ShadowNet: A Single Node ESP32-Based Infrastructure-Free Local Messaging System for Isolated Environments

Satvika S
Department of Computer Science and Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Bangalore, India
bl.sc.u4aie24047@bl.students.amrita.edu

Rishi Kumar
Department of Computer Science and Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Bangalore, India
bl.sc.u4aie24042@bl.students.amrita.edu

Jeevietha Ramesh Kumar
Department of Computer Science and Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Bangalore, India
bl.sc.u4aie24017@bl.students.amrita.edu

Dr. Yasoomkari D
Department of Computer Science and Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham, Bangalore, India
d_yasoomkari@blr.amrita.edu

***Abstract*—Research or military environment where mass communication cannot take place via the internet due to the emergencies; then communication that is effective becomes crucial. In this paper, we have presented ShadowNet, a no-infrastructure, back to basics, local, single node, messaging system built upon an ESP32 micro-controller. The ESP32 is the Wi-Fi access point as well as the micro web server- it is also the internet free network. A low-weight HTTP/JSON RPC-based communication-model between clients and servers is used, and an optimised polling-process is used when the embedded systems have limited resources. The prototype created under MicroPython is implemented and tested by means of indoor experiments where clients communicate with each other. The system performance is measured against the response time of messages, capacity, and stability. It has been experimentally established that there is an average message latency of 100–300 ms and a communication range of about 15–25 meters indoors. The results suggest that a small microcontroller-based system would be useful in providing functional and infrastructure-free communication among groups in isolated settings. Also, system constraints of scalability and security have been addressed and possible paths of future enhancements outlined.**

***Index Terms*—Embedded Web Server, Emergency Networks, ESP32, Infrastructure-Free Communication, IoT Communication, Offline Messaging, Wi-Fi Access Point**

I. Introduction

Communication networks are very important in coordination during responses, disaster recovery [1], military campaigns [2], and scientific expeditions in the field. In such situations, effective and instantaneous exchange of information has an immediate bearing on the success and safety of operations. However, traditional communication networks are very dependent on infrastructure, including cell bases, broadband infrastructure, and servers. This makes these networks very unreliable in areas where disasters have occurred, remote areas, and areas where communication networks may be deliberately disabled.

To counter these issues, recent progress in the field of embedded systems as well as wireless networking has made it possible to build smaller devices that can facilitate autonomous communication. The availability of microcontroller platforms with Wi-Fi capabilities, like the ESP32 platform [3], has made it possible to develop devices that can perform all the necessary functions related to network setup as well as applications related to networking. These devices can build local networks by taking advantage of access point functionality without relying on external infrastructure.

The communication methods without the need for existing infrastructure usually use either mobile ad hoc networks [4], mesh routing, or specialized radio hardware. Although they provide scalability and range extension, they often add considerable complexity to the configuration process, power consumption [5], or the need to use client software. Such factors have impeded their usage within time-critical and constrained scenarios.

The paper introduces ShadowNet, a local message passing infrastructureless system designed and implemented on an ESP32 microcontroller. The system features a wireless network established by the system itself, along with a message passing chat functionality accessible using a standard web browser on the microcontroller. The system requires no additional applications to be installed on smartphones or laptops, or an internet connection. Messaging with other users or computers is achieved using an HTTP-based server-client protocol.

The primary contribution of this undertaking will be to provide evidence that a single ESP32 module can and will be able to perform communication tasks for multi-users in an interactive way simply through standard web components by acting as a WiFi hotspot and embedded web server.

II. Related Work

Infrastructure-independent communication has gained considerable attention with respect to managing disasters, military communications, and remote area detection [6]. Mobile ad hoc networks (MANETs) and delay-tolerant networks have been suggested to offer connectivity in infrastructure-less environments. However, these methods, although much more reliable, are associated with increased overhead in terms of deployment as well as power consumption because they are usually required to implement dynamic routing, more than one network node, etc.

Low power wireless solutions like Bluetooth Low Energy (BLE) had also been considered [7]. LoRa had also been considered. BLE solutions provide low power consumption, but data transfer rate as well as support for multi-client communication are not satisfactory. Hence, BLE solutions are not preferred for interactive multi-client messaging. LoRa solutions allow a larger communication range. However, low data transfer rates along with high latency do not support interactive communication.

