

Queuing Modeling Approach to Developing Conceptual Framework for Smart Health Care System by using IoT and Cloud Computing Technologies

S S Mishra* and Mohit Kumar

Department of Mathematics & Statistics,
Dr Rammanohar Lohia Avadh University, Ayodhya, (U.P.), India

*Corresponding Author's Email: smssmishra5@gmail.com

Deepak Verma

Programme of Computer Science,
Dr Rammanohar Lohia Avadh University, Ayodhya, (U.P.), India
Email: deepak.discrete@gmail.com

(Received November 15, 2024, Accepted December 08, 2024)

Abstract: Healthcare IoT explodes with data from wearables and medical devices from wearable sensors and medical devices. Healthcare wrestles with mountains of data from wearables. Fog computing, processing data closer to its source, emerges as a solution for real-time patient insights can help to address this challenge. One key aspect of fog computing is load balancing, which distributes data Processing tasks across multiple servers to prevent overload and ensure smooth operation. In the context of health IoT, a “server busy” policy can be used to re-queue data when all servers are occupied. This ensures that critical health information is not lost and can be processed as soon as resources become available. With this in mind, a queuing modeling approach has been introduced to develop the conceptual framework for smart health care system in this paper for effectively computing the performance measures of the system.

Keywords: Queueing Models, Server Busy Policy, IoT, Cloud Computing, Smart Health Care System.

AMS Subject Classification: 90B22, 60K25

1. Introduction

Building upon the disruptive technologies of the fourth industrial revolution, this study delves into the concept of “smart hospitals”. It identifies and prioritizes critical technologies like IoT, robotics, and AI through a fuzzy

logic approach, paving the way for a novel evaluation model. This model assesses a hospital's readiness for embracing smart infrastructure, ultimately aiming to revolutionize healthcare delivery and enhance patient outcomes. By bridging the gap between theory and practice, the research offers valuable insights for both academics and healthcare institutions seeking to navigate the digital transformation journey, vide Ronaghi¹.

Lost lipstick found future the untold story of IoT, has blossomed into a vast network of everyday devices – fridges, phones, and more seamlessly communicating. While this *smart environment* unlocks exciting possibilities, it also raises concerns. Security looms large, with billions of connected devices presenting potential vulnerabilities. Privacy and data sharing also demand careful consideration as the IoT evolves, vide Kizza *et al.*².

This version condenses the idea while incorporating the challenge of managing the data, they generate becomes challenging. Cloud of Things (CoTs) emerges as a solution, integrating cloud computing with Highlights the small sensors and interconnected network. CoTs offers efficient management of data and resources, enabling valuable services and a wider range of IoT applications. This technology holds significant potential for the future, but its architecture, operation, and potential issues need careful consideration. A novel fog computing resource management (frm) paradigm for vital healthcare applications that uses hybrid load balancing and scheduling , vide Mutlag *et al.*³.

Wireless Sensor Networks (WSNs) have seen widespread use in critical sectors like healthcare, facing challenges in energy, processing, and data management. Integrating WSNs with cloud computing into sensor clouds offers a solution. This article explores data collection techniques in WSNs, emphasizing their significance in healthcare. This paper explores the world of Wireless Sensor Networks (WSNs) and their cloud partners. It breaks down how these networks collect data, analyzes it in the cloud, and reveals the hidden, it aims to enhance the efficiency of healthcare systems through real-time data processing and storage vide Ali *et al.*⁴.

The General Data Protection Regulation (GDPR) grants users the right to move their personal data (RtDP), allowing users to transfer their personal data between services. However, applying RtDP to the Internet of Things (IoT) poses challenges due to technical difficulties, legal issues, and lack of user awareness. It highlights the difficulty of implementing new legislation due to low user awareness, stakeholder understanding gaps, and technical hurdles. Recent EU legislation like the Digital Markets Act (DMA) and Data Act fight back, Easy data transfers and control the misuse. Challenges by guaranteeing data portability and preventing data misuse. While the UK currently upholds

RtDP, it may diverge from the EU's approach in the future. However, the impact of EU legislation might still influence UK practices due to compliance considerations, vide Turner and Tanczer⁵.

In this paper, we propose weaving the web a smart healthcare framework takes shape by introducing queueing modeling approach with IOT and cloud computing technologies.



Figure 1: Flow Chart of Conceptual Framework

Server busy policy of queueing models will be used to tackle dynamic data loading challenges in fog computing for health IoT devices. By employing adaptable queueing strategies, data replication, and potential offloading to the cloud, it optimizes resource allocation and ensures uninterrupted data delivery even during server overload. To faster processing, lower latency, and improved responsiveness for critical healthcare data, ultimately enabling better patient monitoring and decision-making, vide Evangelin and Vidhya⁶.

