Performance Analysis of Internet of Things Interactions via Simulation-Based Queueing Models

Article

Dr.Meriam Afif, Mehdi Jerbi, Mohamed Ayadi

National Institute of applied sciences and technology, 676 INSAT Centre Urbain Nord BP، Tunis Cedex 1080

Abstract

A variety of middleware APIs and protocols exist to help in the development of Internet of Things (IoT) applications. These apps can rely on either reliable or less reliable protocols that offer different Quality of Service (QoS) levels. Using these protocols, APIs, and varying QoS levels can help meet the requirements of critical IoT applications, such as those used in emergency response situations. To study QoS in these IoT applications, we need to employ a method of performance analysis. This is where queueing network models come in - they provide a framework for analyzing and modeling IoT interactions in terms of QoS through either analytical or simulation models. In this paper, we present various queueing models that can represent different QoS scenarios in IoT interactions. Specifically, we propose queueing models that can illustrate factors such as message drop probabilities, inconsistent mobile connectivity, message availability or validity, prioritization of crucial information, and the processing or transmission of messages. Our simulation models demonstrate that altering QoS settings significantly impacts both the success rates of delivery and response times.

Keywords: *Internet of Things; queueing models; middleware; QoS analysis*

Introduction

The Internet of Things (IoT) is all about connecting the physical world with computer systems. IoT devices, which have sensors, are found everywhere - from community spaces and smart buildings to high-tech transportation systems[1,2]. These devices can make our lives safer and better by providing useful information. For example, some studies have used IoT devices to keep an eye on offices or homes for any signs of earthquakes. This kind of application gives us important information, and we expect it to work quickly and reliably. Therefore, it's really important for us to find a way to analyze and model the performance of IoT applications. IoT devices are usually cheap, use little power, and can be moved around. Depending on where they're used, the applications that use them can have different features. Some challenges we can pinpoint are: (i) low-cost devices don't have a lot of memory, (ii) there might be limited data available or the data might not always be valid, (iii) the device might not always be connected (especially if it's mobile), and (iv) there might be a need to deliver data quickly. Moreover, IoT applications are built on APIs and protocols, which

can be either reliable or unreliable, and these can also impose additional QoS constraints[3,4]. So, these challenges are really important when we're trying to analyze the performance of IoT systems.Past work [5–7] on both designing and evaluating mobile systems with certain constraints (like limited resources or inconsistent availability) have relied on something called queueing theory [8]. Important IoT protocols have been looked at in terms of metrics like how often deliveries are successful and how quick the response times are [9,10]. But these studies focused on specific protocols. So, researchers have started to look at how well-known interaction models [11,12] perform, which can represent different IoT protocols. They've looked at the publish/subscribe model using formal models [13] and evaluated it using queueing networks [14] or queueing Petri nets [15]. You can think of queueing networks as a series of connected service centers that can provide simulation or analytical solutions for various performance measures (like response time).When we link different service centers to create a queueing network, we can model an IoT interaction. This paper discusses some queueing models that we've looked at in earlier work. Specifically, in studies [16–19], we developed a framework where different middleware protocol nodes (like clients, servers, and brokers) are looked at as queues, and the messages they exchange are seen as jobs being completed. We looked at a separate queueing model to model Quality of Service (QoS) settings like the inconsistent connection [20] of mobile devices, using an ON/OFF queueing model [16,17]. We showed how our models could be used in studies [16–18,21–23] to measure the performance of publish/subscribe and data-streaming systems. In another study [19], we used our approach to measure the performance of IoT devices with different QoS settings. This paper builds on our earlier work [24] that provides a summary of queueing models representing the QoS constraints mentioned earlier in IoT applications. In this paper, we've included two queueing systems that model the processing or transmission of different types of IoT messages (like video data versus temperature data) and the prioritization of important data in critical IoT applications (like situations involving a building fire).

This paper makes several key contributions:

1. It explains a general ON/OFF model that covers the chances of losing messages and the probabilities of messages joining.
2. It introduces two queueing systems that model the transmission, processing, and prioritization of different types of messages based on how urgently they need to be delivered.
3. It describes additional features for queueing models that represent the availability of messages and devices with limited resources.
4. It compares different queueing models in terms of how often deliveries are successful and how quick the response times are, using experiments based on simulations.

The rest of this paper is set out as follows: we discuss related work in Section 2. In Section 3, we describe the queueing models that represent different Quality of Service (QoS) settings in IoT interactions and applications. In Section 4, we compare the performance of queueing models considering message availabilities, probabilities of messages joining, and limited buffer capacities. We also show how you can use the queueing models we present to correctly set up and fine-tune an IoT system. In Section 5, we conclude the paper and suggest possible future developments.

Related Work

Nowadays, IoT applications are made up of devices that use existing APIs and IoT protocols like CoAP, MQTT, DPWS, XAMPP, and ZeroMQ [10,25] to share data. Each protocol comes with certain

Quality of Service (QoS) features to ensure specific response times and success rates for data delivery between devices. Initially, each protocol takes on different characteristics from the transport mechanisms (like TCP/UDP) it's built on. After that, it supports different ways of delivering messages. For example, CoAP lets you choose between "confirmable" and "nonconfirmable", while MQTT gives you three options: "fire and forget", "delivered at least once", and "delivered exactly once" [4]. Plus, devices that use these protocols can be mobile (like wearable devices), and IoT applications might need to make sure the information they provide is fresh by dropping (possibly less important) messages (like mission-critical info for public safety [2]).

Studies [3,9,10] have looked at the balance between response times and delivery success rates when using key IoT protocols. But these methods are specific to certain protocols, which can limit the options for application designers when they're introducing a new IoT protocol. When it comes to middleware protocols, queueing Petri nets (QPNs) were used in one study [15] to predict performance accurately. But while QPNs are great at representing parallelism, they're better suited for small to moderate size systems and can use a lot of computational resources [26]. A number of past efforts focused on designing and evaluating mobile systems have tried to ensure QoS requirements are met under different constraints (like inconsistent availability and limited resources) [27–29]. The evaluation methods they used mainly came from the field of queueing theory.

Several existing systems like RabbitMQ, ActiveMQ, and mosquitto message brokers handle message dropping through maximum queue capacity [30]. To ensure the timeliness of messages, some messages are lost due to the validity or availability periods that can be set for each message through pub/sub protocols and APIs (like the JMS API). More advanced methods support semi-probabilistic delivery [31] or consider subscriber preferences [32–34]. For reliable and timely data exchange, current solutions manipulate data at both the middleware and network layers. Earlier middleware-based solutions [35–37] supported prioritizing or bandwidth allocation based on system capacity, data relevance, and data importance. More recent solutions prioritize based on the validity of published data and subscriptions [38], or delay and reliability requirements [39]. Now, standardized message brokers like RabbitMQ and ActiveMQ allow priorities to be assigned at the publisher side before a message is sent.

