



AI-Based Self-Healing Communication Network

Shruti Kharbade, Rakhi Sarode, Leena Barange, Vaishnavi Pillay Guide: [Dr.M.K.Demde]

Department of Electronics and Telecommunication Engineering, Priyadarshini College of Engineering, Nagpur, Maharashtra, India

Abstract — Reliable data exchange is essential in IoT deployments, yet a faulty device or a degraded wireless link can halt it. This paper reports a prototype that heals its own communication paths with the help of machine learning, built on ESP32 boards, Wi-Fi, MQTT and Python. The boards send periodic status reports covering signal strength, uptime, free memory and connection state to a Python backend through an MQTT broker, where a trained model rates each node's health. Dijkstra's algorithm proposes candidate paths, and each is scored on predicted health, signal quality, delay, hop count and cost. When a node disappears or a link weakens, the backend switches to the best remaining path and informs the affected nodes. Text and images are transferred with acknowledgments and resending of lost chunks, and a web dashboard presents node state, routes, predicted health and recovery events live. Dashboard observations from the five-node prototype confirm that monitoring, health scoring and route selection work together.

Index Terms — ESP32, IoT, machine learning, self-healing network, MQTT, dynamic routing, network monitoring.

I.

INTRODUCTION

IoT devices are increasingly used for automation, monitoring and transport, forming networks of many cooperating nodes. Dependability is difficult to guarantee because nodes can suffer from poor wireless coverage, congestion, power faults or broken hardware. Several studies therefore apply artificial intelligence to strengthen network resilience and to handle faults automatically [1], [3], [4].

Many conventional networks send data along one fixed route. When a relay on that route stops working, a new route has to be found, and the time needed to notice the fault and set up the replacement can lead to dropped packets or a short outage. Self-healing designs attempt to catch such faults early and divert traffic by themselves [2], [4], by constantly observing node condition. Measures like RSSI, delay, availability, uptime, memory state and earlier failures describe how the network is behaving.

This project transfers that concept to ESP32 hardware, joining Wi-Fi links, MQTT messaging, a Python monitoring service, machine learning and dynamic route choice. Where [1]–[4] address cellular and mesh systems, the work here is a compact Wi-Fi prototype made of ESP32 boards.

II.

LITERATURE REVIEW

Earlier work on AI-assisted and self-healing networks is summarised below.

- **Outage management in cellular networks:** Raza et al. [1] propose an AI-based way of dealing with outages in dense cellular deployments.
- **Self-healing mesh networks:** Sahin and Arslan [2] design a mesh that repairs itself with no global clock synchronisation, keeping nodes connected through adaptive multi-hop routing as conditions vary.
- **Overview of AI-based self-healing:** Farmani and Khalil Zadeh [3] review AI techniques for self-healing in cellular systems.
- **Self-healing for mobile network automation:** Asghar et al. [4] present AI-driven self-healing as a route to automated mobile networks.
- **Fault prediction with routing in IoT:** The study in [5] treats a node whose state looks like an outlier as likely to fail, using a Local Outlier Factor detector, and lets a Q-learning agent steer traffic away from such nodes.

- **Deep-learning fault prediction:** ENADL [6] models the network as a graph and trains a deep network on node and edge attributes to flag likely faulty devices before data is routed over the revised topology.

Studies [1]–[4] centre on cellular and mesh settings, while [5] and [6] show that predicting node faults before routing improves IoT performance, but they rely on reinforcement learning or deep learning and are evaluated mainly on simulated or dataset-based testbeds. This project instead brings Random Forest health prediction [8] and Dijkstra-based adaptive routing [7] to an inexpensive, small ESP32 and Wi-Fi network, supported by MQTT monitoring and a web dashboard.

III. PROBLEM STATEMENT AND SIGNIFICANCE

A dead relay or a weak link in an IoT network can sever the route in use. If the fault is spotted late and a replacement is searched only afterwards, data is delayed or lost and service is briefly interrupted. This work aims to keep traffic moving without human help whenever a node vanishes or link quality drops.

Monitoring and automation applications depend on uninterrupted data flow, so this topic has practical value. Building the prototype from ordinary, inexpensive ESP32 boards shows that small networks can gain fault tolerance through software-based monitoring and smart routing.

IV. OBJECTIVES

- Track the state of ESP32 nodes continuously using RSSI, uptime, memory and heartbeat data.
- Automatically identify node failures and reduced communication quality.
- Estimate node health with a machine-learning model.
- Create alternative routes with Dijkstra's algorithm and pick the best one.
- Restore communication by moving to another route and sending the new route to the nodes.
- Move text and image data reliably using acknowledgments and retransmission.
- Display network state, routes and recovery events on a web dashboard.

