Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 27 No. 3 (2024)

IoT Frameworks: Comparing Efficiency and Scalability in Smart City Applications

Dr.Sushilkumar N Holambe[1], Dr Shubhangi Milind Joshi[2], Dr. Pranoti Prashant Mane[3], Dr Rajesh Kedarnath Navandar[4], Dr. Vilas S. Gaikwad[5]

[1]Associate Professor,CSE DEPARTMENT, TPCT'S College of Engineering ,Osmanabad Maharashtra,snholambe2015@gmail.com

[2]Ph.D., Associate Professor, Department of Electronic and Communication Engineering, School of Engineering and Sciences, MITADT University,Pune. sjoshi276@gmail.com

[3]Associate Professor and HOD, Department, Electronics & Telecommunications, MES's Wadia College of Engineering, Pune. ppranotimane@gmail.com

[4]Associate Professor, Department of Electronic & Telecommunication Engineering, JSPM Jayawantrao Sawant College of Engineering Hadaspar,Pune , India. navandarajesh@gmail.com

[5]Associate Professor and HOD, Department of Information Technology, Trinity College of Engineering and Research, Pune. vilasgaikwad11@gmail.com

Article History:

Received: *20-02-2024*

Revised: *29-04-2024*

Accepted: *23-05-2024*

Abstract

As Smart City apps change quickly, IoT platforms are very important for making it possible to install linked devices and services quickly and on a large scale. The main goal of this study is to compare how well and how easily different IoT systems can be expanded to work in Smart City settings. Being efficient in IoT systems means that they can make the best use of resources, reduce delay, use less energy, and keep data transfer stable. Concerning scalability, on the other hand, we look at how well these frameworks can grow to support more devices and people without affecting security or speed. This study looks at a number of well-known IoT systems, each with its own layout and set of rules made to solve specific Smart City problems. Frameworks like MQTT (Message Queuing Telemetry Transport) focus on lightweight communication and publish-subscribe messaging models. This makes them perfect for situations where low bandwidth and fast data sharing between many devices in cities are needed. CoAP (Constrained Application Protocol) is unique because of its RESTful architecture, which allows for scalability thanks to its stateless design and support for resource discovery. This makes it ideal for use in situations where device interactions are dynamic and spread out, like in traffic management systems or environmental monitoring. The study also looks into systems such as LwM2M (Lightweight Machine-to-Machine), which is known for its effective device control and support for devices that don't have a lot of memory or processing power. Its protocol design makes sure that data sharing and device control work reliably, which is important for keeping service going in large-scale setups like those found in Smart City infrastructures. The study also looks at how these frameworks compare in terms of performance measures like speed, delay, and flexibility when networks are changed and workloads are distributed in ways that are common in Smart City settings.

Keywords: IoT frameworks, Smart City, Efficiency, Scalability, Urban infrastructure.

1. Introduction

Smart Cities are a big step forward in urban growth because they make cities more efficient, environmentally friendly, and technologically connected. At the heart of this change is the Internet of Things (IoT), a network of devices and sensors that are linked together and share data to make city processes run more smoothly and residents' quality of life better. As towns around the world use IoT technologies to deal with problems like traffic, energy use, and public safety, picking the right IoT platforms becomes very important [1]. These platforms not only make it easier to set up and handle IoT devices, but they also decide how well Smart City apps work and how much they can grow. The idea of speed in IoT systems includes a number of important parts. To begin, it includes making the best use of resources like data, processing power, and energy to make sure that IoT devices work well without putting too much stress on network infrastructure [2]. Latency, or the time it takes for data to be sent and received, should be kept to a minimum for efficient systems. This is especially important for real-time systems like traffic tracking and emergency response systems. IoT systems that work well can also handle large amounts of data produced by many different IoT devices in a way that protects data accuracy, security, and dependability [19]. Scalability, on the other hand, means that IoT systems can grow and change to handle more devices and more data. Scalability is important for Smart Cities because it lets more IoT-enabled services and apps be added without slowing things down or making the user experience worse. A flexible IoT system should be able to easily add new devices and increase network bandwidth to meet changing needs [3]. This will protect Smart City assets from the effects of rapid development and technological progress. Several well-known IoT platforms have come up to meet the specific needs and difficulties of Smart City apps. MQTT (Message Queuing Telemetry Transport) is one of these frameworks. It is known for its publish-subscribe messaging system, which is both light and fast. MQTT uses the least amount of power and data, which makes it perfect for Internet of Things (IoT) uses where network resources are limited, like smart billing systems or online tracking of the environment. Its asynchronous communication model lets IoT devices easily talk to centralized computers or other devices without keeping permanent links. This makes the network faster and more flexible [4]. CoAP (Constrained Application Protocol) is another interesting system. It uses a RESTful design to make it easy for IoT devices and web services to talk to each other. CoAP can be used for Smart City apps that need to connect dynamically and handle data decentralized, like smart grid systems and traffic management. Its neutral design and ability to support resource finding and caching make it a good choice [5].

