Received: 8 June 2020 | Revised: 24 October 2020 | Accepted: 24 November 2020
DOI: 10.1002/dac.4709

RESEARCH ARTICLE

WILEY

Using priority queuing for congestion control in IoT-based technologies for IoT applications

Stephen S. Oyewobi[1] | **Karim Djouani**[1,2] | **Anish Matthew Kurien**[1]

[1]FSATI/Tshwane University of Technology, Pretoria, South Africa
[2]University Paris-Est Creteil (UPEC)/LSSI Lab, Creteil, France

Correspondence
Stephen S. Oyewobi, FSATI/Tshwane University of Technology, Pretoria 0001, South Africa.
Email: oyewobistephen@gmail.com

Funding information
National Research Foundation of South Africa, Grant/Award Number: 90604

Summary
The Internet of Things (IoT) connect millions of devices in diverse areas such as smart cities, e-health, transportation and defense to meet a wide range of human needs. To provide these services, a large amount of data needs to be transmitted to the IoT network servers. However, the IoT networks suffer from limited resources such as buffer size, node processing capabilities, and server capacities adversely affecting throughputs, latency, and energy consumption. Additionally, the ensuing heavy network traffic due to large amount of data transmitted results in congestion which degrades IoT network performance. Therefore, innovative congestion control techniques, e.g., queue management approach needs to be developed to overcome congestion problems in IoT networks. In this paper, a novel priority queuing technique (*Npqt++*) is developed to control congestion in IoT networks. The *Npqt++* implements a preemptive/nonpreemptive discipline with a discretion rule to classify network traffic based on their real-time requirement into priority groups. If the discretion rule for low priority packets is satisfied, high priority packets are pushed to the front of the queue; otherwise, they wait in the queue. Our approach significantly outperforms existing techniques in terms of throughput, delay, and energy consumption.

KEYWORDS
congestion control, Internet of Things, IoT applications, preemptive/nonpreemptive priority, queuing

1 | INTRODUCTION

The Internet of Things (IoT) is the connection and exchange of data between sensors/devices.[1–3] The vision of IoT is to make every single device in the globe a part of the internet such that its position and status can be uniquely identified and accessible to the network.[4] The exponential growth in the number of devices being deployed as part of IoT has driven the application of IoT in many fields such as smart metering, homes, and cities, agriculture, asset tracking, transport, and defense.[3] However, existing technologies have not been adapted for the deployment and unique requirements of IoT such as long-range communications, low data rate, and low energy consumption as well as cost-effectiveness. For example, ZigBee and Bluetooth are not designed to handle long-range communication whereas cellular communications suffer from high power consumption, high deployment cost, and high complexity. Similarly, Wi-Fi does not support the massive deployment of sensors with a minimum power consumption over an extended range. For all of those reasons, the low power wide area network (LPWAN), a new paradigm in communication, was designed to fill the gap between cellular and short-range wireless technologies to address the diversity and support the deployment of IoT.[3,5] For instance, LPWAN defines a set of unique features that are well-matched for IoT specific requirements and deployment such as extended range and massive scale connectivity for low power, low data rates, and low-cost machine-to-machine (M2M) communication. Additionally, the rise of LPWAN has made the vision and future IoT application scenarios very reachable.[5] Conversely, LPWAN uses a single-hop star topology to connect devices to the base station using ALOHA-based MAC protocols.[6] Generally, random access MAC, e.g., ALOHA-based MAC protocol, is uncontrolled leading to packet collision resulting in reliability and scalability problems in dense networks like IoT.[7] Similarly, technology advances that implement internet protocol stack (e.g., IPv6) that integrate the "Things" to the internet also suffer challenges such as bandwidth and energy limitation as well as limited buffer resources.[8] The contributions in this paper attempt to address these shortcomings in the IoT-based technologies to alleviate packet collision within the finite buffer space of the nodes in IoT network. Therefore, we have proposed a novel priority queuing technique for congestion control in IoT-based technologies for IoT applications using preemptive/nonpreemptive discipline (*Npqt*++). Hence, the main contributions of this paper can be summarized as follows:

1. We implement a preemptive and nonpreemptive priority queuing model for the packets of each priority group to differentiate the traffic type to prevent packet collision and congestion.
2. We implement a preemptive and nonpreemptive priority queuing model to delineate the periods when packet being served out of the buffer can be interrupted or not.
3. We implement a discretion rule for preemption/nonpreemption to prevent the nodes from a selfish behavior when transmitting their packets.
4. We implement the discretion rule for preemption/nonpreemption to protect the low priority packets from interruption by higher priority packets during nonpreemptive periods based on its elapsed service time.

The rest of the paper is organized as follows. In Section 2, we discuss related works in congestion control and priority queuing in IoT network. Section 3 details the network setup and problem formulation. Section 4 presents the proposed preemptive/nonpreemptive priority queuing model for congestion control in IoT network. In Section 5, the analysis of the queuing delay of the proposed method is presented. Furthermore, results are presented and discussed in Section 6. Then, Section 7 concludes our findings in this the paper.