V. SYSTEM DESIGN AND METHODOLOGY

A. System Architecture

The design goal is to keep nodes connected when one of them fails or when links degrade. The pieces are the ESP32 nodes, Wi-Fi, an MQTT broker [9], a Python backend that acts as the AI controller, a machine-learning module, a routing module and a dashboard (Fig. 1). Nodes connect via a Wi-Fi router to the Mosquitto broker, which hands their messages to the backend. A web page lets the sender and receiver submit and fetch text and images, while a Flask dashboard shows the network state. Logs, the dataset, the trained model and received images are stored on the backend side.

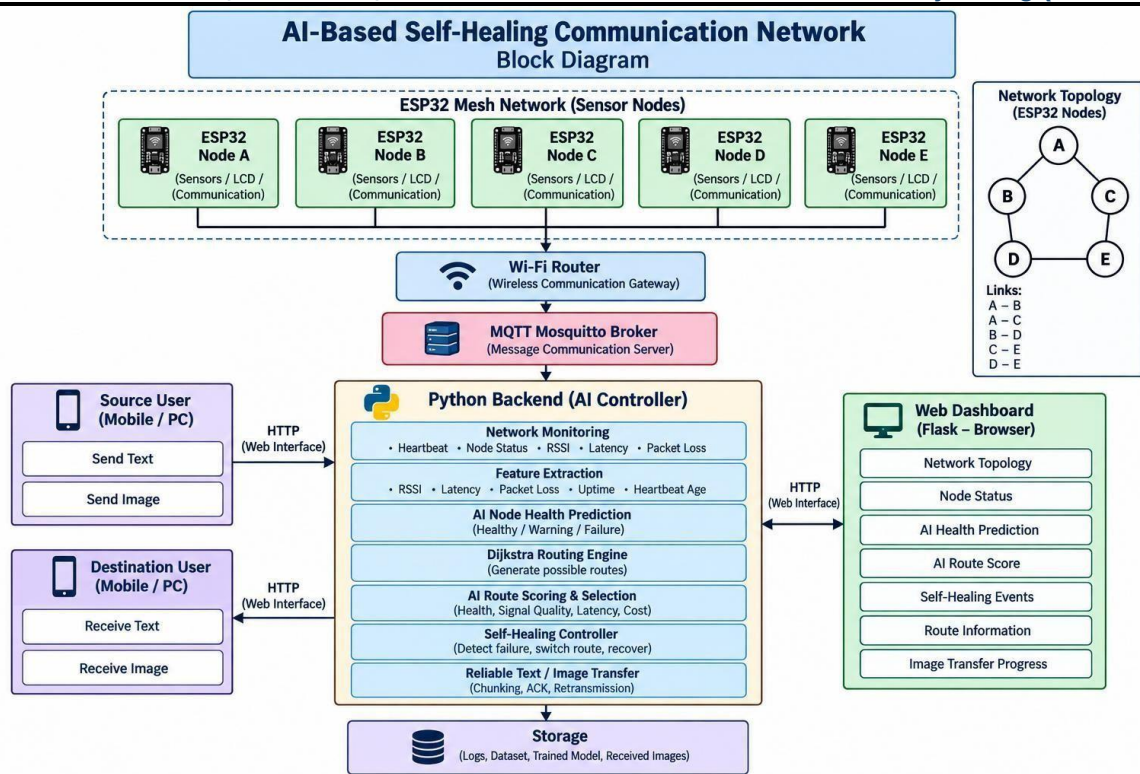


Fig. 1. Block diagram of the AI-based self-healing communication network

B. Network Structure

Five ESP32 nodes, A through E, form the test network over Wi-Fi. Its links are A-B, A-C, B-D, C-E and D-E, which gives more than one possible route between sender and receiver, such as $A \rightarrow C \rightarrow E$ (Route 1) and $A \rightarrow B \rightarrow D \rightarrow E$ (Route 2).

C. Network Monitoring

Each ESP32 publishes a heartbeat at fixed intervals through the broker to the backend. The message reports RSSI, IP and MAC addresses, Wi-Fi channel, free heap, uptime, status, failure count and recovery information.

From these, the backend maintains an up-to-date picture of the network and also follows latency and packet loss (Fig. 1).

D. Failure Detection

Every active node is expected to send heartbeats, and one that stays silent beyond the configured interval is flagged as unavailable. If Node C drops out, for example, the route $A \rightarrow C \rightarrow E$ becomes unusable and the backend starts recovering a route.