2. Server Busy Policy

2.1. Threshold-based: Define a resource utilization threshold (CPU, memory, etc.) to trigger “busy” state. When busy, dynamic data loading can be throttled, paused, or completely stopped. This paper proposes a decentralized load balancing algorithm for Edge/Fog computing focusing on service

latency. It aims to create a level playing field for users by minimizing latency variance across nodes. Unlike traditional methods like CPU or RAM usage, it directly considers user experience for fairer QoS. Both simulations and real-world tests on Raspberry Pies demonstrate its effectiveness in balancing latency across heterogeneous nodes, vide Mattia *et al.*⁷.

2.2. Queueing: Maintain a queue for incoming data loads, processing them when server resources are available. Consider prioritizing critical data or using dynamic queuing strategies based on data relevance.

2.3. Tuning parameters: The policy might have tunable parameters like queue size, timeout values, or priority weights that need to be set appropriately for optimal performance. Two new queue models cater to different priorities—customer-friendly with short waits but limited capacity, and manager-friendly with unlimited customers but longer waits. Both adapt to unpredictability, empowering diverse service systems to optimize based on their focus, vide Alam *et al.*⁸.

2.4. Impact on latency: Re-queueing can introduce additional latency for customer re requests. The policy should be designed to minimize this impact while maintaining efficient resource utilization. Overall, a “server busy” policy with good re-queueing strategies and potential offloading mechanisms can be an effective way for a fog computing load balancer to optimize resource distribution and handle spikes in traffic. However, careful monitoring and tuning are necessary to ensure good performance and user experience, Nakamura *et al.*⁹.

3. Single-Queue Models

We imagine a simple line with one server, like an M/M/1 model where arrivals and service times follow specific patterns. Queueing network models: These model more complex scenarios with multiple servers and interconnected queues, like a Jackson network simulating a real-world data center.

4. M/M/C Queuing Model

We manage crowds with multiple servers, vide Evangelin and Vidhya⁶. This model assumes several servers engaged in providing service to the demand raised by clients, vide Kembe *et al.*¹⁰. Understanding Performance with queuing models—studying cloud systems can be complex, so researchers turn to modeling for efficient analysis, vide Ghomi *et al.*¹¹. One powerful tool is queuing theory, which uses mathematical models to simulate how tasks wait and get processed.

5. Operating Characteristics of Queuing models for Measuring Performance of Cloud computing

Queuing Model, Double Server Formula, vide Basak *et al.*¹²

$$P_0 = \text{Prob}[\text{system is empty}] = \frac{1}{1 + \frac{\lambda}{\mu} + \left[\frac{\lambda}{\mu}\right]^2 \left[\frac{\mu}{2\mu - \lambda}\right]}$$

$$P_b = \text{Prob} \left[\begin{array}{c} \text{both servers} \\ \text{are busy} \end{array} \right] = \text{Prob} \left[\begin{array}{c} \text{new arrivals must} \\ \text{wait in queue} \end{array} \right]$$

$$= \frac{1}{2} \left[\frac{\lambda}{\mu} \right]^2 \left[\frac{2\mu}{2\mu - \lambda} \right] p_0$$

$$L_q = \text{average number in the system} = \frac{\gamma \mu \left[\frac{\lambda}{\mu} \right]^2 P_0}{(2\mu - \gamma)^2}$$

$$L = \text{average number in the queue} = L_q + \frac{\lambda}{\mu}$$

$$W_q = \text{average number in system} = \frac{\mu \left[\frac{\lambda}{\mu} \right]^2 P_0}{(2\mu - \lambda)^2}$$

$$w = \text{average time in system} = W_q + \frac{1}{\mu}$$

Note: λ is the arrival rate. μ is the service rate (per server).

Performance Measures	Single server (M/M/1)	Multiple Server (M/M/c)
Average server utilization in the system (ρ)	λ/μ	$\lambda/c\mu$
Probability that there are no patient in the system i.e. the servers are idle (P_0)	$1 - \lambda/\mu$	$k! + (\lambda/\mu) \sum_{c=0}^{c-1} \frac{c!}{c!} \mu$
Average of number of patients in the system (L_s)	λ/μ	$L_q + \lambda/\mu$
Average time a patient spends in the system (W_s)	$1/\mu$	L_s/λ
Average of number of patients in the queue (L_q)	$\lambda^2/\mu(\mu - \lambda)$	$(\lambda/\mu) \sum_{c=0}^{c-1} \frac{c!}{c!} \mu$
Average time a patient spends in the queue (W_q)	$\lambda/\mu(\mu - \lambda)$	L_q/λ

5.1. Cloud Data Centers: This is a powerhouse of internet services. The Cloud's Engine Room: Massive data centers, packed with servers, power the virtual world, vide Kizza *et al.*². These centers process service requests

through virtual machines (VMs), managed by software on the underlying servers. Imagine the data center in three layers:

Traffic Management: It controls data flow in and out.

