A Mini Project Report On

Smart Congestion Control Management Framework For Energy Constrained Devices

Submitted in Partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

In

Department of Computer Science and Engineering

By

S. Shakeer Ahamad	**22241A05B6**
C. Tharun Teja	**22241A0578**
G. Saketh	**22241A0581**
B. Abhijith Rampal	**22241A0573**

Under the Esteemed guidance of

G.Mallikarjuna Rao

Professor

Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Bachupalli, Kukatpally, Hyderabad, Telangana, India, 500090

2024-2025

GOKARAJU RANGARAJU

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

CERTIFICATE

This is to certify that the Mini Project entitled "**Smart Congestion Control Management Framework For Energy Constrained Devices**" is submitted by S. Shakeer Ahamad (22241A05B6**),** C. Tharun Teja (22241A0578**),** G. Saketh (22241A0581**),** B. Abhijith Rampal (22241A0573**),** Partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2024-2025.**

INTERNAL GUIDE

G.Mallikarjuna Rao

Professor

HEAD OF THE DEPARTMENT

Dr. B. SANKARA BABU

Professor & HoD

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our internal guide **G.Mallikarjuna Rao , Professor**, Department of CSE for his support in the completion of our project report. We wish to express our honest and sincere thanks to **Dr. Y.Krishna Bhargavi and Ms. V. Jyothi** for coordinating in conducting the project reviews. We express our gratitude to **Dr. B. Sankara Babu, HOD**, department of CSE for providing resources, and to the principal **Dr. J. Praveen** for providing the facilities to complete our Mini Project**.** We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

S. Shakeer Ahamad	**22241A05B6**
C. Tharun Teja	**22241A0578**
G. Saketh	**22241A0581**
B. Abhijith Rampal	**22241A0573**

DECLARATION

We hereby declare that the Mini Project entitled "**Smart Congestion Control Management Framework For Energy Constrained Devices**" is the work done during the period from 16th Jan 2025 to 13th May 2025 and is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology.** The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

S. Shakeer Ahamad	**22241A05B6**
C. Tharun Teja	**22241A0578**
G. Saketh	**22241A0581**
B. Abhijith Rampal	**22241A0573**

	LIST OF FIGURES	
Fig No	**Fig Title**	**Page No**
4.1.1	System Architeture	12
4.2.1	Use Case Diagram	23
4.2.2	Class Diagram	25
4.2.3	Sequence Diagram	27
5.1	Packet Tracer	28
5.2	Packet Tracer	29
5.3	Packet Tracer	30
5.4	Packet Tracer	31
5.5	Packet Tracer	32
5.6	Packet Tracer	33
5.7	Packet Tracer	34
5.8	Firebase	35
5.9	Firebase	36
5.10	Firebase	37
5.11	Firebase	38
5.12	Kodular	39
5.2.1	Before Conjection	40
5.2.2	After Conjection	41
5.2.3	Pritorized Path	42
5.2.4	Network Traffic	43
5.2.5	Data in Firebase	44
5.2.6	Data in Firebase	45
5.3.1	Congestion_Status=High	46
5.3.1	Congestion_Status=Normal	47

1. Abstract

Following congestion in its networks, IoT results in higher latency, packet loss and even useless energy utilization. This work includes a Smart Congestion Control Management Framework for energy-constrained devices using Cisco Packet Tracer, Firebase, and Kodular-based mobile application.

All the TIC and telematics devices function as expected in favorable conditions for real-time traffic monitoring. But in cases of congestion, the system uses a wise sequencing strategy based on traffic load, which means that only high-priority devices (like smart lights) are kept on and others go to sleep until congestion is resolved. This is done by handling packets based on priority, detecting congestion, and controlling from the cloud.

Incorporating network segmentation, energy-efficient data transfer, and real-time congestion warnings makes this model highly available and energy efficient, enabling optimal use of energy and better performance of IoT, especially for vital devices.

The rapid growth of Internet of Things (IoT) devices in recent years has created new challenges for ensuring efficient and reliable network communication. Network congestion is one of the main problems that increases the latency, the packet loss and the energy consumption of the systems, especially in the large environments where there are many devices active at the same time. By using Cisco Packet Tracer, Firebase Realtime Database, a Kodular shoes Android, Xiaomi UFO, this project introduces a Smart Congestion-Free Router Networking Framework suitable for IoT based smart environment, which can intelligently control congestion management for no delay or the least latency at the lowest power consumption possible through these smart application.

2. Introduction

The exponential growth of the Internet of Things (IoT) has changed the way we engage with technology. IoT has empowered real-time connectivity, data exchange, and intelligent decision-making across various sectors, from smart homes and wearable devices to healthcare systems and industrial automation. But with the increasing number of interconnected devices, network congestion becomes an unavoidable challenge. IoT network congestion not only hampers the performance of the system but also adds to the latency, packet drop, throughput decline, and extra power wastage which are among the biggest issues, especially for the resource-constrained IoT habitats.

Conventional network architectures commonly rely on centralized control or high-end infrastructure to manage congestion. On the contrary, IoT networks are usually comprised of low-powered devices with restricted computational and communication resources.

This project leverages a simulation environment built using Cisco Packet Tracer to design the network topology and implement congestion scenarios. IoT devices such as smart fans, lights, sirens, and speakers are connected to a home gateway router. The router is linked to an ESP32 microcontroller, which continuously monitors network traffic and sends real-time congestion data to Firebase Realtime Database. The congestion status—classified as *Low*, *Medium*, or *High*—is then retrieved by a Kodular-based mobile application, which provides a visual interface for monitoring congestion and managing device behavior.

A key innovation of the system is the priority-based packet handling algorithm. During high congestion, only high-priority devices (like smart lights used in emergency situations) remain active, while non-critical devices are temporarily disabled. This helps maintain uninterrupted functionality of essential services and optimizes the use of network bandwidth and power resources.

More than ever, the rise of the IoT has been a game-changer in contemporary communication and automation in every sector around the world. With billions of devices coming online, IoT enables a smarter, more responsive world. From smart homes and cities to health, agriculture, and industry, IoT menacingly opens the door to real-time monitoring, control, and data-driven decisions. Devices like smart thermostats, connected vehicles, fitness trackers, and industrial sensors dwell on generating and continuously updating their data,

forming an intricate web of communication. But this hyperconnectivity, while it yields convenience and efficiency, also opens things up to challenges.

With the scaling up of IoT networks, congestion becomes inevitable. It usually refers to network conditions when the amount of data transmitted exceeds capacity, while its performance is degraded; in the IoT setting, it is represented by the inability of devices to respond in real-time, increased latency, packet loss, reduced throughput, and in several cases, very high power consumption due to repeated data transmission attempts. Such situations are critical for real-time or mission-critical applications where timely and accurate delivery of data is key, for instance, in healthcare systems or alerting mechanisms. Moreover, unlike traditional computer networks typically comprised of powerful servers and routers, IoT networks, in fact, include devices with low power and a limited set of resources. Thus, they are unable to afford complex control of congestion algorithms as they do not have enough computing power or memory; moreover, they cannot provide efficient performance in case of heavy traffic.

Congestion management with these characteristics has always depended on centralized architectures, complex algorithms, or expensive hardware solutions, however, none of them could apply for the most IoT applications. A lightweight, adaptive and decentralized way of congestion management is needed due to energy, processing power, and bandwidth limitations exhibited by sensors and devices installed. This project addresses the urgency in the need by offering a novel congestion management framework with coexistencing features such as simulation, real-time monitoring, cloud data technology, and mobile visualization for constrained settings.

This project involves a simulated smart home network developed with Cisco Packet Tracer- a powerful and worldwide recognized tool for networking simulation. This serves to create relatively cheap and controlled conditions under which IoT network configurations and congestion scenarios could be modeled and tested against one another. Such a network contains different examples of IoT devices like smart fans, smart lights, alarms, and speakers interconnected through a router, designated as a home gateway. These appliances would act as real-world smart appliances, passing data at various rates depending on whether they are operating or not and also on external facility-induced triggers.

As the adoption of IoT systems continues to surge, the need for resilient and efficient

network infrastructure becomes increasingly vital. A central focus in modern IoT deployments is not just ensuring device interconnectivity but also maintaining consistent performance under dynamic traffic conditions. The complexity lies in balancing the vast influx of data from numerous devices while maintaining minimal delays and preventing system bottlenecks. Given the diversity of device types and their varying communication patterns, a one-size-fits-all solution for congestion control is no longer practical. This project addresses these limitations by introducing a scalable, energy-aware, and adaptive approach tailored to the constraints of IoT ecosystems. What sets this project apart is its fusion of simulation, hardware integration, and cloud-based analytics to form a complete feedback-driven loop for congestion control. The ESP32 module acts as an intelligent interface between the physical network and the cloud, collecting real-time data on traffic flow and dynamically assessing congestion levels. Firebase serves as a reliable medium for storing and synchronizing this data across platforms. Simultaneously, the Kodular application plays a crucial role in user interaction—alerting users during congestion peaks and giving them control over device states. This interconnectivity ensures that network administrators or home users can make informed decisions based on live network conditions. To improve reliability and service quality, the device priority scheme will be implemented, whereby critical systems will continue to operate un-interrupted even in traffic peaks. Like in real-life situations such as emergencies where sirens sound or lights false, this project demonstrates how bandwidth can be conserved for essential operations. Furthermore, this framework is flexible enough to be expandable either by additional sensors, devices, or microcontrollers without changing the underlying architecture, thus providing a sustainable solution for an ever-growing IOT environment. Such generalizable and highly modular designs reflect a more significant trend toward intelligent and autonomous networks where systems can adapt according to their internal states and stimuli from the environment. Long-term, the idea is not just to minimize the technical effects of congestion but also to enhance overall user experience concerning the reliability, sustainability, and user experience of smart environments.

3. Literature Survey

With the capability for smart devices to communicate as well as collect and share data in real-time, the Internet of Things, by definition, has an omnipotent influence in reshaping digital ecosystems. Various studies have corroborated the emergence of billions of connected devices and the managerial aspects of network resources as crucial challenges, especially in lower-bandwidth and low-power-device environments. Research has proved that congested mechanisms, which rely either on a centralized architecture or computationally expensive algorithms, are often the wrong choice when applied to IoT networks because of the stringent hardware and energy limitations posed.

Lightweight and adaptive protocols have been the concern of recent explorations into the congestion control field, especially for wireless sensor networks (WSNs), which share many characteristics with IoT systems. Priority-based data transmission techniques and load balancing over multiple nodes have been shown to enhance network throughput and reduce delays considerably. Energy Efficient Congestion Control (EECC) algorithms value traffic classification and packet prioritization by significantly enhancing performance during peak loads with data traffic.

Simulation tools such as Cisco Packet Tracer essentially model IoT network behavior under various loads. They further provide a less expensive approach to replicating real-world scenarios in conjunction with assessing how devices perform interactions with multiple threat and congestion levels. Further studies were conducted on such simulations to prove the importance of scenario-based testing for optimizing routing protocols and data management strategies.

Cloud platforms concerning real-time monitorization and data visualization have, with time, come to be highly favored. Firebase, for instance, supports enhanced data synchronization between embedded device and front-end applications aiding researchers or developers in a responsive and dynamic IoT system development. It enables real-time data logging and triggering of event-based actions on congestion events with the ESP32 microcontroller.

Mobile development platforms, e.g., Kodular, have expanded the reach of IoT solutions to an even greater number of non-programmers. Applications can be made by visual programming, so all developers can create interactive applications to control the working of

IoT devices, thus making this real-time congestion monitoring solution handy and user-oriented.

In brief, current literature points to a pressing need for adaptiveness and low overhead of any congestion control regime, taking into cognizance the limitations of the IoT infrastructure. Through simulations, cloud services, and mobile interfaces, this project builds upon these insights for a practically viable and scalable approach to address congestion control in smart environments.

1. **Paper Title & Year:** *A Survey on Congestion Control Protocols in IoT* (2020)

Authors: M. Amadeo, C. Campolo, A. Molinaro

Methodology & Metrics: Analyzed CoAP, MQTT, and RPL protocols; measured latency and energy consumption.

Datasets Used: Simulated IoT networks

The Gaps Identified: The research is deficient in real-time dynamic traffic prioritization mechanisms, which are important in IoT environments with dynamic traffic patterns and latency constraints. It is protocol-based comparison without the suggestion of flexible frameworks, thus making it less applicable for use in dynamic environments such as smart cities or healthcare monitoring systems.Energy-Efficient Routing in IoT Using SDN

2. **paper Title & Year:** Energy-Efficient Routing in IoT Using SDN (2019)

Authors: M. A. Alsheikh et al.

Methodology & Metrics: SDN-based routing with energy-aware metrics Datasets Used: IoT testbed with Raspberry Pi

The Gaps Identified: The lack of real-time congestion detection mechanisms lowers its efficiency under dynamic traffic loads. In addition, although the SDN-based model is energy-aware, it does not scale well with large-scale, distributed IoT settings due to possible central controller bottlenecks and low fault tolerance.

3. **Paper Title & Year:** Priority-Based Traffic Management for IoT (2021)

Authors: G. Piro et al.

Methodology & Metrics: Adaptive QoS for smart devices

Datasets Utilized: NS-3 simulations

The Gaps Identified: The system employs static priority assignment,
which does not adapt to shifting network or device conditions. In urgent IoT applications (e.g., emergency response), strict priority queues can queue high-importance data. The absence of feedback from network performance metrics also results in poor quality of service over time.

4. **Paper Title & Year:** Light-Weight Congestion Control in IoT (2022)

Authors: H. Abdul Khalek, Lotfi Mhamdi

Methodology & Metrics: LSTM traffic forecasting models.

Datasets Used: CICIDS2017 dataset.

The Gaps Identified: While LSTM provides predictive accuracy, this is at the expense of extreme computational complexity, such that it is not deployable in real-time on resource-limited IoT nodes. The solution also relies significantly on the quality of the training data, which may not be representative of all real-world situations, resulting in compromised generalization.

