Power System Technology
ISSN:1000-3673

Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

# Improving Machine to Machine Communication in Integrated MQTT and COAP IoT Network

**Reshma N. Bhai**
Professor, Department of Electronics and Telecom. Engg., Institute of Civil & Rural Engineering Gargoti, India.
reshmabhai78@gmail.com

**Dr. Mahadev S. Patil**
Professor, Department of Electronics and Telecom. Engg. Rajarambapu Institute of Technology, (Affilaited to Shivaji University) Islampur, India
mahadev.patil@ritindia.edu

**Abstract:** This study presents a pioneering integration of MQTT and CoAP protocols within a Software-Defined Networking (SDN) framework, complemented by a Support Vector Machine (SVM)-based flag status indicator, to enhance interoperability and performance in IoT networks. The primary goal is to address the escalating demand for seamless communication in heterogeneous IoT environments. The integration of MQTT and CoAP through the SDN controller signifies a significant advancement, providing centralized control and programmability for dynamic management of communication protocols and network resources. This approach fosters flexibility, essential for accommodating diverse IoT devices with varying communication requirements, streamlining communication pathways and elevating network efficiency. A key innovation is the introduction of an SVM-based flag status indicator, leveraging SVM's capabilities in handling non-linear relationships and resisting over fitting. This indicator proves instrumental in accurately determining flag status, surpassing alternative classifiers like Decision Trees, Naive Bayes, and Random Forests. The study's experimental results validate the success of the integrated approach, demonstrating superior throughput, sustainable residual energy levels, reduced end-to-end delays, and a higher message delivery rate compared to alternative classifiers. These outcomes affirm the methodology's efficacy in achieving seamless interoperability, efficient communication, and robust decision-making across diverse IoT scenarios.

**Keywords:** IoT Integration, MQTT and CoAP Protocols, Software-Defined Networking (SDN), Support Vector Machine (SVM).

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

## 1. Introduction

The IoT envisions a future where countless devices, commonly known as "smart objects," are seamlessly interconnected across various environments. These smart objects are inherently diverse, varying widely in hardware capabilities (like processing power and memory), software stacks (including operating systems and installed applications), and communication standards [1]. The complexity of IoT systems intensifies as conventional Internet components such as desktops, smartphones, and cloud infrastructures interact with these heterogeneous devices. Initially, IoT deployments were isolated and tailored to specific domains, functioning primarily as proofs of concept rather than scalable, interoperable ecosystems capable of supporting cross-domain applications [2]. However, there is a growing consensus on the need for unified and scalable architectures. In recent times, the proliferation of connected devices via IoT technologies has accelerated significantly. A wide range of sectors including agriculture, healthcare, industrial automation, security systems environmental monitoring, smart transportation, and home automation are increasingly integrating IoT functionalities [3]. These implementations typically focus on data exchange or remote control through cloud-supported communication protocols. Within the IoT layered model, the application layer plays a crucial role, acting as the bridge between user-facing applications and the underlying network, and offers several communication standards [4]. As more IoT use cases emerge, there is a pressing need to evolve or design protocols that support flexible adaptation to varying network states and ensure compatibility across systems. Nonetheless, customizing protocols for each individual application remains a complex and resource-intensive task [5]. Machine Learning (ML) offers promising solutions by enabling intelligent and adaptive protocol behavior.

To ensure device-level interoperability, Internet Protocol (IP)-based stacks such as HTTP and CoAP are often utilized as foundational components in building a Web of Things. However, managing the diversity of device capabilities and establishing uniform IP-based communication necessitates network components that can interlink disparate systems, allowing seamless end-to-end data exchange [6]. To achieve smooth interaction with applications, intermediary components like proxies may be needed, especially when translating between different communication protocols or data formats.

Addressing these requirements, we introduce an IoT Hub architecture strategically positioned at the edge of interconnected networks. This hub extends the functionality of the network by serving as a border router, intermediary proxy, content cache, and resource registrar. The aim is to enhance communication efficiency and support interoperability among diverse IoT devices. We validate this concept through the design and deployment of a practical IoT Hub within a real-world test bed and evaluate its performance, offering valuable insights for the development of cohesive and robust IoT infrastructures.

The integration of MQTT and CoAP-based IoT network devices on a common platform through the use of an SDN controller represents a significant advancement in the field of IoT.

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

By unifying communication protocols and leveraging the capabilities of Software-Defined Networking (SDN), this integration streamlines device interoperability and communication within the IoT ecosystem. The SDN controller plays a pivotal role in managing and orchestrating the interactions between MQTT and CoAP devices, ensuring a cohesive and efficient network architecture.

To enhance the system's intelligence and decision-making capabilities, a machine learning-based approach is introduced. This approach involves designing a system for determining the status flag in the IoT network. The integration of Support Vector Machines (SVM), Decision Trees (DT), Naive Bayes (NB), and Random Forest (RF) based classifiers adds a layer of intelligence to the system. These classifiers are trained using a comprehensive traffic dataset representative of the IoT network, allowing them to learn and adapt to the intricate patterns and behaviors exhibited by connected devices.

The machine learning classifiers are employed to evaluate the best suitability of each classifier in the system based on their performance metrics and accuracy in predicting and deciding status flags. This intelligent decision-making process assists in optimizing the overall performance of the IoT network, ensuring efficient resource allocation, and enhancing the system's responsiveness to dynamic changes.

Furthermore, the integration of these classifiers into the system serves to provide personalized and adaptive services to IoT devices from the server thereby providing interoperability to the IoT devices that are protocol dependent. The classifiers dynamically assess the state of each device and determine the most appropriate service or action, contributing to a more responsive and tailored user experience.

By combining the robust integration of MQTT and CoAP, the centralized control facilitated by the SDN controller, and the intelligent decision-making capabilities of machine learning classifiers, this comprehensive approach represents a significant leap forward in creating a more resilient, efficient, and adaptive IoT ecosystem. The utilization of diverse classifiers allows for a thorough evaluation of the system's performance, ensuring the selection of the most suitable classifier for the unique characteristics of the IoT network traffic dataset. This integrated framework holds the potential to redefine the landscape of IoT networks by enhancing their intelligence, scalability, and overall performance.