E. Machine-Learning-Based Health Prediction

The backend derives features such as RSSI, latency, packet loss, uptime and heartbeat age from the collected data (Fig. 1). A Random Forest classifier [8] takes these features and predicts whether each node is likely to be ONLINE or OFFLINE; this prediction is one more input to route selection. In the block diagram the health result is shown as Healthy, Warning or Failure.

F. Dynamic Route Generation

Whenever a route is needed, the routing module lists the possible paths between the two endpoints. Dijkstra's algorithm [7] then finds suitable paths using the defined cost. For the example topology the candidates are $A \rightarrow C \rightarrow E$ and $A \rightarrow B \rightarrow D \rightarrow E$. Similar adaptive multi-hop routing is used in self-healing mesh studies [2], and fault-aware routing for IoT is examined in [5], [6].

G. Route Evaluation

The candidate paths are compared on the parameters in Table 1, and the selection step picks the one that best suits present conditions.

In the implemented dashboard, a route's final score is 60% average predicted node health, 25% health of its weakest node and 15% route cost. If RSSI stays below -70 dBm, or optional application latency reaches 200 ms, over three consecutive checks, a new AI route evaluation is started.

Table 1. Parameters used for route evaluation

Parameter	Role in route selection
Predicted node health	Machine-learning output for each node on the path
RSSI / signal quality	Reflects the quality of the wireless link
Latency	Delay experienced along the path
Number of hops	Length of the path
Route cost	Cost value used by Dijkstra's algorithm
Node availability	Whether each node is currently sending heartbeats

H. Self-Healing Operation

Healing starts when the current route can no longer be trusted. Traffic normally follows $A \rightarrow C \rightarrow E$; if Node C goes down, the backend notices, assesses the other path, and may choose $A \rightarrow B \rightarrow D \rightarrow E$ and send it to the nodes concerned. A returned acknowledgment confirms that they received and applied it. This matches the self-healing principle in earlier work [1], [2], [4].

I. Network Degradation Handling

Gradual deterioration is handled as well as complete node loss. A RSSI that stays weak, for instance, suggests the present route is unsuitable, and once a measured value crosses its threshold the backend assesses the alternatives.

J. Reliable Data Transfer

Text and images can both be transferred. An image is cut into small chunks, each confirmed by an acknowledgment, and any chunk that does not arrive is sent once more.

K. Monitoring Dashboard

The Flask web dashboard presents node status, topology, active route, predicted node health, route score, recovery events, route version and image-transfer progress.

L. System Workflow

ESP32 nodes $\rightarrow$ Wi-Fi $\rightarrow$ MQTT broker $\rightarrow$ Python backend $\rightarrow$ network monitoring $\rightarrow$ ML health prediction $\rightarrow$ route generation $\rightarrow$ route evaluation $\rightarrow$ route selection $\rightarrow$ self-healing $\rightarrow$ data transmission $\rightarrow$ dashboard.

VI. HARDWARE AND COMPONENTS

Low-cost parts keep the prototype affordable, so several nodes can be built for testing. Table 2 lists them.

Table 2. Hardware components and their function

Component	Function
ESP32	Main controller of each node; supplies Wi-Fi, communicates over MQTT, monitors network parameters and receives updated routes
16×2 LCD	Displays local details such as node ID, connectivity, Wi-Fi signal level, next hop and route status
I2C interface module	Cuts down the GPIO pins needed for the LCD; typically GPIO 21 (SDA) and GPIO 22 (SCL)
Connecting wires	Jumper wires joining the ESP32, LCD, I2C module, battery and power parts
3.7 V Li-ion battery	Rechargeable portable power, letting a node run without a fixed supply
Power bank module	Regulated supply for the prototype and battery charging, keeping each node portable

VII. RESULTS AND DISCUSSION

The prototype ran with all five ESP32 nodes and was observed on the live Flask dashboard (Figs. 2 and 3). The overview reports 5 of 5 nodes online, none offline, and 711 heartbeats received. The AI status is ACTIVE, with 96.67% average node health and 90.67% for the weakest node.