2 | RELATED WORK

Of late IoT has become a leading focus in the research community; therefore, lots of studies to improve various aspects of IoT have been conducted. In fact, several literatures have pointed to congestion control as a foremost subject in IoT network. In Al-Kashoash et al.,[9] a congestion control technique for IoT paradigm also known as packet discarding-based node clustering (PDNC) was developed. In this method, all the nodes deployed in a particular area of interest are clustered into several groups, and a cluster head is selected for each group. Then, the PDNC is implemented at each node to reduce the number of packets contributing to congestion. Their results suggest that the proposed mechanism reduced congestion while improving overall performance. By taking congestion over the internet into account, Mishra et al.[10] developed an adaptive congestion control strategy that adjusts transmission rate every time the available bandwidth and delay fluctuates. The proposed technique implements TCP cubic to maintain fairness and steady-state to reduces packet drop. Their experimental results showed significant improvement regarding throughput and inter-protocol fairness for the proposed approach. In Zhou et al.,[11] a proposed improvement over TCP westwood (TCPW) called polling-TCPW which is an adaptive sliding window algorithm was investigated for narrow band-IoT (NB-IoT). The proposed technique is to enhance the status report policy in the RLC protocol stack of the radio link control layer of the NB-IoT to regulate data transmission and to achieve automatic repeat-request retransmission. The polling-TCPW achieved enhanced throughput and reduced transmission delay of RLC with a guaranteed system stability. Sukjaimuk et al.[12] implemented a dynamic congestion control for a hierarchical information-centric network model for IoT sensor network. Similarly, literature abounds with lots of research efforts where queuing models have been deployed to solve congestion problems. For example, in the work, Tabassum et al.[13] did a comparative study of three queuing algorithms comprising first-in first-out (FIFO), priority queuing, and weighted fair queuing. Then, they investigated the quality of service (QoS) effects of their study on the IoT network traffic. They evaluated and discussed the performance of the study based on metrics such as jitters, latency, packet loss over VoIP, and video and FTP traffic. Also, in Huang et al.,[14] an admission control model for M2M communication was developed. This approach first sorts all M2M request into delay-sensitive and delay-tolerant then it routes all delay-tolerant request into a low priority queue. The objective of this method is to minimize the amount of request coming from different devices in the IoT network to the access point, in addition, to avert access collision as well as to improve QoS performance. Additionally, in order to transmit critical data with minimum delay constraint, Ambigavathi and Sridharan[15] deployed an energy efficient and load balancing priority queuing algorithm to classify packets based on the location of the devices generating the packets. This method schedules packets generated from within based on their priority whereas packets generated remotely are scheduled based on their deadline. A hardware scheduler then schedules and transmits data as high, medium, or low priority data. This approach showed better performance in terms of throughput, packet delivery ratio, and power consumption when compared with existing mechanisms. Walraevens et al.[16] analyzed a discrete-time priority queue in respect of the delay experienced in a train-arrival process. They extended a previous study with only two traffic classes where one class has priority over the other to a generalized number of M classes with N arbitrary priority classes where ($1 \leq N \leq M$). They used probability generating functions to compute the moments and tail probabilities of the steady-state packet delays of all traffic classes. Then, they went on to demonstrate the usefulness of partitioning of traffic classes in priority classes for some specific scenarios. Undoubtedly, the motivation in each of the above-mentioned studies/works is that (1) IoT is a platform hosting different types of applications over the internet with each application having a different real-time requirements and (2) that owing to their different real-time requirements, different traffic types should have different priorities. Consequently, these studies have each implemented different priority models/approaches/algorithms that have assigned different priority to the different traffic types to prevent/ameliorate collision/congestion. However, one major shortcoming of the aforementioned reviewed works is that during congestion, each node transmits in a selfish manner without any rule describing the manner in which packets are transmitted. Again, to the best of our knowledge, none of the proposed priority queuing algorithms have implemented a preemptive/nonpreemptive priority queuing discipline to address congestion problems of IoT network. Also, none of the existing algorithms in congestion control for IoT network has defined a discretion rule that describes periods where packet transmission can be interrupted or not.

3 | NETWORK SETUP AND PROBLEM FORMULATION

This section discusses the network setup as well as the problem formulation of the proposed technique.

3.1 | Network setup

The network setup consists of three different types of nodes, namely, the leaf nodes (sensor nodes), intermediate nodes, and the sink nodes. The sink node serves as the gateway between the LPWAN network and IoT-based end nodes. Whereas the intermediate nodes link the sink and the leaf nodes as shown in Figure 1. The network topology is constructed based on the directed acyclic graph (DAG) concept and the IPv6 routing protocol for low-power and lossy (RPL) networks as shown in Figure 2. Each node in the RPL is organized as a destination-oriented directed acyclic graph (DODAG) to form a network topology. To start a network topology, the sink broadcasts routing metrics and constraints through the DODAG information object (DIO) to neighboring nodes according to its objective function (OF). Then, based on their OF and local policy when a node receives a DIO message from neighboring nodes, it constructs a routing topology by selecting a neighboring node with the best rank as its parent. The building of the network topology continues until the DIO message reaches the leaf nodes.[9]

3.2 | Problem formulation

Now, let us consider a network scenario that operates the random access ALOHA-based mechanism comprising 1 sink node S, 8 intermediate nodes I, and N leaf nodes $L_1,...,L_k,....,L_N$ as demonstrated in Figure 2. The link between nodes L_k and I is denoted as $\mathcal{R}_{ki}$, whereas the link between any two intermediate nodes I is denoted as $\mathcal{R}_{ii}$. Also, the link between nodes I and S is denoted as $\mathcal{R}_{is}$. Again, we assume that each link between the nodes has a channel capacity of CC_b bit/sec. However, the intermediate nodes I have a channel capacity of $CC_b/2$ bit/sec since the radios of intermediate nodes I are transmitting and receiving simultaneously. Also, let us assume that a buffer of B packet size is assigned to

