JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

REAL-TIME DYNAMIC BANDWIDTH MANAGEMENT IN IOT USING ESP32

[1]Abhiram K V, [2]Advith Rai M, [3]Asmi, [4]Deekshith M Rai Author
[1234]Final Year B.E. Students, Department of Computer Science & Engineering,
Srinivas Institute of Technology, Mangaluru, India

Abstract: In modern Internet of Things (IoT) ecosystems, managing bandwidth among a growing number of devices has become a critical challenge. This paper introduces NETGENIE, a real-time bandwidth optimization framework using the ESP32 microcontroller. The system enables dynamic prioritization of devices and smart bandwidth allocation based on real-time metrics. Administrators can control device behavior via a web interface, enabling a hands-on approach to managing traffic demands. The platform tracks parameters such as latency, packet loss, and device activity to adapt bandwidth allocation accordingly. This ensures high-priority devices operate with low latency, while non-critical devices are handled efficiently. With its real-time monitoring, energy-efficient operations, and fault-tolerant features, NETGENIE significantly improves the scalability, responsiveness, and robustness of IoT networks. Its application spans across domains such as smart homes, industrial automation, and connected healthcare.

Index Terms **- IoT, Bandwidth Management, ESP32, Real-Time Monitoring, Device Prioritization, Network Optimization.**

1.INTRODUCTION

With The exponential rise of smart devices has put an increasing strain on network bandwidth in IoT environments. Conventional allocation techniques typically assign fixed or static bandwidth per device, regardless of the criticality or nature of its operation. This rigid model can lead to inefficiencies, particularly when time-sensitive applications compete with background processes for bandwidth. To address this, NETGENIE was developed as a flexible bandwidth management system leveraging an ESP32 microcontroller. The core idea revolves around assigning dynamic priority levels to devices and redistributing bandwidth based on ongoing usage and task importance.

Devices with critical real-time functions receive more bandwidth and reduced latency, while non-urgent operations operate in power-saving modes. NETGENIE also incorporates an easy-to-use web dashboard that enables administrators to visualize device performance and modify priority levels in real time. This adaptability ensures effective network resource distribution and forms the foundation for smarter, more scalable IoT infrastructures.

2. LITERATURE SURVEY

Literature surveys serve as a foundation for any research by reviewing prior studies, theories, and technological implementations relevant to the topic. They provide a contextual understanding of the current landscape, highlight unresolved challenges, and identify areas where new contributions are needed. In the field of IoT network optimization, existing literature offers valuable insights into traffic management, energy efficiency, and system scalability. Several studies have addressed the complexity of managing large-scale IoT infrastructures. The most common areas of focus include improving communication protocols, reducing energy consumption, enhancing security, and handling diverse traffic patterns. The findings from these works offer both theoretical and practical guidance for developing adaptive systems like NETGENIE. Below are four key contributions that influenced the direction and design of our project.

2.1 Network Optimization in the Internet of Things

With the evolution of the Internet from fixed endpoints to billions of connected devices, managing traffic efficiently in IoT networks has become a significant concern. IoT devices often have limited power and processing capabilities, yet they generate a continuous stream of data. This situation creates pressure on bandwidth and demands new methods to optimize routing, manage congestion, and ensure seamless scalability. One study conducted a comprehensive analysis of optimization strategies in IoT environments. The research focused on solving problems such as packet loss, energy usage, and QoS (Quality of Service) by employing dynamic and adaptive algorithms.

These methods aim to ensure that limited network resources are allocated intelligently based on data sensitivity and real-time requirements. The uniqueness of this work lies in its structured review of optimization techniques and its ability to compare different solutions across multiple network parameters. It highlights the pressing need for adaptable systems capable of adjusting to changing traffic patterns. NETGENIE draws from this perspective by using real-time traffic metrics to control bandwidth allocation and device priorities dynamically.

2.2 Edge-Intelligent Networking for Smart Cities

Smart cities rely on massive IoT networks that demand highly reliable and secure communication. Traditional topologies fall short in environments that are constantly changing or vulnerable to cyber threats. One paper addressed this issue using edge

intelligence and distributed learning to improve robustness in network architecture. The authors proposed a system that offloads decision-making to the edge of the network, reducing the dependency on central servers.