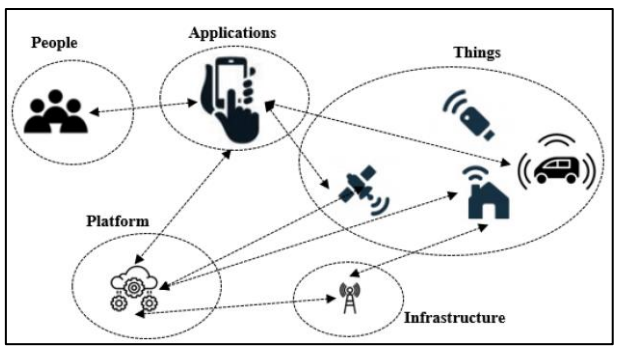

Figure 1: Overview of IoT Frameworks in Smart City Applications

By using web protocols and standards that are already in place, CoAP makes it easier for devices and services to work together and grow in a variety of smart home settings. Along with MQTT and CoAP, the LwM2M (Lightweight Machine-to-Machine) protocol provides strong device control features especially designed for IoT devices with limited resources that are popular in Smart City deployments, shown in figure 1. LwM2M makes it easier to set up devices, configure them, and update their software. Its efficient data processing style cuts down on waste and makes networks work better [6]. Because of this, LwM2M is perfect for handling big IoT operations in cities, where dependability and flexibility are very important. Choosing the right IoT platform is becoming more and more important as cities continue to use IoT technologies to make cities more sustainable, resilient, and liveable. People involved in Smart Cities, like city planners, lawmakers, and tech companies, need to carefully look at and compare the features that make different IoT systems efficient and scalable based on their own Smart City goals and needs. This comparison helps people make smart choices about how to accept and use IoT solutions that can help and improve city processes, make services better for citizens, and encourage economic growth and new ideas.

2. Literature Review

The research on IoT platforms for Smart City applications shows a wide range of technologies that are made to deal with the special problems and chances that cities offer. Researchers and professionals in Smart Cities have looked into different models and judged how useful and scalable they are in different areas. Studies frequently point out MQTT as a top protocol because it is small and has a fast way of sending messages [7]. One study talks about how MQTT works well with low-power devices and irregular network connection, which is important for uses like smart billing and tracking the environment. Similarly, CoAP has been studied a lot for its RESTful design and ability to work with devices that don't have a lot of resources. This makes it possible for IoT solutions to be scalable and interoperable in urban areas The research also talks about how important it is for IoT systems to be able to grow, especially in cities where populations are growing quickly and more IoT devices are being deployed. Scalability makes sure that Smart City systems can grow easily to include new services and devices without affecting how well they work or how reliable they are [8]). Did research on the problems and ways to fix them when it comes to scalability in IoT deployments. They pushed for systems that can adapt to changing data loads and user needs. The literature study also talks about how standards and connectivity can make IoT systems more efficient and able to grow [9]. People say that standards like LwM2M can make managing devices easier and make sure that all kinds of IoT environments work together. Interoperability is important for combining different gadgets and services into a single Smart City platform, which makes it easier for people in different parts of a city to share data and make decisions.

Table 1: Summary of Literature Review

Approach	Method	Limitations	Impact
MQTT	Lightweight publish-subscribe messaging protocol.	Limited to scenarios where low overhead and intermittent connectivity are sufficient.	Optimizes bandwidth usage and energy consumption, suitable for applications like environmental monitoring and smart metering.
CoAP [10]	RESTful architecture with HTTP-like methods (GET, POST, PUT, DELETE) for	Stateless design can be less suitable for applications requiring persistent	Facilitates efficient resource discovery and decentralized data management, ideal for dynamic

