, allowing us to digitize and analyze the audio for further processing.

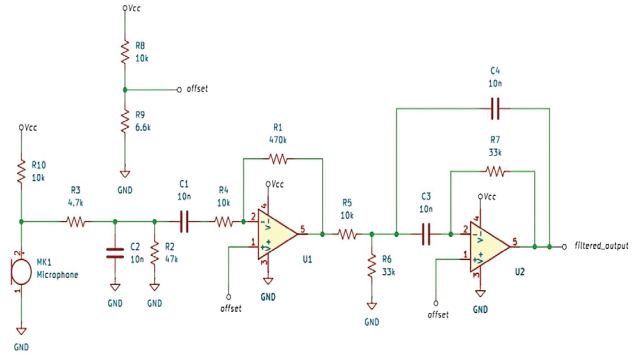

Fig. 9 Amplifier and Filter Circuit

As shown in Figure 9, the voice signal is captured by an electret microphone, which converts sound vibrations into low-amplitude fluctuating voltage levels. This signal is first amplified through an amplifier circuit and then passed through a bandpass filter with a 300Hz-4kHz passband (typical range for human speech).

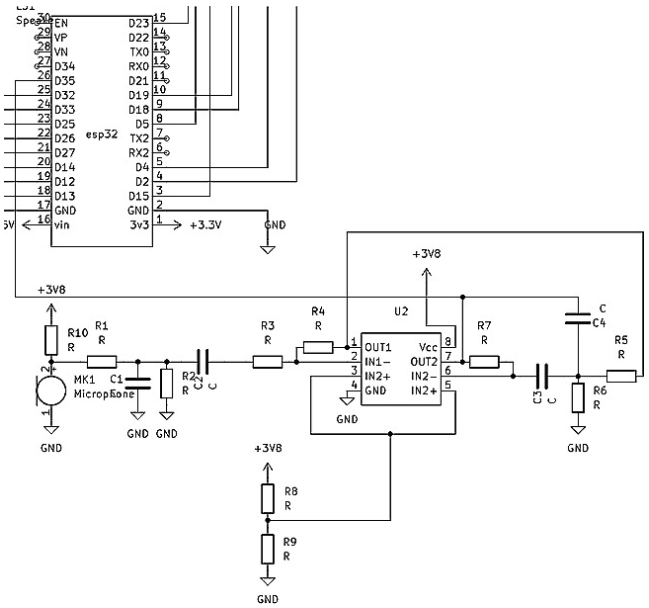

Fig. 10 Schematic Diagram of the Model

Then as showned in the schematic diagram in Figure 10, the filtered and amplified signal is fed to the micro- controller's analog input pins, where it is digitized into 8-bit data using the built-in ADCs. This process, known as Pulse Code Modulation (PCM), is performed at a sampling rate of 8kHz. The resulting digital voice data is packaged into packets and transmitted to the recipient. Upon reception, the recipient converts the digital data back to analog using a Digital-to-Analog Converter (DAC) at the same 8kHz, 8-bit resolution and outputs it through a speaker.

C. *Interfacing the Output Speaker*

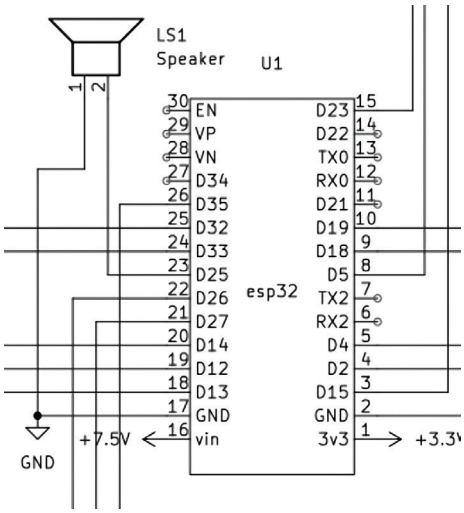

Fig. 11 Speaker Interfacing with ESP32

Since the ESP32 microcontroller has an internal 8-bit DAC, the speaker is simply directly connected to the D25 pin and GND pin of the microcontroller. This is shown in Figure

11. **Results**

First, the hardware was designed in printed circuit boards, and the protocol was programmed on it. The circuit required a two-layer PCB but due to unavailability and cost issues, it was possible on a single layer PCB by adding extra wires. The hardware can be seen in Figure 12. Four of these devices were made and tested.