## 2. Related Work

Over the last ten years, numerous studies have examined the IoT, exploring its uses, supporting technologies, system architectures, and related concerns such as privacy and security [7]. Zikria et al. [8] focused on ML techniques applied to IoT, addressing both practical implementations and obstacles, as well as their role in 5G communication and protection mechanisms. The works in have analyzed multiple IoT communication protocols across different layers, including application, network, and physical. Al et al. [9] reviewed

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

conventional protocols like the CoAP, Hypertext Transfer Protocol (HTTP), Extensible Messaging and Presence Protocol (XMPP), MQTT, and Data Distribution Service (DDS). Sethi et al. [10] provided a concise comparison between HTTP, CoAP, and MQTT alongside other layered protocols. A column article in [8] emphasized the significance of application-layer communication protocols in IoT systems. Salman et al. [11] offered a basic comparative analysis between application-layer and other networking layer protocols. Palattela et al. [9] provided a comparative study between CoAP and HTTP, highlighting their respective advantages without proposing new research directions. MQTT and CoAP were evaluated under unreliable and delayed transmission conditions [12], indicating MQTT's superior performance for IoT contexts. Aijaz et al. investigated [13] IoT communication protocols specifically for cognitive machine-to-machine (M2M) interactions, where they concentrated solely on application-level communication through CoAP. Granjal et al. [14] focused on the security challenges inherent in CoAP while neglecting other relevant application-level protocols and future improvement scopes. A broad overview of legacy application-layer protocols including HTTP, CoAP, XMPP, AMQP, MQTT, and DDS was given [15], albeit without identifying their limitations or proposing future advancements. Mijovic et al. [16]analyzed the computational efficiency of CoAP, MQTT, and WebSocket in varied environments. A comparative review of legacy application-layer protocols was also conducted by Saritha et al. [17], though the study lacked detailed discussion on research gaps. Interoperability concerns among HTTP, CoAP, MQTT, and AMQP were investigated [18]. Safaei et al. [19] explored reliability trade-offs, concluding that MQTT is well-suited for IoT. Practical evaluations involving MQTT and CoAP are presented [20]. Pohl et al. [21]assessed the performance of CoAP, HTTP, XMPP, WebSocket, MQTT, and AMQP through diverse network configurations. A detailed examination of IoT application-level protocol performance was carried out [22]. Sandeli et al. [23]presented an analysis of application-level coding strategies for IoT, outlining their implementations, pros, and cons. Finally, they assessed the performance of HTTP, CoAP, and MQTT under various operational conditions, with MQTT emerging as the most efficient and lightweight option.

The Constrained Application Protocol (CoAP), specified in RFC-7252, is a lightweight communication protocol tailored for low-power and low-bandwidth IoT environments. Developed by the IETF CoRE working group, CoAP draws inspiration from HTTP but operates over the User Datagram Protocol (UDP). Despite its efficiency, CoAP encounters challenges in managing congestion, particularly due to its use of a simplistic Binary Exponential Backoff (BEB) strategy [24]. When traffic load increases and bandwidth remains limited, congestion results in repeated transmissions, which in turn lead to higher energy consumption, increased delays, packet drops, and reduced system throughput and Packet Delivery Ratio (PDR) [25].

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

To improve congestion handling, an enhanced end-to-end control mechanism known as CoCoA (Congestion Control/Advanced) was introduced as an extension of CoAP. A key improvement in CoCoA lies in its dynamic calculation of the Retransmission Timeout (RTO), replacing the static RTO used in CoAP with a variable one based on Round-Trip Time (RTT), similar to methods used in TCP. In a study by Betzler et al. [26], testbed experiments comparing CoAP and CoCoA across different node configurations revealed that CoCoA improved the PDR by 14–45%.

Further development led to CoCoA+, which enhances performance by replacing the BEB with a Variable Backoff Factor (VBF), calculated using the initial RTO to prevent excessive retransmissions in short durations [27]. Both CoCoA and CoCoA+ use strong and weak RTO values to adjust retransmission behavior. Building on these, CoCo-RED was proposed by Suwannapong et al. [28], incorporating a Fibonacci Pre-increment Backoff (FPB) strategy. Akpakwu et al. [29] later proposed a context-sensitive congestion control technique that adapts the RTO based on dynamic RTT values.

More recent advancements include Precise CoCoA (pCoCoA) [30] and CoCoA++ [31], which further refine the RTO mechanism. pCoCoA simplifies operations by maintaining a single smoothed RTO instead of dual RTOs. It leverages the retransmission counter from acknowledgments of Confirmable (CON) messages to control repeated transmissions, achieving lower computational demand than its predecessors. CoCoA++, similarly using a single RTO, integrates CAIA Delay-Gradient (CDG) and a Probabilistic Backoff Function (PBF) in its RTO estimation process. Unlike earlier versions, CoCoA++ does not rely on per-packet RTT and replaces VBF with PBF, obtaining congestion insights through TCP's congestion window metrics. In essence, these developments aim to enhance CoAP's resilience to congestion by implementing more adaptive and optimized RTO algorithms. Based on various literature following challenges are seen in IoT networks:

### 2.1 Challenges in Interoperability

Interoperability is a critical aspect in the context of the IoT, where a multitude of devices, protocols, and platforms coexist. Achieving interoperability ensures that different components within the IoT ecosystem can communicate seamlessly, exchange data, and collaborate effectively. However, interoperability challenges are common, especially when dealing with diverse IoT devices and protocols.

1. Protocol Diversity: The details of several conventional application-layer protocols in the IoT domain, including HTTP, CoAP, MQTT, and AMQP. While each protocol may serve specific use cases efficiently, the coexistence of multiple protocols can lead to interoperability challenges. Ensuring that devices using different protocols can communicate without issues becomes crucial.

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

2. Comparison Studies: Some of the referenced studies briefly compare different protocols, including HTTP, CoAP, and MQTT, but they may not delve deeply into the intricacies of ensuring seamless interoperability between devices using these protocols. Achieving true interoperability involves addressing not only basic communication but also understanding and accommodating the nuances of each protocol.
3. IoT Application Layer Protocols: The interoperability issues of various application layer protocols are covered. However, the specific challenges and potential solutions for ensuring these protocols work harmoniously are not extensively discussed. Different devices may have varying levels of support for these protocols, requiring careful consideration.
4. Context-Aware Congestion Control: While not explicitly discussed as an interoperability challenge, the context-aware congestion control mechanisms introduced for CoAP, such as CoCo-RED, suggest an awareness of the need to adapt to different network conditions. This kind of adaptability is crucial for ensuring interoperability across diverse network environments.
5. Open Challenges: The passage suggests that some of the literature does not list open challenges for future research, indicating a potential gap in addressing interoperability challenges. Identifying and documenting these challenges is essential for the IoT community to work collaboratively towards solutions.

**2.2 Challenges in Energy Efficiency**

Energy efficiency is a paramount concern in IoT, where many devices operate with limited power resources, such as batteries. Optimizing energy consumption ensures prolonged device lifetimes, reduces maintenance needs, and contributes to the overall sustainability of IoT deployments.

1. Low-Powered Protocols: CoAP is considered as low powered low bandwidth protocol. This feature aligns with the overarching goal of energy efficiency in IoT devices. The choice of lightweight and efficient protocols is crucial for devices that may have stringent power constraints.
2. Congestion-Related Energy Consumption: The passage mentions that congestion in CoAP can lead to increased energy consumption. Congestion-related issues, such as network retransmissions, can impact the energy efficiency of devices. The introduction of CoCoA, with a focus on mitigating congestion, aims to address this challenge and improve energy efficiency.
3. Variable Retransmission Timeout (RTO): The transition from a fixed RTO in CoAP to a variable RTO in CoCoA is designed to enhance energy efficiency. A variable RTO based on Retransmission Time (RTT) calculation allows for more adaptive and energy-

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

conscious retransmission strategies, reducing unnecessary energy consumption during periods of network stability.