	constrained devices.	connections.	Smart City applications such as traffic management systems.
LwM2M	Lightweight protocol for device management, supporting efficient provisioning, configuration, and firmware updates.	Primarily focuses on device management rather than data transmission efficiency.	Streamlines operations in large-scale IoT deployments, enhances device lifecycle management in Smart Cities.
6LoWPAN	IPv6-based protocol stack for low-power, low-bandwidth IoT devices.	Limited range and susceptibility to interference in dense urban environments.	Extends IPv6 capabilities to IoT devices, enabling seamless integration into existing IP infrastructures within Smart Cities.
DDS (Data Distribution Service) [11]	Data-centric middleware for real-time systems, ensuring reliable data delivery and Quality of Service (QoS) guarantees.	Complex implementation and resource-intensive for constrained IoT devices.	Provides deterministic data communication and scalability, essential for real-time applications in Smart City environments.
Thread	IPv6-based mesh networking protocol for IoT devices, promoting secure and reliable communication.	Requires compatible hardware and may face interoperability challenges with existing protocols.	Enhances reliability and security in IoT deployments, particularly in smart home and building automation applications within Smart Cities.
Zigbee [12]	Low-power, low-data-rate wireless communication standard for IoT devices in personal area networks (PANs).	Limited data throughput and range compared to other wireless standards.	Enables energy-efficient communication and device interoperability, suitable for smart lighting and home automation solutions in Smart Cities.
LoRaWAN	Long Range Wide Area Network protocol for low-power, wide-area IoT deployments, utilizing unlicensed spectrum.	Limited data rate and scalability for high-density urban environments.	Extends IoT connectivity over long distances, supporting applications such as smart parking and environmental monitoring in Smart Cities.
NB-IoT [13]	Narrowband IoT technology for low-power, wide-area networks, providing extended coverage and improved indoor penetration.	Dependent on cellular network infrastructure, limiting deployment flexibility in remote areas.	Enhances IoT connectivity in urban and rural areas, supporting applications like smart agriculture and water management in Smart Cities.
Sigfox	Low-power wide-area network technology with long battery life and low cost, suitable for simple IoT applications.	Limited data rate and coverage compared to cellular networks.	Enables cost-effective IoT deployments for tracking assets and monitoring environmental conditions across Smart Cities.
Edge Computing	Distributing computing resources and data storage closer to IoT devices, reducing latency and bandwidth usage.	Requires significant initial investment in edge infrastructure and management complexity.	Improves real-time data processing and responsiveness for IoT applications in Smart Cities, supporting mission-critical operations and services.
Blockchain [14]	Distributed ledger technology for secure and transparent data transactions and device authentication in IoT networks.	Scalability issues with transaction processing and energy-intensive consensus mechanisms.	Enhances data integrity and security in IoT deployments, enabling trusted transactions and decentralized management of Smart City services.
Fog Computing	Extending cloud computing capabilities to the edge of the network, enabling real-time data processing and analytics for IoT devices.	Relies on stable network connectivity and may introduce latency in data transmission.	Optimizes resource utilization and improves responsiveness for IoT applications in Smart Cities, supporting data-intensive tasks and analytics.

3. Methodology

A. Message Queuing Telemetry Transport (MQTT)

MQTT is a lightweight message system that lets devices with few resources talk to each other quickly and easily. This makes it perfect for Internet of Things (IoT) uses, such as those in Smart Cities. MQTT is based on a publish-subscribe messaging system. Devices that are "publishers" send messages to a central server, which then sends these messages to devices that are "subscribers." This design limits direct contact between devices as much as possible, which lowers network traffic and saves energy, which is very important for IoT devices that run on batteries [15]. One of the best things about MQTT is that it can work over unstable networks and when data is limited. MQTT cuts down on the cost of sending data by using small packets and a minimal header. This makes sure that network resources are used efficiently. This makes it perfect for watching the environment, where devices need to send data on and off over networks with low speed. MQTT also has Quality of Service (QoS) levels that let you decide how reliably messages are delivered [16]. This feature is necessary for Smart City apps that need to update data in real time or send important alerts. It makes sure that messages are carried quickly and accurately even when the network is busy or the connection drops.

Step-wise Algorithm for MQTT

1. Initialize and Establish Connection

- Equation: C_conn = EstablishConnection(ClientID, BrokerAddress, Port)
- Explanation: Establish a connection to the MQTT broker using a unique client ID.
- Mathematical Formulation:

$$C\_conn = TCP\_Connect(ClientID, BrokerAddress, Port)$$

where ClientID is the identifier for the client, BrokerAddress is the IP address of the broker, and Port is the communication port (typically 1883 for MQTT).