Using reinforcement learning, the network topology could adjust based on environmental changes, failures, or load shifts. Importantly, the solution was designed to work on ordinary devices, avoiding the need for specialized or expensive hardware like GPUs. This concept aligns with NETGENIE's design principle of decentralized control. By allowing the ESP32 to handle bandwidth decisions at the node level, the system mirrors edge-based frameworks in functionality but with a simpler, more accessible implementation. It proves that intelligent networking does not require high complexity to deliver reliable performance.

2.3 Network Optimization in Industrial IoT (IIoT)

In industrial environments, IoT systems face stricter demands for reliability, low-latency communication, and secure data transfer. A noteworthy paper modelled IIoT as a geometric graph and introduced a virtual coordinate system (VCS) to improve node placement for optimal communication. The key challenge addressed was selecting reference nodes that minimize error across the network. The authors developed a mathematical formulation for node placement and proposed an iterative approach for choosing new reference nodes. Their method enhanced packet delivery rates and reduced total energy consumption compared to random placement techniques.

These improvements were verified using simulation results, showing clear performance gains. While NETGENIE does not use VCS, the idea of intelligent node behaviour and adaptive structure inspired the system's load balancing logic. Instead of optimizing physical layout, NETGENIE focuses on virtual optimization by dynamically reallocating bandwidth and adjusting device priority in real-time to maintain communication efficiency.

2.4 Green Communication in IoT Using Hybrid Algorithms

As IoT networks grow, their cumulative energy usage becomes a concern, especially for battery-operated sensor nodes. One study proposed a hybrid optimization algorithm combining the Whale Optimization Algorithm and Moth Flame Optimization (MFO) to improve energy efficiency. It selected cluster heads based on parameters like residual energy, network load, and device temperature. The system dynamically re-evaluated which nodes should handle communication to extend overall network life. Compared to conventional models, the hybrid approach demonstrated better results in simulations by maintaining high performance with reduced energy drain.

This showed that smart algorithmic control could significantly impact sustainability. NETGENIE adopts similar principles of green communication but applies them through device prioritization and power-saving modes. When devices are marked as low priority, they operate in energy-efficient states, reducing unnecessary transmission. This not only conserves power but also helps avoid network congestion by focusing resources on critical tasks.

3. SYSTEM DESIGN

3.1 Data Flow Diagram

The data flow diagram (DFD) serves to map the movement of information within the NETGENIE system. It visually outlines how inputs from different sources are processed and how results are stored or acted upon. The DFD begins with inputs from the administrator—such as priority changes—and from the IoT devices, which continuously send usage metrics and performance data to the system. These inputs move through various internal processes, including traffic monitoring, bandwidth allocation, and energy optimization. Each process updates a central database that stores metrics, configuration data, and device statuses. This centralized storage plays a crucial role in enabling the system to make informed decisions quickly and accurately. Real-time data is also fed back to the administrator via the dashboard, ensuring complete transparency.