In this paper, we're using queueing-network models (QNMs) [14,40] to model the performance of data exchange in IoT. QNMs have been used a lot to represent and analyze communication and computer systems. They're straightforward and effective tools for application designers when it comes to evaluating and predicting system performance. While in [27,28] the authors evaluated the performance of WiFi mobile users with an ON/OFF queue, this paper uses and adapts this queue as part of a QNM that represents how IoT devices behave when mobile. To represent the max capacity of parts of the IoT system like pub/sub message brokers (RabbitMQ, ActiveMQ), you can apply a specific buffer size to any queue in the QNM. Also, to represent message losses that happen in brokers via APIs or protocols (like JMS), you can apply a validity or availability (lifetime) period to each message that enters the QNM. To represent losses that happen because of the protocol used for data exchange (like CoAP deployed over UDP [25]), the ON/OFF loss model can be used, which drops messages when the queueing server isn't active. We're also introducing a model for message dropping that uses probabilistic techniques. Finally, to represent IoT systems that consider the preferences of data recipients [32–34] to deliver different types of data (message sizes, importance, etc.) [35,36,38,39], we're introducing multi-class and priority queues where different classes of messages can be assigned different priorities. In short, application designers can analyze and configure certain system

aspects (middleware QoS delivery modes, network and user connectivity, message dropping rates, system resources, priority levels, message sizes) by combining the provided queueing models to adjust and ensure the right response time and delivery success rate between IoT devices.

Queueing Models

In this section, we're going to go over some key definitions of the queueing models that we used in our simulation-based approach.

1. M/M/1 Model

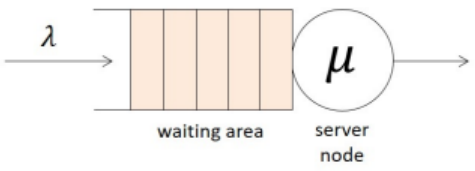
This queue is used to model continuous service of messages (transmission, reception or processing) as part of an IoT interaction from end to end. It's based on the classical M/M/1 queue (Figure 1a) where Poisson arrivals are serviced by a single server for a service time that's exponentially distributed.

An M/M/1 queue (qm/m/1) can be defined with the tuple:

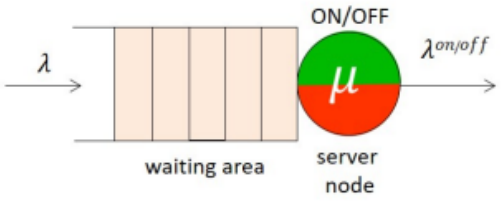
$$qm/m/1 = (\lambda, \mu). \quad (1)$$

Here, λ is the rate at which messages arrive at the queue and μ is the rate at which messages are serviced. Let D = 1/μ be the service demand for the processing delay (service time) of a message. Based on [14], the time that a message stays in an M/M/1 system (the sum of queueing time and service time, which we also call mean response time) is calculated with:

$$\Delta^{m/m/1} = \frac{D}{1 - \lambda D}. \quad (2)$$



(a) M/M/1 queue.



(b) ON/OFF queue.

Figure 1. M/M/1 and ON/OFF queues.

2. ON/OFF Model

In Figure 1b, we introduce the intermittent (ON/OFF) queue, which models the connections and disconnections of a mobile peer. Messages arrive in the system following a Poisson process with rate λ > 0, and are placed in a queue waiting to be "served" with rate μ > 0 (also exponentially distributed).

As for the server, we assumed that its operation followed an on–off procedure. Specifically, the server stays in the ON state for an exponentially distributed time, with parameter θON. While in that state, the server services messages. Let TON = 1/θON be the time during which the server is ON. When this time is up, the server enters the OFF state for an exponentially distributed period with rate θOFF. While in that state, the server stops servicing messages. Let TOFF = 1/θOFF be the time during which the server is OFF. Like the M/M/1 queue, an ON/OFF queue qon/off can be defined with the tuple:

$$q_{on/off} = (\lambda, \lambda^{on/off}, \mu, T_{ON}, T_{OFF}). \quad (3)$$

Here, λ is the rate at which messages arrive, λ on/off is the rate at which messages leave, and μ is the service rate for processing messages during TON. The output process λon/off is intermittent because no messages leave the queue during TOFF intervals. As for when TON expires while a message is still being serviced, we assumed that the server stops processing the message and continues in the next TON period.

We use Δon/off to represent the mean response time for the qon/off queue, which is the time that a message spends in the system.

In [16,17,19], we provided the following formula to estimate Δon/off:

$$\Delta^{\text{on/off}} = \frac{E(n)_{\text{on/off}}}{\lambda}. \quad (4)$$

Here, E(n)on/off refers to the average number of messages in the system (both in the server and the queue) [17].

3. Probabilistic ON/OFF Model

The queueing delays that may occur in the ON/OFF model can be addressed via the probabilistic ON/OFF model (Figure 2a).

In this model, the arrival process is still Poisson, and messages wait to be served while the server is ON. However, when the server is OFF, messages can either leave the system with probability 1 - ζ or enter the queue with probability ζ. The probabilistic ON/OFF queue q-prob-on/off can be defined via the tuple:

$$q^{prob}_{\text{on/off}} = (\lambda, \lambda^{\text{on/off}}, \mu, \zeta, T_{\text{ON}}, T_{\text{OFF}}). \quad (5)$$

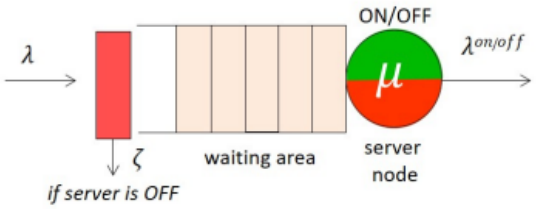
Here, λ is the rate of message arrival, ζ is the probability of entering the queue, λ

on/off is the rate at which messages leave, and μ is the service rate (during TON) for message transmission. When ζ = 1, we get the ON/OFF model of Section 2. (i.e., during OFF periods, all messages enter the queue).

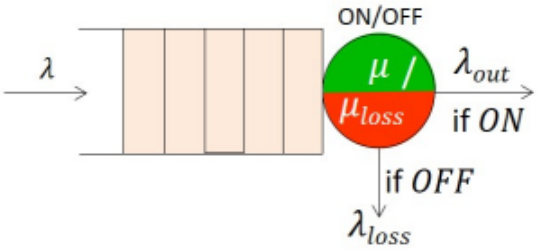
Δ-prob-on/off is the mean response time (the time a message stays in the system) for the q-prob-on/off queue. In [41], we provided the following formula to estimate Δ-prob-on/off:

$$\Delta^{prob}_{\text{on/off}} = \frac{E(n)_{\text{on/off}}}{\lambda_{eff}} \quad (6)$$

where λeff is the rate of messages served in the probabilistic ON/OFF queue q-prob-on/off.



(a) Probabilistic ON/OFF queue.



(b) ON/OFF loss queue.