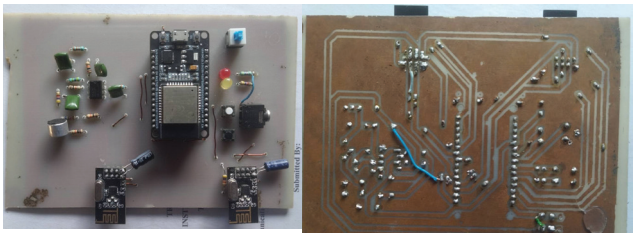

Fig. 12 Front and Back of the Hardware Device

A. *Comparison of FMTG With Existing Routing Protocols*

This section compares the performance of the FMTG protocol with commonly used ad hoc routing protocols such as OLSR, BATMAN, AODV, and DSR. The comparisons focus on control overhead, route setup cost, packet header size, multi-hop behavior, and power consumption. All numerical values from the original protocols are retained, and the figures from our system measurements are included for clarity.

1) Idle-Time Control Overhead: One of the most important differences between FMTG and traditional routing protocols appears during idle network conditions. In OLSR, every node sends HELLO packets every 2 seconds and broadcasts topology control (TC) messages every 5 seconds, which still results in a steady amount of background signaling. In a typical 10-node network, this generates approximately 2–4 kbps of continuous control overhead per node even when no user data is being transmitted. BATMAN shows similar behavior, transmitting 52-byte Origi- nator Messages (OGMs) at 1 Hz. In a 10-node mesh, this leads to roughly 4.16 kbps of unavoidable background traffic.

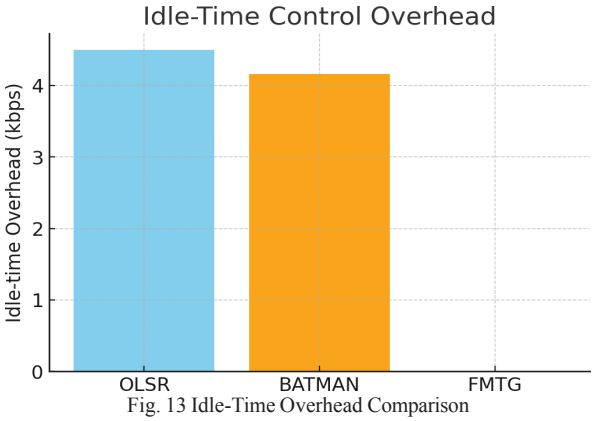

Fig. 13 Idle-Time Overhead Comparison

In contrast, FMTG generates *zero periodic overhead*. When no call is active, the network remains completely silent. Routing information is created only during call setup. The routing state stored for each active route is very small, staying under 40 bytes in total (2 B source, 2 B destination, 2 B intermediate sender, 2 B intermediate receiver, and a few bytes for counters and flags). Overall, FMTG reduces idle overhead by 100% compared to OLSR and BATMAN.

2) Route Setup Cost Compared to AODV: AODV uses a network-wide flooding approach for route discovery. Each Route Request (RREQ) packet is typically 24-32 bytes, and each Route Reply (RREP) is 20-28 bytes. In a 10-node network, this usually adds up to about 300–400 bytes of control traffic for a single route discovery attempt.

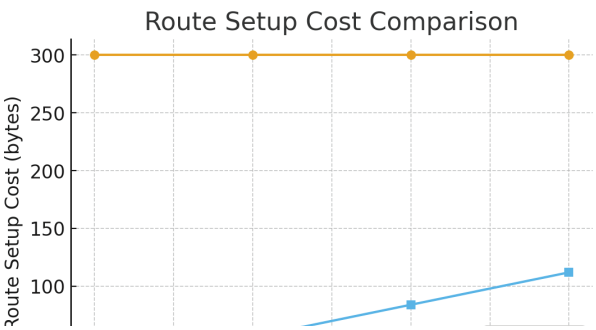

Fig. 14 Routing Cost Comparison

FMTG uses a much simpler mechanism. It broadcasts only one 28-byte discovery packet. This is followed by a chained acknowledgement process, where each acknowledgement is also 28 bytes. The total setup cost for an H-hop route is therefore $28H$ bytes. For example, a 4-hop discovery requires only $28 \times 4 = 112$ bytes, which is roughly 3-5 times smaller than the AODV discovery cost.

In case a relay node fails, AODV must repeat the entire flood procedure. FMTG instead performs a local reconnect using just one or two 28-byte broadcast packets (28-56 bytes), completely avoiding a full rediscovery.

3) Packet Header Size Compared to DSR: DSR includes the full source route inside every packet header, adding 4-6 bytes for each hop. Thus, a 4-hop path results in a 16–24-byte source-route header, and this overhead grows linearly with the number of hops.