The ESP32 microcontroller has been widely used for Internet of Things projects due to its integrated Wi-Fi functionality, cost-effectiveness, and versatility in networking options. Researchers have previously utilized the ESP32 platform for designing smart home automation networks [8], sensors for data acquisition [9], and the hosting of web servers on the local network [10]. Various studies have also been conducted to employ communication over the ESP32 platform through the use of mesh networks [11], the ESP-NOW communication protocol [12], time synchronized mesh protocol mechanisms [13], and optimized communication strategies for constrained IoT devices [14], including edge computing implementations [15].

On the other hand, the approach taken by ShadowNet involves a one-node architecture, where the ESP32 module acts as a wireless access point as well as an embedded web server. As a result, the solution provided by the proposal not only overcomes the shortcomings of the previous solution, i.e., the requirement of a separate routing infrastructure, but also provides access using web technology. Therefore, the system is ideal for short-range communication among small groups of people in an isolated site or a disaster scenario. Prior works have also highlighted the importance of securing such lightweight IoT deployments, particularly authentication in resource-constrained environments [16], lightweight security at the edge [17], and anomaly detection within IoT networks [18]. Furthermore, reliable communication architectures for remote and disaster-prone deployments [19] and predictive link modelling in isolated sensor networks [20] reinforce the need for robust yet simple systems such as ShadowNet.

A. Research Gaps Identified

Effective communication becomes difficult in situations where the usual networking infrastructure is either unavailable or has been destroyed or hindered by the very situations being studied or observed. Natural disasters often result in the destruction of cellular networks and internet backbone cables, and other situations such as military expeditions or scientific missions are often carried out in areas lacking network access.

Most modern communication platforms rely on routers, centralized servers, or continuous Internet connectivity and hence cannot be readily deployed in an isolated environment. Many portable communications available to date are either expensive, power-hungry, or too cumbersome to configure. This creates a need for a compact, low-power, easily deployable communication system capable of operating autonomously while it supports real-time multi-user interaction.

B. Main Contribution

The main objectives that the proposed ShadowNet framework aims for are as follows:

1) To design a fully autonomous communication system that operates without internet connectivity or external routing infrastructure.
2) To configure the ESP32 microcontroller as a wireless access point and embedded message server.
3) To provide a browser-based, real-time chat interface accessible from standard smartphones and laptops.
4) To utilize lightweight HTTP-based communication suitable for resource constrained embedded devices.
5) To implement automatic device presence detection to improve situational awareness.
6) To evaluate system performance in terms of latency, stability, and user capacity.

III. System Architecture

The ShadowNet system adopts a compact architecture for client–server interaction implemented fully on embedded hardware. A single ESP32 microcontroller acts as the central communication node, hosting wireless networking and Web hosting combined into one device for message coordination. The overall architecture of the system is given in Fig.1.

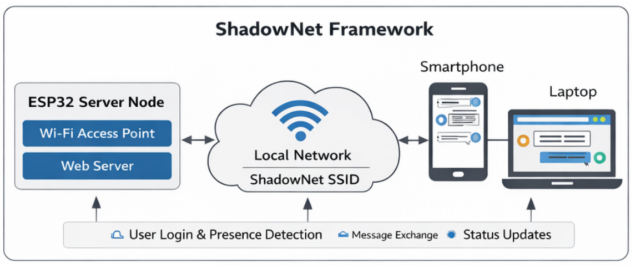

Fig. 1. System architecture of the proposed ShadowNet framework.

A. Architectural Overview

In this, the ESP32 is set up during initialization to act as a Wi-Fi access point, hence creating a wireless network within its reach. This network can serve as a type of local area network in isolation without any need for internet connectivity, external routers, or additional infrastructure. Client devices

such as smartphones or laptops connect directly to this access point using standard Wi-Fi capabilities.

After establishing a connection, client devices access the browser-based chat interface directly hosted on ESP32. The interface is stored in the microcontroller's flash memory and delivered using HTTP responses. This design avoids application installation and enables participating with any modern web browser directly.

B. Client–Server Communication Model

The system follows a light-weight client server communication infrastructure using HTTP over Wi-Fi.

1) **Server (ESP32):** The ESP32 serves as an embedded HTTP server, managing the registration of clients, the storage of messages, the retrieval of messages, and the management of the presence of users. In-memory storage is employed for the storage of active clients and the last messages sent.
2) **Clients (User Devices):** Connected devices act as clients of the HTTP protocol and as such, they start and end interactions by sending requests to deliver messages, get updates or send disconnection notifications. The interactions are all founded on the request response pattern, and therefore are compatible with the traditional browser context.

C. Message Handling and Presence Detection

To ensure that the messages are delivered on time, the system uses a polling update process. The clients send HTTP requests to the server at periodic intervals to query messages that have been received and display the availability status of the contacts. The server delivers the data in a JSON array format that holds incremental changes.