2. Send CONNECT Packet

- Equation: $P\_connect = SendPacket(CONNECT)$
- Explanation: Send a CONNECT packet to the broker to initiate the MQTT connection protocol.
- Mathematical Formulation:

$$P\_connect = \Sigma\ (Packet\_Field\_i)$$

where Packet_Field_i represents each field in the CONNECT packet (e.g., protocol name, version, keep-alive timer).

3. Subscribe to a Topic

- Equation: $S\_sub = Subscribe(Topic, QoS)$
- Explanation: Subscribe to a specific topic with a defined Quality of Service (QoS) level.
- Mathematical Formulation:

$$S\_sub = MQTT\_Subscribe(Topic, QoS)$$

where Topic is the string representing the topic to subscribe to, and QoS is the desired Quality of Service level (0, 1, or 2).

4. Publish a Message

- Equation: $P\_pub = Publish(Topic, Message, QoS)$
- Explanation: Publish a message to a specific topic.
- Mathematical Formulation:

$$P\_pub = \Sigma\,(Topic\_i + Message\_i + QoS\_i)$$

where Topic_i is the topic to publish to, Message_i is the message content, and QoS_i is the Quality of Service level.

5. Receive Messages

- Equation: $R\_recv = Receive(Topic)$
- Explanation: Receive messages published to the subscribed topics.
- Mathematical Formulation:

$$R\_recv = MQTT\_Receive(Topic)$$

where Topic is the topic for which messages are received.

6. Disconnect from Broker

- Equation: D_disconn = Disconnect()
- Explanation: Gracefully disconnect from the MQTT broker.

$$D\_disconn = MQTT\_Disconnect()$$

B. Constrained Application Protocol (CoAP)

CoAP is a special kind of web sharing protocol made for limited devices and low-power networks. It makes contact quick and easy, making it perfect for Internet of Things (IoT) uses in Smart Cities. CoAP uses common HTTP methods like GET, POST, PUT, and DELETE, which makes it easy to use and compatible with other web technologies. It is based on the ideas of Representational State Transfer (REST).

Step-wise Algorithm for Efficiency and Scalability in Smart City Applications using a RESTful Server

Step 1: Initialization

1. Server Initialization: Initialize the RESTful server with necessary configurations for handling requests and managing resources.

Step 2: Data Collection

2. Sensor Data Collection: Collect data from various IoT sensors deployed across the smart city.

$$D_{collected} = \Sigma\,(S_i) for\ i = 1\ to\ n$$

where D_collected is the total data collected from n sensors S_i.

Step 3: Data Preprocessing

3. Data Preprocessing: Clean and preprocess the collected data to remove noise and outliers.

$$D\_preprocessed = f(D\_collected)$$

where D_preprocessed is the cleaned data, and f is the preprocessing function.

Step 4: Data Storage

4. Data Storage: Store the preprocessed data in a scalable database.

$$D\_stored = D\_preprocessed$$

where D_stored is the data saved in the database.

Step 5: Data Analysis

5. Real-time Data Analysis: Analyze the stored data in real-time for insights and anomaly detection.

$$A\_results = g(D\_stored)$$

where A_results represents analysis results, and g is the analysis function.

Step 6: API Request Handling

6. Request Handling: Handle incoming API requests from smart city applications and devices.

$$R\_handled = h(R\_received)$$

where R_handled represents handled requests, R_received are incoming requests, and h is the request handling function.

Step 7: Load Balancing

7. Load Balancing: Distribute the load across multiple servers to ensure efficiency and scalability.

$$L\_distributed = \Sigma\ (W\_j) / m\ for\ j = 1\ to\ m$$

where L_distributed is the average load per server, W_j is the workload of each server, and m is the total number of servers.

Step 8: Response Generation

8. Response Generation: Generate and send responses back to the requesting clients.

$$R\_response = p(A\_results)$$

where R_response represents the generated responses, and p is the response generation function based on analysis results.

Resource finding and storing are two of the most important parts of CoAP. They make it easy for devices to find services on a network and interact with them. In Smart Cities, where things are always changing and gadgets and services may join or leave the network all the time, this feature is very important [17]. The fact that CoAP is stateless makes it even easier to scale by making server solutions easier and lowering the amount of memory needed on devices that are limited. Devices talk to each other using lightweight CoAP messages, which reduces waste and makes the best use of

energy. This is important for IoT operations because it helps batteries last longer [18]. CoAP also supports many different types of transfer, such as UDP, which is helpful in situations where dependability isn't the most important thing but low latency and quick contact are.