4. Extensions for Optimization: The introduction of extensions like CoCoAP+, Precise CoCoA (pCoCoA), and CoCoAP++ with optimized RTOs reflects the ongoing efforts to further optimize energy consumption. The passage suggests that these extensions provide advancements in computing optimal RTOs and managing retransmissions more efficiently.

While the passage touches upon interoperability challenges and energy efficiency concerns in the context of IoT and CoAP, there is a need for more in-depth exploration and documentation of specific challenges and solutions. Addressing these aspects is crucial for fostering a robust and sustainable IoT ecosystem.

## 3. Proposed Work

Combining MQTT (Message Queuing Telemetry Transport), CoAP (Constrained Application Protocol), and SDN can establish a robust and adaptable communication framework tailored for IoT systems and devices. Here's a high-level overview of this proposed integration:

1. MQTT and CoAP Integration: MQTT and CoAP are both lightweight, efficient protocols designed for IoT devices. However, they have different communication paradigms. MQTT follows a publish-subscribe model, whereas CoAP uses a request-response model. To integrate them, a translation layer is needed to facilitate communication between MQTT and CoAP. MQTT Broker with CoAP Plugin: Set up an MQTT broker that supports CoAP translation. When an MQTT message is received, the broker translates it to CoAP and forwards it to the CoAP endpoints. When a CoAP response is received, the broker translates it back to MQTT and publishes it to the appropriate MQTT topic.
2. SDN Integration: SDN allows centralized control of the network through a controller, which can be beneficial in managing IoT device communication and optimizing data flow. For SDN integration, OpenFlow is a widely used protocol to communicate between the controller and the network devices.
   - SDN Controller: Deploy an SDN controller, like OpenDaylight or ONOS, which will manage the network devices and provide a central point of control.
   - OpenFlow-enabled IoT Devices: IoT devices that support OpenFlow protocol are necessary to enable SDN control over them. These devices can include switches, routers, or gateways.
   - SDN Southbound Interface: The SDN controller communicates with the network devices through the southbound interface, which can be OpenFlow in this case.

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

- SDN Northbound Interface: The SDN controller receives commands and instructions from higher-level applications through the northbound interface. This is where the integration with MQTT and CoAP comes into play.

3. Integration of MQTT, CoAP, and SDN: Now that we have an MQTT-to-CoAP translation layer and an SDN controller in place, we can integrate them to create a unified system for IoT communication.
   - MQTT-to-CoAP Translation with SDN: The SDN controller can be programmed to manage the translation process between MQTT and CoAP. It can monitor the incoming MQTT messages and determine the appropriate CoAP endpoints for translation. Additionally, it can keep track of CoAP responses and translate them back to MQTT for publishing.
   - Dynamic Network Configuration: SDN enables dynamic network configuration based on real-time data and conditions. With the combination of MQTT and CoAP, the SDN controller can optimize data flow, prioritize traffic, and handle failures more efficiently.
   - Load Balancing and Traffic Routing: SDN can help distribute the load and traffic across the network by dynamically routing messages between MQTT and CoAP endpoints based on network conditions and device capabilities.
   - Security and Access Control: SDN can enhance security by enforcing access control policies and providing a centralized point for monitoring and managing security measures.

By integrating MQTT, CoAP, and SDN, the IoT communication infrastructure becomes more flexible, scalable, and manageable, enabling efficient data exchange and communication between IoT devices and applications. However, it's important to thoroughly plan, test, and secure the implementation to ensure its reliability and effectiveness in real-world IoT scenarios.

The handshaking Mechanism:

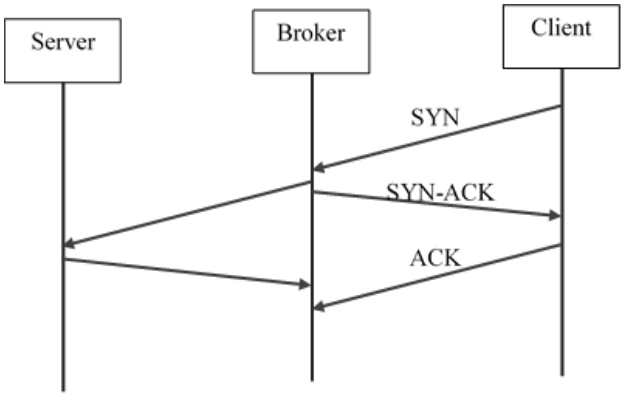



Figure 1: Three way Handhake in MQTT based WSN network

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

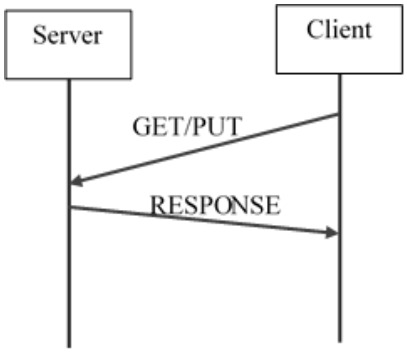



Figure 2: Handshake in CoAP based WSN system

The MQTT Three-Way Handshake is a TCP-based mechanism used to establish a connection between an MQTT client and a broker, as illustrated in Figure 1. This process involves three steps: first, the client initiates the handshake by sending a SYN (Synchronize) message; second, the broker acknowledges with a SYN-ACK (Synchronize-Acknowledgment); and finally, the client sends an ACK (Acknowledgment) to confirm the connection. This is a classic TCP handshake process optimized for MQTT communication.

The CoAP (Constrained Application Protocol) Request-Response handshake operates over UDP and is employed for communication between CoAP clients and servers, as shown in Figure 2. The client sends a request message such as GET or PUT, and the server responds with a corresponding response message, completing the lightweight handshake. In the context of SDN, the controller communicates with OpenFlow-enabled devices using an SDN handshake. This involves sending an OFPT_HELLO message to initiate communication, to which the device responds, thereby establishing a control channel.

To bridge the gap between MQTT and CoAP, a Translation Layer Handshake is introduced. This custom layer ensures compatibility and seamless data exchange between the two protocols by translating their formats and ensuring consistent communication.

### 3.1 Session Management

Session Management plays a vital role in maintaining ongoing communication with IoT devices. It tracks session states, communication parameters, and security credentials. Maintaining these sessions becomes critical when dealing with heterogeneous devices in a large-scale network.

Proposed Machine Learning-Enabled SDN Framework

To intelligently manage IoT communication and protocol selection, a machine learning-based SDN framework is proposed. The SDN controller is empowered with ML capabilities to enhance decision-making processes with system formation as shown in Figure 3.