Figure 2. *Probabilistic and loss ON/OFF queues.*

Such a probabilistic ON/OFF model helps system designers in improving or tuning delays of components that transmit messages during period TON, but may also disconnect for a period TOFF. Specifically, such a component could represent an IoT sensor (which is constrained in terms of memory and energy) that can periodically go to sleep mode, while its probabilistic behavior can be defined at the application layer.

4. ON/OFF Loss Model

The ON/OFF models discussed in the previous subsections don't consider the case of message losses (or message drops). All messages are either buffered before service or immediately served if the server is empty.

To model message losses, we consider the ON/OFF loss queue where messages still arrive in the system according to a Poisson process (Figure 2b). Regarding the server function, we consider that while the server is ON, messages enter the queue and wait to be served with rate μ. When the server is OFF, messages enter the queue, are served with rate μloss > μ, and exit the system (i.e., messages are lost).

The assumption that messages are “served” with rate μloss > μ was adopted because the ON/OFF loss queue is used for the performance analysis of an IoT mobile device that uses a middleware protocol. This protocol introduces (message) losses since it builds atop UDP and doesn't establish a logical sender/receiver session (i.e., there's no QoS guarantee for messages). As for the value of μ, it can be based on the network transmission delay of the end-to-end interaction. As for μloss, we identify the network’s point where message losses occur. For instance, if the ON/OFF loss queue is used to model the intermittent connectivity of a wireless receiver, then we parameterize μloss via the corresponding transmission delay of the network. The definition of such a delay can be based on how the sender interacts with the access point that the receiver connects or disconnects.

An ON/OFF loss queue q-loss-on/off can be defined according to the tuple:

$$q^{loss}_{\text{on/off}} = (\lambda, \lambda_{\text{out}}, \lambda_{\text{loss}}, \mu, \mu_{\text{loss}}, T_{\text{ON}}, T_{\text{OFF}}). \quad (7)$$

where λ is the messages’ arrival rate; λout and μ are the messages’ output and processing
rates during ON periods, respectively; λloss and μloss are the lost and processing rates
during OFF periods. Both λout/λloss rates were exponential since new messages split
according to the TON/TOFF intervals.

5. Multiclass Model

This queue model provides continuous serving of messages from different classes. These classes categorize messages of IoT interactions that have different characteristics, such as sensor data, video streaming services, and transactional messaging, each requiring distinct transmission and processing resources. Each class corresponds to the common M/M/1 queue, which features Poisson arrivals and exponential service times. Multiple M/M/1 queues are used to form the multiclass queue.

A multiclass queue (qmcl) is defined by the tuple:

$$\text{qmcl} = (\lambda, \mu, C) \quad (8)$$

where λ is the input rate of messages to each class's queue, and μ is the service rate for processing messages of each class. The time a message from class ck remains in the system is given by:

$$\Delta^{\text{mcl}} = \frac{1}{\mu_{c_k} - \mu_{c_k} \sum_{c_n \in C} \lambda_{c_n} / \mu_{c_n}}. \quad (9)$$

6. Nonpreemptive Priority and Multiclass Model

The multiclass queue serves messages based on a first-come-first-served policy, regardless of the class each message belongs to. However, IoT systems handle data of different application types such as emergency response, real-time, and video streaming. The nonpreemptive priority and multiclass (NPPM) model enables prioritization of messages from different classes when being served. Messages from n classes arrive according to Poisson processes and are then classified into different queues based on their assigned priority.

A nonpreemptive priority and multiclass queue (qmclpr) is defined by the tuple:

$$\text{qmclpr} = (\lambda, \mu, C, Y) \quad (10)$$

where λ is the input rate of messages to each class's queue, μ is the service rate for processing messages of each class, and Y is the assigned priority to each class. The time a message from class ck with priority yck remains in the system is given by:

$$\Delta^{\text{mcl}} = \frac{L_{c_k, y_{c_k}}(\lambda, \mu)}{\lambda_{c_k}} \quad (11)$$

7. Additional Features

Until now, we considered queueing models with infinite buffer capacity and messages with infinite lifetime. However, in reality, IoT interactions may not always fit these characteristics. For example, long

disconnection periods of an IoT sensor may exceed the buffer capacity, rendering older messages obsolete. To address these constraints, additional features are incorporated into the queueing models.

7.1. Lifetime Messages in Queueing Networks

In a queueing network, messages with a specific arrival rate λ are processed in the first queue. These messages have an attributed lifetime period, after which they are considered expired and leave the queueing network. This model also includes the M/M/1 queue with reneging or impatient customers, which allows for messages to leave if a certain lifetime period passes without service.

7.2. Queues of Finite Capacity

This feature considers a specific buffer size for each queue, preventing the system from handling more than K messages. This feature is especially relevant for devices with limited capacity. If a new message arrives and the condition new_queue_size + message_in_service > K is true, the message is dropped. Otherwise, the message joins the queue and waits to be served. This model is represented as M/M/1/K in literature. In our model, we incorporate system size K in the M/M/1 and the ON/OFF queues by adding it to the corresponding tuples (1),(3), (5), and (7).

Experiment Results

1. ON-OFF Model

In the context of the article, we write a Python script that simulates the mean response time of a queue with a server operating in ON/OFF mode. The function simulate_queue takes the arrival rate, service rate, simulation time, ON time period, and OFF time period as input arguments and returns the mean response time of the system. The main part of the script sets up the necessary parameters, runs the simulations, and generates a plot to visualize the results.

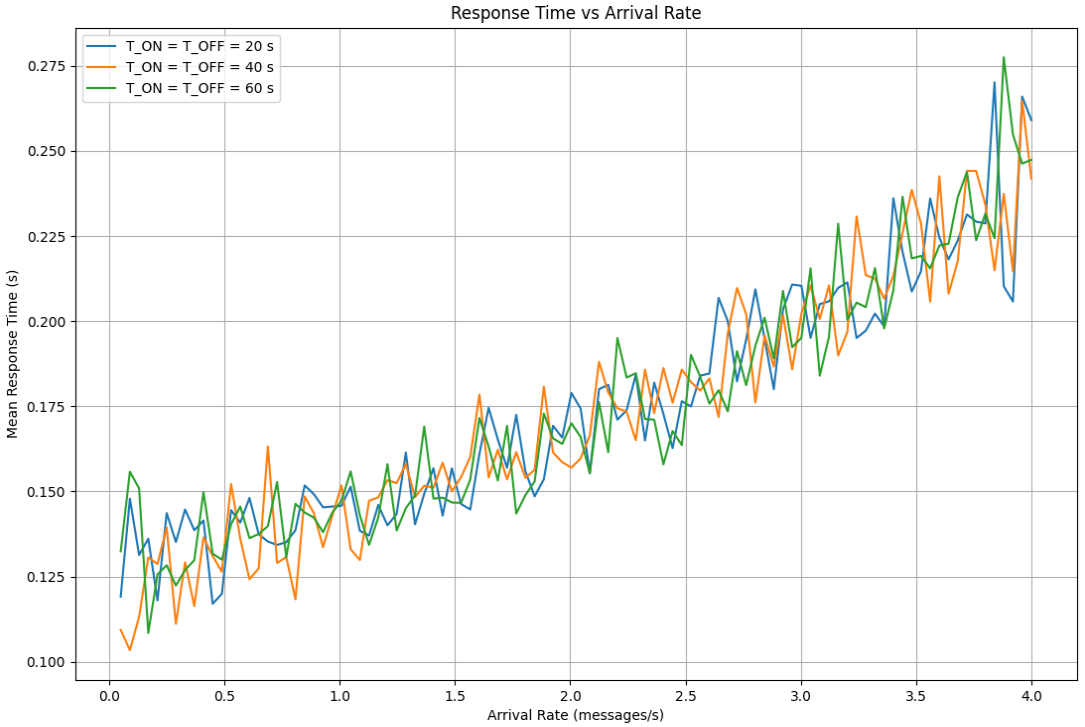


Figure 5: Response time for various T ON , T OFF parameters.