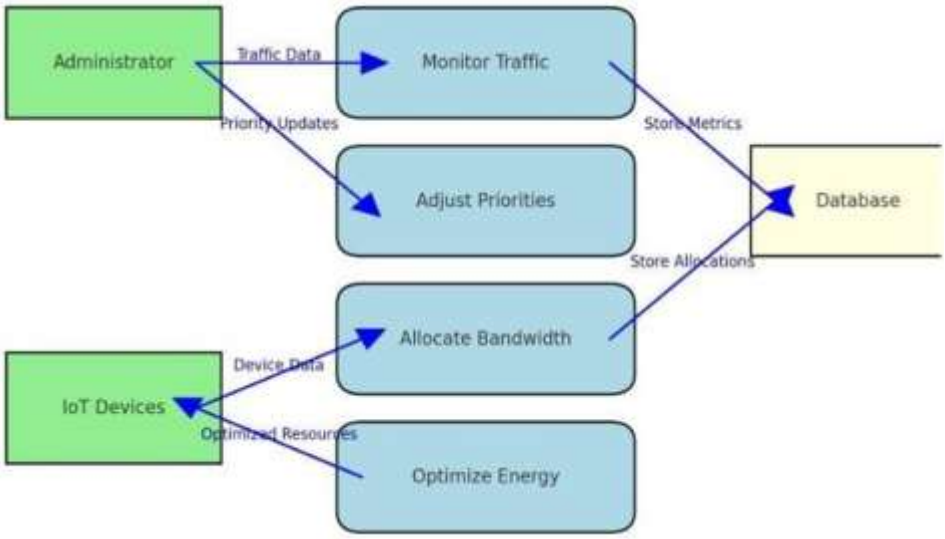

Fig 3.1: -Data Flow Diagram

The DFD underscores the system's modularity and clarity. Each function operates semi-independently but contributes to a unified goal-efficient network management. By breaking the system down into logical processes and data interactions, the DFD simplifies understanding while maintaining a high level of technical detail. This makes it easier to troubleshoot, enhance, or scale the system in the future.

***3.2* Activity Diagram**

The activity diagram illustrates the step-by-step operation of the NETGENIE system, much like a flowchart. It maps both administrator actions and system responses from start to finish. The process starts with system initialization, where all connected

devices are assigned a low-priority status by default. This creates a baseline from which future actions are carried out based on real-time data. Following initialization, the system constantly monitors the traffic generated by each device. These metrics are presented to the administrator through an interactive dashboard. The admin can then adjust device priorities, which prompts the system to reallocate bandwidth in real time. High-priority devices are given more resources, while low-priority ones are shifted into energy-saving modes. If a critical device fails, the system redistributes tasks automatically and notifies the admin.

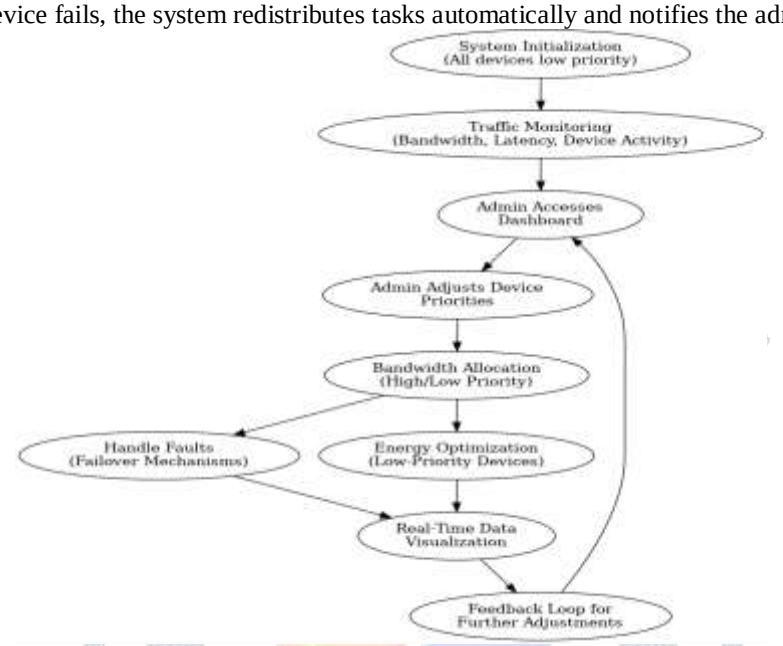

Fig 3.2: -Activity Diagram

This iterative cycle continues with real-time visual updates and system feedback, allowing for continuous optimization. By showing the entire lifecycle of an operational loop, the activity diagram highlights the system's adaptability and fault tolerance. It captures the essence of NETGENIE's design philosophy: to enable intelligent, hands-free network management while retaining admin oversight when needed.

4. IMPLEMENTATION

4.1 Arduino IDE

Arduino IDE served as the central development environment for writing and uploading code to the ESP32 microcontroller. Its simplicity and extensive library support made it an excellent choice for programming tasks such as traffic analysis, priority management, and bandwidth control. Developers used it to write modular C code that enabled seamless interaction between the ESP32 and connected IoT devices. A critical advantage of the Arduino IDE is its built-in serial monitor and debugging tools, which allowed developers to simulate faults and test how the system responded to them. This ensured that NETGENIE would remain stable and responsive under stress conditions.

The IDE's support for real-time data monitoring also allowed developers to verify that bandwidth distribution algorithms were functioning as intended during active testing. In addition to its debugging capabilities, the IDE's library ecosystem helped speed up development by providing pre-built modules for network connectivity, sensor management, and HTTP/Web Socket communication. These features made Arduino IDE not just a programming platform, but an essential tool for ensuring system robustness and expandability.