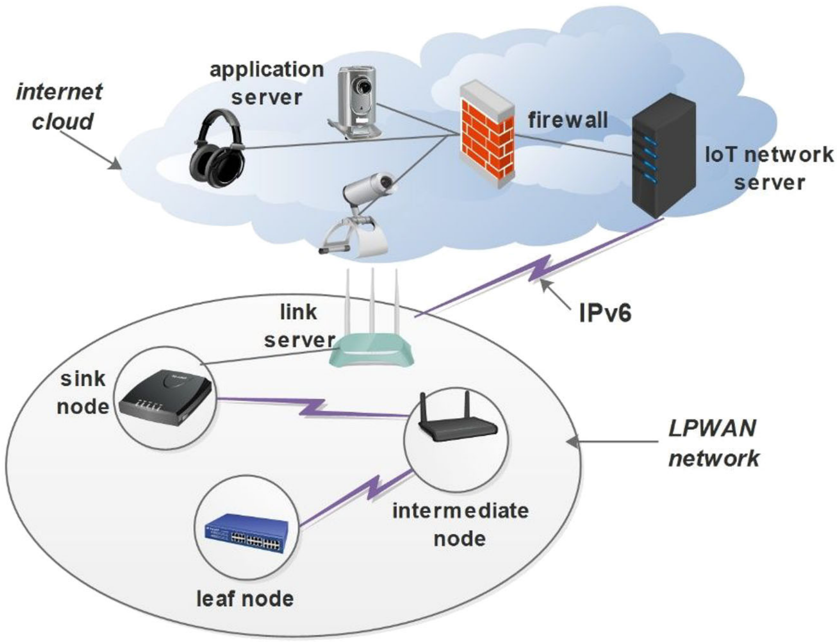

FIGURE 1 Network architecture

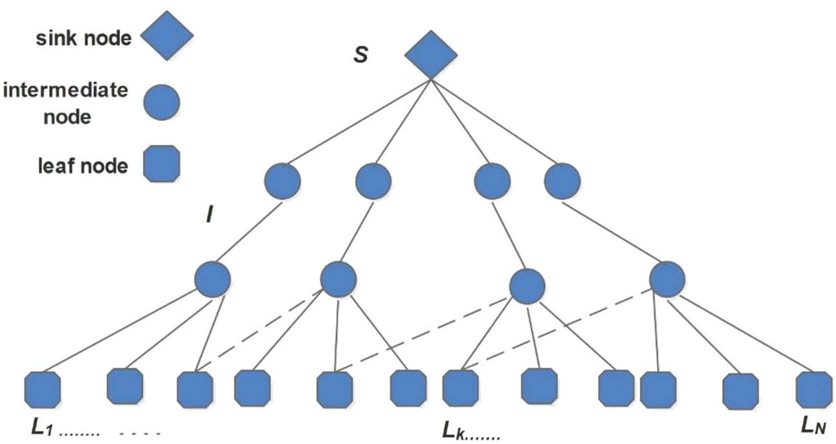

FIGURE 2 RPL-based network topology

each node in the network. Typically, packets are generated at an average data rate say λ_1^L, $\lambda_{k,}^L$, λ_N^L by the applications in leaf nodes L_1,...,L_k,.... ,L_N, and are stored in the MAC buffer. Then, the packets are transmitted by the MAC protocol to the intermediate nodes I at an average departure rate of μ_1^L, $\mu_{k,}^L$, μ_N^L. Largely, several packets are lost on the link before they arrive at the intermediate nodes I with a probability of $P_{ch-loss}^j$ where $j = 1$,.... k,.... N. Therefore, the packet that finally arrives at the intermediate nodes I from leaf nodes L_1,...,L_k,.... ,L_N at an average data rate of λ_1^I, $\lambda_{k,}^I$, λ_N^I is given as[9]

$$\mu_j^I = \left(1 - P_{ch-loss}^j\right)\mu_j^I, \tag{1}$$

where $j = 1$,.... k,.... N. Whereas the total number of packets that arrives at intermediate nodes I is given as

$$\lambda_{total}^I = \sum_{j=1}^{N} \lambda_j^I. \tag{2}$$

Likewise, the intermediate nodes I store the received packets and then transmits these packets to the sink node S with an average departure rate of μ^I. However, a different scenario takes place when congestion occurs. In this case, each application in leaf nodes L_1,...,L_k,.... ,L_N starts to generate packets at a high data rate to the intermediate nodes I without considering the channel capacity, buffer size, and departure rate of the intermediate nodes *I*, as well as the sending rate of other leaf nodes. Therefore, there is stack of buffer overflow, packets collision, and packets loss at the nodes. As a result, most packets are lost due to buffer overflow rather than due to wireless link loss in the RPL networks during congestion. To mitigate such scenarios, novel congestion control techniques should be developed to forestall congestion in IoT-based networks and technologies.

4 | IMPLEMENTING THE PREEMPTIVE/NONPREEMPTIVE PRIORITY QUEUING MODEL FOR CONGESTION CONTROL

In this section, we consider the implementation of a preemptive/nonpreemptive priority queuing model to control congestion at the buffer of the nodes.[17] As a result, the proposed model organizes packets into preemptive and non-preemptive service discipline to be pushed out by link server to prevent buffer overflow at the nodes. In view of this, the nodes are grouped into two groups based on the type of services they execute as (1) nonpreemptive priority nodes (*NPNs*): these are nodes hosting applications with hard or soft real-time requirement. For example, control and factory automation applications with a latency of $0.25 - 10$ ms, as well as safety and alarm systems with a latency of $10 - 100$ ms. These nodes are assigned a high priority in the groups, and their packets cannot be interrupted; (2) preemptive priority nodes (*PPNs*): these are nodes hosting applications with soft or no real-time requirement. For instance, monitoring systems with a latency of $\geq$100 ms. In addition, within the *PPN* group, there exist priority classes based on the time-constraint requirements of the services they execute, and their packets can be interrupted. As a rule, the packets of a *PPN* can be interrupted by packets of a *NPN* or other *PPNs* multiple times before it departs the server. The *Npqt++* is implemented as an M/G/1/B queuing model with a discretion rule (this is discussed in later sections) where *B* is the buffer size. The discretion rule is based on the elapsed service time of the *PPNs*. The discretion rule determines if the packets of a *PPN* can depart the server without any interruption by other priority class *PPNs* or a *NPN* packet. A different FIFO priority queue is implemented for the packets of different priority group. This approach is to differentiate the traffic types and also to prevent head-of-line blocking as shown in Figure 3.