The plot shows the mean response time of a queue with a server operating in ON/OFF mode as a function of the arrival rate of messages. The x-axis represents the arrival rate in messages per second, while the y-axis represents the mean response time in seconds.

There are three curves in the plot, each corresponding to a different ON/OFF time period (T_ON = T_OFF). As the arrival rate increases, the mean response time also increases. However, the rate of increase is different for each ON/OFF time period.

For the ON/OFF time period of 20 seconds, the mean response time increases gradually and smoothly as the arrival rate increases. For the ON/OFF time period of 40 seconds, the mean response time shows a more pronounced increase as the arrival rate increases, with a sharper increase at higher arrival rates. For the ON/OFF time period of 60 seconds, the mean response time increases rapidly and nonlinearly as the arrival rate increases.

Overall, the plot shows that the mean response time of the system is sensitive to the ON/OFF time period and the arrival rate. The variation in the curves may be due to random fluctuations or errors introduced by the simulation or measurement process.

2. ON/OFF model with QoS Features

On one hand, we write a python script that simulates a queueing system with an ON/OFF server model and calculates the mean response time of messages in the queue. The queueing system is simulated for different arrival rates and different QoS features.

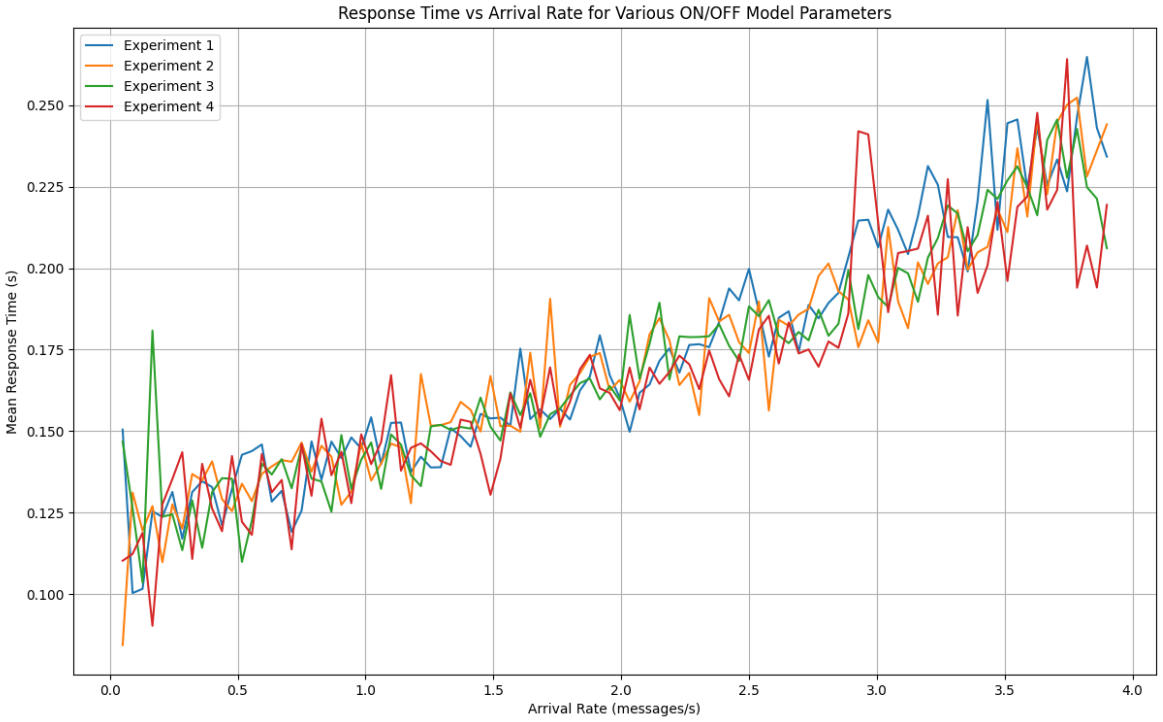


Figure 6: Response time for various ON/OFF model parameters.

The Experiment 1: ζ=1, lifetime = infinit, K = Infini
The Experiment 2: ζ=1, lifetime = 30 sec, K = Infini
The Experiment 3: ζ=0.75, lifetime = Infini, K = Infini
The Experiment 4: ζ=0.75, lifetime = Infini, K = 100

The curve on the plot represents the mean response time of the system as a function of the arrival rate. As the arrival rate increases, the mean response time also increases, which makes sense since the queue becomes longer, and the server takes more time to process messages. The curve also shows that the mean response time is affected by the ON/OFF time periods of the server. As the ON/OFF time periods increase, the mean response time also increases, which indicates that the server's behavior has a significant impact on the overall system performance.

In experiment 1, we notice that the response time rate is increasing exponentially. The reason behind that is the lifetime of the experiment and the size of the queue are infinite. In other words there is huge amount of incoming messages, so the response time will increase exponentially.

In experiment 2, the response time is so low because the long of the experiment is 30 sec. So we don't have many incoming messages.

In experiment 3, the rate of entering the queue is 0.75. Which means the number of incoming messages will be decreased. Thus the value of mean response time is low.

In experiment 4, the size of queue is limited to 100. In other words we decrease the mumber of treated messages. Thus, the value of mean response time is low.
The plot exhibits a lot of variation due to the stochastic nature of the simulation, which involves the generation of random variables like arrival times and service times.

On the other hand, we write a python script that simulates a queueing system with an ON/OFF server model and calculates the mean success rate of messages in the queue. The queueing system is simulated for different arrival rates and different QoS features.