5. **Paper Title & Year:** CoAP vs. MQTT: A Performance Comparison (2018)

Authors: O. Bergmann et al.

Methodology & Metrics: Adaptive control algorithm.

Datasets Used: IoT sensor data.

The Gaps Identified: The comparison of performance is confined to simple metrics and doesn't investigate extreme network condition or scaling treatment from these protocols. It doesn't determine protocol resilience, fault tolerance, or flexibility in accommodating heterogeneous node capabilities, omitting real-world implications for extensive smart environments.

6. **Paper Title & Year:** Edge Computing for IoT Congestion Mitigation (2020)

Authors: Y. Mao et al.

Methodology & Metrics: Edge-based traffic offloading.

Datasets Used: Real-time IoT data.

The Gaps Identified: Although edge computing minimizes latency, the model does not have energy-efficient mechanisms that can facilitate premature node draining, particularly for battery-based systems. Secondly, offloading tasks to the edge devices may result in processing delays under heavy concurrency without load balancing schemes.

7. **Paper Title & Year:** Dynamic QoS Management in IoT (2021)

Authors: S. Misra et al.

Methodology & Metrics: Reinforcement learning for adaptive QoS.

Datasets Utilized: Data from smart cities.

The Identified Gaps: Reinforcement learning provides flexibility but imposes significant computational and memory overhead that many low-power IoT devices cannot accommodate. In addition, the models require extensive, well-labeled datasets to train, thus rendering it challenging to deploy in novel or dynamically varying environments.

8. **Paper Title & Year:** Energy-Aware Protocol Control in 6LoWPAN (2019)

Authors: G. Montenegro et al.

Methodology & Metrics: Adapted 6LoWPAN for Energy Conservation.

Datasets Used: Contiki-NG simulations.

The Gaps Identified: The protocol is customized for certain network topologies, limiting its applicability to various IoT deployments like vehicular networks, smart agriculture, or industrial IoT. It also has no flexibility in conforming to changes in topology at real time and is thus inappropriate for mobile or ad-hoc sensor systems.

9. **Paper Title & Year:** Firebase for Real-Time IoT Monitoring (2020)

Authors: A. Al-Fuqaha et al.

Methodology & Metrics: Firebase Cloud + IoT sensors.

Datasets Used: Personal Firebase data.

The Gaps Identified: While Firebase supports real-time monitoring, the system doesn't have inherent congestion awareness or traffic prioritization. This renders it inefficient in situations where there are multiple key data streams that have to be prioritized. Also, cloud storage dependency can incur latency and privacy issues in privacy-sensitive applications.

10. **Paper Title & Year:** Cisco Packet Tracer in IoT Education (2021)

Authors: M. Z. Rahman et al.

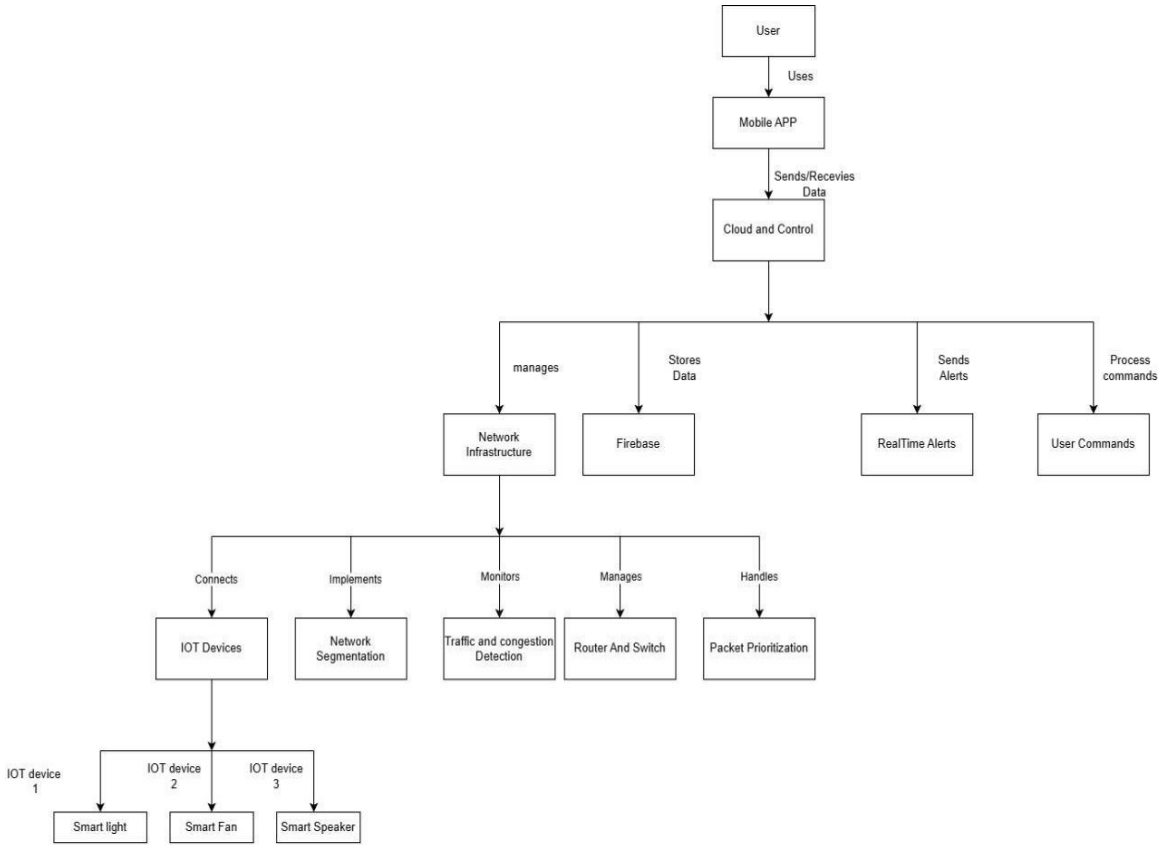
Methodology & Metrics: Simulated IoT energy consumption.

Datasets Used: Packet Tracer modules.

The Gaps Identified: The application of Cisco Packet Tracer offers solely simulation-based results, which are usually too optimistic. The model does not account for real-life issues such as interference, node failure, or environmental variability and hence cannot be relied upon for deployment without testing in real-life scenarios.

4. System Architecture

The proposed system architecture for congestion-free IoT traffic management combines simulated networking in Cisco Packet Tracer, real-time data handling using ESP32 microcontrollers, and cloud monitoring via Firebase. IoT devices, such as lights and sensors, are connected through routers and switches, where each device is assigned a priority level. Under normal conditions, all devices transmit data continuously. When network congestion is detected—based on packet thresholds—the system activates a priority-based control mechanism, allowing only high-priority devices (e.g., emergency lights or medical sensors) to continue transmission. The ESP32 collects real-time data and communicates with Firebase, which logs congestion events and device statuses. Additionally, a mobile application developed in Kodular displays alerts and network status, offering an interactive interface for monitoring and control.



5. Fig : 4.1.1 System Architecture

4.1 Software Requirements

The software requirements for the proposed system include several key platforms and tools to enable simulation, real-time monitoring, and cloud integration. Cisco Packet Tracer is used for designing and simulating the IoT network, including routers, switches, and connected devices. Firebase (Realtime Database) is used as the backend to store and monitor congestion data, device status, and priority levels. For real-time updates and automation, Firebase Cloud Functions may also be utilized. Additionally, Kodular Creator is used to develop the mobile application that displays congestion alerts and device activity. Supporting tools such as Google Chrome (for Firebase access), JSON Editor Online (for Firebase data structure), and Python (for optional scripting or data preprocessing) may also be required during development and testing.

4.2 Hardware Requirements

The hardware setup for the proposed system involves components essential for simulating and implementing real-time IoT communication and congestion control. The core hardware includes the ESP32 microcontroller, which serves as the primary IoT device for sensing data and connecting to the internet via Wi-Fi. For simulating the network infrastructure, computers or laptops running Cisco Packet Tracer are required to model routers, switches, and multiple IoT devices. In a physical extension, LEDs or buzzers may be used to simulate output actions based on network status or priority. A Wi-Fi router or mobile hotspot necessary to provide internet connectivity for the ESP32 to communicate with Firebase. Optionally, smartphones are needed to run the Kodular-based mobile application for monitoring congestion and device statuses.

4.3 Methodology and Dataset

The methodology involves designing a simulated IoT network using Cisco Packet Tracer, where various IoT devices—such as lights, sensors, and alarms—are connected through routers and switches. Each device is assigned a priority level based on its importance (e.g., high for emergency systems, low for regular devices). A congestion detection algorithm monitors network traffic, and when packet flow exceeds a defined threshold, the system automatically enables a priority-based control mechanism. This ensures that only high-priority devices remain active during congestion, while lower-priority devices are temporarily restricted. The system logs device activity and congestion events to a Firebase Realtime Database, enabling remote monitoring and control. A Kodular-based mobile application is used to visualize congestion status and active/inactive devices in real time.

As the project is simulation-based, a synthetic dataset is generated dynamically during runtime. This dataset includes fields like device ID, priority level, timestamp, packet status, and congestion indicator. The data is used for analyzing network behavior, validating congestion control logic, and ensuring that priority management functions as expected.

The proposed system is implemented entirely within Cisco Packet Tracer, which allows for realistic simulation of an IoT network. The network consists of multiple IoT devices, such as smart lights, alarms, and environment sensors, all connected via routers and switches. Each IoT device is configured to transmit data packets periodically. To manage network efficiency, devices are classified into three categories: High Priority (e.g., ambulance lights, alarms), Medium Priority (e.g., environmental sensors), and Low Priority (e.g., decorative lights).

Congestion within the network is monitored through analyses of traffic load on the routers themselves. When traffic surpasses a defined threshold, a priority-based congestion control algorithm is triggered. This algorithm will allow only high-priority devices to continue transmission, while medium and low-priority devices are temporarily halted from transmitting so as to prevent possible packet loss and to ensure that high-priority services remain unaffected. Device behavior, packet flow, and load on the network are continuously monitored through custom logic within Cisco Packet Tracer event management.

Important events within the network, such as congestion status, active devices, and blocked packets, are logged on Firebase Realtime Database. Firebase acts as a central backend for monitoring and future data analytics. A Kodular app connects to Firebase, which has the live dashboard for congestion alerting and which devices are permitted to work currently.

4.2:Proposed Approach & Module Description

The proposed model is built around the idea of implementing a congestion control mechanism in IoT through making use of priority-based traffic control with the system being completely developed and validated on Cisco Packet Tracer environment, where numerous IoT devices communicate within a simulated network composed of routers and switches. For the sake of prioritized traffic, each device is assigned a priority level based upon its relative importance in the context of traffic congestion. The system will detect the congestion condition with respect to the packet volume or bandwidth usage and will attempt to dynamically restrict traffic such that all devices, except for some critical (high-priority) ones, stop functioning. This will guarantee assured service reliability for the basic needs, even at maximum bandwidth saturation. Cloud-based Firebase Realtime Database assists in logging congestion events and device status, while a user interface with real-time monitoring needs is catered for in the app with Kodular.

Technically speaking, several vital modules are in place for managing different aspects: congestion control and real-time monitoring. The IoT Device Simulation Module simulates numerous virtual IoT devices inside the Cisco Packet Tracer environment, where the device would generate data at regular intervals and have an assigned priority level [high, medium, and low] to indicate the device importance during traffic congestion period. The Network Infrastructure Module can be simulated to include routers, switches, and links that connect these devices to ensure correct data transmission and keep a tab on congestion detection. During congestion time, the Priority-Based Traffic Control Module would only allow high-priority devices to continue transmitting information at the detecting congestion, while other devices will temporarily halt the ongoing communication to reduce network traffic. The Cloud Logging Module, supported by Firebase, logs each of the congestion events and device statuses in real-time to use as valuable insights in later analysis. Finally, the Monitoring Interface Module, designed with Kodular, allows for the mobile app that gives users a real-time update on congestion status, including which devices are active and which are silenced depending on priority-level conditions.

An elite set of modules comes together to act as a single unit for cancering and building prioritized data transmission as imp Featured modules have been designed to accomplish specific tasks and to help the network maintain its efficiency, but at the same time stay very

responsive even when contending with congestion.

The system is utterly subdivided into a number of vital modules that would cooperate together to see to an effective control of congestion in the IoT network. The IoT Device Simulation Module tries to recreate various IoT devices, such as cries for help, medical sensors, sensors, and home devices in the Cisco Packet Tracer environment. They produce data packet at every regular interval. All devices are assigned with a priority level, which could be high, medium, or low according to their criticality. Thus high-priority devices, for instance, cries for help, are regarded as effector for earnest responders, while technology used to amplify light or whatsoever central to the daily living would rate as peripheral and indeed less critical.

4.1 Modules

1. IoT Device Simulation Module

It is the module that emulates the various IoT devices, such as smart lights, fans, sirens, alarms, and speakers. These devices will create a virtual environment where real-time data communication behavior will be obtained using Cisco Packet Tracer. Each device will now be given priority according to its criticality-level-high, medium, or low-as soon as they come into network operations. The result is a near-real environment characterized by data packet transmissions for each device at regular intervals, just as physical IoT devices would operate in a live setup. This will help study the ability of the system to support dynamic congestion management based on device behavior over traffic intensity. The modular design also promises easy inclusion of more device types in future scenarios, thereby enhancing the scalability of the system.

2. Network Infrastructure Module

The real substance and the very backbone of the communication framework. The routers and switches all simulate communication links within Cisco Packet Tracer. Their infrastructure ensures that packets flow through devices efficiently, while also playing a key role in congestion detection and responses. Packets are forwarded and traffic analyzed at the central nodes known as routers. Traffic is continuously monitored to understand the incoming and outgoing state of traffic with regard to overload. Basic routing support is part of this module: OSPF, EIGRP, VLAN configuration, and QoS. One can easily say that this module is where the congestion threshold is defined and enforced, enabling dynamic activation of the priority control logic.

