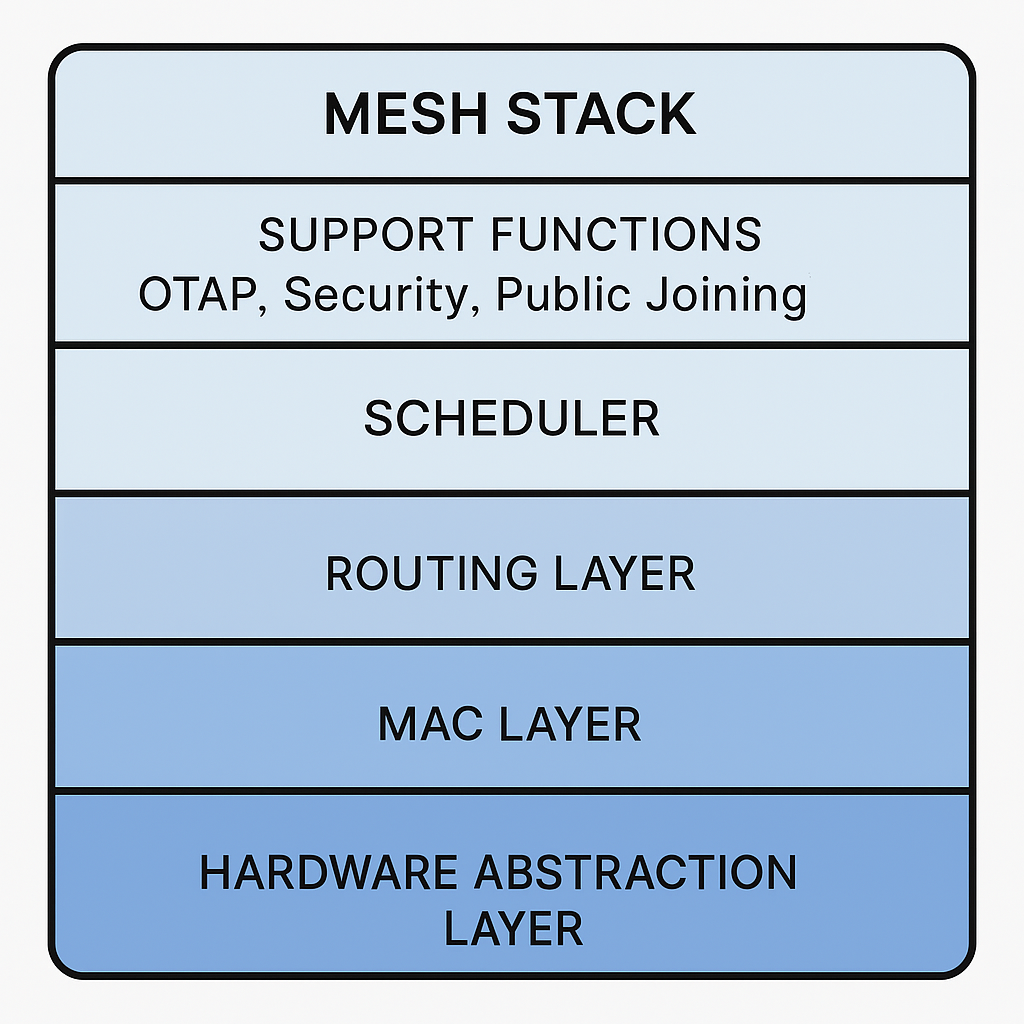
Let’s dive deeper into the Mesh Stack and Application in the context of Single-MCU and Dual-MCU architectures, explaining their components, roles, and how they work together to create a functional IoT system.

### **1. Mesh Stack**

The Mesh Stack is the core networking software provided by Wirepas. It is responsible for enabling communication between devices in the Wirepas Mesh network. Think of it as the "communication brain" of the device.