Now, let us consider that Q_1 is the queue for the packets of the *NPNs* whereas Q_2 is the queue for the packets of the *PPNs* with priority $1 \leq g \leq M$ where M is number of priority class of the *PPNs*. Clearly, *PPN* with priority $g = 1$ has the highest priority in its class whereas $g = M$ has the lowest priority. Most of the time, an *NPN* packet can always interrupt a *PPN* packet and go to the front of the queue to be pushed out of by the server. However, if a *PPN* with priority g (PPN_g) has packets leaving the server and the packet of a higher priority *PPN* say $g = 1$ (PPN_1) arrives at the queue, then the discretion rule has to apply. Therefore, the *Npqt++* first check if the preemptive discretion rule of PPN_g is satisfied. If the preemptive rule is satisfied, the PPN_g packet transmission is interrupted and the PPN_1 packet goes to the front of the line to be pushed out by the server. Nonetheless, the PPN_g has two options if the packet arrival rate of a parent node (intermediate node) is greater than the packet departure rate: (1) it can choose to pause its transmission while the packets wait at the buffer until PPN_1 packets are completely pushed out of the server or (2) it can select another neighboring node as its parent node and resume its packet transmission as shown by the dash lines in Figure 2. However, if PPN_g decides to pause its transmission the packets are pushed to the head of the queue Q_2 and transmitted out of the server once PPN_1 transmission is completed. Else, the packets go to the tail of the queue Q_2. Conversely, if the preemptive rule is not satisfied, PPN_g packet is not interrupted and they are completely pushed out of the server while PPN_1 packets wait in the queue.

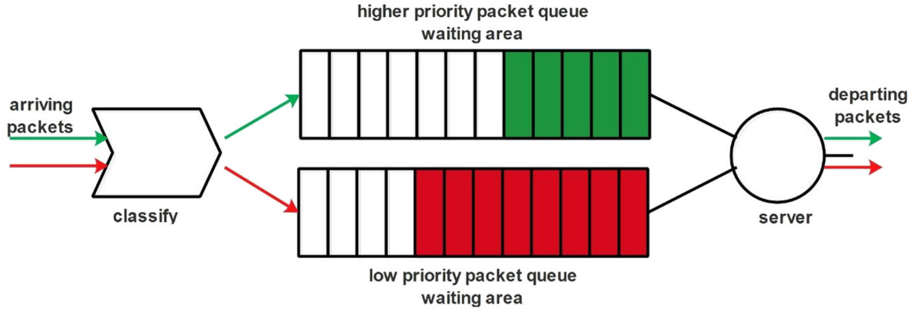

FIGURE 3 Priority queuing model

5 | QUEUING DELAY ANALYSIS

In this section, we define the variables, parameters, and concepts used in the analysis of the preemptive and nonpreemptive priority queuing model.

5.1 | Discretion rule

The discretion rule is defined as the function of the elapsed service time of the PPN_g. The discretion rule is satisfied if the elapsed service time of PPN_g is less than a predefined threshold ϕ_g. If this happens, a *PPN* with higher priority, e.g., PPN_1 packets can interrupt the packets of a PPN_g. Then, PPN_1 packets are pushed to the front of the queue to be transmitted out of by the server. Else, PPN_1 packets will wait in the queue until the PPN_g packets are completely serviced. Clearly, the service time of a PPN_g can be considered as the sum of preemptive and nonpreemptive periods by a high priority *PPNs* as[17]

$$S_g = S_{A_g} + S_{B_g}, \tag{3}$$

where S_{A_g} is the preemptive period of the PPN_g (i.e., period when PPN_g packet transmission can be interrupted) by higher *PPNs* is given as

$$S_{A_g} = min\left\{S_g, \phi_g\right\}, \tag{4}$$

and S_{B_g} is the nonpreemptive period (i.e., period when PPN_g packet transmission cannot be interrupted) by higher *PPNs* is presented as follows:

$$S_{\mathrm{B}_g} = max\left\{0, S_g - \phi_g\right\}. \tag{5}$$

Similarly, the service time of a *NPN* is given as

$$S_{npp} = S_{B_{npp}}, \tag{6}$$

where $S_{B_{npp}}$ is the nonpreemptive period of the *NPN* clearly the preemptive period $S_{A_{npp}} = 0$, this is because S_{npp} is nonpreemptive due to the high priority assigned to *NPN*.

5.2 | Variables and concepts

The concept of the *delay cycle* and Laplace transform is used to analyze the queuing delay of the *PPNs* packets. The *delay cycle* of a *PPN* packet can be divided into two parts: (1) *initial delay*, which is the time it takes to service the initial packets out of the server and (2) *delay busy periods*, which is the time spent to service high priority packets out of the server before a *PPN* packet is considered. Hence, based on the impact a high priority packet may have on the queuing time of a PPN_g packet, the nodes are further classified into three classes: type $-\alpha$, type –g, and type $-\beta$. Henceforth, type $-\alpha$ nodes include all *NPN* and *PPNs* with a high priority $(g - 1)$ than g. Whereas type $-\beta$ nodes comprise *PPNs* with a low priority $g+1$ through *M*. Then, type –g nodes are all *PPN* with priority g. Based on this, three types of delay cycle including type $-\alpha$ delay cycle, type –g delay cycle, and type $-\beta$ delay cycle are considered in analyzing the queuing time of a PPN_g. If we assume that packets arrival follow a Poisson distribution then a type $-\alpha$ delay cycle starts with the arrival of a *NPN* packet or a type $-\alpha$ packet at the server. Whereas a type –g delay cycle starts with arrival of type –g packet at the server. Lastly, a type $-\beta$ delay cycle starts with the arrival of type $-\beta$ packet at the server. Typically, a delay cycle ends when the packet that initiates the delay departs the server and the server is empty of type $-\alpha$ and type –g packets. In general, a typical delay busy period can be considered as a series of mutually exclusive delay cycles (i.e., either type $-\alpha$ delay cycle, or type –g delay cycle or probably numerous type $-\beta$ delay cycles).