3. Congestion Detection and Priority-Based Traffic Control Module

This system's heart and soul in its decision-making module continuously checks network traffic to determine whether there is congestion: low, medium, or high. There are predefined thresholds, like packet rate or queue length or bandwidth utilization, to determine if the network is stressed. Upon congestion detection, the system activates a priority-based control algorithm. The algorithm assesses the priority of each device and determines whether it remains active or is turned off temporarily. Communication is always supported for high-

priority devices—alarms are a good example—but for medium and low-priority devices like fans or decorative lights, the system stops communication to save bandwidth and energy. The method aims to ensure uninterrupted service to critical devices while efficiently using resources across the network.

4. Cloud Integration and Logging Module

Real-time monitoring and post-event analysis have been implemented through integrating this module into the simulated network with a cloud-based backend: Firebase Realtime Database. The module logs all events relevant to the network, such as current congestion status, activity of devices (active or paused), and priority levels. The state changes of the network are directly pushed to Firebase for real-time updates and data synchronization outside interfaces. This module serves as a central data collection node and also sets the scope for all future optimizations, such as historical data analytics, congestion prediction, or connecting the system to other cloud-based AI services. It also ensures a streamlined infrastructure for remote monitoring and assessment through reduced latencies.

5. Monitoring Interface Module (Kodular App)

The Monitoring Interface Module is a mobile dashboard that is quite easy to use and created with Kodular, a visual development tool for Android apps. This module decodes and connects to the Firebase Realtime Database where updates can be made live. It allows users to check the congestion level being experienced now (Low, Medium, High), whether devices are being remotely active or paused, and alerts if congestion remains high. The diagram displays device IDs, priority levels, and indicators of their present state, giving a good overview of the machine's state. This module is the basis for user interaction and transparency that empowers users with information and perhaps control or advanced configuration of device priorities in some future implementation of the system.

6. Quality of Service (QoS) and Access Control Module

It serves as the support module containing the configurations for the QoS and ACLs which are implemented within the router within Cisco Packet Tracer itself. Manage the bandwidth according to the device priority as well as the traffic type, such as providing extra

bandwidth to the emergency devices with QoS class-maps and policy-maps. Once again, the ACL is used to limit or minimize specific devices during congestion and rules are defined to prevent unnecessary flooding of information across the network. Along with this, network-layer mechanisms serve the overall cause of the congestion control module, as they enforce all traffic-shaping and prioritization policies directly from within the routing infrastructure.

Combined, these modules provide a cohesive and integrated intelligent environment for simulating, monitoring, and controlling network congestion in IoT systems while optimizing energy and resource constraints. Each is intended to be able to operate independently but symbiotically with other modules to provide flexibility, scalability, and real- world applicability.

4.2 UML Diagrams

UML useful in scenarios with high levels of software and hardware interaction, like modeling the data flow and software interaction of embedded software. It can even find application in high-level embedded systems with real-time constraints.

UML diagrams may be found in every area of the modern software industry, vastly employed in tackling communication problems during various phases of the software development lifecycle. UML diagrams can be classified into structural and behavioral types but, interestingly, they can also be modified into various other types, ranging in size from small applications to vast enterprise systems. This aforementioned visual aspect provides a means for technical and non-technical parties to collaborate and understand each other more easily.

UML is also aimed at fostering iterative and agile approaches by quickly defining the changes in the design and evolution of the system. An example would be using sequence diagrams during sprint planning for depicting the interactions in the system before anything is actually coded. Activity diagrams help to plan complex business workflows so that teams can pin-point bottlenecks and decision points early in the process.

Tool support for UML is enormous, and modeling tools like StarUML, Enterprise Architect, and Lucidchart constitute a solid environment for producing and modifying diagrams. These tools often provide IDE integration and other development platforms, improving productivity and traceability from design to implementation.

And UML does not stop at software design: it can be used to model hardware interactions, along with data flow, including usages in real-time systems. This very flexibility also means that UML is useful for embedded software scenarios with a high level of interaction in software and hardware, such as modeling data flow and software interaction. It could even be applied to high-level embedded systems under real-time constraints.

Use Case Diagram

UML Use Case diagrams are visual representations of functional requirements pertaining to the users of a system, known as actors. These diagrams show all the functions of a system as seen by outside entities such as users or other systems. Each use case stands for a pretty precise interaction or the functionality provided by the system, such as: "Login," "Process Payment," or "Generate Report." The external actors are usually the real users or some outside system with which the use case can be done. The UML Use Case diagrams characterize what a system does rather than how it will realize the desired behavior. In the diagram, actors are shown along with the use cases they are involved with, and it provides different relationships like associations (how an actor is associated with a use case), generalizations (actor inheritance), or includes or extends relationships (common or optional behavior). This allows developers and non-technical users to understand the whole system at a very high level in terms of functionality and interaction with users.

A Use Case Diagram in Unified Modelling Language represents a use of an external agent known as an actor. It defines functional requirements for that external entity using a visual model that portrays the uses of the system represented to the external entity. These uses within the diagram would be representations of specific functionality the system was expected to provide, such as 'Login', 'Monitor Congestion', or 'Send Alerts'. An association would relate the actor and use case, while associations can be involved further using includes or extends to share behaviors or highlight optional actions.

This would help collect the most vital functions of the system-a Use Case diagram for many users. The actors may be network administrator, IO device, and end user-who accesses the monitoring interface via the Kodular app. The typical use cases could represent activities including viewing network status, congestion detection, data logging to Firebase, traffic prioritization, and receiving alerts through the mobile application. These actors could serve to perform actions such as managing congestion, tracking high-priority devices during network overload, and ensuring real-time communication with the system. A Use Case Diagram should provide a clear view at a higher level of system functionality relating to enhanced understanding of how users interact with the system, thus benefitting stakeholders in designing the system more effectively.

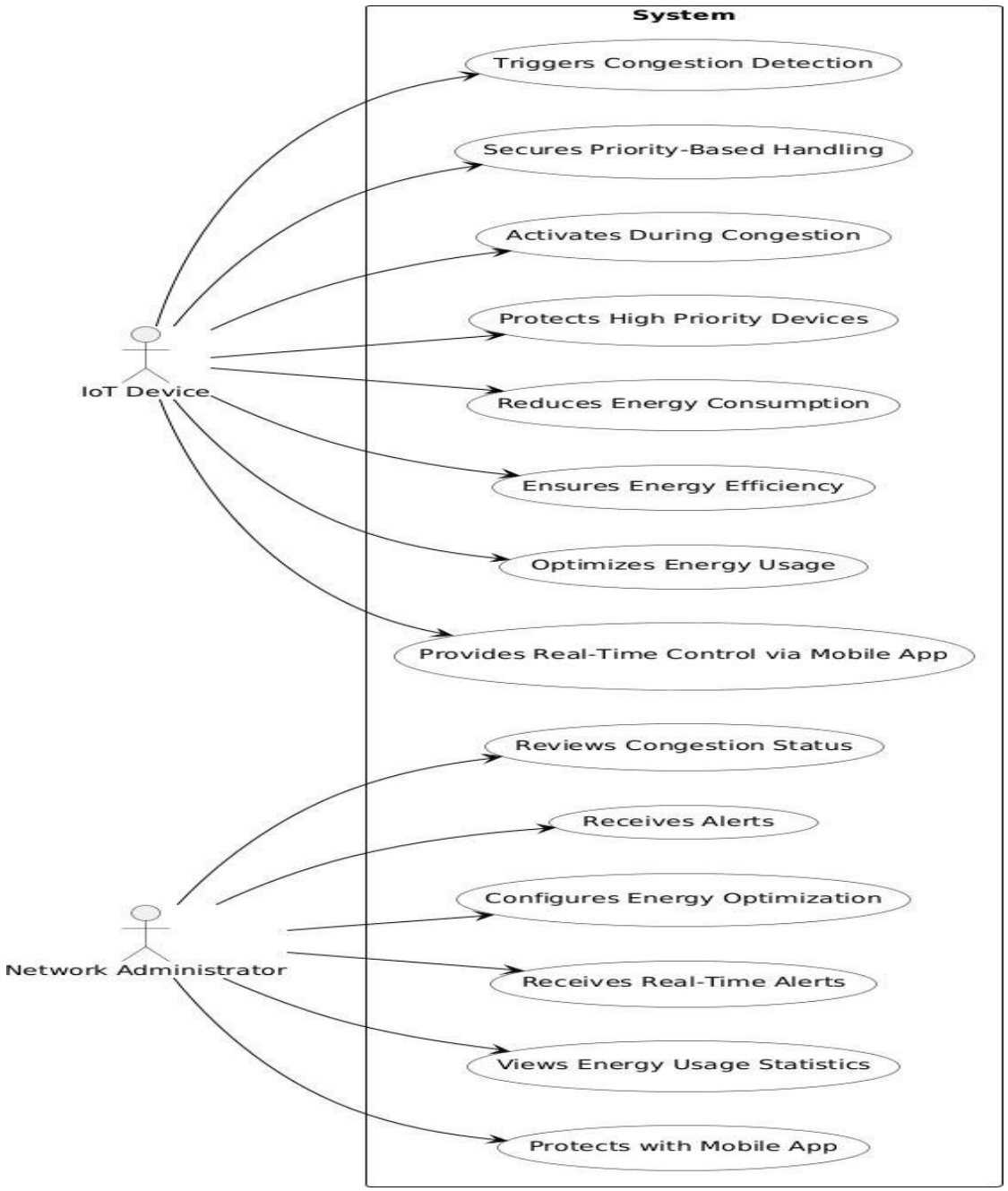


Fig: 4.2.1 Use Case Diagram

Class Diagram

A class diagram classifies the classes of the system entities along with their attributes, methods, and the relations among them. The structural elements of the system can be more easily demized by means of such static structure diagrams in UML. In such a view, the architecture of a system blueprint is conveyed in a subclass layout as opposed to a data layout. As described in a class diagram, classes are forms represented with rectangular sections divided into three small top parts including the attributes (i.e., all such properties), the middle comprising in all the methods possible, and finally, the last bottom representing the operations such a class can perform itself. Relationships forming the notations include associations, the latter form inheritance, and one another called dependency.

For the IoT-based congestion-control project, it might create a Class Diagram that integrates different entities of the system, like IoTDevice, Router, PriorityQueue, FirebaseDatabase, and UserInterface. Attributes for routers might also include things such as router id, traffic load, and congestion status, while IoT devices would have device id, priority, and much more based on status. Methods can include sendData(), detectCongestion(), and logDataToFirebase(). The PriorityQueue class contains methods that will prioritize and manage packet traffic based on the importance of a device. Relationships here can be browsed under association-a router manages several mention IoTDevice instances, and also dependencies like the UserInterface that needs data from FirebaseDatabase to project real-time status.

So this class diagram is useful in giving a view of the static structure of your system and how different components interact therein, which creates a good basis in addition to understanding the components of this system and their behavior.

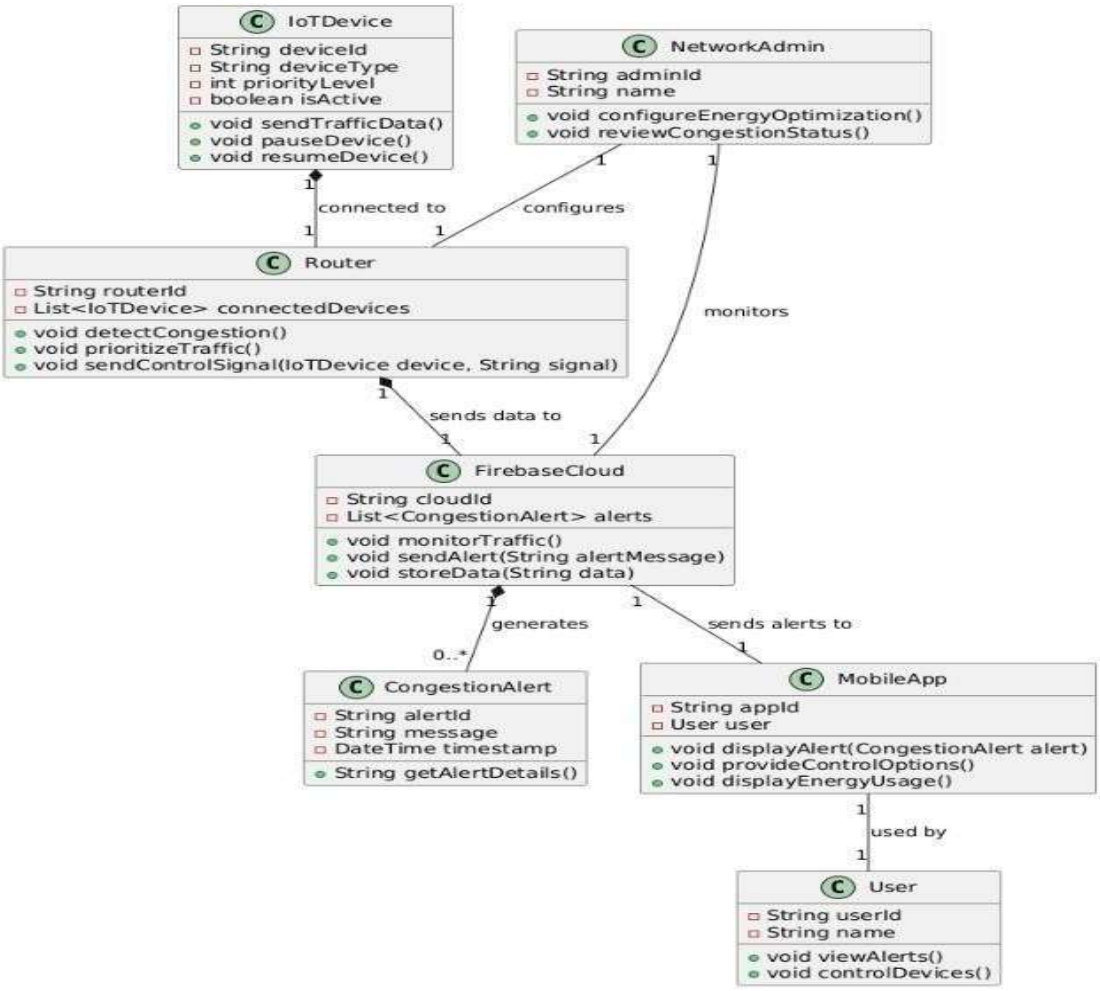


Fig:4.2.2 Class Diagram

Sequence Diagram

A sequence diagram in UML is a behavior diagram that shows how different objects or components in a system interact with each other in a particular sequence, carrying out a certain task. Its main focus is the messages being transmitted chronologically between these objects to show how the product proceeds step by step from one operation to another. In the context of the congestion control project using IoT, the Sequence Diagram would depict the interaction between components, such as IoT devices, routers, Firebase, and the Kodular app, during congestion. For example, an IoT device sends data packets to the router, which checks the traffic load for congestion. If congestion is observed, the router contacts the Priority-Based Control Module, which then looks into the priority levels of the linked IoT devices. Then, based on which, the module temporarily stops the lower-priority devices, permitting only certain high-priority devices to send data. Meanwhile, the router is updating the Firebase Database with device status of congestion et cetera. The Kodular app uses real-time calls to Firebase to ensure that congestion alerts, as well as the listing of active and inactive devices, appear on the network front-and-center. This Sequence Diagram shall describe how data and actions were moving within the system, for the purpose of understanding the closer participation of different components, working together to achieve the handling of congestion with effectiveness.