#### **Components Inside the Mesh Stack**

1. Hardware Abstraction Layer (HAL):
   * Provides a standardized interface to interact with the hardware (e.g., radio chip, timers, interrupts).
   * Abstracts the hardware details so the stack can run on different chips without modification.
2. MAC Layer (Medium Access Control):
   * Manages low-level communication tasks like accessing the radio frequency, avoiding collisions, and ensuring data is sent and received correctly.
   * Handles time-slotted communication to optimize energy efficiency and avoid interference.
3. Routing Layer:
   * Implements multi-hop communication, allowing data to travel across multiple devices to reach its destination.
   * Ensures efficient routing of messages even in large, dense networks.
4. Scheduler:
   * Manages the timing of tasks to ensure the application and the Mesh stack can share the same MCU without conflicts.
   * Prioritizes tasks like sending/receiving data, processing updates, and running application logic.
5. Support Functions:
   * OTAP (Over-The-Air Programming): Enables firmware updates over the network.
   * Security: Ensures encrypted communication and secure key management.
   * Public Joining: Allows new devices to join the network securely.

#### **Why the Mesh Stack is Important**

* Communication: It enables devices to communicate with each other in a decentralized, scalable, and reliable manner.
* Energy Efficiency: The stack is optimized for low-power operation, making it ideal for battery-powered IoT devices.
* Scalability: It supports large networks with thousands of devices, ensuring efficient routing and minimal latency.
* Flexibility: The stack abstracts hardware details, allowing developers to focus on the application without worrying about the underlying communication protocols.

### **2. Application**

The Application is the custom software written by developers to define the specific functionality of the IoT device. It sits on top of the Mesh stack and uses the stack’s APIs to interact with the network.

#### **Components Inside the Application**

1. Business Logic:
   * Defines the specific tasks the device performs, such as:
     + Reading sensor data (e.g., temperature, humidity).
     + Controlling actuators (e.g., turning on lights, motors).
     + Sending or receiving data packets.
2. Peripheral Control:
   * Code to interact with hardware peripherals like GPIOs, I2C, SPI, UART, ADC, etc.
   * For example:
     + Reading data from a temperature sensor via I2C.
     + Controlling an LED via GPIO.
3. API Calls:
   * Uses the Single-MCU API or Dual-MCU API to interact with the Mesh stack.
   * For example:
     + Sending a message to another node in the network.
     + Querying the network status or configuration.
4. Custom Features:
   * Developers can add features specific to their product, such as:
     + Data logging.
     + Custom communication protocols.
     + Integration with external systems (e.g., cloud platforms).

#### **Why the Application is Important**

* Device Functionality: The application defines what the device does, making it useful for specific use cases (e.g., smart lighting, asset tracking, environmental monitoring).
* Customization: Developers can tailor the application to meet the exact requirements of their product or solution.
* Interaction with the Network: The application uses the Mesh stack to send and receive data, enabling the device to participate in the IoT network.

### **How the Mesh Stack and Application Work Together**

The Mesh stack and the application are like two parts of a team:

* The Mesh stack handles all the communication and networking tasks, ensuring the device can send and receive data reliably.
* The Application focuses on the device’s specific tasks, such as reading sensors, controlling actuators, or processing data.

#### **Single-MCU Architecture**

* Both the Mesh stack and the application run on the same chip.
* The scheduler ensures they share the MCU’s resources without conflicts.
* Example: A smart light bulb where the Mesh stack handles communication, and the application controls the light based on received commands.

#### **Dual-MCU Architecture**

* The Mesh stack runs on one chip (Radio IC), while the application runs on a separate chip (Application MCU).
* The two chips communicate via UART or USB.
* Example: A smart thermostat where the Mesh stack handles communication, and the application processes temperature data and controls the HVAC system.

### **How This Helps in IoT Development**

1. Separation of Concerns:
   * The Mesh stack handles complex networking tasks, so developers can focus on building the application.
   * This reduces development time and complexity.
2. Scalability:
   * The Mesh stack is optimized for large networks, allowing developers to build scalable IoT solutions.
3. Flexibility:
   * Developers can choose between Single-MCU and Dual-MCU architectures based on their hardware and performance requirements.
4. Energy Efficiency:
   * The Mesh stack’s low-power design ensures long battery life for IoT devices.
5. Reliability:
   * The Mesh stack’s robust routing and communication protocols ensure reliable data transmission, even in challenging environments.

### **Analogy Recap**

* The Mesh stack is like the kitchen system in a pizza shop, handling all the behind-the-scenes tasks like managing the oven and ensuring pizzas are delivered to the right tables.
* The Application is like the chef, focusing on cooking pizzas and interacting with customers