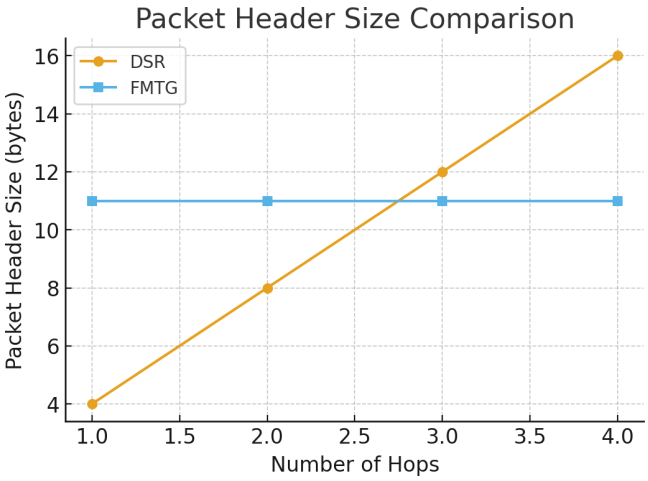

Fig. 15 Header Size Comparison

FMTG uses a fixed-size 11-byte header for all packets (2 B source, 2 B destination, 2 B intermediate sender, 2 B intermediate receiver, 1 B packet type, 1 B IR channel, and 1 B IS channel). This header size does not depend on the number of hops. As a result, FMTG headers are 40–70% smaller than DSR headers in multi-hop routes.

4) Transport Overhead During Voice Transmission: FMTG packets also carry live audio data in the final prototype system. The implementation uses 8 kHz, 8-bit PCM signals, which corresponds to a raw bandwidth of

$8000 \text{ samples/s} \times 8 \text{ bits} = 64 \text{ kbit/s}$.

Each FMTG packet carries 16 bytes of audio. Therefore, the system must transmit:

$8000/16 = 500$ packets per second.

With a fixed packet size of 28 bytes, the on-air bandwidth becomes:

$500 \times 28 = 14{,}000$ B/s $= 112$ kbit/s

The overhead factor is thus: 112/64 = 1.75.

This means FMTG requires roughly 75% more bandwidth than the raw audio, which is still relatively efficient and significantly lighter than DSR's hop-dependent expansions or OLSR's periodic control traffic.

B. Range Tests

Range tests of the device with the protocol were performed by programming the device to make 100 iterations of making connection with the other node, i.e., by sending 100 discovery packets per minute and counting the number of acknowledgments received.

Starting with two nodes and as in Figure 16, it can be seen that each node has about 14 meters before it starts losing packets.

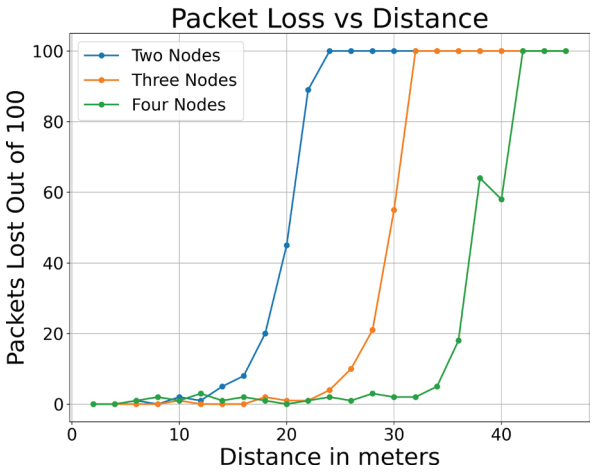

Fig. 16 Ranges for Different Number of Nodes

A third node was added between, and the range increased upto 23 meters with acceptable packet loss. Similarly, with a fourth node between, about 33 meters.

Tests on bidirectional speech signal of 64kbps PCM audio from the microphone were also performed by using frequency duplexing, but the results were not very satisfactory, and it was not possible to gain significant range gains.

C. Speech Transmission Tests

With 64kbps speech data, or 8KBps, and a packet payload size of 16 bytes, it should be 8000/16 or 500 packets per second. The test was performed by simultaneously transmitting speech at 500 packets per second and recording the number of packets received from the other end. Node break direction was disabled for this text. The following data were collected, for two, three, and four nodes at 10m, 20m and 30m respectively, which is depicted in Figure 17.