5.3 | Occupancy time and completion time

The occupancy time R_g of a PPN_g packet can be considered as the sum of N_g interruptions (breakdowns) plus preempted services times S_{pg} as well as one successful service time, S_{sg} as[17]

$$R_g = \sum_{n=1}^{N_g} \left(D_g + S_{pg}\right)^n + S_{sg}. \tag{7}$$

However, during the nonpreemptive period S_{B_g} of the successful service time S_{sg} of a PPN_g packet, there may be packets of high priority nodes waiting in the queue. Accordingly, each of these packets should be served before the next PPN_g packet is served. Therefore, the completion time C_g of a PPN_g packet consists of the occupancy time R_g plus a delay busy period Y_g initiated by the high priority packets waiting in the queue during S_{B_g}. Note that Y_g is the combination of the separate breakdown time D_g generated by the high priority packets. Additionally, the length of the breakdown time D_g is identically distributed for each interruption. As a result, the busy period elapsed from the moment a PPN_g packet arrives at the server until the instant the server is emptied of any PPN_g packets and high priority packets can be denoted as B_g. Hence, the Laplace transform of B_g can be represented as[17,18]

$$B_g^*(s) = C_g^*\left(s + \lambda_g - \lambda_g B_g^*(s)\right). \tag{8}$$

The length of the breakdown time D_g initiated by a type $-\alpha$ packet (in this case, *PPNs* with priority $g-1$) is equivalent to B_{g-1}. However, if the breakdown time is generated by a *NPN* packet (i.e., nodes with nonpreemptive priority), then an initial delay cycle D_{g-1} is initiated. In addition, during the initial delay cycle D_{g-1}, each type $-\alpha$ packet waiting in the server generates a subbusy period of B_{g-1}. Nonetheless, each breakdown time happens with probability of $\lambda_{g-1}/\Lambda_{g-1}$ and $\Lambda_{g-1} - \lambda_{g-1}/\Lambda_{g-1}$ respectively. Therefore, the Laplace transform of D_g in a recursive form can be represented as[17,18]

$$D_g^*(\mathrm{s}) = \frac{\lambda_{g-1}}{\Lambda_{g-1}} B_{g-1}^*(s) + \frac{\Lambda_{g-1} - \lambda_{g-1}}{\Lambda_{g-1}} D_{g-1}^*\left(s + \lambda_{g-1} - \lambda_{g-1} B_{g-1}^*(s)\right), \tag{9}$$

where $g \geq 2$, $D_1^*(\mathrm{s}) = 1$, and $\Lambda_g = \sum_1^g \lambda_g$. Following this, the number of interruptions that a PPN_g packet encounters before it completely departs the server is equal to the number of high priority packets arriving during S_{A_g}. For the most part, the conditional probability that n interruptions will occur is obtained as follows[19]:

$$P\left[N_g = n | S_g\right] = \frac{\left(\Lambda_{g-1} S_{A_g}\right)^n}{n!} e^{-\Lambda_{g-1} S_{Ag}}. \tag{10}$$

Accordingly, the completion time can be considered as a delay cycle plus an initial delay of S_g during which high priority packets waiting in the queue generates a subbusy period of D_g. So then, the Laplace transform of the completion time can be expressed as follows[17]:

$$C_g^*(s) = S\left(s + \Lambda_{g-1} - \Lambda_{g-1} D_g^*(s)\right). \tag{11}$$

5.4 | Analysis of the queuing time

The steady-state probability π_g that a PPN_g packet will arrive at the server which is in a state j where $j \in \{0, \alpha, \mathrm{g}, \beta\}$ (using the earlier described delay cycles and the assumption that packet arrival follows a Poisson) can be denoted as[17,18]

$$\pi_0 = 1-\rho,\ \ \pi_\alpha = \rho_\alpha(1-\rho)/\left(1-\rho_g-\rho_\alpha\right),\ \ \pi_g = \rho_g(1-\rho)/1-\rho_g-\rho_\alpha),\ \ \pi_{\beta=}\rho_\beta/\left(1-\rho_g-\rho_\alpha\right). \tag{12}$$

The utilization factor of the *PPNs* $\rho = \lambda_g E[S_{eg}]$ where S_{eg} is the effective service time of PPN_g packets. Also, note that we assume that the server is in state 0 when the server is empty. Hence, the Laplace transform of the queuing time can be given as[17]

$$W_g^*(s) = \pi_0 + \pi_\alpha W_{g/\alpha}^*(s) + \pi_g W_{g/g}^*(s) + \pi_\beta W_{g/\beta}^*(s) + W_{g/j}^*(s), \tag{13}$$

where $W_{g/j}^*\ j \in \{\alpha, g, \beta\}$ is the conditional waiting time of a PPN_g packet when it arrives at the server which is in state *j*. Indeed, $W_{g/j}^*\ j \in \{\alpha, g, \beta\}$ can be regarded as the waiting time of a PPN_g packet which arrives the server in a delay cycle *j* (i.e., a type $-\alpha$ delay cycle or type –*g* delay cycle or several type $-\beta$ delay cycles). Therefore, if $\psi_{g/j}$ represents the initial delay whereas C_g denotes the service time of the type *j* delay cycle where $j \in \{\alpha, g, \beta\}$ then the conditional waiting time $W_{g/j}^*(s)$ can be obtained as[17,19]

$$W_{g/j}^*(s) = \frac{\left(1-\lambda_g E\left[C_g\right]\right)\left(1-\psi_{g/j}^*(s)\right)}{E\left[\psi_{g/j}\right]\left(s-\lambda_g+\lambda_g C_g^*(s)\right)}. \tag{14}$$