Essential Services: Handles tasks like domain management and load balancing.

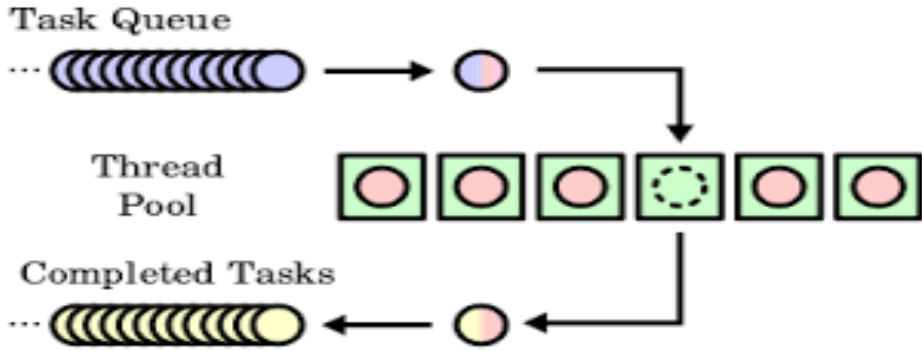


Figure 2: Queuing Model

Compute Layer: This is layer where hundreds of servers, physically organized in racks, handle your applications and data. This complex infrastructure ensures the smooth delivery of cloud services you use daily.

- i. **Queue behavior:** How tasks arrive, wait, and get processed in the cloud.
- ii. **Performance bottlenecks:** Identifying areas causing delays or inefficiencies.
- iii. **Resource allocation:** Optimizing server usage and resource distribution. Through this modeling approach, researchers can understand and improve the performance of cloud systems, ensuring smooth operation and efficient delivery of your favorite online services.
- iv. **Detection:** Define what constitutes a “busy” state. This could be based on CPU usage, memory consumption, number of active connections, or other metrics. Specify thresholds for triggering the policy.
- v. **Prioritization:** Prioritize critical requests over non-critical ones. This might involve defining service tiers or assigning weights to different request types.
- vi. **Rate limiting:** Limit the number of requests accepted per user, per second, or other relevant measure.
- vii. **Queueing:** Queue incoming requests and process them as resources become available.
- viii. **Back off Strategies:** Implement mechanisms for clients to automatically retry requests after a delay, instead of bombarding the server with repeated attempts.

- ix. **Rejection:** For extreme situations, reject some requests with appropriate error messages to prevent further overload.
- x. **Communication:** Inform users about the busy state with clear messages and expected wait times. Provide an estimated recovery time, if possible. Offer alternative routes or solutions if available.
- xi. **Monitoring and Evaluation:** Monitor the effectiveness of the policy and track user experience metrics during busy periods. Adjust thresholds and response actions as needed based on observations and user feedback.

6. Formation of Queue in Server Busy Policy

The basic structure focuses on two main aspects: the data structure used to store the queued items and the scheduling/processing logic. Here are some common data structures for queuing, vide Adeniran et al.¹³ :

- i. **FIFO (First-In-First-Out):** The simplest option, where new requests join the back, processed sequentially. Fairness prevails, but prioritization is a puzzle.
- ii. **LIFO (Last-In-First-Out):** Opposite of FIFO, newer requests are processed first. This can be useful for real-time tasks but lacks fairness for older requests.
- iii. **Priority Queue:** Each item has a priority associated, and higher-priority items are processed first. This allows for flexible prioritization but requires setting and managing priorities effectively.
- iv. **Multi-level Queue:** Combines different queue types based on priority or other criteria. This offers more granular control but adds complexity.
- v. **Scheduling/Processing Logic and Server availability:** The queue waits until a server becomes free before processing the next item.
- vi. **Batch processing:** Process multiple items together to improve efficiency, especially for smaller tasks.
- vii. **Timeouts:** If an item waits too long, remove it from the queue or retry processing.
- viii. **Dead-letter queue:** Store undeliverable or failed items for later handling or analysis.
- ix. **Queue length limits:** Prevent the queue from growing indefinitely by setting capacity limits.
- x. **Overflow strategies:** Choose how to handle situations when the queue is full, like dropping requests, rejecting new ones, or using alternative processing methods.
- xi. **Monitoring:** Track queue length, wait times, and processing duration to optimize performance and identify potential issues. By choosing the