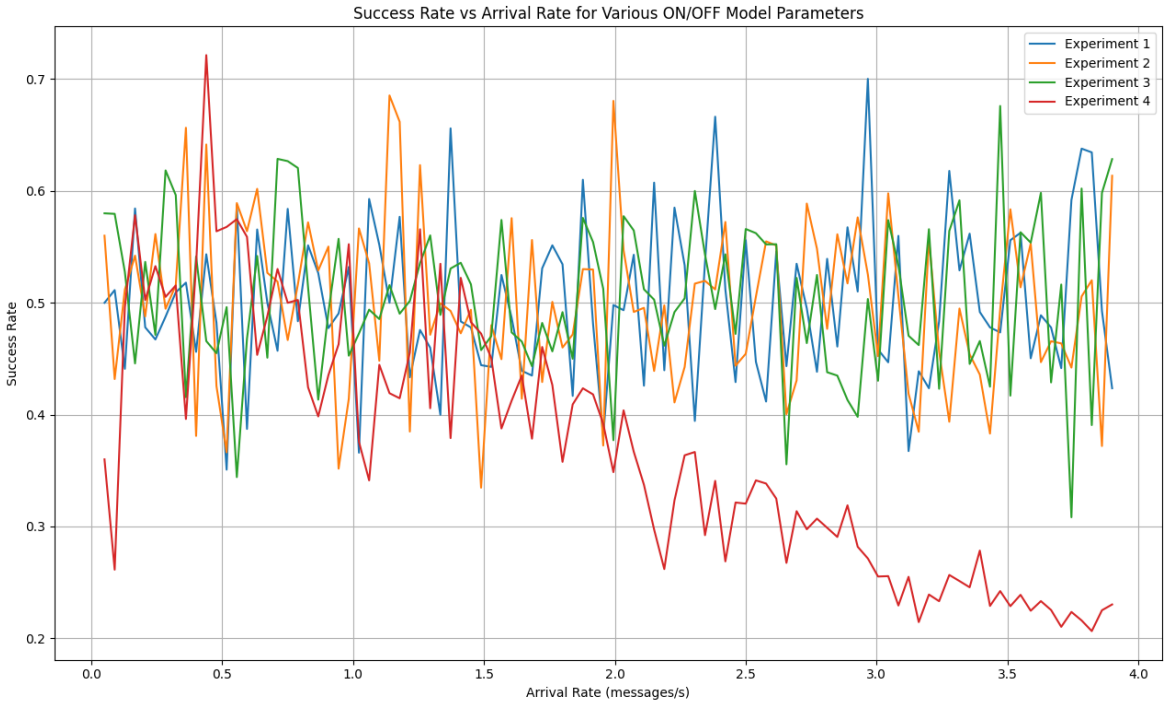


Figure 7: Success rates for various ON/OFF model parameters.

The Experiment 1: ζ=1, lifetime = infinit, C = Infini
The Experiment 2: ζ=1, lifetime = 30 sec, C = Infini
The Experiment 3: ζ=0.75, lifetime = Infini, C = Infini
The Experiment 4: ζ=0.75, lifetime = Infini, C = 100

The curve on the plot shows the relationship between the arrival rate and the success rate of the system under different ON/OFF model parameters. As the arrival rate increases, the success rate decreases, which is expected since the queue becomes longer, and the server has to process more messages. However, the impact of the ON/OFF model parameters on the success rate is not consistent across the experiments.

For Experiment 1, where there are no buffer capacity or message lifetime constraints, **the success rate decreases gradually** as the arrival rate approaches 4 messages/s,
For Experiment 2, where there is a message lifetime constraint of 30 seconds, the success rate decreases exponentially as the arrival rate increases.
indicating that the server is able to handle the traffic up to a certain point before the lifetime constraint starts to affect the success rate.

For Experiment 3, where there is a reduced buffer capacity, the success rate decreases even more gradually as the arrival rate increases, indicating that the reduced buffer capacity allows the server to handle higher traffic..

For Experiment 4, where there is a limited buffer capacity of 100 messages, the success rate decreases rapidly as the arrival rate approaches the buffer capacity limit, indicating that the **buffer capacity** is a critical factor in the system's performance.

Overall, the plot provides insights into how different ON/OFF model parameters affect the system's success rate under different QoS features and helps identify the optimal values of these parameters that result in the highest success rate.

3. Message Classes Assigned with Priorities

The plot shows the mean response time for each class in two different scenarios. The x-axis represents the class priority, and the y-axis represents the mean response time in seconds.

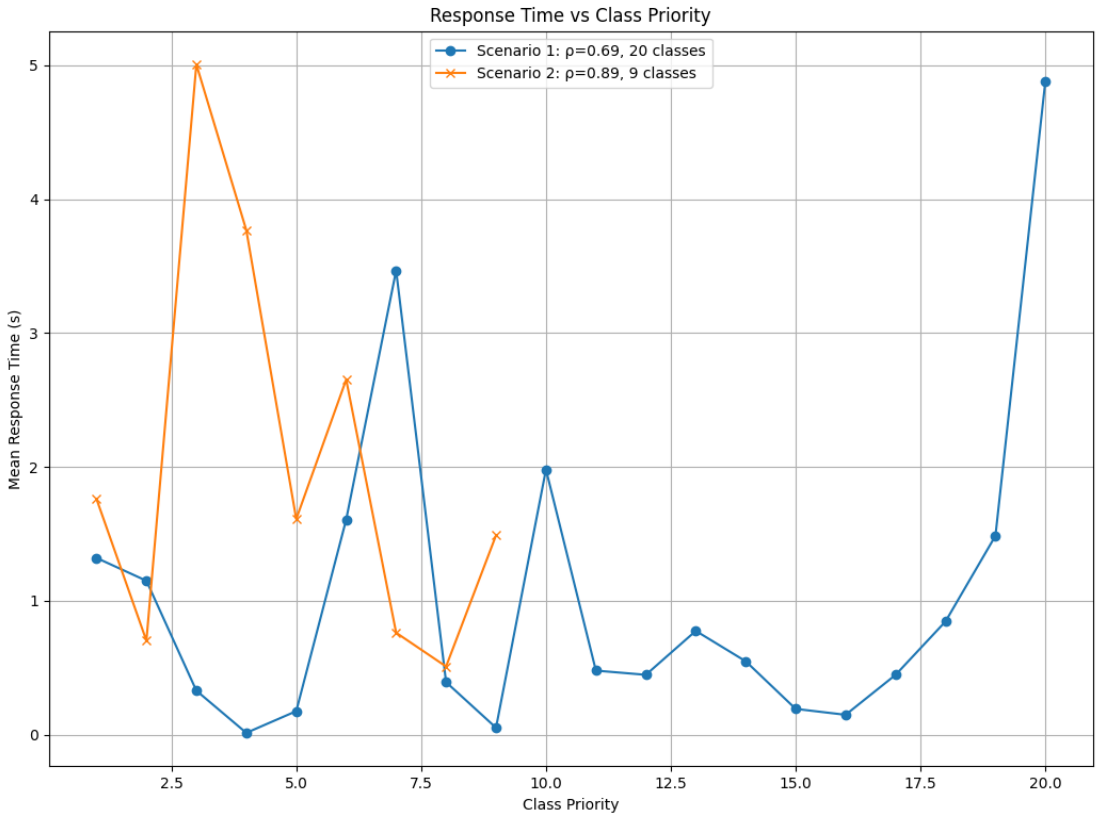


Figure 8: Response time for various classes of messages assigned with different priority parameters.

In the plot, we see two curves representing the mean response time for two different scenarios. The x-axis shows the class priority, which means that the classes are arranged in ascending order of priority from left to right. The y-axis shows the mean response time for each class.