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

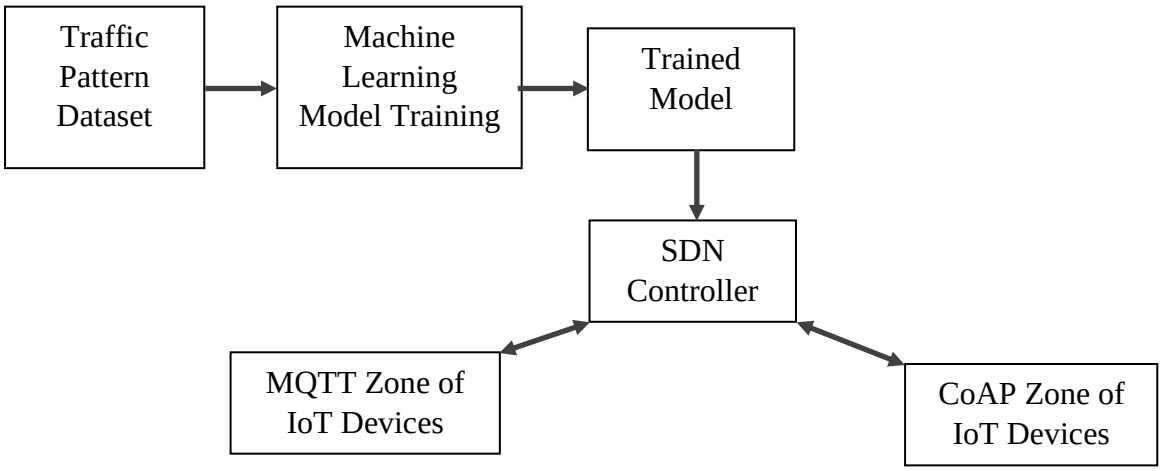



Figure 3: Proposed Machine Learning Based Optimization Framework

### 3.2 Dataset Integration

The process begins with the integration of the IoT Traffic Generation Patterns Dataset, which contains parameters such as data rate, start time, sampling interval, delay constraints, and the total number of active devices. These parameters are mathematically modeled as follows:

Let:

- $D$: Total number of IoT devices
- $R_d$: Data rate of device $d$
- $T_s$: Sampling time interval
- $T_{\text{start},d}$: Start time of device $d$
- $\delta$: Delay constraint

The total traffic at time $t$, denoted $\tau(t)$, is given by:

$$\tau(t) = \sum_{d=1}^{D} R_d \cdot 1_{\{T_{\text{start},d} \le t \le T_{\text{start},d}+T_s\}} \quad \ldots(1)$$

Here, $1_{\{.\}}$ is an indicator function that returns 1 if the device is active during time $t$, and 0 otherwise.

Machine Learning Facility

Multiple classifiers are considered for traffic pattern learning, including Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree (DT), and Random Forest (RF). The classifiers are trained using labeled data from the traffic dataset.

The classification task aims to predict the optimal protocol (MQTT or CoAP) or management action based on the input features. The learning process minimizes the cross-entropy loss function:

$$\mathcal{L} = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) \quad \ldots(2)$$

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

Where:

- $N$: Total number of training samples
- $C$: Number of classes
- $y_{i,c}$: Actual label for class $c$ of sample $i$
- $\hat{y}_{i,c}$: Predicted probability for class $c$ of sample $i$

Feature Engineering and Protocol Decision

Important features such as traffic load, device count, and delay requirements are extracted and selected. Based on the incoming traffic pattern $x$, the model predicts the optimal protocol using:

$$\hat{P} = \arg \max_{p \in \{\text{MQTT,CoAP}\}} f_p(x) \quad \ldots(3)$$

Where $f_p(x)$ is the classifier's output score for protocol $p$.

### 3.3 Classifier Integration with SDN Controller

The trained ML models are integrated into the SDN controller, which queries them in real-time to decide on routing, load balancing, and protocol assignment. These dynamic predictions drive adaptive decisions in the controller logic.

Dynamic SDN Management with Utility Optimization

To formalize adaptive SDN control, we define a utility function:

$$U(a) = \alpha \cdot \text{Throughput}(a) - \beta \cdot \text{Delay}(a) - \gamma \cdot \text{Energy}(a) \quad \ldots(4)$$

Here,

- $a$: Network action (e.g., route, protocol, priority)
- $\alpha, \beta, \gamma$: Weighting coefficients

The SDN controller selects the optimal action:

$$a^* = \arg\max_a U(a) \quad \ldots(5)$$

This enables the controller to make data-driven, optimal decisions based on real-time ML predictions. Integrating machine learning classifiers within an SDN-enabled IoT environment provides significant enhancements in network adaptability, protocol optimization, and traffic-aware resource management. By leveraging real-time classification and predictive analytics, the system dynamically selects appropriate communication protocols (MQTT or CoAP), optimizes traffic flow, and reduces latency and energy overhead. Mathematical modeling, as shown in the equations (1) to (5), ensures formal grounding of the architecture and guides the controller toward optimal decision-making.

Status flags serve as fundamental control elements in integrated systems, offering an effective, lightweight method for representing the real-time status or condition of different components within a large-scale or distributed architecture. These flags typically adopt binary representations and are implemented as single-bit indicators, denoting whether a particular

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

module, subsystem, or feature is currently active or inactive, functioning correctly, experiencing an error, or requiring attention.

Such flags significantly contribute to the efficiency and reliability of communication across modules, system diagnostics, coordination between interdependent processes, and user feedback mechanisms. Their usage becomes even more vital in complex systems where simultaneous operations must be synchronized, monitored, and controlled. A mathematical modeling approach allows for a precise, formal representation of how these flags behave and interact with other system elements.

Let us denote the status flag for a specific component $i$ at time $t$ as $S_i(t)$. It is defined as:

$$S_i(t) \in \{0,1\}, \quad \forall i \in \{1,2,\dots,N\} \qquad \dots(6)$$

Here:

- $S_i(t) = 1$: implies that component $i$ is active, enabled, or operating correctly at time $t$,
- $S_i(t) = 0$: signifies that component $i$ is inactive, disabled, or non-operational.

The following subsections describe the various critical functions that status flags support within an integrated system, alongside corresponding mathematical expressions to demonstrate their operational logic.

### 3. 4Functional Applications of Status Flags

1. System Monitoring

One of the primary uses of status flags is to monitor the operational state of various modules within an integrated system. By aggregating the status of individual components, the system can build a real-time health profile.

Define the system-wide status vector as:

$$\mathbf{S}(t) = [S_1(t), S_2(t), \dots, S_N(t)] \qquad \dots(7)$$

This vector effectively captures the operational footprint of all components at time $t$. To quantify overall system health, we define a health function:

$$H(t) = \sum_{i=1}^{N} S_i\,(t) \qquad \dots(8)$$

A higher value of $H(t)$ indicates more active components, suggesting a healthy and fully functional system, whereas a lower value can flag possible outages or inactive subsystems.