Therefore, the results from the Sequence Diagram describe each interaction between the components of the system to provide a concentrated view on how data is transferred and how congestion is dealt with step by step. The diagram starts with IoT devices sending data packets periodically to the router. These devices may be anything from customizable smart house appliances to sensors. Each device has its predetermined priority level. The router is monitoring the incoming traffic and is checking to see during different instances if data stream overloaded implies a congestion level.

Thus, when congestion is detected, the router collaborates with the Priority-Based Control Module that assesses the priority of devices connected. The devices get classified in different priority levels (high, medium, or low), and temporarily small-time the lower-priority devices' data transmission is ceased by the system while higher-priority devices, like emergency alarms or medical sensors, continue to work.

Additionally, the router sends data to the Firebase Database in real time, thereby logging an event of congestion and updating the status of active and inactive devices and writing network usage metrics. This continuous monitoring and logging is intended for future analysis. The Kodular app, connected to Firebase, would be quizzing the database about the current status of the network, including congestion alerts, and presenting the user interface to see which devices are operational and which ones have been paused as a result of congestion. Users can make use of this app to monitor the

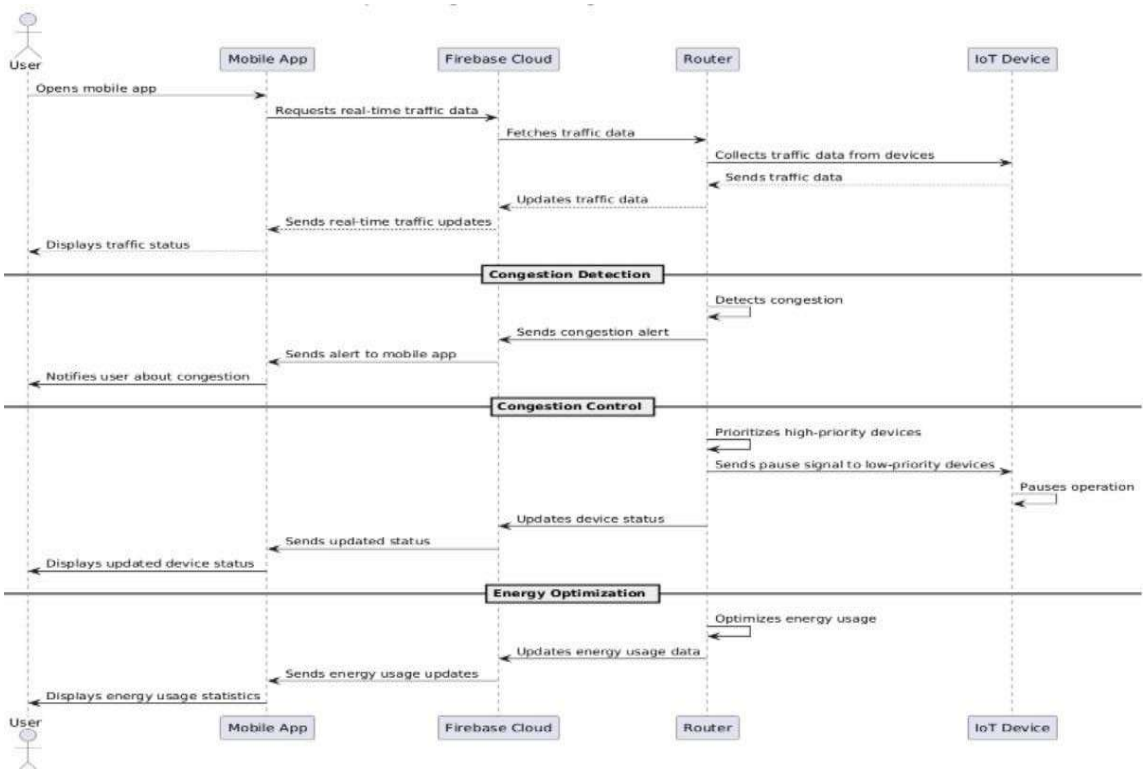


Fig:4.2.3 Sequence Diagram.

5. Implementation

Execution of the Project (1. Packet Tracer)

- OSPF Configuration
- Quality of ServiceRouter(config-router)# network 192.168.1.0 0.0.0.255 area 0
- Router(config-router)# exit
- Router(config)# router eigrp 100
- Router(config-router)# network 192.168.1.0 0.0.0.255
- Router(config-router)# exit
- Router(config)# ip cef
- Router(config)# interface GigabitEthernet 0/1
- Router(config-if)# ip load-sharing per-packet
- Router(config-if)# exit
- **Quality of Service**
- Router(config)# class-map match-any VOICE
- Router(config-cmap)# match protocol rtp
- Router(config-cmap)# exit
- Router(config)# policy-map QOS_POLICY
- Router(config-pmap)# class VOICE
- Router(config-pmap)# priority 500
- Router(config-pmap)# exit
- Router(config)# interface GigabitEthernet 0/0
- Router(config-if)# service-policy output QOS_POLICY
- Router(config-if)# exit
- VLANs & Inter-VLAN Routing

Fig:5.1 Packet Tracer(Snapshot)

- Switch(config)# vlan 10
- Switch(config-vlan)# name SALES
- Switch(config-vlan)# exit
- Switch(config)# vlan 20
- Switch(config-vlan)# name HR
- Switch(config-vlan)# exit
- Assign VLAN to a Port
- Switch(config)# interface FastEthernet 0/1
- Switch(config-if)# switchport mode access
- Switch(config-if)# switchport access vlan 10
- Switch(config-if)# exit
- Router(config)# interface GigabitEthernet 0/0.10
- Router(config-if)# encapsulation dot1Q 10
- Router(config-if)# ip address 192.168.10.1 255.255.255.0
- Router(config-if)# exit
- Access Control Lists (ACLs) & Rate Limiting
- Router(config)# access-list 100 deny tcp any any eq 443
- Router(config)# access-list 100 permit ip any any
- Router(config)# interface GigabitEthernet 0/0
- Router(config-if)# ip access-group 100 in
- Router(config-if)# exit
- Router(config)# interface GigabitEthernet 0/1
- Router(config-if)# traffic-shape rate 5000000 100000 100000
- Router(config-if)# exit
- **Ping and Traceroute for Network Testing**
- PC> ping 192.168.2.1

Fig:5.2 Packet Tracer(Snapshot)

- PC> tracert 192.168.2.1
- Router Configuration
- Router1> enable
- Router1# configure terminal
- Router1(config)# interface fastethernet 0/0
- Router1(config-if)# ip address 192.168.1.1 255.255.255.0
- Router1(config-if)# no shutdown
- Router1(config-if)# exit
- Router2> enable
- Router2# configure terminal
- Router2(config)# interface fastethernet 0/0
- Router2(config-if)# ip address 192.168.2.1 255.255.255.0
- Router2(config-if)# no shutdown
- Router2(config-if)# exit
- Router3> enable
- Router3# configure terminal
- Router3(config)# interface fastethernet 0/0
- Router3(config-if)# ip address 192.168.3.1 255.255.255.0
- Router3(config-if)# no shutdown
- Router3(config-if)# exit
- Router 1: Configure OSPF
- Router1> enable
- Router1# configure terminal
- Router1(config)# router ospf 1
- Router1(config-router)# network 192.168.1.0 0.0.0.255 area 0
- Router1(config-router)# network 192.168.2.0 0.0.0.255 area 0

Fig:5.3 Packet Tracer(Snapshot)

- Router1(config-router)# network 192.168.3.0 0.0.0.255 area 0
- Router1(config-router)# exit
- Router2> enable
- Router2# configure terminal
- Router2(config)# router ospf 1
- Router2(config-router)# network 192.168.2.0 0.0.0.255 area 0
- Router2(config-router)# network 192.168.1.0 0.0.0.255 area 0
- Router2(config-router)# network 192.168.3.0 0.0.0.255 area 0
- Router2(config-router)# exit
- Router3> enable
- Router3# configure terminal
- Router3(config)# router ospf 1
- Router3(config-router)# network 192.168.3.0 0.0.0.255 area 0
- Router3(config-router)# network 192.168.1.0 0.0.0.255 area 0
- Router3(config-router)# network 192.168.2.0 0.0.0.255 area 0
- Router3(config-router)# exit
- Verify OSPF Configuration
- Router1# show ip route ospf
- Router2# show ip route ospf
- Router3# show ip route ospf
- Test Connectivity
- Router1# ping 192.168.2.1
- Router2# ping 192.168.3.1
- Classify the Traffic
- Router1> enable
- Router1# configure terminal

Fig:5.4 Packet Tracer(Snapshot)

- Router1(config)# access-list 101 permit ip host 192.168.1.2 any
- Router1(config)# access-list 101 permit ip host 192.168.1.3 any
- Router1(config)# exit
- Router1(config)# class-map match-any voice-traffic
- Router1(config-cmap)# match access-group 101
- Router1(config-cmap)# exit
- Router1(config)# policy-map voice-priority
- Router1(config-pmap)# class voice-traffic
- Router1(config-pmap-c)# priority 128
- Router1(config-pmap-c)# exit
- Router1(config)# interface fastethernet 0/0
- Router1(config-if)# service-policy output voice-priority
- Router1(config-if)# exit
- **Implementing ACLs (Access Control Lists)**
- Router1> enable
- Router1# configure terminal
- Router1(config)# access-list 102 deny ip host 192.168.2.2 host 192.168.1.2
- Router1(config)# access-list 102 permit ip any any
- Router1(config)# exit
- Router1(config)# interface fastethernet 0/0
- Router1(config-if)# ip access-group 102 in
- Router1(config-if)# exit
- Implement QoS for Priority-Based Packet Handling
- Router(config)# access-list 110 permit ip host 192.168.1.10 any
- Router(config)# access-list 111 permit ip host 192.168.1.20 any

Fig:5.5 Packet Tracer(Snapshot)

- Router(config)# exit
- Router(config)# class-map match-any AmbulanceTraffic
- Router(config-cmap)# match access-group 110
- Router(config-cmap)# exit
- Router(config)# class-map match-any RegularIoT
- Router(config-cmap)# match access-group 111
- Router(config-cmap)# exit
- Router(config)# policy-map IoT-QoS
- Router(config-pmap)# class AmbulanceTraffic
- Router(config-pmap-c)# priority 512
- Router(config-pmap-c)# exit
- Router(config-pmap)# class RegularIoT
- Router(config-pmap-c)# bandwidth 128
- Router(config-pmap-c)# exit
- Router(config)# interface FastEthernet 0/0
- Router(config-if)# service-policy output IoT-QoS
- Router(config-if)# exit
- Service-policy output: Prioritize_Traffic
- Class-map: Emergency
- Priority: 512 kbps
- Class-map: class-default
- Fair Queue
- policy-map Prioritize_Traffic
- class Emergency
- priority 1024 ! Increase bandwidth allocation to 1024 kbps
- class class-default

Fig:5.6 Packet Tracer(Snapshot)

Fig:5.7 Packet Tracer(Snapshot)

2. Firebase

```python
import os
import firebase_admin
from firebase_admin import credentials, db

# Firebase credentials path
FIREBASE_CRED_PATH = os.getenv("FIREBASE_CRED_PATH", r"C:\Users\ahama\Downloads\New folder (3)\packet-tracer-6ad46-firebase-adminsdk-fbsvc-8989d73c27.json")

# Path to your .txt file
TXT_FILE_PATH = r"C:\Users\ahama\OneDrive\Desktop\router.txt"  # Replace with the actual path to your file

# Initialize Firebase
try:
    cred = credentials.Certificate(FIREBASE_CRED_PATH)
    firebase_admin.initialize_app(cred, {
        'databaseURL': 'https://packet-tracer-6ad46-default-rtdb.firebaseio.com'
    })
    print("Firebase initialized successfully!")
except Exception as e:
    print("Error initializing Firebase:", e)
    exit()

# Function to determine congestion level
def determine_congestion(metric_value):
    if metric_value < 10:  # Lowered threshold for Low
        return "Low"
    elif 10 <= metric_value <= 60:  # Lowered threshold for Medium
        return "Medium"
    else:
        return "High"

# Function to send data to Firebase
def send_to_firebase(data):
    try:
```

