DOI: https://doi.org/10.53052/9788367652148.25

Michał RAJZER[1], Bartłomiej SZKODNY[2]

Opiekun naukowy: Jacek IZYDORCZYK[3]

ROZWINIĘCIE MOŻLIWOŚCI PROTOKOŁU SIECI MESH POPRZEZ POWIĘKSZENIE ILOŚCI MOŻLIWYCH POŁĄCZEŃ Z UŻYCIEM WARSTWY OPROGRAMOWANIA

Streszczenie: Podczas gdy wiele protokołów sieciowych jest ogólnodostępnych, wiele z nich ma swoje ograniczenia. Ten artykuł skupia się na rozwiązaniu problemów dostępnego komercyjnie protokołu ESP-Now. Proponowane rozwiązanie jest dostępne online, na platformie GitHub. Jest działającą korektą na ograniczenia ESP-Now, jak na przykład ograniczona liczba węzłów w sieci.

Słowa kluczowe: ESP-NOW, sieć mesh, szyfrowanie.

EXTENDING THE ABILITIES OF A MESH NETWORK PROTOCOL BY EXTENDING THE POSSIBLE CONNECTIONS WITH A SOFTWARE LAYER

Summary: While many network protocols are available, many of them have limitations. This paper explores a solution to the issues of a commercially available protocol, ESP-Now. The proposed solution is open-sourced and available on GitHub. It is a viable fix for the limitations of ESP-Now, such as a small number of maximum nodes in a network.

Keywords: article, ESP-NOW, mesh network, encryption.

1. Introduction

Many mesh networks for use in IoT devices, such as Zigbee and Thread, have been devised recently. As such, many manufacturers have developed standards that aim to repurpose existing solutions to compete in the quickly evolving space. For example, Espressif have released a protocol called ESP-NOW, which allows the device manufacturer to connect up to 20 devices. This is a significant limitation of the

[1] Politechnika Śląska, AEiI, Informatics, michal.rajzer03@gmail.com
[2] Politechnika Śląska, AEiI, Informatics, szkobart@outlook.com
[3] Prof. dr hab. inż., Politechnika Śląska, AEiI, jacek.izydorczyk@ieee.org

protocol, which is otherwise well-rounded, especially for electronics amateurs, as it requires little preparation and is pretty cheap to implement. With this in mind, there was an apparent need to expand the system's capabilities. The authors of this paper created such an expansion of the system, and it has been open-sourced at *https://github.com/lrajzer/ESP-Meshed*.

2. ESP-NOW technical specification

As stated above, ESP-NOW allows up to twenty devices to communicate. This is accomplished using a predefined packet structure pictured below [1].

Table 1. ESP-NOW Message format

MAC Header	Category Code	Organization Identifier	Random Values	Vendor Specific Content	FCS
24 bytes	1 byte	3 bytes	4 bytes	7~257 bytes	4 bytes

Table 2. The Vendor Specific Content

Element IDr	Length	Organization Identifier	Type	Version	Body
1 byte	1 byte	3 bytes	1 byte	1 byte	0~250 bytes

Furthermore, the ESP-NOW API allows for a so-called broadcast mode. All nodes on the same network receive the message. This is accomplished by setting the MAC Header only to comprise 1's. This mode is used in the presented solution to increase the number of available devices and allow us to add several additional features.
In the ESP-NOW API, the only easily accessible fields of the packet are MAC Header, Message Length, and Body. In this situation, there are at most 274 bytes to manipulate, but changing the MAC header is impossible due to the use of broadcast mode. Therefore, there are only 250 bytes available. With this in mind, it was decided to minimize the memory overhead when creating the "packet structure" for the solution. The scheme illustrated below was created to balance the connected devices and the data being transmitted.

Table 3. Devised header format

Message ID	Control Code	Receiver Address	Sender Address	Message Data
13 bits	3 bits	12 bits	12 bits	0~245 bytes

This scheme was deemed optimal as it allows for a network of up to 4096 devices, with the flexibility to enable self-healing, acknowledgements, and automatic network configuration. The exact functions and mechanisms of the proposed solution are outlined in the following sections.

3. Mechanics of the system

3.1. Message ID

Each message has its semi-unique ID. The message ID is created when the message is sent by creating a random 12-bit number using the sender ID as the initial seed. The last bit of the ID is used for message acknowledgements, which are defined in detail below.

3.2. Control code bits

The devised header format allocates space for a 3-bit control code. It is used for network connection, encryption and speed optimisation. Each of these processes is described in detail below. The specific codes have the meanings as follows:

Table 3. Control codes (in binary) and their definitions

Control code (in binary)	Definition
000	Normal packet
001	Extended mode packet
010	Ping request
011	Ping response
100	Possible speeds request
101	Possible speeds response
110	Public key request
111	Public key response

3.3. Extended mode packets

Due to planned expansions to the system, the control code 001 was reserved as the extended mode packet. This type of packet is used now for exchanging private encryption keys but may be changed later.

3.4. Message relaying

The main distinguishing feature between the software layer presented and the standard mode of operation is the possibility to connect more devices and multi-hop communication. In the presented implementation, it was opted to use a crude and naïve approach to reduce the program complexity and the processing time needed for each node. This allows the system to omit computationally or memory-expensive routing algorithms[4][5]. To send data, the device broadcasts a message to its neighbours, which they then broadcast to their respective neighbours, who then repeat the process. To ensure that two or more nodes will not send the same message between themselves indefinitely, each node stores a list of message IDs it has transmitted in a user-defined period.

3.5. Device connection

New nodes are added to the network in two modes, either with a designated ID or a self-assigned one. The self-assigned ID process is quite complex, so a complimentary flowchart was created to illustrate the steps.