4.2 Canva API

To enhance the visual quality of the admin interface, the Canva API was employed to create visually appealing dashboard components. Though Canva is primarily a design tool, its API allows developers to create custom templates for user interfaces, which helped streamline the layout of metrics like latency, bandwidth, and device statuses. Using Canva's templates, the admin dashboard was equipped with charts, graphs, and color-coded indicators to display real-time system status. This visual layer made it easier for administrators to make decisions based on clearly presented data. Integration with ESP32 meant that these visuals updated automatically as device behaviour changed.

Moreover, the Canva API enabled interface customization based on deployment context—whether for smart homes, healthcare, or industrial use. Its flexible design options ensured that NETGENIE's dashboard remained accessible and informative across different domains, while also maintaining a professional aesthetic.

4.3 ESP32

The ESP32 microcontroller served as the heart of the NETGENIE system. With built-in Wi-Fi and Bluetooth support, it handled both communication and processing functions without needing additional hardware. At start up, the ESP32 connects to a designated network, authenticates itself, and establishes a secure connection with the admin dashboard. Throughout operation, the ESP32 monitors connected devices and collects data on bandwidth usage, response times, and network status. This information is used to dynamically allocate bandwidth based on device priority.

High-priority tasks are given more processing resources, while low-priority devices are assigned energy-efficient settings. This smart management reduces latency and extends device lifespan. To maintain resilience, the ESP32 supports advanced features such as deep sleep modes, watchdog timers for auto-recovery, and firmware-over-the-air (FOTA) updates. These capabilities ensure the system remains efficient, secure, and up to date, even in large-scale, mission-critical IoT environments like smart cities or healthcare networks.

4.4 Programming Languages and Interface Technologies

The NETGENIE system integrates several programming languages and web technologies to support both device-level operations and user interaction through a dashboard. At the core of the system, the **C programming language** is used to develop firmware for the ESP32 microcontroller. C provides the necessary control over memory and hardware to manage real-time data processing, bandwidth allocation, and energy optimization. Its efficiency makes it ideal for resource-constrained devices like the ESP32, allowing smooth execution of tasks with minimal delay or resource usage.

For the user-facing interface, a combination of **HTML, CSS, and JavaScript** is utilized to build a responsive and interactive web dashboard. **HTML** serves as the structural foundation, organizing elements such as tables, graphs, and input forms into a clear layout. This structure enables system administrators to navigate the panel easily and view live data about connected IoT devices. Meanwhile, **CSS** enhances the visual appearance of the interface. It defines colours, spacing, font styles, and responsive design features to ensure the dashboard is aesthetically consistent across different screen sizes and devices.

To enable live updates and dynamic content rendering, **JavaScript** is employed as the primary scripting language. It handles communication with the backend using Web Sockets and AJAX, allowing real-time updates without reloading the page. JavaScript libraries like Chart.js are used to create visualizations that reflect bandwidth usage, latency patterns, and priority changes. This integration of technologies ensures that administrators can make immediate adjustments and receive instant feedback, resulting in a system that is not only efficient but also intuitive to operate.

5. RESULTS AND DISCUSSION

The NETGENIE system was tested in a simulated IoT network environment to evaluate its real-time bandwidth management capabilities. The admin dashboard, shown in Figure 5.1, provided a comprehensive view of all connected clients, including metrics such as IP address, latency, bandwidth consumption, and priority levels. Each device was assigned a unique socket ID, and the administrator had the ability to adjust its priority in real time using a dedicated control button. During testing, default settings categorized devices as low priority, while the administrator upgraded selected clients based on performance needs.