The initial delay of the types $-\alpha$ and –*g* delay cycles can be obtained respectively as

$$\psi_{g/\alpha}^*(s) = D_g^*(s), \tag{15}$$

$$\psi_{g/g}^*(s) = C_g^*(s). \tag{16}$$

So to obtain the conditional waiting time $W_{g/\alpha}^*(s)$ and $W_{g/g}^*(s)$ substitute 15 and 16 into 14 respectively. After this, only the conditional waiting time $W_{g/\beta}^*(s)$ needs to be estimated to obtain the queuing time in 13. Hence, to obtain $W_{g/\beta}^*(s)$, we consider the type $-\beta$ delay cycle. In view of this, $W_{g/\beta}^*(s)$ can be considered as a type $-\beta$ busy cycle initiated by the arrival of a type h packet where $h \in \{\beta\}$. Of course, this type $-\beta$ busy cycle ends when the type h packet departs the server and the server is not in a type –*α* or type –*g* delay cycle. Therefore, $W_{g/\beta}^*(s)$ can be considered as the breakdown time initiated by the packets of a type *β* node with a priority *g*+1 through $h-1$.

5.5 | Estimating the expected queuing delay if PPN_g chooses another parent node

The PPN_g has two options if the packet arrival rate of a parent node (intermediate node) is greater than the packet departure rate. The PPN_g can decide to select a new intermediate node as its parent node. In this case, the PPN_g packet will arrive at the server when the server is either in an empty state or a busy period. Take for example, if the packet arrives at time t_m when the server is in a type $-\alpha$ delay cycle, it will wait in the queue until the entire type $-\alpha$ packets are delivered out of the queue. Therefore, the expected queuing delay of the PPN_g packet is equivalent to the queuing time given in 13 given a steady-state probability π_g also assuming that the server is in a state *j* when the packets arrive where $j \in \{0, \alpha, g, \beta\}$.

5.6 | Estimating the expected queuing delay if PPN_g stays with parent node

On the other hand, if the PPN_g decides to stay with its parent node PPN_g packet will be served immediately the server is emptied of the type $-\alpha$ packets. Therefore, the expected queuing delay of the PPN_g in this case consists of only a type $-\alpha$ delay cycle. Hence, the Laplace transform of the expected queuing delay can be obtained as[17]

$$W_g^{!*}(s) = W_{g-1/\alpha}^*(s). \tag{17}$$

6 | ANALYSIS OF THE RESPONSE TIME

The total time expected to push a PPN_g packet that encounters n interruptions out of the server consists of the queuing delay caused by the n interruptions and the PPN_g packet service time. Therefore, assuming a Poisson arrival, the response time T_g consists of two independent random variables, the queuing time W_g and the occupancy time R_g. Thus, the Laplace transform of response time can be obtained as[17]:

$$T_g^* = W_g^*(s)R_g^*(s). \tag{18}$$

7 | RESULTS AND DISCUSSION

To analyze and evaluate the performance of the *Npqt++*, we set up the simulation scenario as illustrated in the network topology in Figure 2. The leaf nodes serve as the sensor nodes and the simulation is set to 800 s. However, to give time for the network topology to be fully constructed as shown in Figure 2, the sensor nodes only starts sending packets after 80 s. Similarly, to generate congestion during simulation, the sensor nodes start sending packets at high data rate (i.e., 8 packets/s). The discretion parameter is selected with the aim that the combine discipline shifts towards the nonpreemptive discipline of the low priority packets. Therefore, preemption is permitted only when the remaining service time of the low priority packet is greater than the waiting delay threshold of high priority packets. The performance of the *Npqt++* is evaluated in terms of the following parameters throughput, average delay, and power consumption. Then, using the Contiki OS and Cooja simulator, we compare the *Npqt++* with two other congestion control algorithms in our related works[11,16] to test the performance of our approach: (1) a hybrid priority/FIFO scheduling discipline (priorityFIFO)—this scheme considers a discrete-time single sever queueing model with two-layered arrival process and a hybrid priority/FIFO scheduling where the traffic types are grouped into different priority classes,[16] and (2) the polling-TCPW-in this algorithm, the status report policy in the RLC protocol stack of the radio link control layer of the NB-IoT is enhanced to control data transmission and to realize automatic repeat-request retransmission.[11] The key parameters and protocols used in the simulation are presented in Table 1.

TABLE 1 Key parameters used in simulation

Parameter	Parameter Value
MAC protocol	ALOHA-based
Channel capacity	50 kbps
Number of nodes	$S = 1, I = 10, L = 15,$
Buffer size	10 packets
Transport layer	UDP
Network layer	IPv6, RPL
Arrival rate	Poisson
Inter-arrival rate	Random
Simulation time	800 s
Arrival rate	5 packets/s
Service rate	6 packets/s

7.1 | Throughput

Throughput is estimated as the total number of packets that is successfully transmitted from the leaf nodes to the sink every second. It is clear from Figure 4 where the y-axis represents the throughput in packets/second, and the x-axis represents the time in second that the *Nqpt++* outperforms the priority FIFO and polling-TCPW in terms of throughput performance. The explanation for this is that the *Nqpt++* implements a preemptive and nonpreemptive discipline that allows the high priority packets to be pushed to the front to be transmitted by the server. Similarly, the discretion rule ensures that low priority packets transmission is not interrupted by high priority packets during their nonpreemptive periods. Therefore, the total number of packets received at the application server increased massively by exploiting the preemptive/nonpreemptive discipline with discretion rule. Conversely, the priorityFIFO shows a better performance than the polling-TCPW. The reason is that the priorityFIFO implements a priority class for different traffic types whereas the polling-TCPW only utilizes a status report policy with a control data transmission and automatic repeat-request retransmission.