The curve for Scenario 1 (ρ=0.69, 20 classes) has a general upward trend as we move from left to right, indicating that the mean response time increases as the class priority increases. However, there is a lot of variation in the curve, with some classes having much higher response times than others. This variation could be due to the randomness introduced in the simulation or due to differences in the arrival and service rates of the classes.

The curve for Scenario 2 (ρ=0.89, 9 classes) also has a general upward trend, but the variation is much smaller than in Scenario 1. This could be due to the smaller number of classes or the higher utilization factor (ρ) of the system, which means that the system is operating closer to its capacity and there is less room for variation.

Discussing Results

In analyzing the results of our series of experiments, the performance and data gathered from curve 1, 3, and 4 stand out as highly successful. These curves yielded consistent results, validating our initial hypotheses and demonstrating robustness in our methods.

However, The variation in the curves may be due to random fluctuations or errors introduced by the simulation or measurement process.

On the other hand, the outcomes of Experiment 2 were not as close to our expectations. The results deviated from our hypothesis, which prompts further investigation. This deviation could potentially be due to a number of factors, such as experimental error. Future work will focus on scrutinizing and understanding the discrepancies in the results of curve 2.

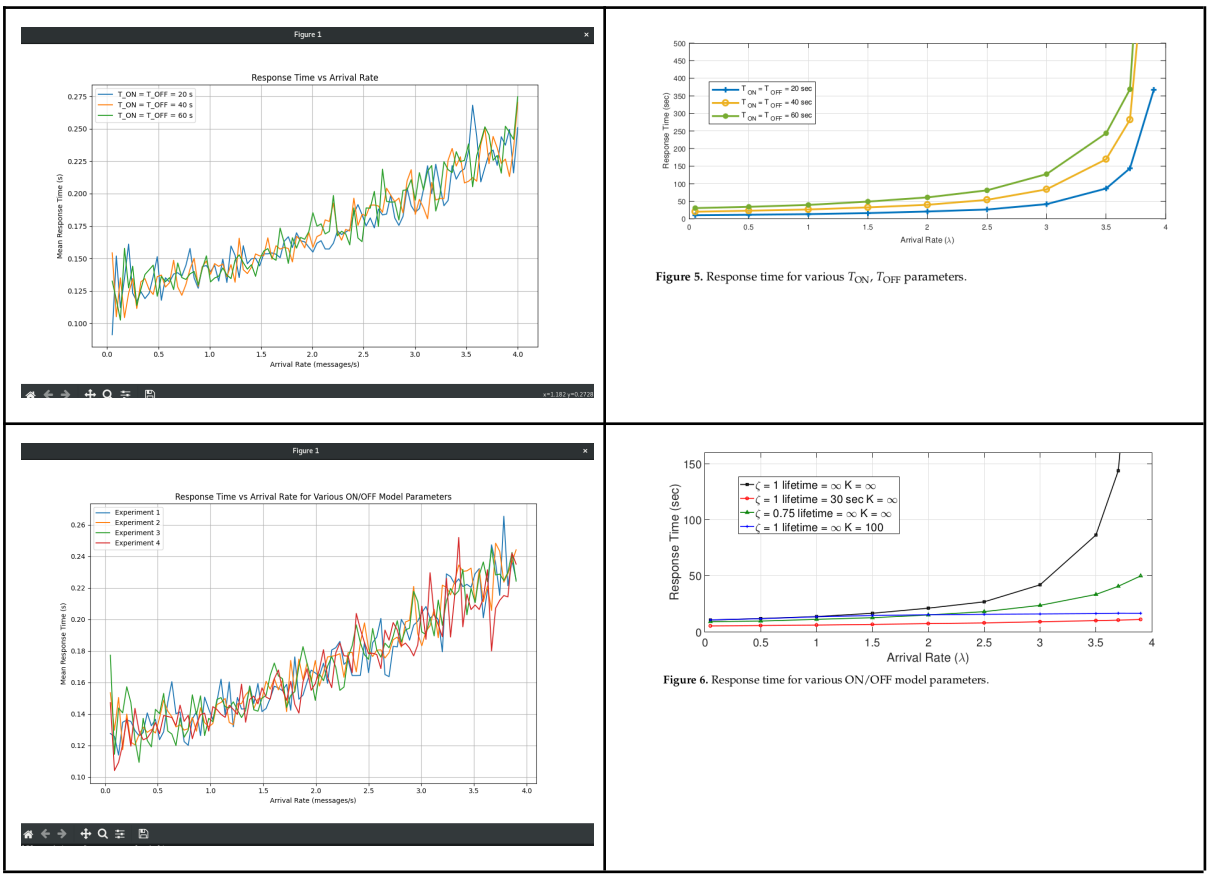


Figure 5. Response time for various T_{ON}, T_{OFF} parameters.

Figure 6. Response time for various ON/OFF model parameters.