The presence detection system is developed through each client being given a distinct identity when registration is done. When a client joins and leaves the network, an automated system message is created by the server to be broadcasted to all linked users. The system improves awareness of a situation and is very helpful when dealing with an emergency or mission.

D. Design Rationale

We observe that the design emphasizes simplicity, portability, and reliability. HTTP communication is favored because of its lower complexity and availability in browsers as an already implemented functionality. Polling is implemented as an alternative for avoiding connections as it is used here for lower memory utilization and system stability on lower hardware. By integrating the access point, web hosting services, and message coordination into one ESP32 board, ShadowNet has implemented an overall self-reliant communication system.

IV. Hardware and Software Implementation

Functions in the ShadowNet system are implemented using small hardware components in order to ensure the system's portability. Simplicity in the system's design has been emphasized. At the same time, the system's complexity is reduced, enabling the system to support multi-user communication efficiently.

A. Hardware Components

The hardware configuration is intentionally minimal to support rapid deployment and field usability.

1) **ESP32 Development Board:** The ESP32 acts as the processing unit for networking. Its built-in Wi-Fi module allows for access point functionality, while its dual-core processor supports web hosting in the embedded system, as well as message coordination. The ESP32 has low memory, which affects how the system handles message buffering or polls for communication.
2) **Power Supply:** For the ESP32, the power supply is provided through the standard USB interface, so it is also able to run from the power bank. Therefore, the ESP32 project does not require fixed power infrastructure to work.
3) **Client Devices:** Any WiFi-enabled device, whether it is a Smartphone, tablet, or laptop, can be used as a client. The interactions are carried out solely using any web browser. Software installation is not required.

B. Software Components

1) **MicroPython Firmware:** MicroPython shall be used to realize the implementation of access point configuration and HTTP server functionality, request handling, and storage. Its lightweight runtime environment supports fast development while remaining compatible with resource constraints of ESP32.
2) **Web-Based User Interface:** It is designed with standard HTML, CSS, and JavaScript, and the source of this interface is stored in the flash memory of the ESP32. The interface will support user registration, display messages, and periodic updates without the use of external libraries.
3) **Communication Protocol:** In the case of the application-layer protocol, the protocol to be employed is the HTTP, and the messages, user lists and system notifications are managed in the JSON format. This enables the application to be well deployed in browser based deployment and reduced labor to disassemble the server that carries the application.

V. Implementation Details

This part describes the operation workflow process used in the ShadowNet system with specific references to how the entrenchment of networking ideas and concepts are useful in the system in order to conduct real-time communication.

A. Network Initialization

When the ESP32 module is started, it will be configured in the Wi-Fi access point mode using the Networking Module in MicroPython. The ESP32 transmits a preset network name and a predetermined password that the client devices can readily be connected to even without routers or the internet connection.

B. Embedded Web Server Operation

After the initialization of the network, the ESP32 starts a simple HTTP server on a standard port. The server receives and processes incoming requests based on the request headers, extracts the service destination, and sends back a suitable response to the client. For a client connecting to the server for the very first time, it provides a chat service through a browser-based interface stored on the client.

C. Client Registration and Identification

In order to register as a client, each such entity submits their username to the server through an HTTP request. The server will then assign the user a unique connection identifier to be stored together with their last activity time. This is useful for tracking the user sessions.

D. Message Exchange Mechanism

Message passing in ShadowNet is achieved through HTTP client-server communication. The clients send message data to the ESP32 server through HTTP POST requests. The server, upon receiving a message, locally stores this message in a memory buffer message queue and increments the global message counter.

For the clients to receive new messages and update notifications, they send periodic "polling" requests to the server. In return, the server delivers a JSON object packing only those messages created after the previous update request sent by the client. Thus, the JSON object avoids the retransmission of redundant information.

This polling form of interaction makes it compatible with the standard web browsers and predictable with regard to memory consumption even on the resource constrained ESP32 platform. The complete sequence for client registration, message submission, polling, and broadcast updates can be seen in Fig.2.

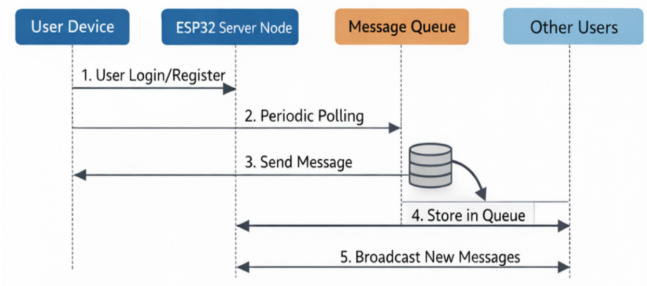

Fig. 2. Polling-based message exchange flow between client devices and the ESP32 server.