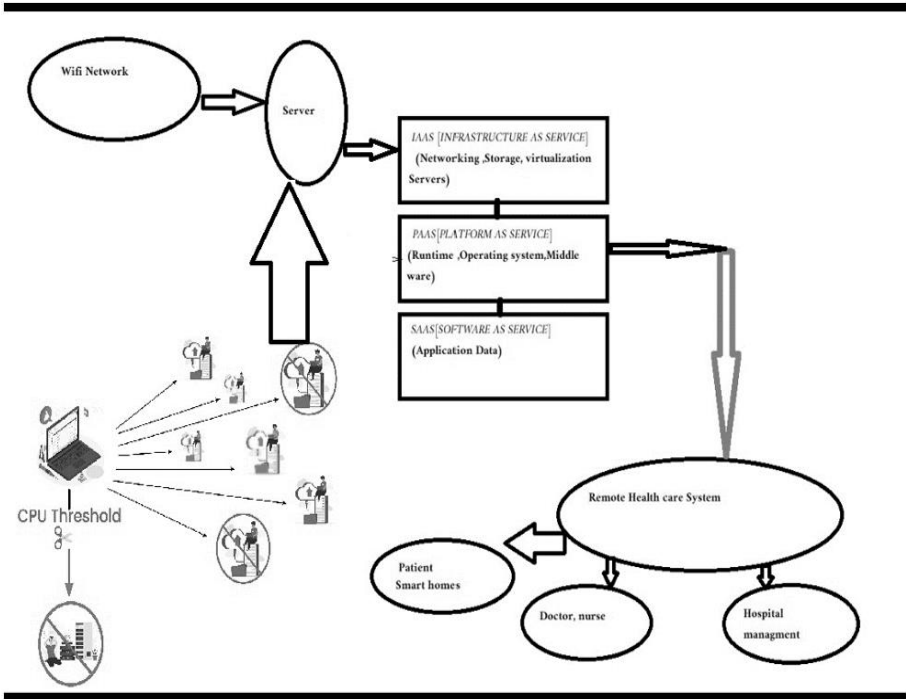
appropriate data structure, scheduling logic, and addressing these considerations, you can design an effective queueing system within your server busy policy.

7. Requeueing and Loading

In requeueing, customer data is simply placed back at the end of the queue to wait for a server to become available. This approach is easy to implement but can lead to increased queuing times and potentially dropped requests if the queue gets overloaded. Long wait times in clinics hurt patients and cost money. This study suggests a way to optimize staffing levels to reduce wait times and costs, vide Kembe *et al.*¹⁰.

Priority-based re-queueing: Customer data can be assigned priorities based on factors like urgency, service type, or resource requirements. Higher priority data gets placed closer to the head of the queue, ensuring they get processed sooner.

Adaptive re-queueing: The re-queueing strategy can be dynamically adjusted based on real-time conditions like server load, queue length, and resource availability. This helps optimize resource utilization and prevent queue overload.



Offloading to cloud: If fog servers are overwhelmed, the load balancer can offload some tasks to the cloud for processing. This can be helpful for computationally intensive tasks or when local resources are insufficient.

Rejection: In extreme cases, the load balancer might need to reject some requests to prevent system overload. This can be done based on pre-defined rules or dynamically depending on real-time conditions. A new queuing model accounts for two-stage service, mixed vacations, and optimizes costs using PSO and ABC algorithms. It's validated and analyzed for sensitivity, vide Kumar *et al.*¹⁴.

8. Integration with Healthcare Platforms

The load balancer should integrate seamlessly with existing healthcare platforms and data management systems for efficient data analysis and utilization. Fault tolerance Mechanisms for graceful degradation and handling server failures should be implemented to ensure continuous data processing and minimal service disruption.

Resource scaling: The system should be able to scale up or down its resources (fog servers, cloud resources) depending on the dynamic data load from health IoT devices, vide Baburao¹⁵. By addressing these challenges and considerations, you can effectively utilize a server busy policy in fog computing to optimize resource management and improve health care data processing for health IoT devices. Ensuring fault tolerance for dynamic data loading on a server under a busy policy requires a well-designed strategy to handle server overload and data loading failures, vide Mohammad *et al.*¹⁶.