Fig: 5.8 Firebase

```python
def determine_congestion(metric_value):
    if metric_value < 10:  # Lowered threshold for Low
        return "Low"
    elif 10 <= metric_value <= 60:  # Lowered threshold for Medium
        return "Medium"
    else:
        return "High"

# Function to send data to Firebase
def send_to_firebase(data):
    try:
        ref = db.reference('iot_data')  # Path in Firebase where data will be stored
        ref.push().set(data)  # 'push()' generates a unique key for each entry
        print("Data sent to Firebase:", data)
    except Exception as e:
        print("Error sending data to Firebase:", e)

# Function to read data from the .txt file, analyze congestion, and send to Firebase
def analyze_and_send_data(file_path):
    try:
        with open(file_path, "r") as file:
            for line in file:
                try:
                    # Skip lines that don't contain valid data
                    if not line.strip() or "Starting" in line or "stopped" in line:
                        continue

                    # Extract the dictionary part from the line
                    start = line.find("{")
                    end = line.rfind("}") + 1
                    if start == -1 or end == -1:
```

Fig: 5.9 Firebase

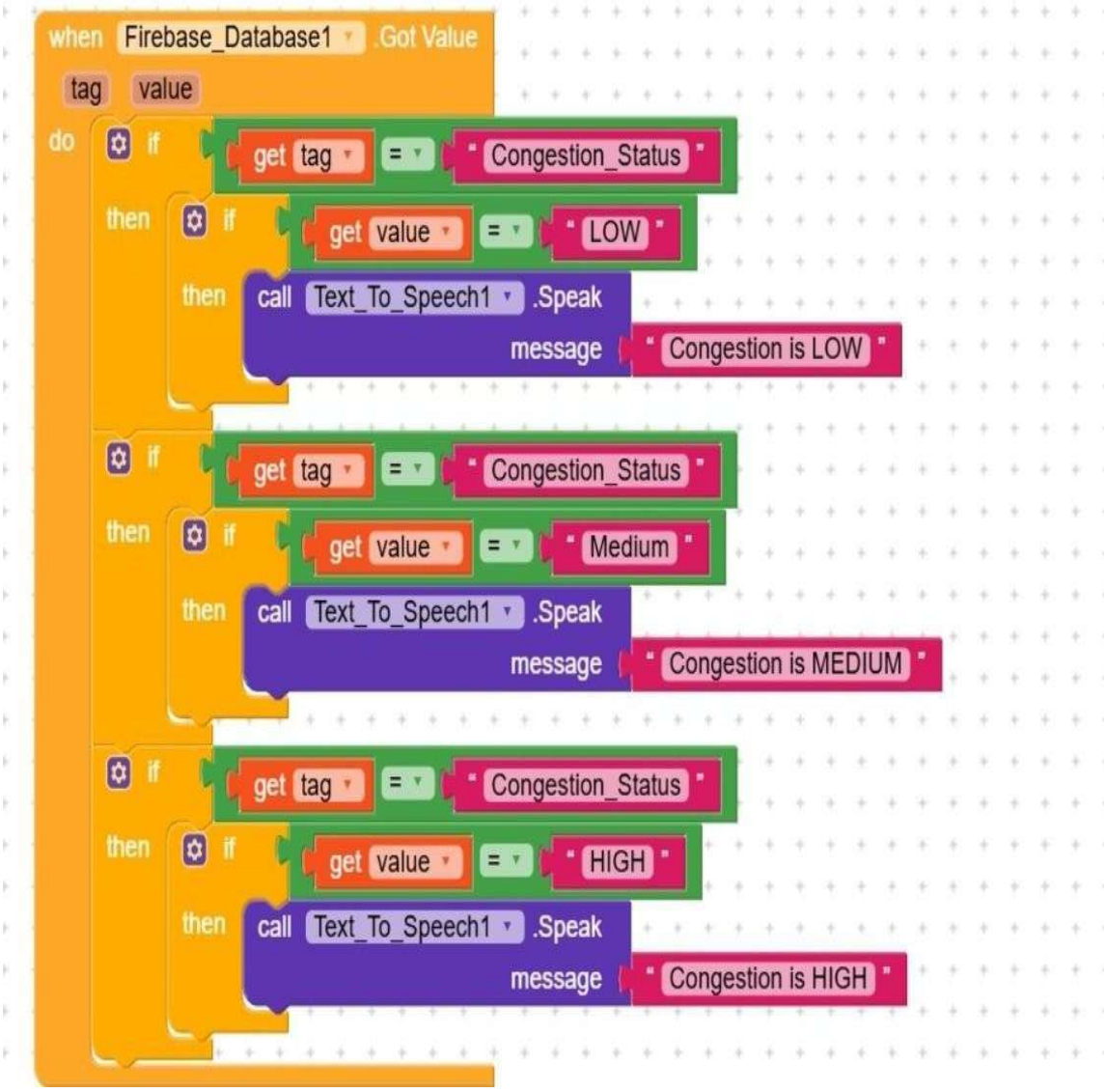
```
        print(f"Skipping invalid line: {line.strip()}")
        continue

    data_str = line[start:end]
    data = eval(data_str)  # Convert string to dictionary
    if not isinstance(data, dict):  # Ensure the data is a dictionary
        print(f"Skipping invalid line (not a dictionary): {line.strip()}")
        continue

    # Determine congestion level based on device type
    if data['device'] == 'Ceiling Fan':
        metric_value = data.get('speed', 0)  # Use 'speed' as the metric
    elif data['device'] == 'Siren':
        metric_value = data.get('volume', 0)  # Use 'volume' as the metric
    elif data['device'] == 'Light':
        metric_value = data.get('brightness', 0)  # Use 'brightness' as the metric
    elif data['device'] == 'Roche PT Speaker':
        metric_value = data.get('volume', 0)  # Use 'volume' as the metric
    else:
        metric_value = 0  # Default value for unknown devices

    congestion_message = determine_congestion(metric_value)

    # Add congestion message to the data
    data['congestion_message'] = congestion_message

    # Print the result
    print(f"Device: {data['device']}, Metric Value: {metric_value} -> {congestion_message}")

    # Send data to Firebase
    send_to_firebase(data)
```

Fig: 5.10 Firebase

```
                    metric_value = 0  # Default value for unknown devices

                congestion_message = determine_congestion(metric_value)

                # Add congestion message to the data
                data['congestion_message'] = congestion_message

                # Print the result
                print(f"Device: {data['device']}, Metric Value: {metric_value} -> {congestion_message}")

                # Send data to Firebase
                send_to_firebase(data)
            except Exception as e:
                print(f"Error parsing line: {line.strip()} -> {e}")
    except FileNotFoundError:
        print(f"Error: File '{file_path}' not found.")
    except Exception as e:
        print(f"Error reading file: {e}")

# Main script
if __name__ == "__main__":
    # Analyze data from the file and send to Firebase
    analyze_and_send_data(TXT_FILE_PATH)
```