7.2 | Average delay

Average delay is measured as the time elapsed from the moment a packet is generated by the application in the leaf nodes until it reaches the IoT application server. The average delay of the high priority packet in the Npqt++ which is more important for time-critical applications is computed. Figure 5 compares the average delay of the *Npqt++* with the average delay of the priorityFIFO and the polling-TCPW. From the comparison, the *Npqt++* has lower average delay than both the priorityFIFO and the polling-TCPW algorithms. The reason is that when packets arrive at the link server in the *Npqt++* they are immediately classified based on their priority and high priority packets are directly moved to the front of the queue. Also, if the discretion rule is satisfied, the high priority packets are pushed out of the link server while low priority packets wait in the queue. Obviously, this has resulted in a low average delay for the high priority

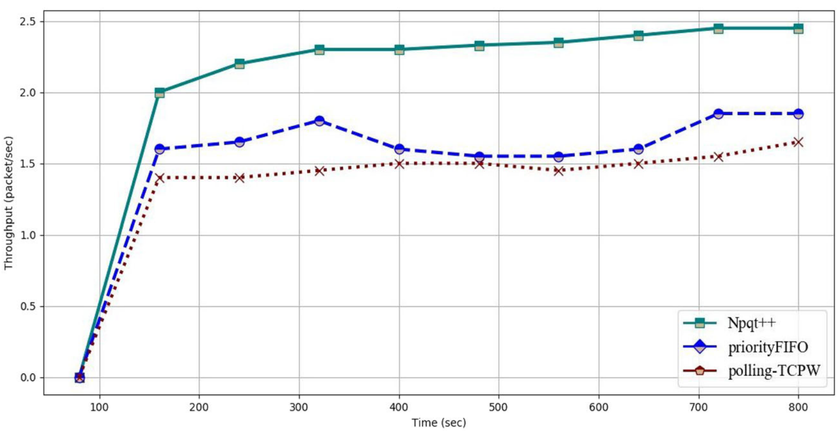

FIGURE 4 Throughput

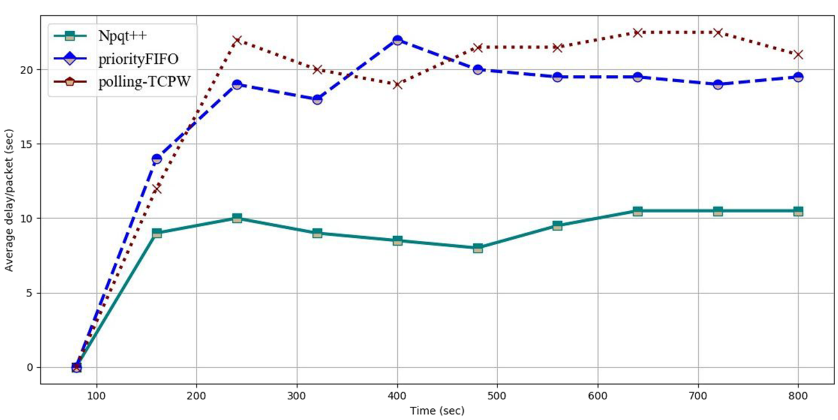

FIGURE 5 Average delay

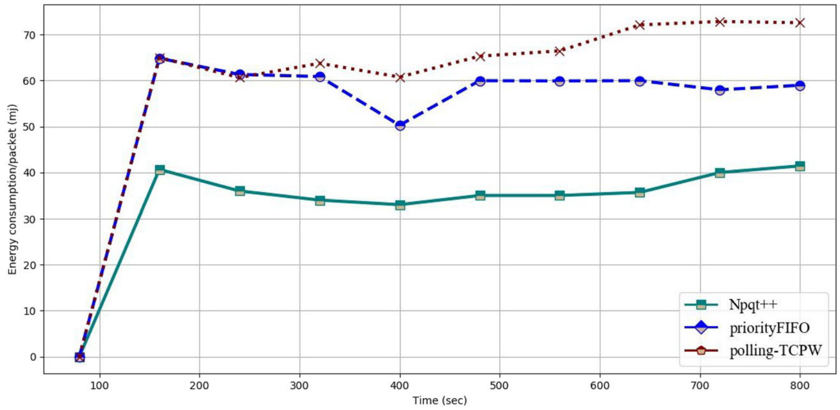

FIGURE 6 Energy consumption

packets and is shown in Figure 5, a better average delay performance for the *Npqt++* in comparison to the priorityFIFO and the polling-TCPW algorithms.

7.3 | Energy consumption

In Figure 6, the energy consumption of the priorityFIFO and the polling-TCPW algorithms is compared with the energy consumption of the *Npqt++*. The *Npqt++* consumes lower energy than the priorityFIFO and the polling-TCPW algorithms. For instance, at 400 s while the polling-TCPW consumes 60 mj of energy of energy per packet transmitted and the priorityFIFO consumes 50 *mj* of energy per packet transmitted, the *Npqt++* consumes just 32 *mj* of energy per packet transmitted. The explanation is that in the *Npqt++*, the number of packets loss during congestion is minimal and therefore does not require retransmission which consumes additional energy.