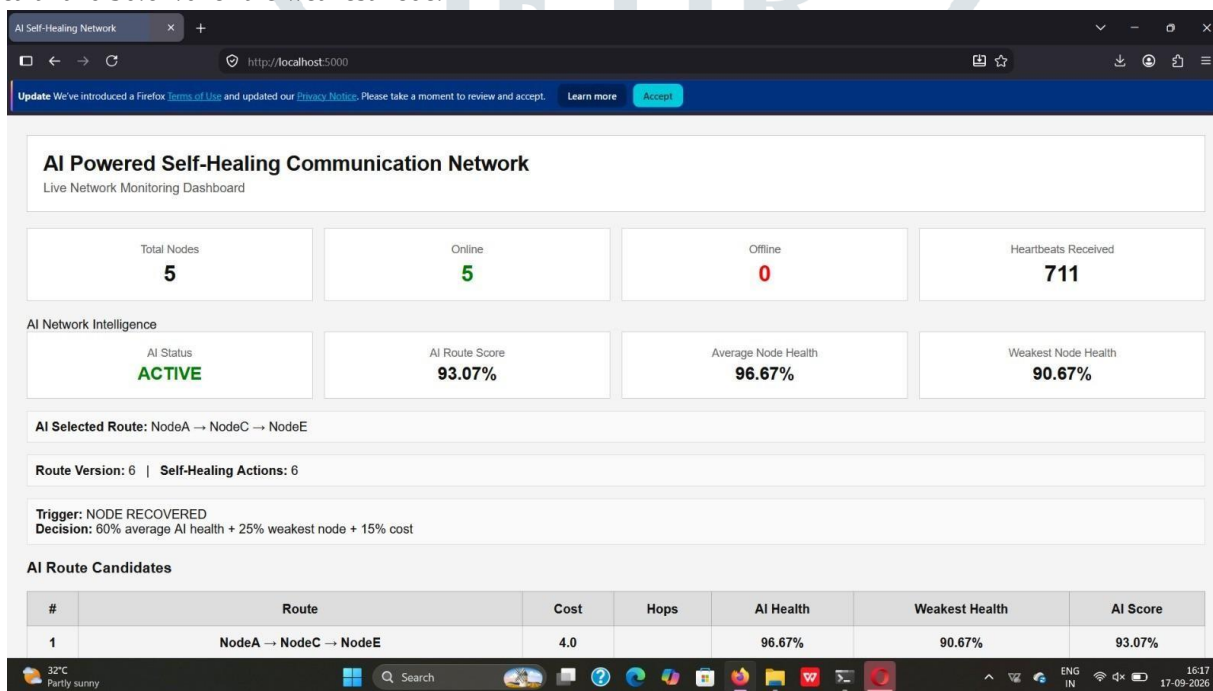


Fig. 2. Live dashboard showing node counts, AI health, selected route, route version and self-healing actions

The backend picked Node A $\rightarrow$ C $\rightarrow$ E with an AI route score of 93.07%. The dashboard shows route version 6, six self-healing actions and a last trigger of NODE RECOVERED, i.e. the route was re-evaluated after a node returned. Table 3 gives the two candidate routes for this topology.

Table 3. AI route candidates (from the dashboard)

#	Route	Cost	AI health	Weakest health	AI score
1	Node A $\rightarrow$ C $\rightarrow$ E	4.0	96.67%	90.67%	93.07%
2	Node A $\rightarrow$ B $\rightarrow$ D $\rightarrow$ E	5.0	97.35%	91.36%	90.83%

Although Route 2 has marginally higher predicted health, its larger cost lowers its final score, so Route 1 wins. Selection therefore weighs health and cost together.