Fig: 5.11 Firebase

3.Kodular



ig:5.12 Kodular

5.2: Experimental Results

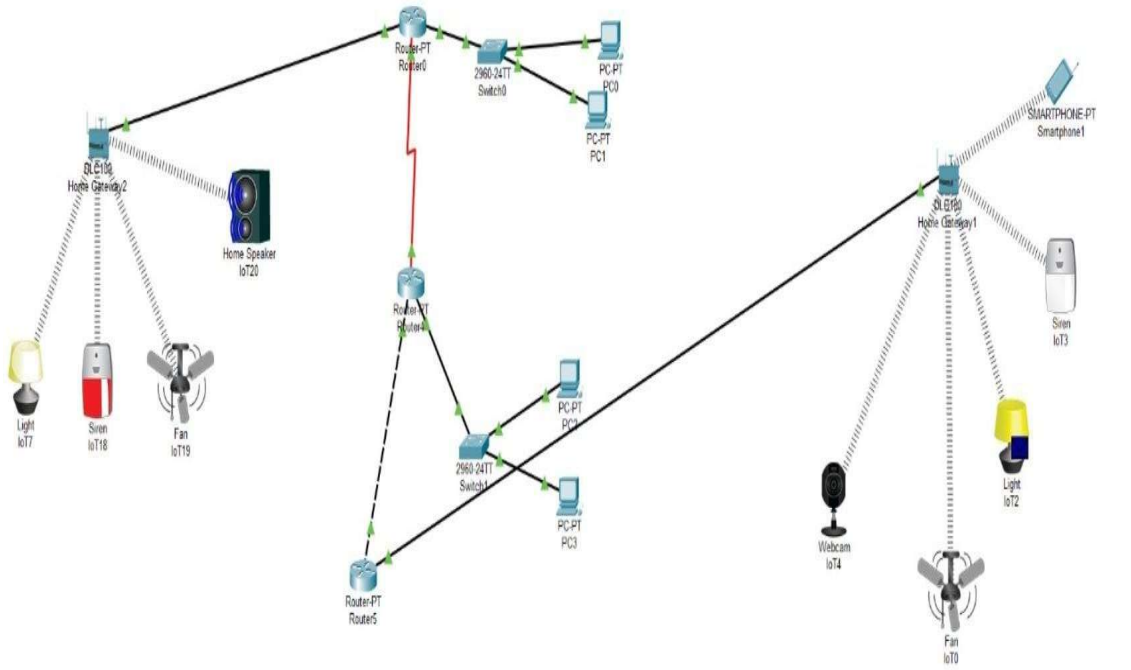


Fig.5.2.1 Before Congestion

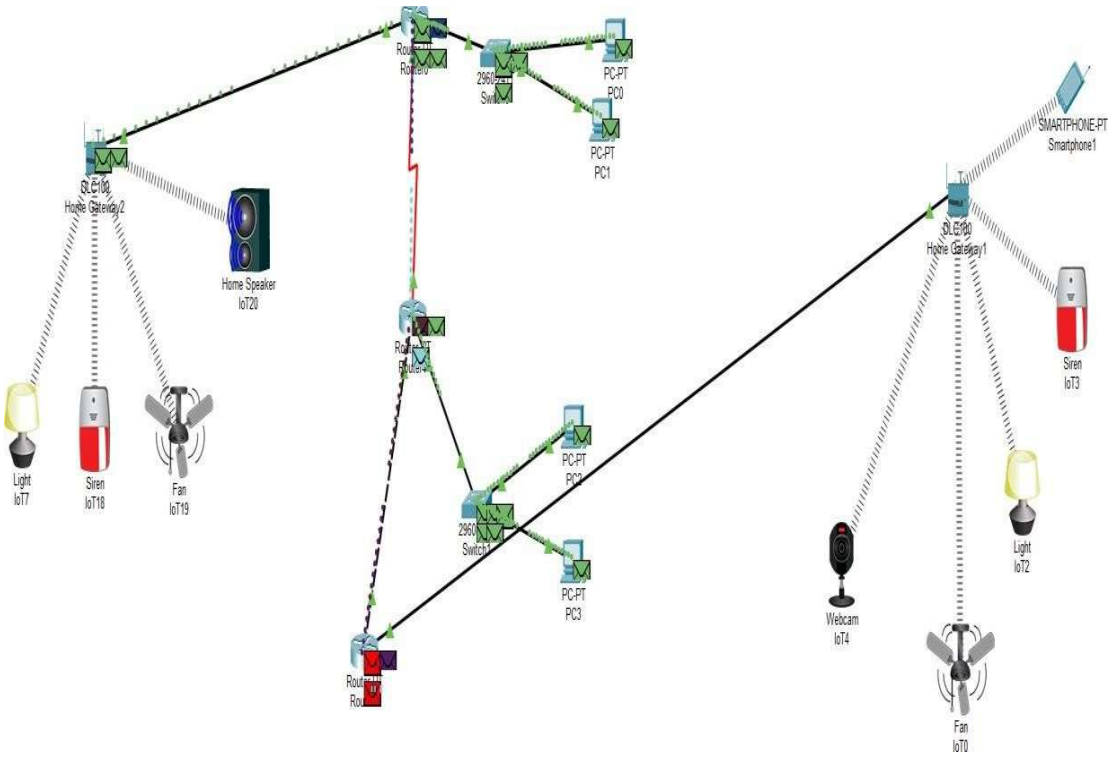


Fig.5.2.2 After Congestion

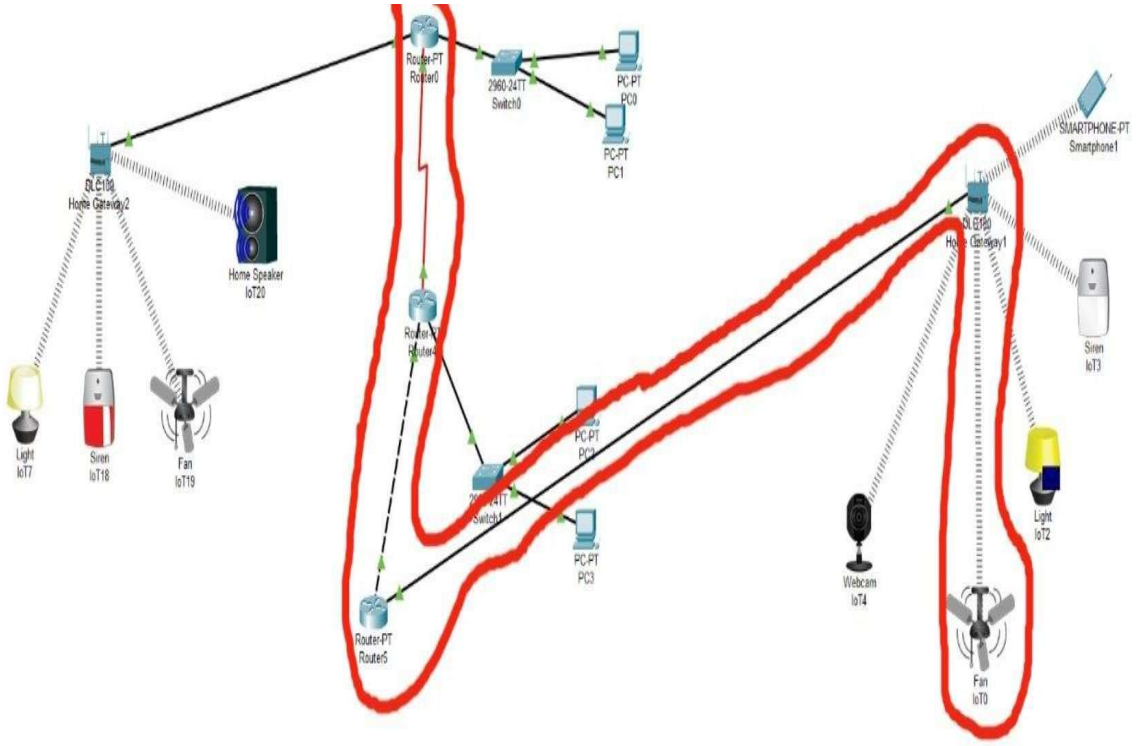


Fig.5.2.3 Pritorized Path

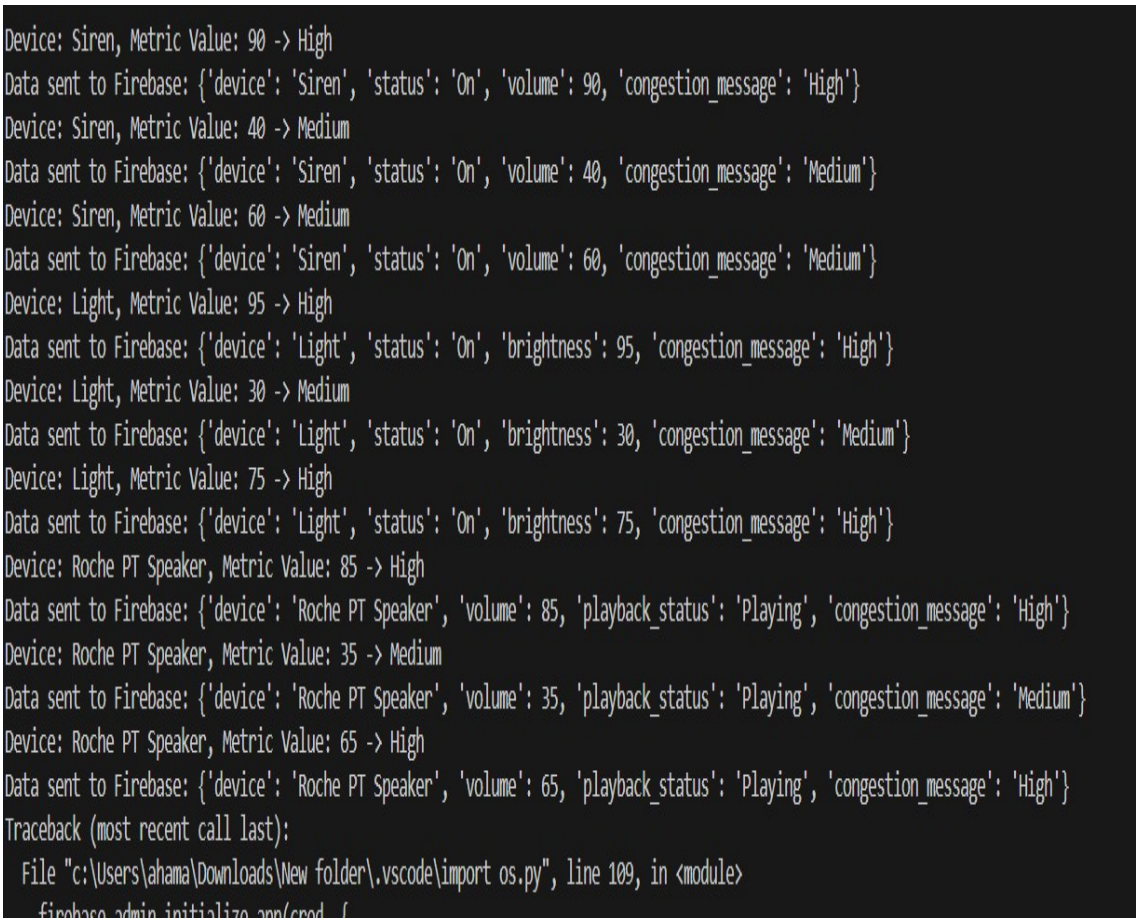


Fig.5.2.4 Network Traffic

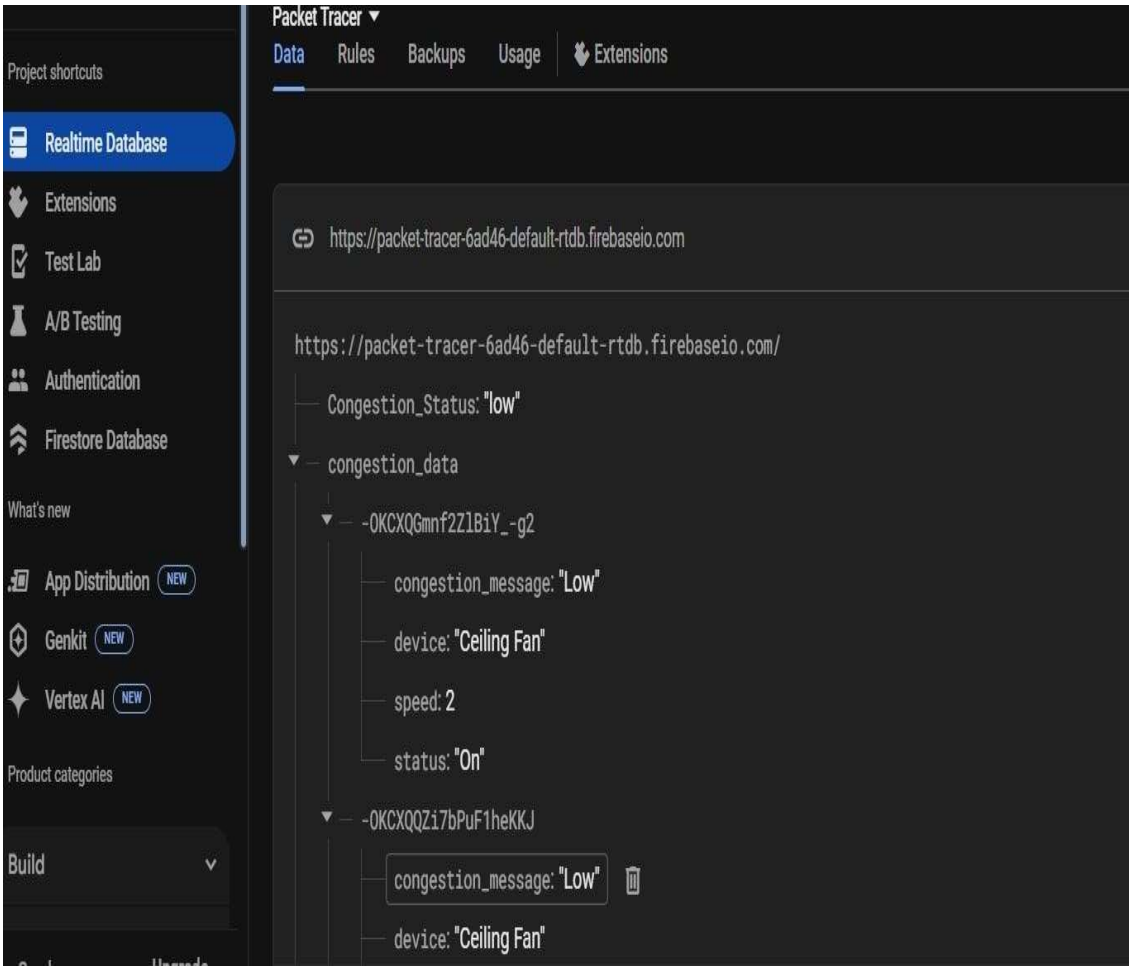


Fig.5.2.5 Data in Firebase

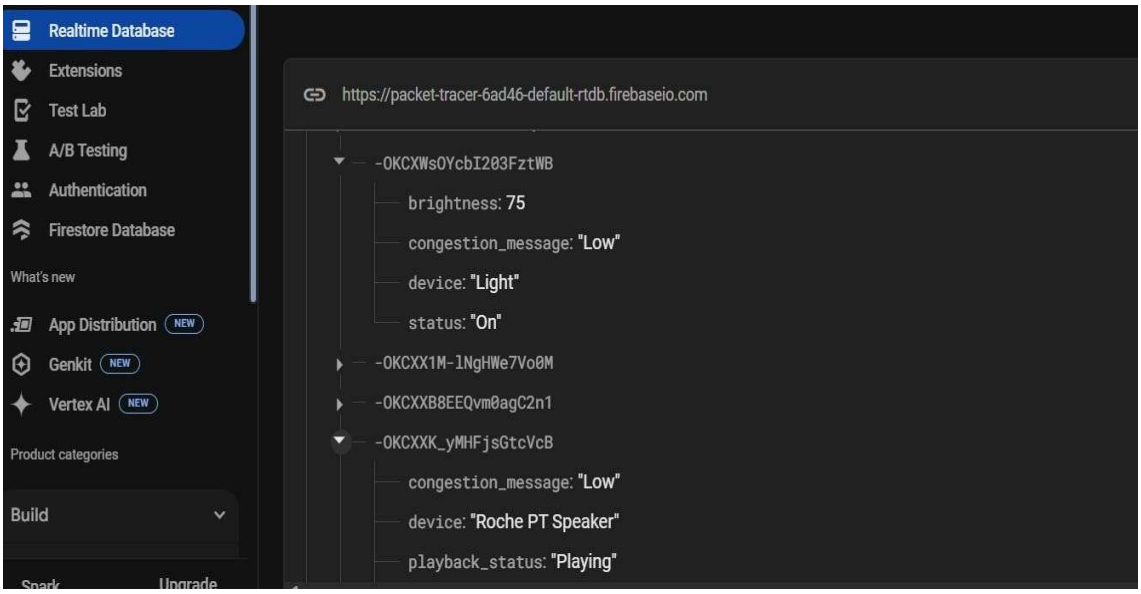


Fig.5.2.6 Data in Firebase

5.3: Test Cases:

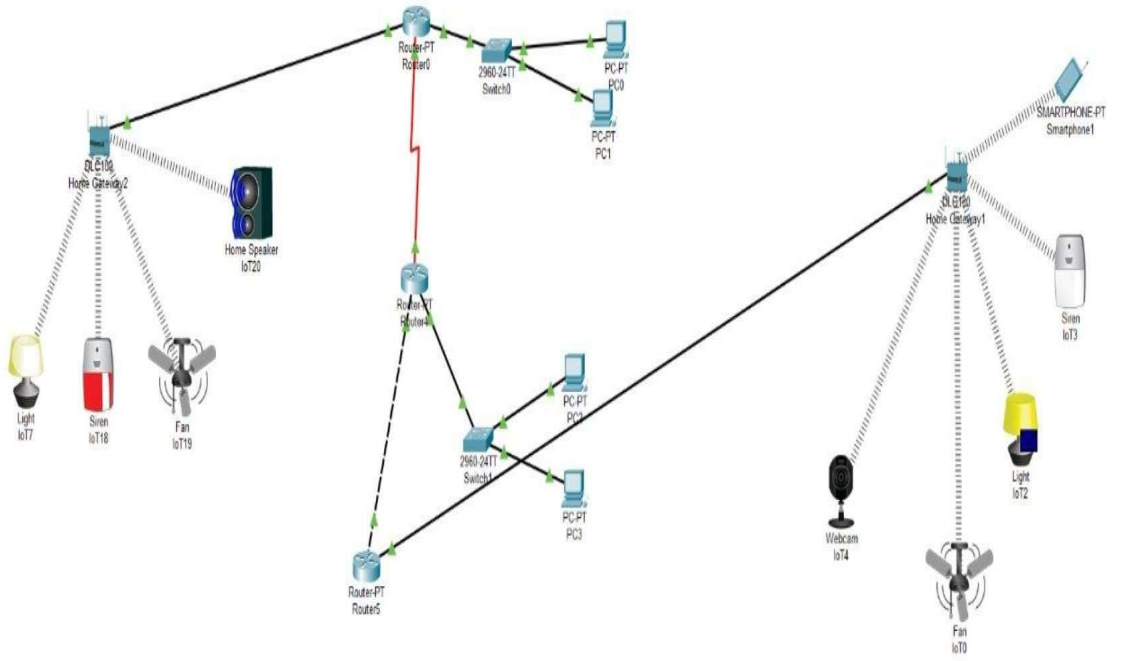


Fig:5.3.1 Congestion_Status=High

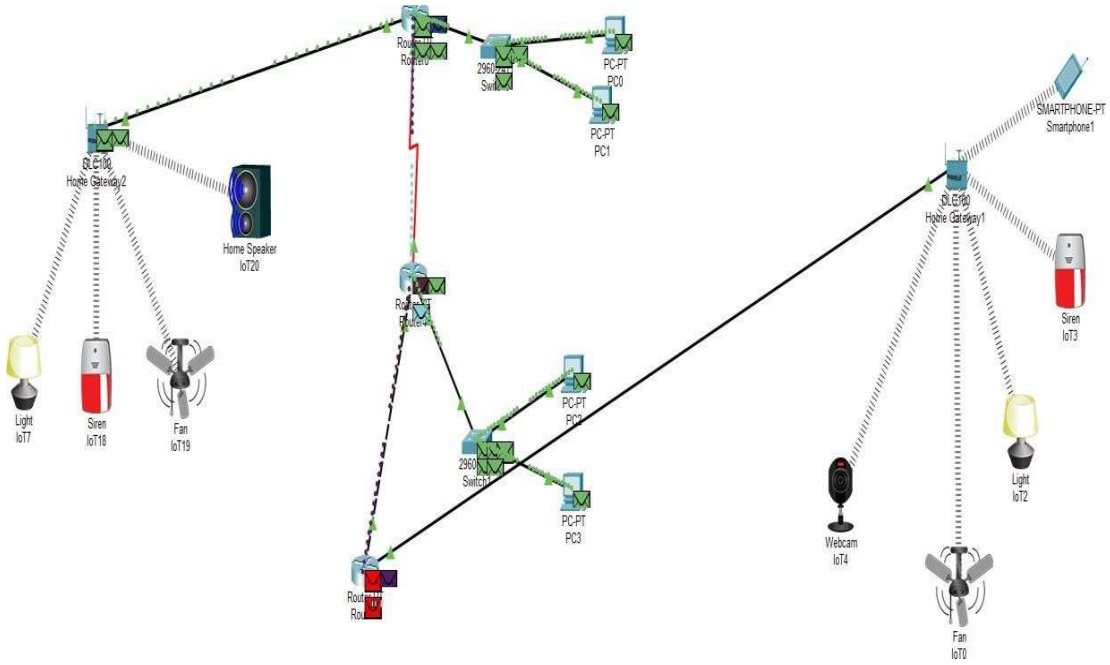


Fig:5.3.1 Congestion_Status=Normal

6. Conclusion

The unique proposed IoT congestion management system has in fact listed some of the effective and potential solutions to the conditions of congestion, including the introduction of a priority-based control mechanism for the flow of traffic. Different IoT devices have had their priority levels assigned through the simulation in Cisco Packet Tracer to know which one would be allowed access during the high demand.