E. Presence Detection and Disconnection Handling

User presence is indicated through client activity tracking. When a client enters the system, a system notice is automatically triggered from the server. When a client disconnects or remains idle for a predetermined amount of time, the server deletes an entity and sends a leave notice to all other entities.

F. System Testing

The functionality/program is validated using a test with multiple clients at the same time. The tests involve validation of message correctness, response, and system stability within the wireless communication coverage area for ESP32 within the wireless coverage range of the ESP32.

VI. Results and Discussion

The ShadowNet system was tested under indoor conditions in a controlled environment, with regards to its use as an infrastructure-free solution for local communication. Various experiments were conducted using multiple client devices that were enabled with Wi-Fi connected simultaneously to the ESP32 access point.

A. Functional Assessment

The system performed in the following ways:

1) Creation of a wireless network locally with no access to the internet.
2) Browser-based access to the chat interface across multiple devices.
3) Message exchange.

The browser-based interface was responsive throughout, across all the connected clients, and system-generated notifications about presence improved the awareness of participant activity.

B. Metrics of Performance

System performance was assessed in terms of metrics relevant to short-range communication systems. Table I shows the observed performance metrics.

TABLE I
Observed Performance Metrics

Parameter	**Observed Result**	**Test Conditions**
Wi-Fi Coverage Range	250–500 m	Typical academic building, concrete walls
Message Latency	100–300 ms	5–8 concurrent users, 2 s polling interval
Maximum Stable Clients	5–8 users	MicroPython runtime, ESP32 Dev Board
Network Stability	Stable	Continuous operation for >30 minutes
Power Consumption	Low	USB power bank, continuous operation

C. Discussion

This experiment depicted that ESP32 microcontroller can be sufficiently used as a communication hub by small teams at remote locations. Communication using HTTP also guarantees uses with widely used web browsing instead of the specialization programming software.

Scalability is limited by the capabilities of the hardware under use but the resulting throughput is sufficient when

responding to an emergency, carrying out research, and executing tactical missions at close ranges. The architecture can run on manageable hardware, but is more expensive as more users are added.

VII. Conclusion and Future Work

ShadowNet is presented in this paper as a single-node, infrastructureless local communication network that should be used in the environment where the internet is inaccessible or unreliable. Using the ESP32 programmed to operate as a wireless access point, embedded web server, and message coordination module at once, real-time communication between or without an existing network infrastructure can be achieved between multiple users. The experimental findings reveal consistent system performance, reasonable message latency and proper multi-user interaction between the wireless network coverage of the system and this means that the resource-efficient use of HTTP-based communication with polling mechanism as a means of real-time interaction over resource-constrained embedded systems is possible. The present implementation has minimum scalability and powerful security, though this ensures that further improvement is possible. In the future efforts, encrypted communication protocols will be incorporated, event-based approaches like WebSockets adopted, a multi-node mesh networking deployed to enhance the range of a system, and storage and sensor-based context awareness will be introduced to enhance the functionality of the system.

Even though ShadowNet is feasible, it is limited in its scalability because the number of simultaneous users is limited by the relatively limited memory and processing capability of the ESP32 and MicroPython, which is aggravated by the polling-based communication mechanism. Also, the lack of encrypted communication, the reliance on the short range of the Wi-Fi of ESP32, and the temporary nature of message storage, make it less applicable to security-sensitive and large-scale applications, which suggests further optimization.

References

[1] S. K. Singh and P. Singh, "Decentralized and infrastructure-free communication solutions for disaster management," *Int. J. Communication Systems*, vol. 33, no. 6, 2020, doi: 10.1002/dac.4466.

[2] M. Matracia, N. Saeed, M. A. Kishk, and M.-S. Alouini, "Post-disaster communications: Enabling technologies, architectures, and open challenges," *arXiv preprint arXiv:2203.13621*, 2022.

[3] A. A. Nascimento, M. A. Teixeira, and L. Pirmez, "Performance evaluation of ESP32 microcontroller for IoT applications," *IEEE Latin America Transactions*, vol. 18, no. 9, pp. 1544–1551, Sep. 2020, doi: 10.1109/TLA.2020.9154126.

[4] J. A. Stankovic, T. He, Y. Tian, T. F. Abdelzaher, C. Lu, and L. Sha, "Real-time communication and coordination in wireless sensor networks," *Proceedings of the IEEE*, vol. 91, no. 7, pp. 1002–1022, Jul. 2003, doi: 10.1109/JPROC.2003.814918.