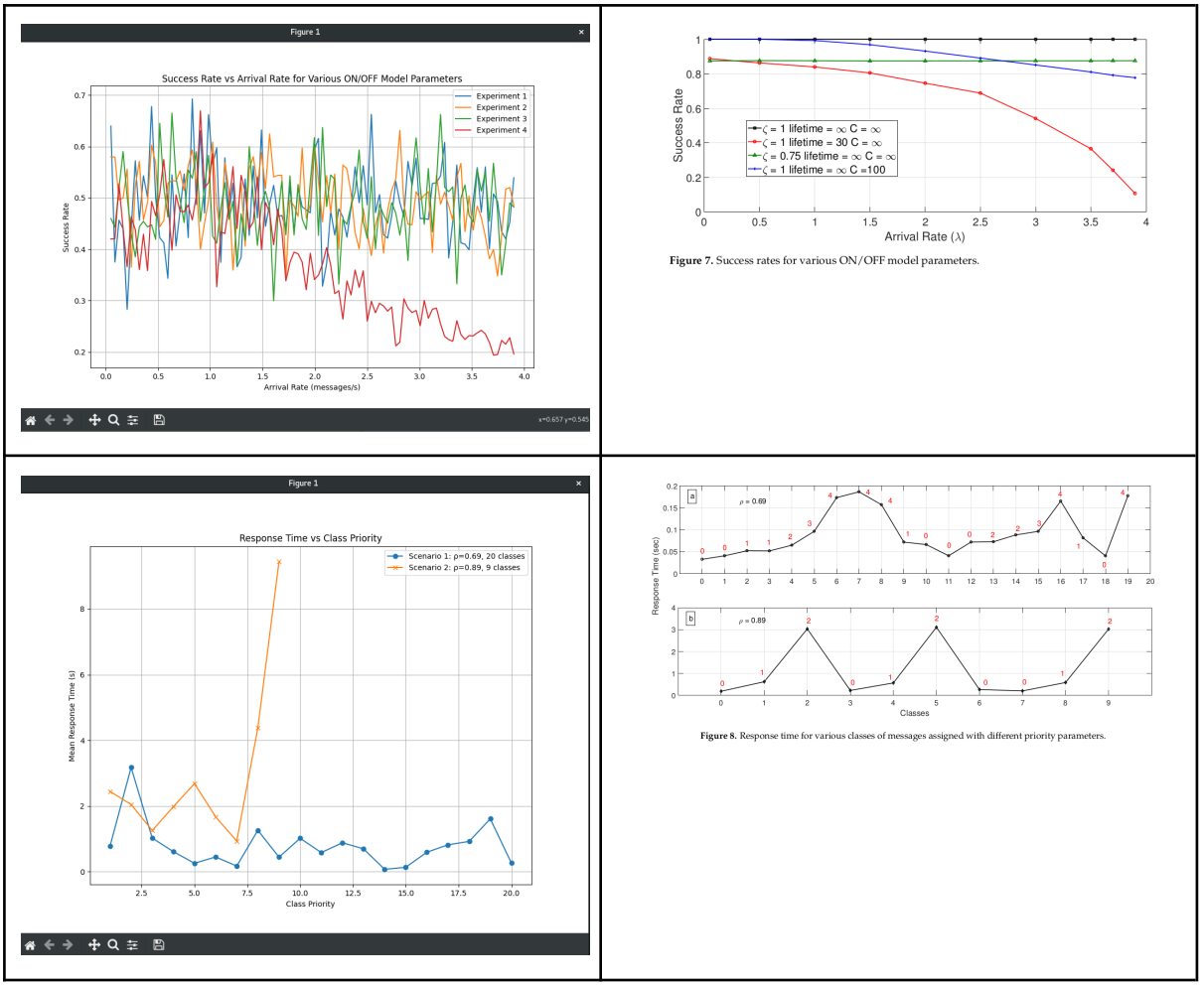


Figure 7. Success rates for various ON/OFF model parameters.

Figure 8. Response time for various classes of messages assigned with different priority parameters.

Conclusion

Trying to model the performance of IoT applications can be a real pain. IoT messaging might result in lost or delayed messages due to different Quality of Service (QoS) settings. These could be things like devices with limited resources, the availability of messages, or inconsistent connections. In this paper, we've gone over simulation-based queueing models that represent these kinds of QoS settings. We've looked at some of these models in previous papers, where they've been validated and evaluated in real-world settings and using actual data. With the simulator we've built, we can check the balance between how often messages are successful and how quick the response times are for various queueing models. This could help people designing systems to create queueing networks that represent IoT interactions and tweak their performance.

In the future, we're planning to look at analytical models for queueing networks. These will include things like finite capacity buffers, lifetime periods, losses, ON/OFF probability, and prioritizing messages (using the chance of dropping them).

Acknowledgements

We would like to extend our gratitude to Dr Meriam Afif, who provided the initial idea that sparked this research project. Her contribution served as the bedrock upon which we built our exploration into the intricacies of IoT and queueing theory.

A significant portion of our inspiration came from the groundbreaking work authored by Georgios Bouloukakis. His meticulous research and innovative approach to the field have been a guiding light for us, and we wish to express our sincere appreciation for his seminal contributions to this domain.

References

1. Community Seismic Network. Available Online: http://www.communityseismicnetwork.org (accessed on 1 May 2015).

2. Cochran, E.; Lawrence, J.; Christensen, C.; Chung, A. A novel strong-motion seismic network for community participation in earthquake monitoring. IEEE Instrum. Meas. Mag. 2009, 12, 8–15. [CrossRef]

3. De Caro, N.; Colitti, W.; Steenhaut, K.; Mangino, G.; Reali, G. Comparison of two lightweight protocols for smartphone-based sensing. In Proceedings of the IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT), Namur, Belgium, 21 November 2013.

4. Lee, S.; Kim, H.; Hong, D.k.; Ju, H. Correlation analysis of MQTT loss and delay according to QoS level. In Proceedings of the

International Conference on Information Networking (ICOIN), Bangkok, Thailand, 28–30 January 2013.

5. Mehmeti, F.; Spyropoulos, T. Performance analysis of “on-the-spot” mobile data offloading. In Proceedings of the 2013 IEEE

Global Communications Conference (GLOBECOM), Atlanta, GA, USA, 9–13 December 2013.

6. Lee, K.; Lee, J.; Yi, Y.; Rhee, I.; Chong, S. Mobile Data Offloading: How Much Can WiFi Deliver? Available online: https://conferences.sigcomm.org/co-next/2010/CoNEXT_papers/26-Lee.pdf (accessed on 27 March 2021).

7. Wu, H.; Wolter, K. Tradeoff analysis for mobile cloud offloading based on an additive energy-performance metric. In Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools, Bratislava, Slovakia, 9–11 December 2014.

8. Gross, D.; Shortle, J.; Thompson, J.; Harris, C. Fundamentals of Queueing Theory; Wiley: Hoboken, NJ, USA, 2008.

9. Durkop, L.; Czybik, B.; Jasperneite, J. Performance evaluation of M2M protocols over cellular networks in a lab environment. In Proceedings of the 18th International Conference on Intelligence in Next Generation Networks (ICIN), Paris, France, 17–19 February 2015.

10. Fysarakis, K.; Askoxylakis, I.; Soultatos, O.; Papaefstathiou, I.; Manifavas, C.; Katos, V. Which IoT protocol? Comparing standardized approaches over a common M2M application. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016.

11. Aldred, L.; van der Aalst, W.M.; Dumas, M.; ter Hofstede, A.H. On the notion of coupling in communication middleware. In Proceedings of the OTM Confederated International Conferences on the Move to Meaningful Internet Systems, Agia Napa, Cyprus, 31 October–4 November 2005.

12. Kattepur, A.; Georgantas, N.; Bouloukakis, G.; Issarny, V. Analysis of timing constraints in heterogeneous middleware interactions. In Proceedings of the International Conference on Service-Oriented Computing, Goa, India, 16–19 November 2015.

13. He, F.; Baresi, L.; Ghezzi, C.; Spoletini, P. Formal analysis of publish-subscribe systems by probabilistic timed automata. In Proceedings of the International Conference on Formal Techniques for Networked and Distributed Systems, Tallinn, Estonia, 27–29 June 2007.

14. Lazowska, E.D.; Zahorjan, J.; Graham, G.S.; Sevcik, K.C. Quantitative System Performance: Computer System Analysis Using Queueing Network Models; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1984.

15. Kounev, S.; Sachs, K.; Bacon, J.; Buchmann, A. A methodology for performance modeling of distributed event-based systems. In Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), Orlando, FL, USA, 5–7 May 2008.

16. Bouloukakis, G.; Georgantas, N.; Kattepur, A.; Issarny, V. Timeliness Evaluation of Intermittent Mobile Connectivity over Pub/Sub Systems. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, L'Aquila, Italy, 22–26 April 2017.

17. Bouloukakis, G.; Moscholios, I.; Georgantas, N.; Issarny, V. Performance Modeling of the Middleware Overlay Infrastructure of Mobile Things. In Proceedings of the IEEE International Conference on Communications, Paris, France, 21–25 May 2017.