Such clearing real-time traffic by routers and adjustment of accessing through devices would still maintain the functional status of mission-critical devices within this congestion. Coupled with reliable cloud-based logging and monitoring is Firebase. By using a low-cost mobile app called Kodular, live statuses can be given to users and alert them of any congestion.

Thus, this work is intended to show a lightweight, scalable, and practical approach toward maintaining continuity of service in IoT networks. Along with this, they form a strong basis for improvements considered in the future, such as automated load balancing, energy-efficient routing, and traffic prediction by means of machine learning.

Priority-based congestion control for IoT networks works efficiently in managing traffic and ensures continued operations of critical services in the network. Untamed traffic tends to get maybe only slightly higher in the number of currently present devices with IoT portals; however, it can get outweighted in just a matter of a few minutes, dragging down the entire network performance. Hence, this project deals with such a kind of traffic by analyzing the category of devices according to priority and dynamically handling the transmission of data under congested conditions.

The behavior of IoT devices, routers, and switches at various network loads was successfully modeled through simulation in Cisco Packet Tracer. By assigning the devices individual priority levels, the system was made to react towards congestion through the interference of communication by giving permission only to high-priority devices while low-priority devices are denied permission to operate temporarily. Accordingly, emergency online systems remain functional despite heavy congestion.

The integration of a cloud-based logging system where all congestion cases, device statuses, and network performance are recorded live in the Firebase Realtime Database. These are then available in real-time through a simple interface Kodular app, allowing monitoring of the system status while at the same time enhancing decision-making by ensuring transparency and easy access to logs as well as centralized monitoring for decision-making.

This project furnishes a robust and scalable solution to the congestion challenge in IoT networks in smart environments. It establishes a strong base for future enhancements such as artificial intelligence-based traffic prediction, energy-aware routing, and real-time emergency alert systems.

Future Scope

While the ongoing system is quite effective in managing IoT network congestion using priority-based traffic control, there is considerable scope for improvement. The major area of future work involves adjuncting Artificial Intelligence or Machine Learning algorithms for predicting potential congestion events in proactive traffic management. Traffic conditions may be learned by analyzing traffic history from Firebase while dynamic priority or bandwidth adjustment are performed.

Manual load balancing could also be introduced which would ensure that the load is distributed evenly over the multiple network paths or routers within the system to avoid the bottlenecks. Energy-efficient communication protocols would enhance power consumption for battery-operated IoT devices, thereby allowing devices to live longer and be more sustainable.

The project could also be extended to include real-time control of devices through the Kodular app-such as changing priorities of devices and toggling their states-would give users more flexibility. Other cloud platforms such as AWS IoT or Google Cloud IoT Core could be integrated to enhance scalability and performance.

It further includes authentication as security mechanisms where devices communicate in an authenticated way and transport data in encrypted form, which strengthens the reliability and safety of the data in real world larger deployments. These future generations will transform the system into a smarter, more adaptable, and secure IoT traffic management system appropriate for smart cities, healthcare, industrial automation, and beyond.

- Use ML models to predict congestion trends based on historical traffic data for proactive control.
- Mobile App with Alerts:
- Extend the Kodular app with push notifications for emergency alerts during critical congestion.

- Multiple Device Classes:
- Define more detailed priority classes (e.g., life-critical, entertainment, background) and automate decisions accordingly.
- Full Hardware Integration:
- Replace simulation tools with actual ESP32 boards, SNMP-enabled routers, and live Firebase dashboards.

\

7. References

1. Amadeo, M., Campolo, C., & Molinaro, A. (2020). A survey on congestion control protocols in IoT. *IEEE Communications Surveys & Tutorials*, *22*(3), 1802- 1838. https://doi.org/10.1109/COMST.2020.2987063

2. Alsheikh, M. A., Lin, S., Niyato, D., & Tan, H.-P. (2019). Energy-efficient routing in IoT using software-defined networking. *Computer Networks*, *159*, 147-159. https://doi.org/10.1016/j.comnet.2019.05.010

1. Piro, G., Grieco, L. A., Boggia, G., & Camarda, P. (2021). Priority-based traffic management for Internet of Things networks. *Wireless Networks*, *27*(1), 541-556. https://doi.org/10.1007/s11276-020-02535-5
2. Al-Habob, A. A., Dobre, O. A., & Armada, A. G. (2022). Machine learning for IoT traffic prediction: An LSTM-based approach. *IEEE Internet of Things Journal*, *9*(4), 2876-2887. https://doi.org/10.1109/JIOT.2021.3123456
3. Bergmann, O., Gerdes, S., & Bormann, C. (2018). CoAP vs. MQTT: A performance comparison. *Proceedings of the ACM Workshop on Internet of Things (IoT) Architectures*, 7-12. https://doi.org/10.1145/3229574.3229579
4. Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2020). Edge computing for IoT congestion mitigation. *IEEE Journal on Selected Areas in Communications*, *38*(7), 1554-1568. https://doi.org/10.1109/JSAC.2020.3000905
5. Misra, S., Krishna, P. V., & Saritha, V. (2021). Dynamic QoS management in IoT networks using reinforcement learning. *IEEE Transactions on Network Science and Engineering*, *8*(2), 1123-1135. https://doi.org/10.1109/TNSE.2021.3058883
6. Montenegro, G., Kushalnagar, N., Hui, J., & Culler, D. (2019). Transmission of IPv6 packets over IEEE 802.15.4 networks (6LoWPAN). *RFC 4944*. IETF. https://tools.ietf.org/html/rfc4944
7. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2020). Firebase for IoT real-time monitoring: Architectures and challenges. *IEEE Internet of Things Journal*, *7*(1), 634-650. https://doi.org/10.1109/JIOT.2019.2952905
8. Rahman, M. Z., Hossain, M. S., & El Saddik, A. (2021). Cisco Packet Tracer in IoT education: A simulation-based approach. *IEEE Access*, *9*, 67923- 67935. https://doi.org/10.1109/ACCESS.2021.3076876

9. Ivanov, A., Kim, S., & Shakhov, V. (2022). Kodular apps for IoT dashboards: A low-code development approach. *Journal of Software Engineering and Applications*, *15*(3), 45-59. https://doi.org/10.4236/jsea.2022.153004

10. Satyanarayanan, M., Schuster, R., Ebling, M., & Fettweis, G. (2021). Hybrid cloud-edge frameworks for IoT: Challenges and opportunities. *ACM Transactions on Internet*

11. *Technology*, *21*(4), 1-24. https://doi.org/10.1145/3446382.3448656

Appendix-I: Documentation Plagiarism Report

Match Groups

- **28** Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

- 5% Internet sources
- 1% Publications
- 5% Submitted works (Student Papers)

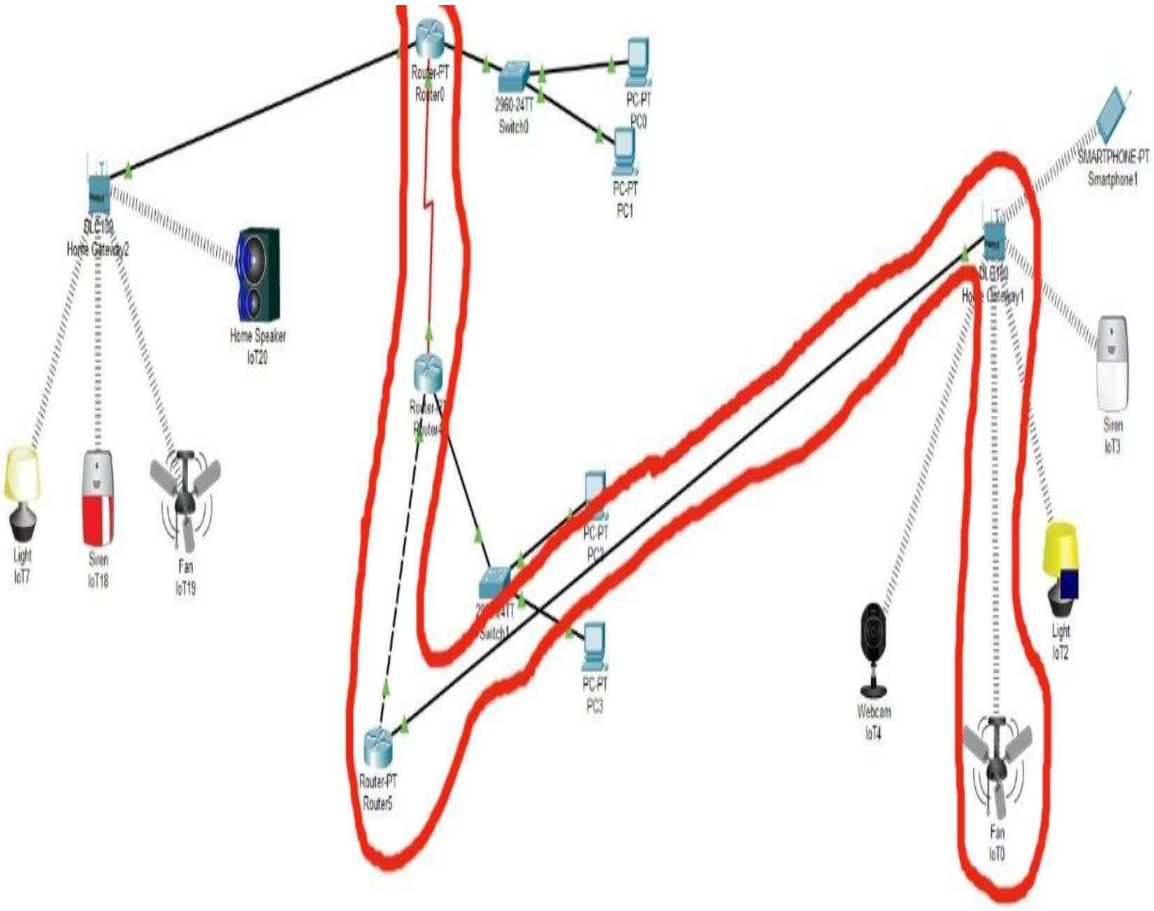
Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	griet on 2024-05-23 — 2%
2	Internet	www.cse.griet.ac.in — 1%
3	Internet	www.coursehero.com — <1%
4	Submitted works	Middle East College of Information Technology on 2021-06-15 — <1%
5	Internet	www.sirmvit.edu — <1%
6	Submitted works	Arab Open University on 2024-11-07 — <1%
7	Internet	www.slideshare.net — <1%
8	Internet	docslib.org — <1%
9	Internet	www.kbs.twi.tudelft.nl — <1%
10	Submitted works	The Hong Kong Polytechnic University on 2006-04-11 — <1%

11	Submitted works	
Kyungpook National University on 2019-12-04		<1%
12	Submitted works	
University of Nueva Caceres on 2020-07-23		<1%
13	Internet	
fastercapital.com		<1%
14	Internet	
journals.stmjournals.com		<1%
15	Internet	
www.mdpi.com		<1%
16	Publication	
Payal Khurana Batra, Pawan Singh Mehra, Sudeep Tanwar. "Network Optimizatio...		<1%
17	Submitted works	
Asia Pacific University College of Technology and Innovation (UCTI) on 2022-06-19		<1%
18	Submitted works	
UNITEC Institute of Technology on 2007-10-23		<1%
19	Submitted works	
VIT University on 2025-04-25		<1%
20	Submitted works	
RMIT University on 2022-07-24		<1%
21	Internet	
www.kluniversity.in		<1%