2. Error Detection and Handling

Status flags are critical in capturing system faults and initiating error-handling routines. Let $E_i(t)$ denote the error status of component $i$, where:

$$E_i(t) = \begin{cases} 1, & \text{if component } i \text{ encounters an error at time } t \\ 0, & \text{otherwise} \end{cases}$$

A global error function can then be defined as:

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

$$E(t) = \bigvee_{i=1}^{N} E_i(t) \quad \ldots(9)$$

If $E(t) = 1$, this implies that at least one component is experiencing an error, thereby prompting the invocation of the system's error management and logging protocols.

3. Process Synchronization

In systems with multiple interdependent tasks, maintaining order and execution flow is vital. Status flags enable synchronization by signaling task completion or readiness. Suppose task $T_j$ depends on the completion of task $T_i$. Then its execution condition can be written as:

$$T_j(t) = \begin{cases} \text{Execute}, & \text{if } S_i(t) = 1 \\ \text{Wait}, & \text{if } S_i(t) = 0 \end{cases} \quad \ldots(10)$$

This approach helps avoid race conditions and ensures the correct sequence of operations, particularly in embedded systems or real-time operating environments.

4. Inter-Module Communication

When different modules operate in tandem, there arises a need to signal readiness, data availability, or other states between them. Let $F_{i\to j}(t)$ denote the flag used by module $i$ to indicate readiness to module $j$:

$$F_{i\to j}(t) = \begin{cases} 1, & \text{if module } i \text{ is ready to send data to module } j \\ 0, & \text{otherwise} \end{cases} \quad \ldots(11)$$

This flag helps regulate communication and prevents unnecessary polling or idle waiting, optimizing system efficiency.

5. Resource Allocation and Conflict Avoidance

Shared resources, such as memory buffers, I/O buses, or network interfaces, require coordinated access. Let $R_k(t)$ represent the usage status of a shared resource $R$ by component $k$. To avoid resource contention:

$$\sum_{k=1}^{M} R_k(t) \le 1 \quad \ldots(12)$$

This condition ensures exclusive access, preventing two or more modules from using the same resource simultaneously, which could result in errors or data corruption.

6. User Interface Feedback and Visualization

Flags can also be linked to visual indicators or dashboards to communicate system state to human operators or end-users. Let $U_i(t)$ be the visual status indicator for component $i$:

$$U_i(t) = S_i(t) \quad \ldots(13)$$

Here, an active flag (1) might display a green icon, while an inactive flag (0) could trigger a warning symbol or turn red, improving situational awareness and system transparency.

7. Conditional Execution and Logic Control

Flags can be incorporated into logical control structures in software to direct program flow. Suppose a controller must decide between two routines based on system conditions:

$$D(t) = \begin{cases} \text{Routine A}, & \text{if } S_1(t) = 1 \text{ and } S_2(t) = 0 \\ \text{Routine B}, & \text{if } S_3(t) = 1 \\ \text{No Action}, & \text{otherwise} \end{cases} \quad \ldots(14)$$

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

Such logical branching provides flexibility and adaptability in system behavior under changing conditions.

8. Power Management in Embedded Systems

Power-sensitive devices benefit greatly from dynamic control of their power states using flags. Let $P_i(t)$ denote the power status of component $i$:

$$P_i(t) = \begin{cases} 1, & \text{if } S_i(t) = 1 \\ 0, & \text{otherwise} \end{cases} \quad \ldots(15)$$

Total power consumption of the system can be modeled as:

$$P_{\text{total}}(t) = \sum_{i=1}^{N} P_i\,(t) \cdot C_i \quad \ldots(16)$$

Where $C_i$ is the constant power consumption rate of component $i$ when it is active. This model facilitates the implementation of smart energy-saving strategies based on real-time system activity.

This way, status flags represent a core mechanism that supports intelligent system behavior, especially in real-time and distributed environments. They encapsulate the system state in a compact binary form, making it easy to store, transmit, and process. Through the use of mathematical models, the application and management of these flags can be optimized for performance, reliability, and scalability. Their implementation aids in monitoring component health, detecting and responding to errors, managing shared resources, synchronizing processes, and ensuring energy efficiency. As integrated systems become increasingly complex and interconnected, the structured use of status flags remains a vital tool for ensuring coherence, responsiveness, and adaptability.

## 4. Results and Analysis

### 4.1 Performance Parameters

The performance parameters for the integration of MQTT to CoAP protocol with the use of SDN can be evaluated based on several key metrics. These metrics help assess the efficiency, scalability, reliability, and overall effectiveness of the implemented system. Here are some important performance parameters to consider:

1. Throughput: Evaluate the rate at which messages can be successfully transmitted over the network. High throughput is essential to support a large number of connected IoT devices and maintain efficient data exchange.

   T = (Number of successfully delivered messages) / (Time taken for transmission) …(18)

2. Message Delivery Rate: Assess the percentage of successfully delivered messages compared to the total sent messages. A high message delivery rate indicates a reliable communication system.

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

MDR = (Number of successfully delivered messages) / (Total number of sent messages) * 100

3. Energy Efficiency: Assess the energy consumption of the IoT devices, especially important for low-power IoT devices operating on limited battery capacity. The analysis is done by considering the number of alive nodes with respect to number of execution iterations. Also, residual energy and energy consumed is analyzed with respect to number of iterations and number of devices in the network
4. End-to-End Delay: Measure the total time it takes for a message to be transmitted from a source IoT device, translated between MQTT and CoAP, and received by the destination IoT device.

EED = (t_destination_received - t_source_sent) …(19)

To evaluate these performance parameters, rigorous testing and monitoring of the integrated system are required. Simulated and real-world scenarios can be used to validate the system's performance under various conditions and loads. Additionally, performance tuning and optimization might be necessary to enhance the system's efficiency and address any bottlenecks identified during testing. Table 1 shows the parameter configuration used in the experimentation in Mininet, which is a popular open-source network emulator that enables the creation of a virtual network on a single machine. Mininet provides a lightweight and efficient platform for developing, testing, and experimenting with SDN and network function virtualization (NFV) applications. The configured network in Mininet is shown in Figure 4.