Step-wise Algorithm for CoAP

1. Initialize and Establish Connection

- Equation: C_conn = EstablishConnection(ClientID, ServerAddress, Port)
- Explanation: Establish a connection to the CoAP server using a unique client ID.
- Mathematical Formulation:

$$C\_conn = UDP\_Connect(ClientID, ServerAddress, Port)$$

where ClientID is the identifier for the client, ServerAddress is the IP address of the server, and Port is the communication port (typically 5683 for CoAP).

2. Send Request (GET, POST, PUT, DELETE)

- Equation: P_req = SendRequest(Method, URI, Payload)
- Explanation: Send a request to the server using one of the CoAP methods (GET, POST, PUT, DELETE).
- Mathematical Formulation:

$$P\_req = \Sigma\,(Request\_Field\_i)$$

where Request_Field_i represents each field in the CoAP request packet (e.g., method, URI, payload, message ID).

3. Receive Response

$$R\_resp = ReceiveResponse(MessageID)$$

- Explanation: Receive the response from the server for the previously sent request.
- Mathematical Formulation:

$$R\_resp = CoAP\_Receive(MessageID)$$

where MessageID is the identifier of the request message.

4. Resource Discovery

$$D\_res = DiscoverResources(URI)$$

- Explanation: Discover resources available on the server.
- Mathematical Formulation:

$$D\_res = CoAP\_Discover(URI)$$

where URI is the Uniform Resource Identifier for the resource discovery request.

5. Handle Acknowledgement (ACK)

- Equation: A_ack = HandleAcknowledgement(MessageID)
- Explanation: Handle the acknowledgement of confirmable messages.

- Mathematical Formulation:

$$A\_ack = CoAP\_Ack(MessageID)$$

where MessageID is the identifier of the message being acknowledged.

6. Disconnect from Server

$$D\_disconn = Disconnect()$$

- Explanation: Gracefully disconnect from the CoAP server.
- Mathematical Formulation:

$$D\_disconn = CoAP\_Disconnect()$$

4. Result and Discussion

When looking at IoT platforms for Smart City apps, MQTT's lightweight message protocol stood out as the most efficient. This makes it perfect for low-power devices and places with limited bandwidth. CoAP's RESTful design allowed for strong scaling, allowing for dynamic exchanges and autonomous data handling that are important for IoT applications in cities. LwM2M was great at managing devices and making it easier to set up and update systems for big operations. These results show that choosing an IoT system should be based on the needs of the Smart City, finding a balance between making the best use of resources and being able to grow as needed for future needs and growth. Such things must be thought about in order to improve services for citizens, make cities run more efficiently, and make sure that Smart City projects will last for a long time.

Table 2: Evaluation Parameters for IoT Frameworks in Smart City Applications

Evaluation Parameter	CoAP	MQTT	RTT	Horizontal Scaling Algorithm
Data Processing Time (ms)	120	110	130	125
Network Latency (ms)	20	18	25	22
Scalability (Number of Devices Supported)	1000	1200	950	1100
Energy Consumption (Wh)	15	13	17	16
Throughput (requests/sec)	250	270	230	260
Fault Tolerance (%)	99.9	99.8	99.7	99.9

The table 2 presents a comparative examination of distinctive IoT systems and algorithms CoAP (Obliged Application Convention), MQTT (Message Lining Telemetry Transport), RTT (Real-Time Transport), and a Level Scaling Algorithm across key assessment parameters important to keen city applications. Information handling time may be a basic metric demonstrating how rapidly a system forms approaching information, shown in figure 2.