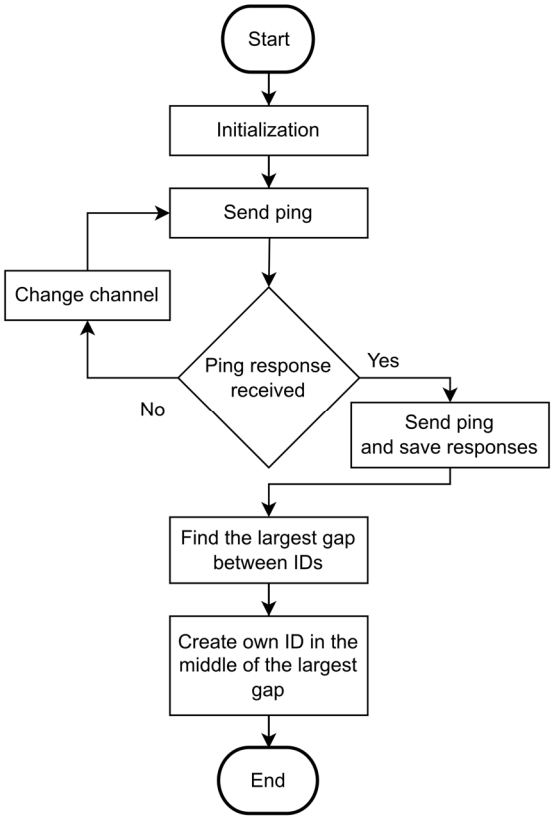

Figure 1. Connection algorithm

The designated ID connection protocol is quite simple. The node first searches through the channels to find a network. It may look for a network with a specific ID if several separate networks are detected. It then joins the network using its specific ID.

3.6. Network creation

A mesh network created using broadcast mode is easily resizable and can easily cover large areas. The connection of the devices can theoretically span up to 1km[2] in ideal conditions. Due to the use of a naïve routing algorithm, the user does not need to change the structure of the code. The network also does not need to reconfigure itself. Node has two initial modes: MJ (Mesh Join) and MC (Mesh Create). Upon boot up, the node is in the MJ mode described in the Device connection section. If the device

fails to acquire a connection, it then switches to the MC mode. It searches for the least used channel. The least used channel is chosen as the channel of the new network.

3.7. Speed selection

Depending on the device, the network speed necessary is quite different. For example, a node that creates a massive amount of data needs a high throughput network, whilst a battery-powered node with an e-paper display would be better served with a lower speed to conserve energy. Thus, there exists a great need for speed negotiation. In this system, it was implemented by the nodes exchanging a handshake upon first contact, where they exchange their preferred maximum speed and default to the lower of the two.

3.8. Acknowledgements

It is quite possible for a packet to get lost in a mesh network, so an acknowledgement system was implemented. After sending a packet, the sender node waits for a user-determined time for an acknowledgement packet. If such a packet does not reach the sender node, the node retransmits the packet, and the cycle repeats a user-determined number of times. On the receiver node, once a packet is received, it is assumed that the network is intact, and the acknowledgement is sent only once without an acknowledgement for the acknowledgement.

4. Security of the system

4.1. End-to-end encryption

The proposed solution has an inbuilt system to establish end-to-end encryption between two nodes. When two nodes need a secure connection, the first one to communicate creates a key that will be used for symmetric encryption. It then uses control codes to access the public key of the second node and encrypts the symmetric key using the public key. It is transmitted over the network just as any other packet would be. Upon receiving the acknowledgement, both nodes switch to using the symmetric encryption between themselves.

4.2. Possible vulnerabilities

Understanding mesh network vulnerabilities is essential in creating a secure environment. Mesh network vulnerabilities include, but are not limited to, DoS, masquerading, man-in-the-middle attacks, repudiation, replaying, and snooping. Most of these are described below.

DoS is an active kind of attack. It can slow down or interrupt the service of the network. The attacker targets one of the mesh network nodes and pumps too many fake requests so that the node effectively becomes disconnected.

Masquerading is when one impersonates another node. The proposed system will be protected from this kind of attack by encryption.

Modification is a tactic where the attacker gains access to a message, trying to modify the information for their benefit. They can delay or delete the messages to harm the system.

Repudiation is the type of attack where the attacker impersonates the node already using encryption in the mesh network. The proposed system is semi-vulnerable to repudiation since the nodes to which the impersonated node was connected will enter a locked state as two nodes with one address will try to set differing symmetric encryption keys.
Replaying, the attacker anyhow obtains the copy of the sent message. He can then emit the message, showing himself as a valid service provider. The user can overcome it, but it was not implemented in the proposed solution.
Snooping is the most common attack vector, which relies on an attacker listening to messages coming through the network. In the proposed system, encryption is strictly necessary to protect against this attack.

5. Conclusion

Extending a mesh network protocol's capabilities is possible using a software layer. The proposed solution can accommodate up to 4096 nodes in a single network, while the original protocol only allowed 20. At the same time, the solution allows for low power consumption by using a naïve packet routing system while at the same time being resilient against many types of attacks.

LITERATURE

1. ESP32 Documentation – ESP-NOW, *https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html,* 20.09.2023
2. ESP32 Documentation – WiFi modes, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html*,* 10.10.2023
3. GUPTA S.K., et al.: Wireless Mesh Network Security, Architecture, and Protocols. Security and Privacy Issues in Sensor Networks and IoT, IGI Global, Hershey, PA, 2020, 1-27.
4. CILFONE A., DAVOLI L., BELLI L., FERRARI G.: Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies", Future Internet 2019, 11(4), 99.
5. ALAMERI A., KOMARKOVA J., AL-HADHRAMI T.: Fuzzy-based optimization of AODV routing for efficient route in wireless mesh networks, PeerJ Computer Science, 2023, 9(1), 1508.