**Table 1: Configuration of Parameters in Experimental Setup**

| Category | Parameter | Example Value(s) |
|---|---|---|
| Communication Protocols | MQTT Broker Address | mqtt://broker.example.com |
| | MQTT Port | 1883 |
| | CoAP Port | 5683 |
| Device Characteristics | Device ID | Device001, Device002 |
| | Device Type | Sensor, Actuator |
| | Sensor Sampling Rate | 1 sample/sec |
| Network Topology | Node Placement | Random, Grid, Hierarchical |
| | Connectivity Model | Mesh, Star, Tree |
| SDN Controller Settings | Controller IP | 192.168.1.100 |
| | Controller Port | 6653 |
| | Flow Table Entries | Priority, Match Criteria, Action |
| ML Model Configuration | Training Data | IoT_Dataset.csv |
| | SVM Kernel Type | Radial Basis Function (RBF) |

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

| | | |
|---|---|---|
| | Decision Tree Depth | 10 |
| Security Parameters | Authentication Key | XYZSecretKey |
| | Encryption Algorithm | AES256 |
| Experimental Metrics | Performance Metrics | Latency, Throughput, Packet Loss |
| | Data Logging Format | CSV, JSON |
| Service Parameters | Service Activation | Enabled/Disabled |
| | Service-Specific Settings | Configuration parameters for specific apps |
| Simulation Parameters | Simulation Duration | 3600 seconds (1 hour) |
| | Time Step | 1 second |
| QoS Settings | Prioritization Parameters | High-Priority Traffic, QoS Level 2 |
| Energy Consumption | Power Management Settings | Sleep Mode, Wake-up Interval |
| | Battery Levels | 20%, 50%, 80% |
| SDN Controller Settings | Controller Type | OpenDaylight |
| | Controller IP | 192.168.1.100 |
| | Controller Port | 6653 |
| | Controller API Version | OpenFlow 1.3, RESTCONF 1.0 |
| | Controller Security | SSL/TLS Enabled, Username/Password |
| | Controller Topology Discovery | LLDP, BDDP |
| | Controller Flow Programming | Reactive, Proactive |
| | Controller Load Balancing | Round Robin, Weighted |
| | Controller Fault Tolerance | Active-Standby, Clustered |

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

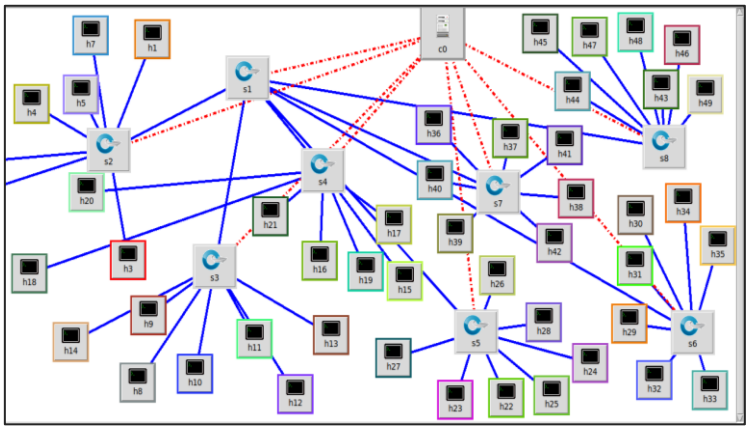
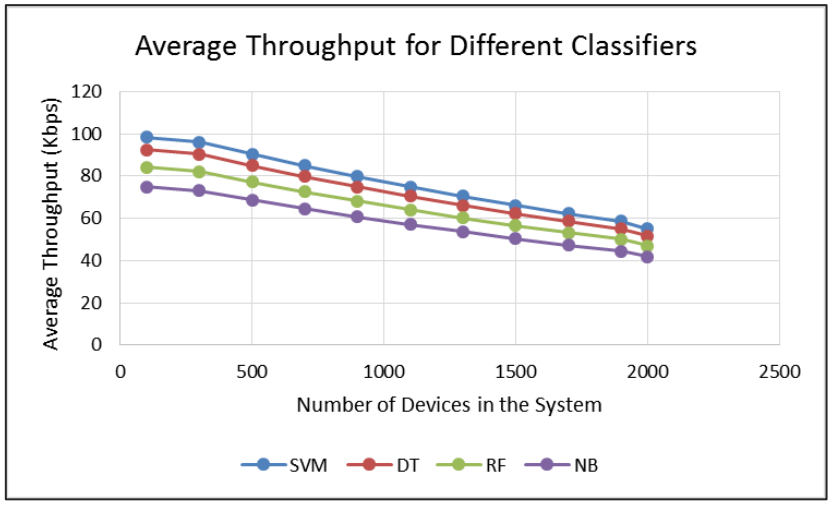
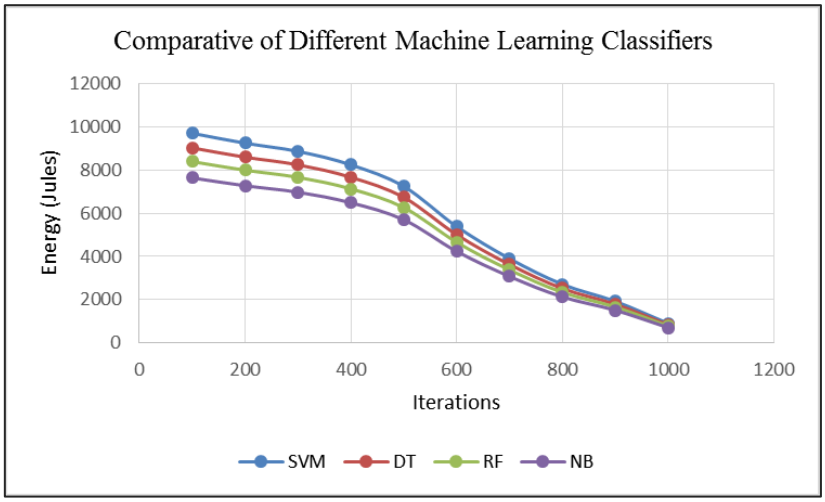