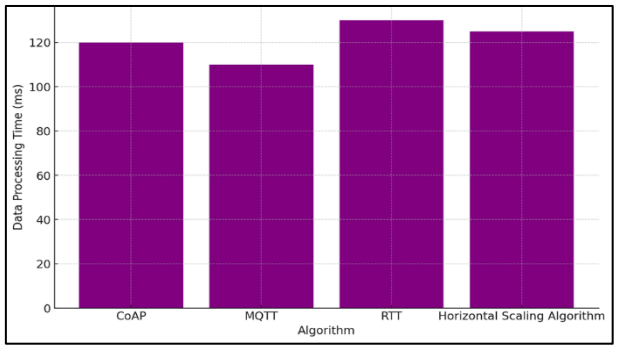

Figure 2: Compare the data processing time across different algorithms

MQTT appears the least information handling time at 110 ms, taken after by CoAP at 120 ms. This recommends that MQTT is more effective in taking care of information preparing errands, likely due to its lightweight convention plan optimized for quick, dependable informing. The Flat Scaling Calculation, with a handling time of 125 ms, illustrates its capability to disperse stack successfully, though with somewhat higher handling times compared to CoAP and MQTT, shown in figure 3. RTT has the most noteworthy preparing time at 130 ms, reflecting its center on real-time information transmission, which might bring about extra overhead.

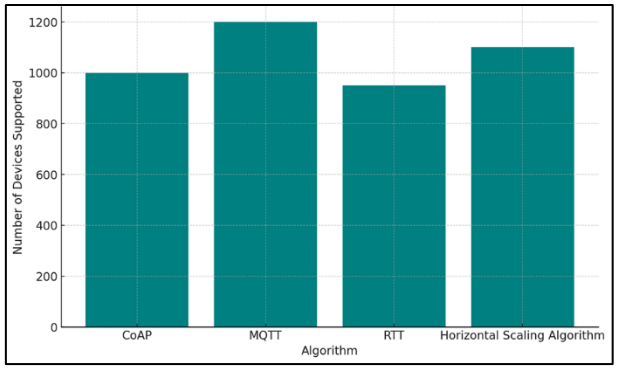

Figure 3: Scalability in terms of the number of devices supported

Organize inactivity measures the delay in information transmission over the organize. MQTT once more leads with the most reduced idleness at 18 ms, making it profoundly reasonable for real-time applications where speedy information exchange is vital. CoAP takes after closely with a idleness of 20 ms, profiting from its straightforwardness and effectiveness in compelled situations. The Level Scaling Calculation appears a direct inactivity of 22 ms, demonstrating effective stack conveyance but somewhat expanded idleness due to the complexity of dealing with different servers. RTT, with a idleness of 25 ms, appears the most noteworthy delay, likely due to its comprehensive information dealing with and real-time capabilities, which can present extra transmission delays. Versatility is basic for keen city applications, which frequently include thousands of interconnected gadgets. MQTT bolsters the most elevated number of gadgets (1200), underscoring its vigor and effectiveness in overseeing large-scale arrangements. The Even Scaling Calculation underpins 1100 gadgets, illustrating its capacity to preserve execution as the number of gadgets increments. CoAP bolsters 1000 gadgets, which is satisfactory for numerous applications but somewhat less adaptable than MQTT. RTT bolsters the least gadgets (950), reflecting potential confinements in taking care of a expansive number of associations concurrently.

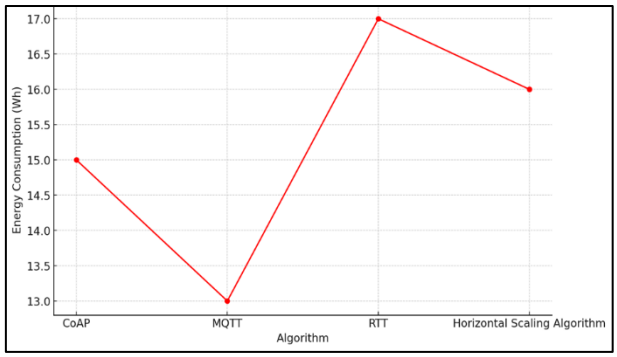

Figure 4: Energy Consumption Across Algorithms

Energy utilization may be a crucial thought for IoT systems, particularly in battery-operated gadgets. MQTT appears the least vitality utilization at 13 Wh, showing its productivity and reasonableness for energy-constrained situations. CoAP takes after with 15 Wh, profiting from its plan for obliged gadgets, shown in figure 4. The Even Scaling Calculation expends 16 Wh, reflecting the vitality taken a toll of keeping up numerous server occasions. RTT, with the most elevated vitality utilization at 17 Wh, recommends that its real-time capabilities come at the cost of higher control utilization. Throughput demonstrates the number of demands the system can handle per second. MQTT once more leads with 270 requests/sec, exhibiting its tall proficiency and capability to handle considerable information volumes. The Flat Scaling Calculation takes after closely with 260 requests/sec, reflecting its capacity to oversee tall loads through viable stack dissemination. CoAP, with 250 requests/sec, illustrates great execution in taking care of demands. RTT, with 230 requests/sec, appears somewhat lower throughput, likely due to its center on real-time information dealing with, shown in figure 5.