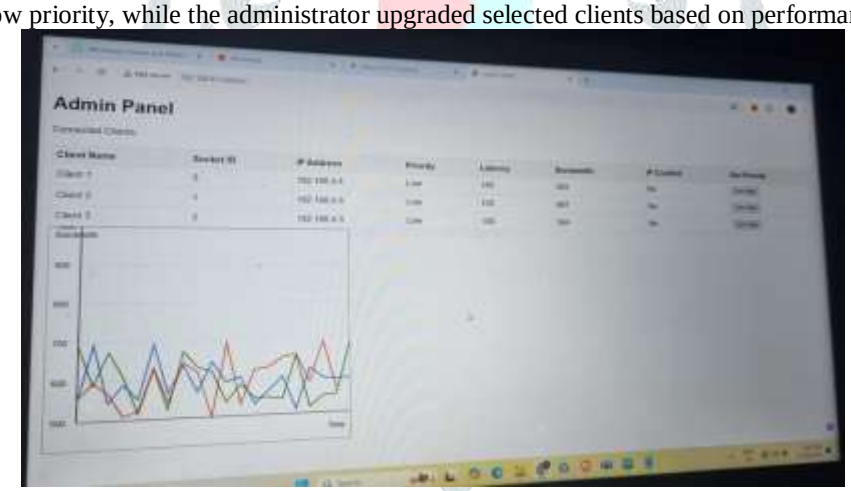

Fig 5.1: -Admin Panel

The table section in the admin panel played a central role in managing client behaviour. It allowed the administrator to monitor real-time changes in latency and bandwidth for each connected device. This helped validate the system's ability to reallocate bandwidth dynamically. For instance, when a client was set to high priority, the system responded instantly by increasing its bandwidth and reducing its latency. The IP conflict status also remained stable throughout, indicating that the system effectively handled network identity without overlaps or address collisions.

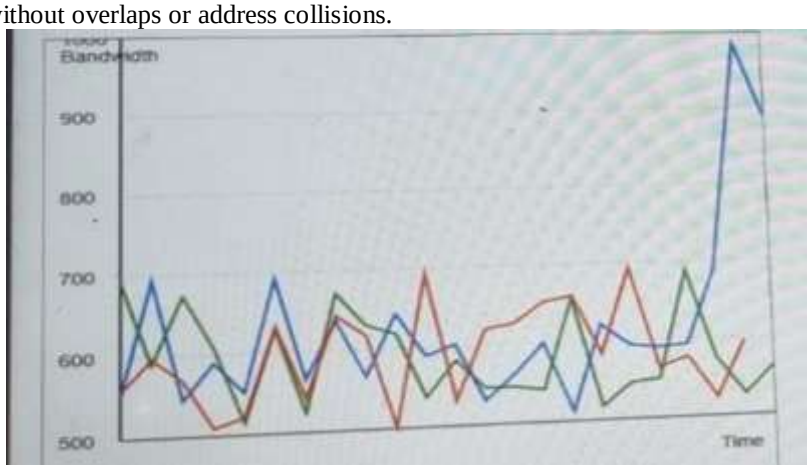

Fig 5.2: -Graph Section

Figure 5.2 illustrates the graphical output of the system's bandwidth usage over time. The line graph displayed trends for each client, with different colours used to distinguish priority levels. As priorities changed, the graph reflected corresponding adjustments in bandwidth allocation. High-priority clients consistently showed rising bandwidth curves, while low-priority clients experienced reduced throughput. This visual confirmation of bandwidth reallocation, based on priority inputs, highlights the system's responsiveness and real-time adaptability. The results confirm that NETGENIE successfully optimizes bandwidth distribution and maintains efficient performance across a multi-client IoT network.

6.ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance and support provided by Prof. Neetha throughout the development of this project. Her insights and feedback played a crucial role in shaping the system architecture and ensuring its practical relevance.

Sincere thanks are also extended to Dr. Sandeep Bhat, project coordinator, for his consistent encouragement and timely suggestions, which helped streamline the research and implementation phases. The team also appreciates the leadership of Dr. Suresha D, Head of the Department, and Dr. Shrinivasa Mayya D, Principal, for fostering an environment that encourages innovation and hands-on learning.

This work was carried out as part of the final-year undergraduate project in the Department of Computer Science and Engineering at Srinivas Institute of Technology. The team thanks all faculty and staff who contributed indirectly to the successful completion of this system.

References

[1] Srinidhi, N. N. et al., Network optimizations in IoT, Elsevier.
[2]Chen, N. et al., Edge intelligent networking for smart cities, IEEE Access.
[3]Maddikunta, P. K. R. et al., Hybrid energy optimization in IoT networks, Springer.
[4]Bhatia, M. & Sood, S. K., Quantum optimization for IoT bandwidth, IEEE.
[5]Jiang, N. et al., Reinforcement learning in NB-IoT resource management, IEEE.