Figure 4: Mininet based configured IoT System

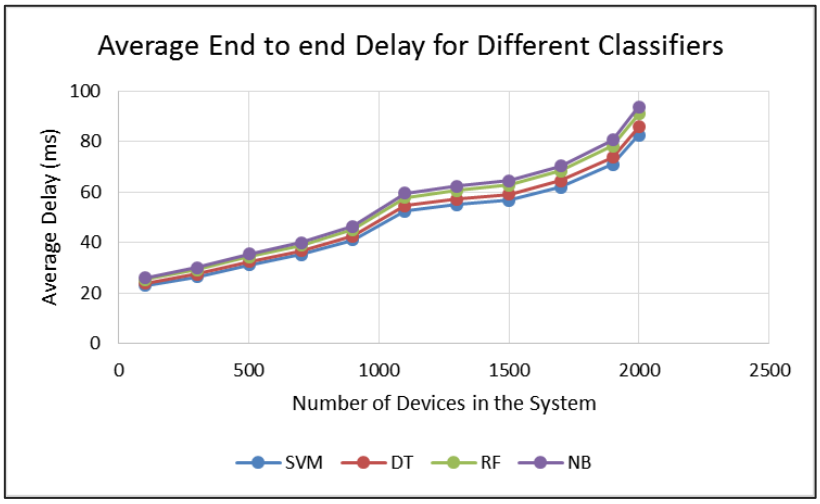



(a) Throughput analysis

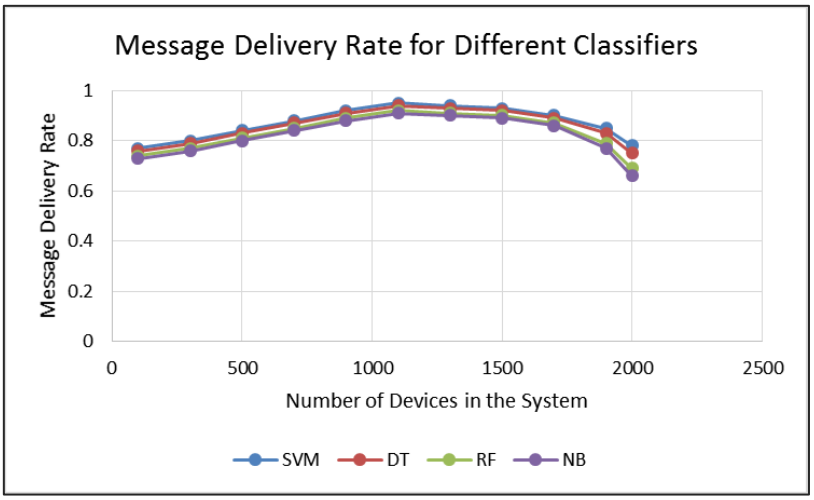



(b) Energy Analysis

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025




(c) Average end to end delay analysis




(d) Message delivery rate analysis

**Figure 5: Performance Characteristics of the network for different classifiers**

### 4.2 Discussion

The superiority of Support Vector Machine (SVM)-based classifiers over Decision Trees (DT), Naive Bayes (NB), and Random Forest (RF) classifiers in deciding flag status can be attributed to their effectiveness in handling crucial network performance parameters. When considering parameters such as throughput of the network, residual energy, end-to-end delay, and message delivery rate, SVMs demonstrate distinct advantages as shown in Figure 5. SVMs are well-known for their ability to handle complex, non-linear relationships in the data, making them particularly effective in scenarios where throughput is a critical metric. Their capacity to create optimal decision boundaries contributes to higher throughput by efficiently classifying data points. Moreover, SVMs are adept at generalization, reducing the risk of overfitting, which

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

is especially crucial for maintaining a balance between high throughput and sustainable residual energy levels.

In the context of residual energy, SVMs showcase resilience to outliers and the ability to handle high-dimensional data. This resilience is essential for ensuring sustainable energy consumption in networked devices. The adaptability of SVMs to non-linear patterns contributes to efficient energy usage, preventing unnecessary depletion of battery resources. Additionally, the robustness of SVMs to outliers and noise in the data makes them well-suited for optimizing message delivery rates, ensuring reliable communication within the network.

SVMs' prowess in mitigating over fitting also aligns with the goal of minimizing end-to-end delay. By creating decision boundaries that generalize well to unseen data, SVMs contribute to lower end-to-end delays, enhancing the responsiveness of the network. Decision Trees, Naive Bayes, and Random Forests may struggle to achieve similar levels of responsiveness, particularly in scenarios with varying network conditions and dynamic data patterns.

In conclusion, the decision by SVM-based classifiers to better determine flag status can be attributed to their ability to optimize network parameters. Their strength in handling non-linear relationships, adapting to high-dimensional data, and minimizing over fitting aligns with the requirements of efficient throughput, sustainable residual energy, low end-to-end delay, and reliable message delivery. This makes SVMs a favorable choice for applications where network performance is a critical determinant of overall system effectiveness.

## 5. Conclusion

In this groundbreaking study, we seamlessly integrated MQTT and CoAP protocols within a SDN framework, enhancing interoperability and performance in IoT networks. Our objective was to address the escalating demand for seamless communication in heterogeneous IoT environments. The integration of MQTT and CoAP through the SDN controller signifies a significant advancement in fostering a cohesive and efficient IoT ecosystem. SDN's centralized control and programmability enable dynamic management of communication protocols and network resources, offering flexibility crucial for accommodating diverse IoT devices with varying communication requirements. The orchestrated synergy between MQTT and CoAP, guided by the SDN controller, streamlines communication pathways, elevating the network's overall efficiency. A pivotal innovation in our integrated system is the introduction of an SVM-based flag status indicator. Leveraging SVM's prowess in handling non-linear relationships and its resistance to overfitting, this indicator proved instrumental in accurately determining flag status. The satisfactory results obtained with SVM, outperforming classifiers like Decision Trees, Naive Bayes, and Random Forests, underscore its significance in optimizing the flag status indicator's performancea critical factor for applications relying on reliable flag status determination. Our experimental results confirm the success of our integrated approach, showcasing superior throughput, sustainable residual energy levels, reduced end-to-end delays,

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

and a higher message delivery rate compared to alternative classifiers. These outcomes affirm the efficacy of our methodology in achieving seamless interoperability, efficient communication, and robust decision-making across diverse IoT scenarios. In conclusion, our work stands as a notable stride in integrating MQTT and CoAP within an SDN framework, complemented by an SVM-based flag status indicator. The superior outcomes and performance over alternative classifiers validate the potential of our approach in shaping the future of IoT networks, emphasizing interoperability, efficiency, and reliable decision-making. This integrated system lays a robust foundation for further research and practical applications in the evolving landscape of IoT technologies.

**References:**

[1] B. Nagajayanthi, "Decades of Internet of Things Towards Twenty-first Century: A Research-Based Introspective," *Wirel. Pers. Commun.*, vol. 123, no. 4, pp. 3661–3697, Apr. 2022, doi: 10.1007/S11277-021-09308-Z/FIGURES/8.

[2] C. Kanzouai, S. Bouarourou, A. Zannou, A. Boulaalam, and E. H. Nfaoui, "Enhancing IoT Scalability and Interoperability Through Ontology Alignment and FedProx," *Futur. Internet 2025, Vol. 17, Page 140*, vol. 17, no. 4, p. 140, Mar. 2025, doi: 10.3390/FI17040140.

[3] V. Choudhary, P. Guha, G. Pau, and S. Mishra, "An overview of smart agriculture using internet of things (IoT) and web services," *Environ. Sustain. Indic.*, vol. 26, p. 100607, Jun. 2025, doi: 10.1016/J.INDIC.2025.100607.

[4] L. Babun, K. Denney, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "A survey on IoT platforms: Communication, security, and privacy perspectives," *Comput. Networks*, vol. 192, p. 108040, Jun. 2021, doi: 10.1016/J.COMNET.2021.108040.

[5] R. Almutairi, G. Bergami, and G. Morgan, "Advancements and Challenges in IoT Simulators: A Comprehensive Review," *Sensors 2024, Vol. 24, Page 1511*, vol. 24, no. 5, p. 1511, Feb. 2024, doi: 10.3390/S24051511.

[6] C. Zanasi, S. Russo, and M. Colajanni, "Flexible zero trust architecture for the cybersecurity of industrial IoT infrastructures," *Ad Hoc Networks*, vol. 156, p. 103414, Apr. 2024, doi: 10.1016/J.ADHOC.2024.103414.

[7] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017, doi: 10.1109/JIOT.2017.2683200.