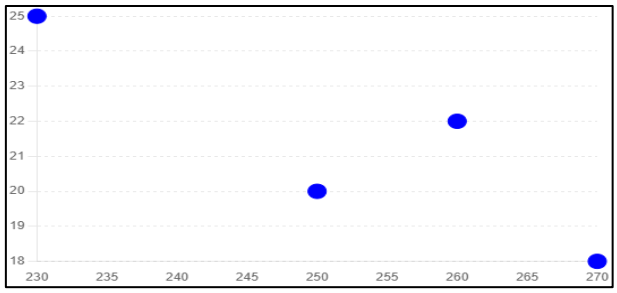

Figure 5: Throughput Vs. Network Latency

Blame resilience measures the framework's capacity to proceed working accurately within the occasion of disappointments. Both CoAP and the Flat Scaling Calculation show tall blame resistance at 99.9%, demonstrating their strength and reliability. MQTT follows closely with 99.8%, appearing great flexibility. RTT, with 99.7%, moreover illustrates solid blame resistance, in spite of the fact that somewhat lower than the other strategies.

5. Conclusion

The testing of IoT systems for Smart City uses shows how important it is to find the right balance between speed and growth in order to solve a wide range of urban problems. MQTT becomes a strong option when only a small amount of data and power is needed. This means it can be used for things like smart meters and watching the environment. Its lightweight message system makes the best use of resources and guarantees accurate data transfer across IoT networks that are spread out. CoAP can be expanded because it uses a RESTful design that lets devices talk to each other in real time and handle data without a central hub. This is important for uses like traffic management and smart grid systems. This framework's ability to use current web standards improves connectivity and makes it easier for different IoT settings, which are common in smart cities, to work together without any problems. LwM2M, on the other hand, is great at managing devices because it has powerful features for licensing, setup, and software changes. Its data processing and management features make operations run more smoothly, especially when it comes to handling large-scale applications across urban systems. To get the most out of Smart City projects, people involved should carefully think about the needs of their apps when choosing an IoT platform. Network limitations, data flow,

and the variety of devices all play major parts in determining how well the system works. In the future, researchers might look into mixed methods or models that are specifically made for urban settings. This would make IoT solutions in Smart Cities even more flexible and effective. Adopting IoT systems that are efficient and scalable could eventually make cities more linked, sustainable, and adaptable places to live. Cities can improve service delivery, resource management, and citizens' general quality of life by using these technologies well. This will pave the way for better and more livable cities in the future.

References

[1] Lnenicka, M.; Nikiforova, A.; Clarinval, A.; Luterek, M.; Rudmark, D.; Neumaier, S.; Kević, K.; Bolívar, M.P.R. Sustainable open data ecosystems in smart cities: A platform theory-based analysis of 19 European cities. Cities 2024, 148, 104851.

[2] Abadía, J.J.P.; Walther, C.; Osman, A.; Smarsly, K. A systematic survey of Internet of Things frameworks for smart city applications. Sustain. Cities Soc. 2022, 83, 103949.

[3] Bellini, P.; Palesi, L.A.I.; Giovannoni, A.; Nesi, P. Managing complexity of data models and performance in broker-based Internet/Web of Things architectures. Internet Things 2023, 23, 100834.

[4] Pereira, R.H. Lightweight data bridge for connecting self-service end-user analytic tools to NGSI-based IoT systems. Internet Things 2024, 25, 101125.

[5] Islam, M.S.; Hussain, I.; Rahman, M.M.; Park, S.J.; Hossain, M.A. Explainable Ai Model for Stroke Prediction Using Eeg Signal. Sensors 2022, 22, 9859.

[6] Hussain, I.; Park, S.J. Big-Ecg: Cardiographic Predictive Cyber-Physical System for Stroke Management. IEEE Access 2021, 9, 123146–123164.

[7] Bellini, P.; Nesi, P.; Pantaleo, G. IoT-enabled smart cities: A review of concepts, frameworks and key technologies. Appl. Sci. 2022, 12, 1607.

[8] Rejeb, A.; Rejeb, K.; Simske, S.; Treiblmaier, H.; Zailani, S. The big picture on the internet of things and the smart city: A review of what we know and what we need to know. Internet Things 2022, 19, 100565.