[5] P. K. Sharma and S. Park, "Lightweight communication protocols for constrained IoT devices," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1050–1077, 2020, doi: 10.1109/COMST.2019.2959702.

[6] A. Adeel, M. Gogate, S. Farooq, C. Ieracitano, K. Dashtipour, H. Larijani, and A. Hussain, "A survey on the role of wireless sensor networks and IoT in disaster management," *arXiv preprint arXiv:1909.10353*, 2019.

[7] S. Sendra, J. Lloret, M. Garcia, and J. Tomas, "Performance evaluation of IoT communication protocols for emergency applications," *IEEE Access*, vol. 8, pp. 55805–55814, 2020, doi: 10.1109/ACCESS.2020.2981153.

[8] R. Silva, J. Sá Silva, and F. Boavida, "Performance evaluation of Wi-Fi-based IoT systems using ESP32," *Sensors*, vol. 21, no. 4, Art. no. 1125, 2021, doi: 10.3390/s21041125.

[9] M. S. Munir, M. A. Khan, and S. A. Khan, "Performance analysis of Wi-Fi enabled microcontrollers for IoT applications," *IEEE Access*, vol. 9, pp. 118734–118746, 2021, doi: 10.1109/ACCESS.2021.3106895.

[10] D. Hercog, T. Lerher, M. Truntič, and O. Težak, "Design and implementation of ESP32-based IoT devices," *Sensors*, vol. 23, no. 15, Art. no. 6739, 2023, doi: 10.3390/s23156739.

[11] M. J. Espinosa-Gavira, A. Agüera-Pérez, J. C. Palomares-Salas, J. M. Sierra-Fernandez, P. Remigio-Carmona, and J. J. González-de-la-Rosa, "Characterization and performance evaluation of ESP32 for real-time synchronized sensor networks," *Procedia Computer Science*, vol. 237, pp. 261–268, 2024, doi: 10.1016/j.procs.2024.05.104.

[12] M. Islam and M. Hossain, "Design of a local wireless communication system using ESP32 access point mode," in *Proc. IEEE Int. Conf. on Informatics, Electronics and Vision (ICIEV)*, 2021, pp. 1–6, doi: 10.1109/ICIEV50855.2021.9381136.

[13] M. Pister and L. Doherty, "TSMP: Time synchronized mesh protocol," in *Proc. IEEE Int. Conf. on Distributed Computing in Sensor Systems (DCOSS)*, 2018, pp. 119–128, doi: 10.1109/DCOSS.2018.00022.

[14] D. C. Smetters, D. J. Troy, J. W. Wills, and T. F. Abdelzaher, "Optimizing push notifications for constrained IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6462–6473, Aug. 2019, doi: 10.1109/JIOT.2019.2914780.

[15] Y.-H. Chang, F.-C. Wu, and H.-W. Lin, "Design and implementation of ESP32-based edge computing for object detection," *Sensors*, vol. 25, no. 6, Art. no. 1656, 2025, doi: 10.3390/s25061656.

[16] K. Nimmy, S. Sankaran, K. Achuthan, and P. Calyam, "Lightweight and privacy-preserving remote user authentication for smart homes," *IEEE Access*, vol. 10, pp. 176–190, 2022, doi: 10.1109/ACCESS.2021.3137175.

[17] M. Singh and S. Sankaran, "Lightweight security architecture for IoT edge devices," in *Proc. IEEE Int. Symp. on Smart Electronic Systems (iSES)*, 2022, pp. 455–458, doi: 10.1109/iSES54909.2022.00066.

[18] P. K. Binu, M. Kiran, and M. V. Sreehari, "Attack and anomaly prediction in IoT networks using machine learning approaches," in *Proc. IEEE 4th Int. Conf. on Electrical, Computer and Communication Technologies (ICECCT)*, 2021, pp. 1–6, doi: 10.1109/ICECCT52121.2021.9616794.

[19] S. Kumar and G. Gutjahr, "QoS in ultra-low memory green IoT nodes for disaster management applications," in *Proc. 3rd IEEE Int. Conf. on Mobile Networks and Wireless Communications (ICMNWC)*, 2023, doi: 10.1109/ICMNWC60182.2023.10435944.

[20] S. Surendran and M. V. Ramesh, "Link characterization and edge-centric predictive modelling in an ocean network," *IEEE Access*, 2023, doi: 10.1109/ACCESS.2023.3235387.