[8] Y. Bin Zikria, H. Yu, M. K. Afzal, M. H. Rehmani, and O. Hahm, "Internet of Things (IoT): Operating System, Applications and Protocols Design, and Validation Techniques," *Futur. Gener. Comput. Syst.*, vol. 88, pp. 699–706, Nov. 2018, doi: 10.1016/J.FUTURE.2018.07.058.

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

[9] M. R. Palattella *et al.*, "Standardized protocol stack for the internet of (important) things," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 3, pp. 1389–1406, 2013, doi: 10.1109/SURV.2012.111412.00158.

[10] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *J. Electr. Comput. Eng.*, vol. 2017, 2017, doi: 10.1155/2017/9324035.

[11] T. Salman and R. Jain, "A Survey of Protocols and Standards for Internet of Things," *Adv. Comput. Commun.*, Feb. 2019, doi: 10.34048/2017.1.f3.

[12] M. Collina, M. Bartolucci, A. Vanelli-Coralli, and G. E. Corazza, "Internet of Things application layer protocol analysis over error and delay prone links," *2014 7th Adv. Satell. Multimed. Syst. Conf. 13th Signal Process. Sp. Commun. Work. ASMS/SPSC 2014*, vol. 2014-January, pp. 398–404, Oct. 2014, doi: 10.1109/ASMS-SPSC.2014.6934573.

[13] A. Aijaz and A. H. Aghvami, "Cognitive machine-to-machine communications for internet-of-things: A protocol stack perspective," *IEEE Internet Things J.*, vol. 2, no. 2, pp. 103–112, Apr. 2015, doi: 10.1109/JIOT.2015.2390775.

[14] J. Granjal, E. Monteiro, and J. Sa Silva, "Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 3, pp. 1294–1312, Jul. 2015, doi: 10.1109/COMST.2015.2388550.

[15] M. B. Yassein, M. Q. Shatnawi, and D. Al-Zoubi, "Application layer protocols for the Internet of Things: A survey," *Proc. - 2016 Int. Conf. Eng. MIS, ICEMIS 2016*, Nov. 2016, doi: 10.1109/ICEMIS.2016.7745303.

[16] S. Mijovic, E. Shehu, and C. Buratti, "Comparing application layer protocols for the Internet of Things via experimentation," *2016 IEEE 2nd Int. Forum Res. Technol. Soc. Ind. Leveraging a Better Tomorrow, RTSI 2016*, Nov. 2016, doi: 10.1109/RTSI.2016.7740559.

[17] S. Saritha and V. Sarasvathi, "A study on application layer protocols used in IoT," pp. 155–159, Apr. 2019, doi: 10.1109/CCUBE.2017.8394143.

[18] V. M. Tayur and R. Suchithra, "Review of interoperability approaches in application layer of Internet of Things," *IEEE Int. Conf. Innov. Mech. Ind. Appl. ICIMIA 2017 - Proc.*, pp. 322–326, Jul. 2017, doi: 10.1109/ICIMIA.2017.7975628.

[19] B. Safaei, A. M. H. Monazzah, M. B. Bafroei, and A. Ejlali, "Reliability side-effects in Internet of Things application layer protocols," *2017 2nd Int. Conf. Syst. Reliab. Safety, ICSRS 2017*, vol. 2018-January, pp. 207–212, Jul. 2017, doi: 10.1109/ICSRS.2017.8272822.

[20] U. Tandale, B. Momin, and D. P. Seetharam, "An empirical study of application layer protocols for IoT," *2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput. ICECDS 2017*, pp. 2447–2451, Jun. 2018, doi: 10.1109/ICECDS.2017.8389890.

[21] M. Pohl, J. Kubela, S. Bosse, and K. Turowski, "Performance evaluation of application

Power System Technology
ISSN:1000-3673
Received: 06-02-2025 Revised: 15-03-2025 Accepted: 10-04-2025

layer protocols for the internet-of-things," *Proc. - 2018 6th Int. Conf. Enterp. Syst. ES 2018*, pp. 180–187, Dec. 2018, doi: 10.1109/ES.2018.00035.

[22] C. Bayılmış, M. A. Ebleme, Ü. Çavuşoğlu, K. Küçük, and A. Sevin, "A survey on communication protocols and performance evaluations for Internet of Things," *Digit. Commun. Networks*, vol. 8, no. 6, pp. 1094–1104, Dec. 2022, doi: 10.1016/J.DCAN.2022.03.013.

[23] M. Sandell and U. Raza, "Application layer coding for IoT: Benefits, limitations, and implementation aspects," *IEEE Syst. J.*, vol. 13, no. 1, pp. 554–561, Mar. 2019, doi: 10.1109/JSYST.2018.2791659.

[24] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An application protocol for billions of tiny internet nodes," *IEEE Internet Comput.*, vol. 16, no. 2, pp. 62–67, Mar. 2012, doi: 10.1109/MIC.2012.29.

[25] P. K. Donta, T. Amgoth, and C. S. Rao Annavarapu, "Congestion-aware data acquisition with Q-learning for wireless sensor networks," *IEMTRONICS 2020 - Int. IOT, Electron. Mechatronics Conf. Proc.*, Sep. 2020, doi: 10.1109/IEMTRONICS51293.2020.9216379.

[26] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoAP congestion control for the internet of things," *IEEE Commun. Mag.*, vol. 54, no. 7, pp. 154–160, Jul. 2016, doi: 10.1109/MCOM.2016.7509394.

[27] A. Betzler, C. Gomez, I. Demirkol, and J. Paradells, "CoCoA+: An advanced congestion control mechanism for CoAP," *Ad Hoc Networks*, vol. 33, pp. 126–139, Oct. 2015, doi: 10.1016/J.ADHOC.2015.04.007.

[28] C. Suwannapong and C. Khunboa, "Congestion Control in CoAP Observe Group Communication," *Sensors 2019, Vol. 19, Page 3433*, vol. 19, no. 15, p. 3433, Aug. 2019, doi: 10.3390/S19153433.

[29] G. A. Akpakwu, G. P. Hancke, and A. M. Abu-Mahfouz, "CACC: Context-aware congestion control approach for lightweight CoAP/UDP-based Internet of Things traffic," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, p. e3822, Feb. 2020, doi: 10.1002/ETT.3822.

[30] S. Bolettieri, G. Tanganelli, C. Vallati, and E. Mingozzi, "pCoCoA: A precise congestion control algorithm for CoAP," *Ad Hoc Networks*, vol. 80, pp. 116–129, Nov. 2018, doi: 10.1016/J.ADHOC.2018.06.015.

[31] V. Rathod, N. Jeppu, S. Sastry, S. Singala, and M. P. Tahiliani, "CoCoA++: Delay gradient based congestion control for Internet of Things," *Futur. Gener. Comput. Syst.*, vol. 100, pp. 1053–1072, Nov. 2019, doi: 10.1016/J.FUTURE.2019.04.054.