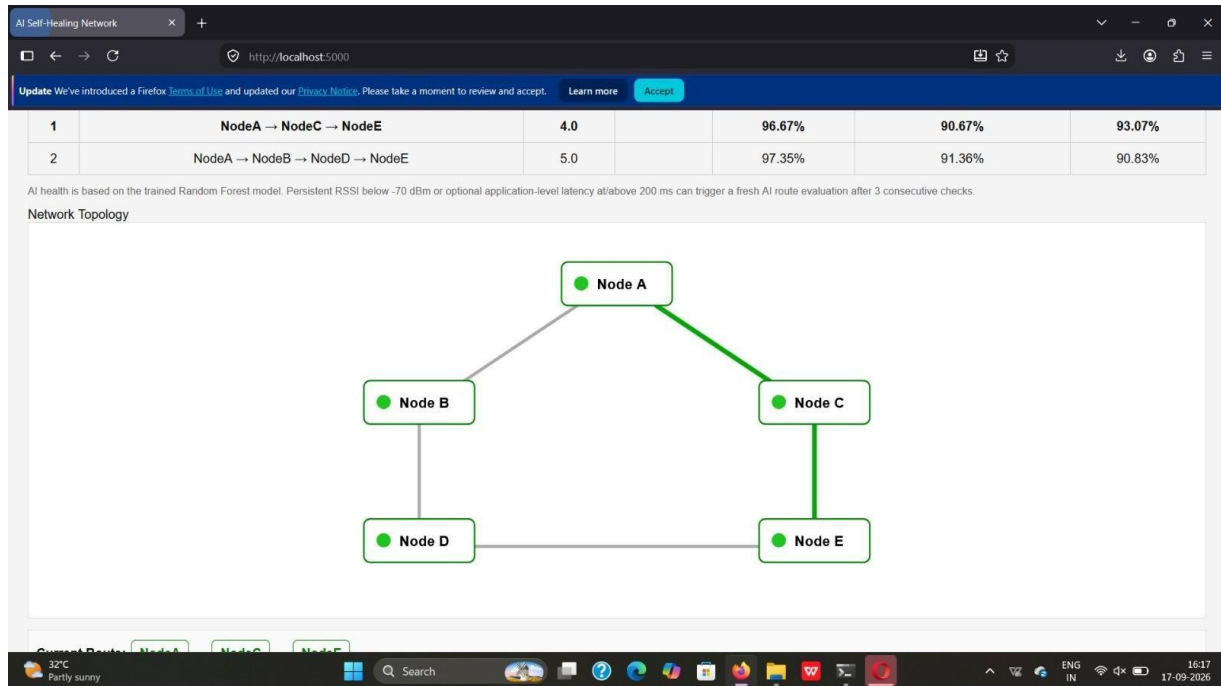


Fig. 3. Network topology on the dashboard; the active route A $\rightarrow$ C $\rightarrow$ E is highlighted in green

The topology view shows every node online with the A-C-E route highlighted. These observations show monitoring, health prediction, route scoring and route selection working together. Measurements such as prediction accuracy, recovery time, packet delivery ratio and latency under controlled failures have not been recorded yet and remain for future experiments.

VIII.

CONCLUSION AND FUTURE SCOPE

This paper presented a self-healing communication network on ESP32 hardware that combines monitoring, machine-learning health prediction and dynamic route selection. Nodes report over MQTT, and a Python backend detects failures or falling link quality. Dijkstra's algorithm supplies the routes, and predicted health plus network measurements guide the choice. If the active route is lost, another is chosen and sent to the nodes involved, while acknowledgments and retransmission keep text and image transfer dependable. In dashboard tests the system monitored all five nodes and selected A $\rightarrow$ C $\rightarrow$ E with a score of 93.07%. The prototype shows how inexpensive ESP32 devices, monitoring software and intelligent routing can make an adaptive IoT network.

Next steps are to measure prediction accuracy, recovery time, packet delivery, latency and reliability under different failures, and to test larger networks with more nodes.

ACKNOWLEDGMENT

The authors are grateful to their project guide and to the faculty of the Department of Electronics and Telecommunication Engineering, Priyadarshini College of Engineering, Nagpur, for their guidance, support and encouragement.

REFERENCES

- [1] M. U. B. Raza, M. U. Farooq, A. Ijaz, M. Manalastas, and A. Imran, "AI-Powered Resilience: A Dual-Approach for Outage Management in Dense Cellular Networks," *Computer Communications*, vol. 236, p. 108129, 2025, doi: 10.1016/j.comcom.2025.108129.
- [2] Sahin and H. Arslan, "A Self-Healing Mesh Network without Global-Time Synchronization," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2024, pp. 256–261, doi: 10.1109/ICC51166.2024.10622330.
- [3] J. Farmani and A. Khalil Zadeh, "AI-based Self-healing Solutions Applied to Cellular Networks: An Overview," arXiv:2311.02390, 2023.
- [4] M. Z. Asghar, F. Ahmed, and J. Hämäläinen, "Artificial Intelligence Enabled Self-healing for Mobile Network Automation," in *Proc. IEEE GLOBECOM Workshops*, 2021, doi: 10.1109/GCWkshps52748.2021.9681937.
- [5] [Authors], "Energy-Efficient and QoS-Aware Data Routing in Node Fault Prediction Based IoT Networks," *IEEE Trans. Netw. Service Manag.*, [vol., no., pp., year].
- [6] [Authors], "ENADL: Towards Performance Improvement of IoT Networks Using Deep Learning-Based Node Fault Prediction," *IEEE Trans. Rel.*, 2025, [vol., no., pp.].
- [7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [8] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [9] OASIS, *MQTT Version 3.1.1*, OASIS Standard, Oct. 2014.