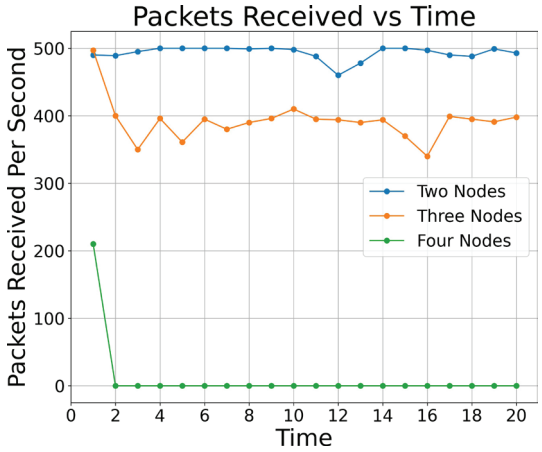

Fig. 17 Speech Packets Received Per Second for Different Number of Nodes

The likely reason behind the failure for speech signal for three and four nodes seems to be due to high interference on the intermediate nodes from nearby nodes. Both the neighbor nodes of the intermediate node send data at the same frequency channel, causing interference and packet corruption at the middle node. It was speculated that this problem could be solved by using time division duplexing instead.

D. Node Break Tests

In the node break test, speech signals were transmitted at full duplex, and the duration between disconnection and reconnection was recorded. Since a broken node needs a replacement, the test could only be performed on three nodes, with only four nodes available. It was also suspected that due to the failure of speech signals at four nodes, the test would have immediately failed for four nodes. Since there was no fifth node to replace, the test was not performed on four nodes.

For three nodes, the test was performed by placing the replacement node within range and disconnecting the connected intermediate node. The following data was collected, shown in Figure 18.

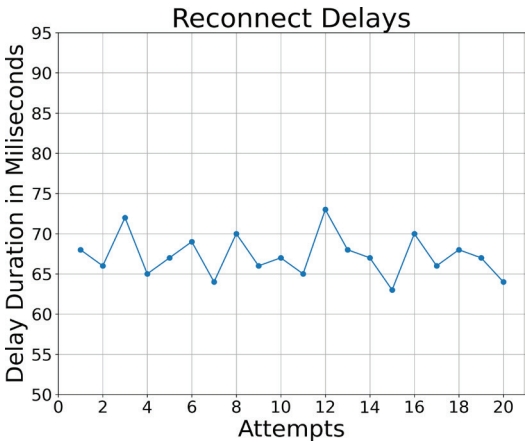

Fig. 18 Reconnect Delay Duration After Node-break

It can be seen that the average delay duration of reconnection is around 67ms.

Future Enhancements

As mentioned in the results, this specific project can be improved by using time division duplexing instead of frequency division duplexing, and with two half duplex channels instead of one full duplex channel.

The system can be enhanced further to better implement in a real-world scenario. The designed hardware can be replaced by existing cell phones which have their own transceivers. The protocol can be implemented in existing cell phones by modifying the drivers of the transceivers at a very low level. Besides this, existing telecommunication companies can also extend their system in a similar way to this project by using a hybrid of existing central systems and distributed systems like this project to further increase robustness and solve problems like poor range in specific areas.

Conclusion

This paper was focused on the development of a distributed telecommunication system consisting of wireless nodes which help other nodes relay their messages and data along with a custom protocol to implement this system. Upto 4 nodes systems were tested, and the range of a single node was extended from 14 metres to 33 metres using 2 other intermediate nodes. Test results for connection establishment were satisfactory but for speech data, it encountered design flaws and interference. Overall, the project was partly successful in the demonstration of the concept. It also highlighted the necessity of time division duplexing.

Acknowledgements

We would like to thank our Department of Electronics and Computer Engineering, Thapathali Campus for providing us with this opportunity to learn and carry our insight into the form of Minor Project. We are additionally grateful to our Supervisor Er. Praches Acharya for his encouragement and mentorship for this project. We would also like to acknowledge the researchers and writers of different papers and the developers of different program- ming libraries which we have used as a reference for the initiation of our project. We would also like to thank our Electronics Lab, Department of Electronics and Computer Engineering, Thapathali Campus for providing us the use of oscilloscopes and help during PCB design, and Robotics and Automation Centre, Thapathali Campus for providing chemicals and drill required for PCB design. Lastly, we would also like to thank our generous friends who provided us with valuable resources and a helping hand in need.

References

[1] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[2] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," RFC Editor, RFC 3626, Oct. 2003.

[3] C. Sommer, R. German, and F. Dressler, "Bidirectional link detection for the b.a.t.m.a.n. routing protocol," *Ad Hoc Networks*, vol. 8, no. 5, pp. 475–489, 2010.

[4] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," RFC Editor, RFC 3561, Jul. 2003.

[5] D. Johnson, D. Maltz, and Y. Hu, "The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4," RFC Editor, RFC 4728, Feb. 2007.

[6] M. E. Van Valkenburg, *Analog Filter Design*. Oxford, UK: Oxford University Press, 1982.

[7] Analog Devices, Inc., "Mt-220: Active filter design techniques," https://www.analog.com/media/en/training-seminars/tutorials/mt-220.pdf, 2018, tutorial, Online.