8 | CONCLUSION

In this paper, a novel congestion control technique for IoT networks has been developed. IoT is a new paradigm that connects millions of devices to the internet to meet human needs in diverse areas of applications such as smart cities, health, transportation, and defense. However, the IoT networks and enabling technologies suffer from shortcomings/problems such as limited buffer size, nodes processing capabilities, and server capacities. Similarly, the large amount of data transmitted from millions of sensor nodes to the IoT network servers leads to congestion in IoT networks. Therefore, congestion control has been recognized has a major focus in IoT networks with several congestion control approaches being proposed. Correspondingly, the technique proposed in this work implements a preemptive/non-preemptive discipline with a discretion rule. This approach is based on priority queuing technique where nodes packets are grouped and transmitted based on the real-time requirements of their IoT applications. The performance of the proposed technique is evaluated in terms of throughput, average delay, and energy consumption. Additionally, the proposed technique is compared to two existing congestion control algorithm to test its performance. Our results showed that our proposed technique outperforms the existing algorithms in terms of performance in terms of throughput, delay, and energy consumption. For future work, we will investigate the possibility of integrating reinforcement learning into the node capabilities. So that the nodes can predict the buffer occupancy status of potential parent node using the previous occupancy statistics.

ACKNOWLEDGEMENTS

This research was funded in part by the National Research Foundation of South Africa (grant number: 90604). Opinions, findings, and conclusions or recommendations expressed in any publication generated by the NRF supported research are those of the author(s) alone, and the NRF accepts no liability whatsoever in this regard. The authors would like to also thank the Telkom Centre of Excellence (CoE) for their support.

DATA AVAILABILITY STATEMENT

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

ORCID

Stephen S. Oyewobi https://orcid.org/0000-0002-1925-6490
Karim Djouani https://orcid.org/0000-0001-6060-8200
Anish Matthew Kurien https://orcid.org/0000-0002-7250-3665

REFERENCES

1. Khan W, Rehman M, Zangoti H, Afzal M, Armi N, Salah K. Industrial internet of things: recent advances, enabling technologies and open challenges. *Comput Electr Eng*. 2020;81:106522.
2. Srinivasan C, Rajesh B, Saikalyan P, Premsagar K, Yadav ES. A review on the different types of internet of things (IoT). *J Adv Res Dynamic Contr Syst*. 2019;11(1):154-158.
3. Ayoub W, Samhat AE, Nouvel F, Mroue M, Prévotet J-C. Internet of mobile things: overview of lorawan, dash7, and nb-iot in lpwans standards and supported mobility. *IEEE Commun Surveys Tutor*. 2018;21(2):1561-1581.
4. Oyewobi SS, Hancke GP, Abu-Mahfouz AM, Onumanyi AJ. An effective spectrum handoff based on reinforcement learning for target channel selection in the industrial internet of things. *Sensors*. 2019;19(6):1395.
5. Naik N. LPWAN technologies for IoT systems: choice between ultra narrow band and spread spectrum. In: *2018 IEEE International Systems Engineering Symposium (ISSE)*. IEEE; 2018:1-8.
6. Adame T, Bel A, Bellalta B. Increasing LPWAN scalability by means of concurrent multiband IoT technologies: an industry 4.0 use case. *IEEE Access*. 2019;7:46990-47010.
7. Oyewobi SS, Hancke GP. A survey of cognitive radio handoff schemes, challenges and issues for industrial wireless sensor networks (CR-IWSN). *J Netw Comput Appl*. 2017;97(Supplement C):140-156. /11/01/2017
8. Gomez C, Minaburo A, Toutain L, Barthel D, Zuniga JC. IPv6 over LPWANs: connecting low power wide area networks to the internet (of things). *IEEE Wireless Commun*. 2020;27(1):206-213.
9. Al-Kashoash HA, Amer HM, Mihaylova L, Kemp AH. Optimization-based hybrid congestion alleviation for 6LoWPAN networks. *IEEE Internet Things J*. 2017;4(6):2070-2081.
10. Mishra N, Verma LP, Srivastava PK, Gupta A. An analysis of IoT congestion control policies. *Proc Comput Sci*. 2018;132:444-450.
11. Zhou C, Zhao J, Liu H. Adaptive status report with congestion control in NB-IoT. In: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE; 2019:1-5.
12. Sukjaimuk R, Nguyen QN, Sato T. Dynamic congestion control in information-centric networking utilizing sensors for the IoT. In: *2018 IEEE Region Ten Symposium (Tensymp)*. IEEE; 2018:63-68.
13. Tabassum M, Tikoicina KM, Huda E. Comparative analysis of queuing algorithms and QoS effects on the IoT networks traffic. In: *2018 8th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*. IEEE; 2018:88-92.
14. Huang J, Xing C-C, Shin SY, Hou F, Hsu C-H. Optimizing M2M communications and quality of services in the IoT for sustainable smart cities. *IEEE Trans Sustain Comput*. 2017;3(1):4-15.
15. Ambigavathi M, Sridharan D. Energy efficient and load balanced priority queue algorithm for wireless body area network. *Future Gener Comput Syst*. 2018;88:586-593.
16. Walraevens J, Bruneel H, Fiems D, Wittevrongel S. Delay analysis of multiclass queues with correlated train arrivals and a hybrid priority/FIFO scheduling discipline. *App Math Model*. 2017;45:823-839.
17. Cho YZ, Un CK. Analysis of the M/G/1 queue under a combined preemptive/nonpreemptive priority discipline. *IEEE Trans Commun*. 1993;41(1):132-141.
18. Oyewobi SS, Hancke GP, Abu-Mahfouz AM, Onumanyi AJ. A delay-aware spectrum handoff scheme for prioritized time-critical industrial applications with channel selection strategy. *Comput Commun*. 2019;144:112-123.
19. Wu Y, Hu F, Zhu Y, Kumar S. Optimal spectrum handoff control for CRN based on hybrid priority queuing and multi-teacher apprentice learning. *IEEE Trans Vehicul Technol*. 2016;66(3):2630-2642.

How to cite this article: Oyewobi SS, Djouani K, Kurien AM. Using priority queuing for congestion control in IoT-based technologies for IoT applications. *Int J Commun Syst*. 2020;e4709. https://doi.org/10.1002/dac.4709