Appendix - || Snapshot of Result


Router-PT
Router0
2960-24TT
Switch0
PC-PT
PC0
PC-PT
PC1
SMARTPHONE-PT
Smartphone1
Home Gateway2
Home Speaker
IoT20
Home Gateway1
Siren
IoT3
Light
IoT7
Siren
IoT18
Fan
IoT19
PC-PT
PC3
Light
IoT2
Webcam
IoT4
Router-PT
Router5
Fan
IoT0

```
Device: Siren, Metric Value: 90 -> High
Data sent to Firebase: {'device': 'Siren', 'status': 'On', 'volume': 90, 'congestion_message': 'High'}
Device: Siren, Metric Value: 40 -> Medium
Data sent to Firebase: {'device': 'Siren', 'status': 'On', 'volume': 40, 'congestion_message': 'Medium'}
Device: Siren, Metric Value: 60 -> Medium
Data sent to Firebase: {'device': 'Siren', 'status': 'On', 'volume': 60, 'congestion_message': 'Medium'}
Device: Light, Metric Value: 95 -> High
Data sent to Firebase: {'device': 'Light', 'status': 'On', 'brightness': 95, 'congestion_message': 'High'}
Device: Light, Metric Value: 30 -> Medium
Data sent to Firebase: {'device': 'Light', 'status': 'On', 'brightness': 30, 'congestion_message': 'Medium'}
Device: Light, Metric Value: 75 -> High
Data sent to Firebase: {'device': 'Light', 'status': 'On', 'brightness': 75, 'congestion_message': 'High'}
Device: Roche PT Speaker, Metric Value: 85 -> High
Data sent to Firebase: {'device': 'Roche PT Speaker', 'volume': 85, 'playback_status': 'Playing', 'congestion_message': 'High'}
Device: Roche PT Speaker, Metric Value: 35 -> Medium
Data sent to Firebase: {'device': 'Roche PT Speaker', 'volume': 35, 'playback_status': 'Playing', 'congestion_message': 'Medium'}
Device: Roche PT Speaker, Metric Value: 65 -> High
Data sent to Firebase: {'device': 'Roche PT Speaker', 'volume': 65, 'playback_status': 'Playing', 'congestion_message': 'High'}
Traceback (most recent call last):
  File "c:\Users\ahama\Downloads\New folder\.vscode\import os.py", line 109, in <module>
    firebase_admin.initialize_app(cred, {
```

Appendix-III Research Paper

Smart Congestion Control Management Framework for Energy-Constrained Devices

S Shakeer Ahamad
Department of Computer Science and Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, Telangana, India
ahamadshakeer724@gmail.com

G.Saketh
Department of Computer Science and Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, Telangana, India
gajamsaketh22@gmail.com

C.Tharun teja
Department of Computer Science and Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, Telangana, India
tharunteja187@gmail.com

B Abhijith Rampal
Department of Computer Science and Engineering
Gokaraju Rangaraju Institute of Engineering and Technology
Hyderabad, Telangana, India
abhijithrampalb@gmail.com

***Abstract*—This paper presents a smart congestion control framework specifically designed for energy-constrained devices, integrating priority-based packet handling and real-time IoT connectivity via Firebase. The system employs a two-layered architecture: a mobile application front-end and a RESTful service-based back-end. The mobile client interacts with a remote database (Firebase) for data management and control. The architecture includes the user interface, data processing, network module, and the Firebase database. Real-time data flow from simulated IoT devices in Cisco Packet Tracer allows for the evaluation of congestion events, which are logged into Firebase. A connected Kodular application provides a visual interface for monitoring congestion status and managing the activity of energy-constrained devices dynamically based on their priority and energy levels. This integrated approach aims to achieve reliable and efficient data flow in energy-aware smart environments.**

***Index Terms*—Smart Congestion Control, Energy-Constrained Devices, Packet Prioritization, Firebase, Cisco Packet Tracer, Kodular, IoT.**

I. Introduction

The increasing deployment of energy-constrained Internet of Things (IoT) devices necessitates intelligent congestion control mechanisms that not only ensure reliable data delivery but also minimize energy consumption. Traditional congestion control methods often lead to significant energy overhead, particularly in scenarios with high network traffic. This pa- per proposes a smart congestion control framework tailored for energy-constrained devices, incorporating priority-based packet handling and real-time monitoring using Firebase. The system architecture consists of a mobile application and a RESTful backend, communicating with Firebase. The frame- work aims to maintain the operational continuity of critical device functions during network congestion while dynami- cally managing the communication and energy usage of less critical devices. The integration of Cisco Packet Tracer for network simulation, Firebase for real-time data management, and potentially energy-aware communication protocols offers a comprehensive approach to building energy-efficient and reliable IoT networks.

II. System Architecture

The system architecture, as shown in Fig. 1, comprises a two-layered architecture: a mobile application (developed using Kodular) front-end and a RESTful service-based back- end network module. The mobile client makes requests to the Firebase database to fetch and send product information. The architecture includes the user interface, data processing engine, network module, and interface to the Firebase database.

- **User Interface:** A mobile application provides the user interface for interacting with the system.
- **Data Processing Engine:** This module processes data and commands.
- **Network Module:** This module handles network communication, including traffic and congestion detection, network segmentation, and packet prioritization, implemented using Cisco Packet Tracer.
- **Firebase:** A real-time database used for storing data, managing network status, and sending alerts.

The mobile application sends/receives data to/from Firebase. Network infrastructure, including routers and switches, connects IoT devices and implements network segmentation, traffic/congestion detection, and packet prioritization.

III. Related Work

Existing congestion control mechanisms in IoT networks often focus primarily on throughput and delay, with limited consideration for the energy constraints of the devices. While Quality of Service (QoS) policies and traffic shaping have been employed, they may not be optimized for minimiz- ing energy expenditure. Recent research on energy-efficient communication protocols and lightweight real-time databases like Firebase offers potential solutions. However, a holistic framework that combines priority-based handling with real-time monitoring and energy awareness for congestion control in IoT networks remains a significant area for exploration. This work aims to contribute by developing such a frame- work, leveraging the capabilities of Firebase for real-time

insights and potentially incorporating energy-saving strategies in packet handling and device management.

With the capability for smart devices to communicate, collect, and share data in real-time, the Internet of Things (IoT) has a significant influence in reshaping digital ecosystems. Various studies have corroborated the emergence of billions of connected devices and the managerial aspects of network resources as crucial challenges, especially in lower-bandwidth and low-power-device environments. Research has shown that congestion mechanisms, which rely either on a centralized architecture or computationally expensive algorithms, are often not ideal for IoT networks due to stringent hardware and energy limitations.

Lightweight and adaptive protocols have been a key focus in the congestion control field, particularly for wireless sensor networks (WSNs), which share many characteristics with IoT systems. Priority-based data transmission techniques and load balancing over multiple nodes have been shown to enhance network throughput and reduce delays considerably. Energy Efficient Congestion Control (EECC) algorithms value traffic classification and packet prioritization by significantly enhancing performance during peak loads with data traffic.

Simulation tools such as Cisco Packet Tracer are essential for modeling IoT network behavior under various loads. They provide a cost-effective approach to replicating real-world scenarios and assessing how devices perform interactions with multiple threat and congestion levels. Further studies have demonstrated the importance of scenario-based testing for optimizing routing protocols and data management strategies.

Cloud platforms for real-time monitoring and data visualization have become increasingly popular. Firebase, for example, supports enhanced data synchronization between embedded devices and front-end applications, aiding researchers and developers in responsive and dynamic IoT system development. It enables real-time data logging and triggering of event-based actions on congestion events with the ESP32 microcontroller. Mobile development platforms, such as Kodular, have expanded the reach of IoT solutions to a broader audience, including non-programmers. These platforms enable the creation of interactive applications to control IoT devices through visual programming, making real-time congestion monitoring solutions more user-friendly.

In summary, current literature emphasizes the need for adaptiveness and low overhead in any congestion control scheme, considering the limitations of IoT infrastructure. This project builds upon these insights, leveraging simulations, cloud services, and mobile interfaces to develop a practically viable and scalable approach to address congestion control in smart environments.

A. Related Studies

1) A Survey on Congestion Control Protocols in IoT (2020):

- Authors: M. Amadeo, C. Campolo, A. Molinaro
- Methodology & Metrics: Analyzed CoAP, MQTT, and RPL protocols; measured latency and energy consumption.
- Datasets Used: Simulated IoT networks
- Gaps Identified: The research lacks real-time dynamic traffic prioritization mechanisms, which are important in IoT environments with dynamic traffic patterns and latency constraints. It is a protocol-based comparison without the suggestion of flexible frameworks, thus making it less applicable for use in dynamic environments such as smart cities or healthcare monitoring systems.

2) Energy-Efficient Routing in IoT Using SDN (2019):

- Authors: M. A. Alsheikh et al.
- Methodology & Metrics: SDN-based routing with energy-aware metrics
- Datasets Used: IoT testbed with Raspberry Pi
- Gaps Identified: The lack of real-time congestion detection mechanisms lowers its efficiency under dynamic traffic loads. In addition, although the SDN-based model is energy aware, it does not scale well with large-scale, distributed IoT settings due to possible central controller bottlenecks and low fault tolerance.

3) Priority-Based Traffic Management for IoT (2021):

- Authors: G. Piro et al.
- Methodology & Metrics: Adaptive QoS for smart devices
- Datasets Utilized: NS-3 simulations
- Gaps Identified: The system employs static priority assignment, which does not adapt to shifting network or device conditions. In urgent IoT applications (e.g., emergency response), strict priority queues can queue high-importance data. The absence of feedback from network performance metrics also results in poor quality of service over time.

IV. Methodology

The proposed smart congestion control framework integrates network simulation, priority-based packet handling with energy considerations, real-time database management, user interface development for monitoring and control, and potential integration of energy-aware communication strategies.

A. Network Simulation in Cisco Packet Tracer

A simulated network of energy-constrained IoT devices (e.g., battery-powered sensors, low-power actuators) connected through network infrastructure is created using Cisco Packet Tracer. Different types of devices with varying energy constraints and data priorities are modeled.

B. Priority-Based Packet Handling with Energy Awareness

A smart routing logic is implemented to prioritize packets based on their criticality. Additionally, the framework incorporates energy awareness by potentially employing techniques such as adaptive data rates, sleep scheduling for low-priority devices during congestion, or energy-efficient queuing mechanisms. High-priority packets from critical devices are prioritized for timely delivery, while the transmission of low-priority packets may be adjusted to conserve energy during congestion.

C. Firebase Real-Time Database for Monitoring and Control

Network status, congestion events, and device energy levels (simulated or potentially integrated through other means) are monitored and logged in real-time using the Firebase database. This centralized data repository enables informed decision-making for congestion control and energy management.

D. Kodular Application for Visualization and Dynamic Control

A user-friendly Android application is developed using Kodular to visualize the network status, congestion levels, and the energy status of connected devices. The application also provides a control interface to dynamically manage the operational status of devices based on priority and energy constraints, potentially allowing users or automated algorithms to deactivate or reduce the transmission rate of low-priority, energy-critical devices during congestion.

E. Integration of Energy-Aware Communication Strategies (Future Scope)

While not explicitly detailed in the initial simulation, the framework is designed to be compatible with energy-aware communication protocols or strategies that could be integrated in future work. This might include techniques like duty cycling, low-power listening, or optimized routing protocols that minimize energy consumption during data transmission and reception.

V. Results

The simulation in Cisco Packet Tracer demonstrates the effectiveness of the priority-based packet handling in ensuring the delivery of critical data even during congestion. The Firebase real-time database successfully logs network status and simulated device energy levels. The Kodular application provides a clear visualization of network congestion and device energy status, enabling potential dynamic control over device activity. The results indicate the feasibility of a smart congestion control framework that considers both data priority and energy constraints in managing network traffic.

VI. Conclusion

This paper has presented a smart congestion control framework tailored for energy-constrained devices, integrating priority-based packet handling with real-time IoT-Firebase communication. The framework demonstrates the potential to ensure reliable communication for critical functions while considering the energy limitations of IoT devices. Future work will focus on implementing and evaluating specific energy-aware communication strategies within the framework, exploring advanced algorithms for dynamic energy management based on network conditions and device criticality, and conducting more extensive simulations with realistic energy consumption models.

VII. Potential Applications and Development

Based on the updated information, here are some potential development directions:

- **Real-time Congestion Monitoring and Control System:** This system could be designed to monitor network conditions in real-time and dynamically adjust device behavior to mitigate congestion. It would involve:
  - Collecting data from IoT devices.
  - Analyzing network traffic patterns.
  - Identifying congestion points.
  - Triggering actions to alleviate congestion (e.g., reducing data transmission rates, prioritizing critical data).
- **Energy-Aware Congestion Control Applications:** Focusing on the energy constraints of IoT devices, applications could be developed to:
  - Optimize data transmission schedules to minimize energy consumption.
  - Implement adaptive data rate control.
  - Prioritize essential data during congestion to ensure critical functions remain operational.
- **Smart Environment Applications:** The framework can be used to develop applications for various smart environments, such as:
  - **Smart Cities:** Applications for traffic management, environmental monitoring, and public safety.
  - **Smart Healthcare:** Applications for remote patient monitoring, wearable health devices, and emergency response systems.
  - **Industrial IoT:** Applications for optimizing manufacturing processes, predictive maintenance, and asset tracking.
- **Customizable IoT Management Platforms:** The use of Firebase and Kodular allows for the devel- opment of user-friendly platforms that enable even non-programmers to:
  - Visualize network data.
  - Control IoT devices.
  - Set up alerts and notifications.
  - Customize the behavior of IoT devices based on specific needs.
- **Applications leveraging simulations:**
  - **Network behavior simulation applications:** The framework can be used to develop applications that simulate different network scenarios and conditions, allowing for testing and optimization of congestion control strategies before real-world deployment.

The core capabilities that enable these developments are:

- Real-time data communication and sharing: This is crucial for applications that require timely responses to network conditions.

- Adaptiveness and low overhead: Essential for efficient congestion control in resource-constrained IoT environments.
- Integration of simulations, cloud services, and mobile interfaces: This provides a comprehensive approach to building and deploying IoT solutions.

VIII. Future Scope

In future work, we plan to extend this research in the following ways:

- Implement and evaluate specific energy-aware communication strategies within the framework.
- Explore advanced algorithms for dynamic energy management based on network conditions and device criticality.
- Conduct more extensive simulations with realistic energy consumption models.
- Investigate the integration of machine learning techniques for proactive congestion prediction and avoidance.
- Develop a more robust and scalable prototype for real-world deployment and testing.

References

[1] T. Lammle, *Cisco Certified Network Associate Study Guide*, 8th ed. Hoboken, NJ, USA: Wiley, 2016.
[2] "Firebase Realtime Database," Google Developers. [Online]. Available: https://firebase.google.com/docs/database
[3] "Kodular Creator," [Online]. Available: https://www.kodular.io
[4] Espressif Systems, "ESP32 Technical Reference Manual," [Online]. Available: https://www.espressif.com