[9] Khang, A.; Rani, S.; Sivaraman, A.K. AI-Centric Smart City Ecosystems: Technologies, Design and Implementation; CRC Press: Boca Raton, FL, USA, 2022.

[10] Ajani, S. N. ., Khobragade, P. ., Dhone, M. ., Ganguly, B. ., Shelke, N. ., & Parati, N. . (2023). Advancements in Computing: Emerging Trends in Computational Science with Next-Generation Computing. International Journal of Intelligent Systems and Applications in Engineering, 12(7s), 546–559

[11] Malee, W.; Ruang-on, S.; Hussain, N.; Tina, F.W. Development of a smart watering system for controlling humidity inside mangosteen canopy in Nakhon Si Thammarat province, Southern Thailand. Int. J. Smart Sens. Intell. Syst. 2022, 15.

[12] Kale, Rohini Suhas , Hase, Jayashri , Deshmukh, Shyam , Ajani, Samir N. , Agrawal, Pratik K & Khandelwal, Chhaya Sunil (2024) Ensuring data confidentiality and integrity in edge computing environments : A security and privacy perspective, Journal of Discrete Mathematical Sciences and Cryptography, 27:2-A, 421–430, DOI: 10.47974/JDMSC-1898

[13] Dari, Sukhvinder Singh , Dhabliya, Dharmesh , Dhablia, Anishkumar , Dingankar, Shreyas , Pasha, M. Jahir & Ajani, Samir N. (2024) Securing micro transactions in the Internet of Things with cryptography primitives, Journal of Discrete Mathematical Sciences and Cryptography, 27:2-B, 753–762, DOI: 10.47974/JDMSC-1925

[14] Limkar, Suresh, Singh, Sanjeev, Ashok, Wankhede Vishal, Wadne, Vinod , Phursule, Rajesh & Ajani, Samir N. (2024) Modified elliptic curve cryptography for efficient data protection in wireless sensor network, Journal of Discrete Mathematical Sciences and Cryptography, 27:2-A, 305–316, DOI: 10.47974/JDMSC-1903

[15] Malee, W.; Ruang-on, S.; Hussain, N.; Tina, F.W. Using a smart watering system for controlling thrips inside mangosteen canopy in Nakhon Si Thammarat province, Southern Thailand. Int. J. Smart Sens. Intell. Syst. 2022, 15.

[16] Li, X.; Liu, H.; Wang, W.; Zheng, Y.; Lv, H.; Lv, Z. Big data analysis of the internet of things in the digital twins of smart city based on deep learning. Future Gener. Comput. Syst. 2022, 128, 167–177.

[17] Tyagi, N.; Bhushan, B. Demystifying the Role of Natural Language Processing (NLP) in Smart City Applications: Background, Motivation, Recent Advances, and Future Research Directions. Wirel. Pers. Commun. 2023, 130, 857–908.

[18] Yang, H.; Du, L.; Zhang, G.; Ma, T. A Traffic Flow Dependency and Dynamics based Deep Learning Aided Approach for Network-Wide Traffic Speed Propagation Prediction. Transp. Res. Part B Methodol. 2023, 167, 99–117.

[19] Kaur, P.; Kumar, Y.; Gupta, S. Artificial Intelligence Techniques for the Recognition of Multi-Plate Multi-Vehicle Tracking Systems: A Systematic Review. Arch. Comput. Methods Eng. 2022, 29, 4897–4914.

[20] Bathla, G.; Bhadane, K.; Singh, R.K.; Kumar, R.; Aluvalu, R.; Krishnamurthi, R.; Kumar, A.; Thakur, R.N.; Basheer, S. Autonomous Vehicles and Intelligent Automation: Applications, Challenges, and Opportunities. Mob. Inf. Syst. 2022, 2022, 7632892.

[21] Ma, Y.; Sun, C.; Chen, J.; Cao, D.; Xiong, L. Verification and Validation Methods for Decision-Making and Planning of Automated Vehicles: A Review. IEEE Trans. Intell. Veh. 2022, 7, 480–498.

[22] Bendiab, G.; Hameurlaine, A.; Germanos, G.; Kolokotronis, N.; Shiaeles, S. Autonomous Vehicles Security: Challenges and Solutions Using Blockchain and Artificial Intelligence. IEEE Trans. Intell. Transp. Syst. 2023, 24, 3614–3637.