18. Bouloukakis, G.; Agarwal, R.; Georgantas, N.; Pathak, A.; Issarny, V. Leveraging cdr datasets for context-rich performance modeling of large-scale mobile pub/sub systems. In Proceedings of the IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Abu Dhabi, United Arab Emirates, 19–21 October 2015.

19. Bouloukakis, G. Enabling Emergent Mobile Systems in the IoT: from Middleware-layer Communication Interoperability to Associated QoS Analysis. Ph.D. Thesis, Inria Paris, Paris, France, August 2017.

20. Bajaj, G.; Bouloukakis, G.; Pathak, A.; Singh, P.; Georgantas, N.; Issarny, V. Toward enabling convenient urban transit through mobile crowdsensing. In Proceedings of the IEEE 18th International Conference on Intelligent Transportation Systems (ITSC), Gran Canaria, Spain, 15–18 September 2015.

21. Gomes, R.; Bouloukakis, G.; Costa, F.; Georgantas, N.; da Rocha, R. Qos-aware resource allocation for mobile iot pub/sub systems. In Proceedings of the 8th International Conference on Internet of Things, Santa Barbara, CA, USA, 15–18 October 2018.

22. Benson, K.; Bouloukakis, G.; Grant, C.; Issarny, V.; Mehrotra, S.; Moscholios, I.; Venkatasubramanian, N. Firedex: A prioritized iot data exchange middleware for emergency response. In Proceedings of the 19th International Middleware Conference, Rennes, France, 10–14 December 2018.

23. Bouloukakis, G.; Benson, K.; Scalzotto, L.; Bellavista, P.; Grant, C.; Issarny, V.; Mehrotra, S.; Moscholios, I.; Venkatasubramanian, N. PrioDeX: A Data Exchange Middleware for Efficient Event Prioritization in SDN-based IoT Systems. Available online: https://hal.archives-ouvertes.fr/hal-03171358/ (accessed on 27 March 2021).

24. Bouloukakis, G.; Moscholios, I.; Georgantas, N.; Issarny, V. Simulation-based queueing models for performance analysis of IoT applications. In Proceedings of the 2018 11th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), Budapest, Hungary, 18–20 July 2018.

25. Karagiannis, V.; Chatzimisios, P.; Vazquez-Gallego, F.; Alonso-Zarate, J. A survey on application layer protocols for the internet of things. Trans. Iot Cloud Comput. 2015, 3, 11–17.

26. Vernon, M.; Zahorjan, J.; Lazowska, E.D. A Comparison of Performance Petri Nets and Queuing Network Models; Technical Report; University of Wisconsin-Madison: Madison, WI, USA, 1986.

27. Mehmeti, F.; Spyropoulos, T. Performance analysis of mobile data offloading in heterogeneous networks. IEEE Trans. Mob. Comput. 2016, 16, 482–497. [CrossRef]

28. Lee, K.; Lee, J.; Yi, Y.; Rhee, I.; Chong, S. Mobile Data Offloading: How Much Can WiFi deliver? IEEE/ACM Trans. Netw. 2012, 21, 536–550. [CrossRef]

29. Phung-Duc, T.; Masuyama, H.; Kasahara, S.; Takahashi, Y. A simple algorithm for the rate matrices of level-dependent QBD processes. In Proceedings of the 5th International Conference on Queueing Theory and Network Applications, Beijing, China, 1 June 2010.

30. John, V.; Liu, X. A Survey of Distributed Message Broker Queues. Available online: https://arxiv.org/pdf/1704.00411.pdf (accessed on 27 March 2021).

31. Gian, P.C.; Costa, P.; Picco, G.P. Publish-Subscribe on Sensor Networks: A Semi-probabilistic Approach. In Proceedings of the 2nd IEEE International Conference on Mobile Adhoc and Sensor Systems, Washington, DC, USA, 7–10 November 2005.

32. Diallo, M.; Fdida, S.; Sourlas, V.; Flegkas, P.; Tassiulas, L. Leveraging caching for Internet-scale content-based publish/subscribe networks. In Proceedings of the 2011 IEEE International Conference on Communications (ICC), Kyoto, Japan, 5–9 June 2011.

33. Pripuzic, K.; Zarko, I.P.; Aberer, K. Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w. In Proceedings of the second international conference on Distributed event-based systems, Rome, Italy, 1–4 July 2008.

34. Salehi, P.; Zhang, K.; Jacobsen, H.A. Popsub: Improving resource utilization in distributed content-based publish/subscribe systems. In Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems, Barcelona, Spain, 19–23 June 2017.

35. Chakravarthy, S.; Vontella, N. A publish/subscribe based architecture of an alert server to support prioritized and persistent alerts. In Proceedings of the International Conference on Distributed Computing and Internet Technology, Bhubaneswar, India, 22–24 December 2004.

36. Maheshwari, P.; Tang, H.; Liang, R. Enhancing web services with message-oriented middleware. In Proceedings of the IEEE International Conference on Web Services 2004, San Diego, CA, USA, 6–9 June 2004.

37. Zhang, R.; Lu, C.; Abdelzaher, T.F.; Stankovic, J.A. Controlware: A middleware architecture for feedback control of software performance. In Proceedings of the 22nd International Conference on Distributed Computing Systems, Vienna, Austria, 2–5 July 2002.

38. Saghian, M.; Ravanmehr, R. Publish/subscribe middleware for resource discovery in MANET. In Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, China, 4–7 May 2015.

39. Wang, Y.; Zhang, Y.; Chen, J. Pursuing differentiated services in a sdn-based iot-oriented pub/sub system. In Proceedings of the 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA, 25–30 June 2017.

40. Baskett, F.; Chandy, K.M.; Muntz, R.R.; Palacios, F.G. Open, closed, and mixed networks of queues with different classes of customers. J. ACM (JACM) 1975, 22, 248–260. [CrossRef]

41. Bouloukakis, G.; Moscholios, I.; Georgantas, N. Probabilistic Event Dropping for Intermittently Connected Subscribers over Pub/Sub Systems. In Proceedings of the IEEE International Conference on Communications (ICC), Shanghai, China, 21–23 May 2019.

42. Montazer-Haghighi, A.; Medhi, J.; Mohanty, S.G. On a multiserver Markovian queueing system with balking and reneging. Comput. Oper. Res. 1986, 13, 421–425. [CrossRef]

43. Abou-El-Ata, M.; Hariri, A. The M/M/c/N queue with balking and reneging. Comput. Oper. Res. 1992, 19, 713–716. [CrossRef]

44. Yue, D.; Zhang, Y.; Yue, W. Optimal performance analysis of an M/M/1/N queue system with balking, reneging and server vacation. Int. J. Pure Appl. Math. 2006, 28, 101–115.

45. Field, T. JINQS: An Extensible Library for Simulating Multiclass Queueing Networks, v1.0 User Guide. Available online: http://www.doc.ic.ac.uk/~ajf/Software/manual.pdf (accessed on 27 March 2021).