9. Conclusion

Queuing theory optimizing flow in lines and cloud Systems is a powerful tool for analyzing and improving systems where people or tasks wait for service. Queuing theory helps understand factors like arrival rate, service rate, and server availability to optimize such situations. The M/M/C model is particularly useful for managing crowds. It assumes multiple servers, random arrival patterns, and first-come, first-served service

This model helps predict wait times, queue lengths, and resource allocation, allowing managers to streamline service delivery and improve overall efficiency. Cloud data centers, the backbone of the internet, rely on queuing theory to understand their performance. By building mathematical models, researchers can analyze how tasks behave in the cloud. These models come in two main categories. Single queue models simulate a basic scenario

like a single server at a store (M/M/1). Queuing network models represent more complex situations with multiple inter connected queues, like a network of servers in a data center (Jackson network). By analyzing these models, researchers gain valuable insights into how tasks flow in the cloud, enabling better optimization and performance improvements, ultimately leading to a smoother and more efficient experience for all users.

Acknowledgement: Authors are thankful for the financial support, vide Department of Higher Education UP Govt., Lucknow (No.-89/2022/1585/seventy-4-2022/001-4-2022 dated 10.11.2022).

References

1. Ronaghi, Mohammad Hossein; Toward a model for assessing smart hospital readiness within the Industry 4.0 paradigm, *Journal of Science and Technology Policy Management* 15(2) (2024), 353-373.
2. Joseph Migga Kizza; *Internet of things (iot): growth, challenges, and security*, *Guide to Computer Network Security*, Springer International Publishing, 2024.
3. Mutlag, Ammar Awad, Mohd Khanapi Abd Ghani, Othman Mohd, Karrar Hameed Abdulkareem, Mazin Abed Mohammed, Meshal Alharbi, and Zaid J. Al-Araji; A new fog computing resource management (FRM) model based on hybrid load balancing and scheduling for critical healthcare applications, *Physical Communication* 59 (2023), 102109.
4. Ali, Ihsan, Hasniuj Zahan and Spyridon Mastorakis; Sensor Clouds: Recent Advancements, Use Cases and Open Challenges, *IEEE Internet of Things Magazine* 7(1) (2024), 98-103.
5. Turner, Sarah, and Leonie Maria Tanczer; In principle vs in practice: User, expert and policymaker attitudes towards the right to data portability in the internet of things, *Computer Law & Security Review* 52 (2024), 105912.
6. Evangelin, K. Ruth and V. Vidhya; Performance measures of queuing models using cloud computing, *Asian Journal of Engineering and Applied Technology*, 4(1) (2015), 8-11.
7. Mattia, Gabriele Proietti, Antonio Pietrabissa, and Roberto Beraldi; A load balancing algorithm for equalising latency across fog or edge computing nodes, *IEEE Transactions on Services Computing*, 16(5) (2023), 3129-3140.
8. S.T. Alam, F.A. Azrin, A.S.M.N. Uddin and E. Sagor; *Formulation and performance assessment for multiple server queueing models*, 2023.
9. Nakamura, Shigenari, Tomoya Enokido and Makoto Takizawa; Load balancing algorithm for information flow control in fog computing model, *International Conference on Emerging Internetworking, Data & Web Technologies*, Springer International Publishing, 2023.

10. M.M. Kembe, E.S. Onah and S. Iorkegh; A study of waiting and service costs of a multi-server queuing model in a specialist hospital, *International journal of scientific & technology research*, 1(8) (2012), 19-23.
11. Jafarnejad Ghomi, Einollah, Amir Masoud Rahmani, and Nooruldeen Nasih Qader; Applying queue theory for modeling of cloud computing: A systematic review, *Concurrency and Computation: Practice and Experience*, 31(17) (2019), e5186.
12. Arpita Basak, and Amit Choudhury; Bayesian inference and prediction in single server M/M/1 queuing model based on queue length, *Communications in Statistics-Simulation and Computation*, 50(6) (2021), 1576-1588.
13. A Adeniran, Muhammad Sani Burodo and Shamsuddeen Suleiman; Application of queuing theory and management of waiting time using multiple server model: empirical evidence from Ahmadu Bello University Teaching Hospital, Zaria, Kaduna State, Nigeria, *International Journal of Scientific and Management Research*, 5(4) (2022), 159-174.
14. Anshul Kumar and Madhu Jain; Cost optimization of an unreliable server queue with two stage service process under hybrid vacation policy, *Mathematics and Computers in Simulation*, 204 (2023), 259-281.
15. D. Baburao, T. Pavankumar and C.S.R. Prabhu; Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method, *Applied Nanoscience*, 13(2) (2023), 1045-1054..
16. M. Aazam, E.-N. Huh, M. St-Hilaire, C.-H. Lung, and I. Lambadaris; Cloud of things: integration of IoT with cloud computing, *Robots and sensor clouds*, (2016), 77-94.