International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

PQ-SDNSch: A Priority Queue- Assisted Scheduling for Emergency Data in SDN-based Mobile Edge Computing

Bibhuti Bhusan Dash[1], Rabinarayan Satpathy[2], Sudhansu Shekhar Patra[*3]

Submitted: 27/01/2024 **Revised**: 05/03/2024 **Accepted**: 13/03/2024

Abstract: A crucial component of the 5th-generation (5G) mobile cellular networks is multi-access edge computing (MEC). MEC allows mobile users to access data and computing resources quickly, enabling Ultra-reliable and Low-latency Communications (URLLC). Researchers have focused on various areas of edge computing as 5G deployments get underway in an effort to take use of the additional capabilities that 5G offers. The sensors gather a wide variety of data, and there is a tremendous quantity of data delivered as well. Numerous researchers have attempted to use MEC to address the problem of excessive traffic caused by the data generated by numerous devices. The MEC infrastructure must be able to handle a large number of interconnected devices, the processing of the enormous amounts of data gathered, and complicated applications to handle the numerous and varied Internet of Things (IoT) devices. In contrast to top-tier servers in the cloud server, the MEC edge server has constrained computing and processing capacity. This paper presents PQ-SDNSch, a scheduling strategy-based software-defined network (SDN) for rapid transmission of emergency data in MEC situations. The MEC incorporates OpenFlow functionality to connect with the SDN controller. In the MEC host with OpenFlow enabled, two queues have been adopted for two classes of high and low-priority events. Both the class of requests are served on a FIFO order within their queue. Various numerical results have been computed for both classes of requests. The system's profit analysis is carried out for both high- and low-priority events.

Keywords: *MEC, SDN, Openflow, High Priority Events, Low Priority Events, Queueing theory, Generating Function*

1. Introduction

MEC has developed to support the Internet of Thing's (IoT) rapid expansion and the many service needs of newly developing vertical sectors, while also evolving to complement cloud computing and be in line with the telecom sector's future. Several researchers have made an effort to tackle this issue by utilising MEC, cloud computing, and fog computing. This is because the data gathered from numerous devices might result in excessive traffic. [1]. The European Telecommunications Standards Institute (ETSI) is now standardizing MEC technology through an Industry Specification Group (ISG) [2]. MEC is a revolutionary paradigm that moves network routing and computations from a centralised cloud to the network edge, enabling services to be delivered close to the clients that uses distributed cloud computing technologies on radio access networks (RANs). MEC makes it possible to analyse, process, and store data at the network's edge rather than transferring it all to the cloud for processing. It is possible to reduce core network congestion and generate new local services by implementing different services and caching material locally to consumers and IoT devices. Low latency real-time data processing is also supported. As a result, MEC may offer real-time services that are efficient, safe, and respond more quickly to the needs of numerous end users, mobile devices, and Internet of Things devices. With its capacity to provide network tractability, scalability, and optimised diversified services that will favourably benefit a variety of vertical industries, MEC is anticipated to play a significant part in the next 5G ecosystem. The MEC system's fundamental architecture is seen in Fig. 1.

1.1. NFV IN MEC

Operators and network service providers are being pressured to adopt Network Function Virtualization (NFV) in order to expedite the delivery of network services and eliminate the need for manual configuration of specialised hardware devices in order to construct service chains. Software operating on conventional servers may now generate, operate, distribute, and control network services thanks to the NFV idea, which offers a novel method of abstracting and virtualizing network operations [3]. The ETSI NFV group leads the efforts to standardise and deploy NFV, working with equipment suppliers and network operators. Considering the skyrocketing demand for the new networking services, NFV promises to enable innovation and provide agility to the delivery of network services. ETSI created a

[1]*School of Computer Applications,* KIIT Deemed to be University, *India*
ORCID ID : 0000-0001-8786-137X
[2] *Faculty of Emerging Technologies, Sri Sri University, India*
ORCID ID : 0009-0008-1442-4759
[3]*School of Computer Applications,* KIIT Deemed to be University, *India*
ORCID ID : 0000-0001-9996-7681
** Corresponding Author Email: sudhanshupatra@gmail.com*

modular NFV architecture to provide scalability, dependability, integration with legacy networks, and communication with currently running systems.

The MEC platform is made to perform flawlessly in the NFV environment in order to meet its goals for high bandwidth, lower latency, and real-time access to the compute, storage, and network capacity. Utilising NFV ideas effectively may be achieved by the incorporation of the MEC into an NFV framework.

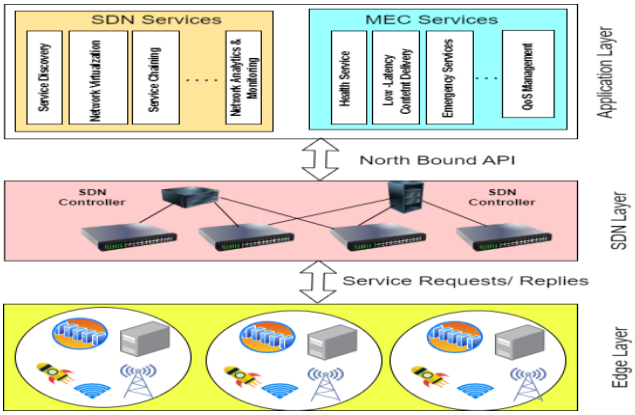

Fig. 1. Architecture of the MEC system

1.2. SDN IN MEC

To accurately identify SDN technology, it is essential to explain both the workings of a conventional network and the evidences that led to the development of the SDN paradigm. The data plane and the control plane are the two primary planes that make up a network in general [4]. The control plane is shown by the requests that are exchanged to carry out these tasks. SDN is designed to give the control plane the adaptability it needs to meet the data plane's traffic forwarding demands. It is a dynamic architecture that promises to automate the network, centralise control tasks, and programme the network via application programming interfaces (APIs), as of Open Network Foundation (ONF). SDN technology is therefore predicated on three pillars in order to accomplish these goals:

(1) separate the control plane from the data plane;

(2) rationally centralising the control plane and

(3) program the control plane.

The data plane components operations are simple. Applications are being used to directly programme SDN controllers as a result of the control plane's centralization.

In order to handle emergency data fast in MEC contexts, this article proposes an SDN-based packet scheduling strategy that takes use of the programmable and decoupled properties of SDN. The OpenFlow protocol function is included in the MEC server in the suggested design so that it may connect with the SDN controller. Two queues are used in the OpenFlow enabled MEC: the normal queue (NQ) for ordinary packet transmission and the emergency queue (EQ) for data transfer in an emergency. Data that is not urgently needed to be transmitted via the EQ is transmitted via the NQ.

The following are the paper's primary contributions:

(1) The suggested model manages the MEC server's resources using two queues;

(2) When an emergency packet enters the queues, the model processed it first; and

(3) The MEC server and MEC hosts use SDN ideas to minimise transmission latency.

The rest part of the article is organized as follows. Section 2 gives the related work in this area, the model description is shown in section 3, Section 4 represents the profit analysis of the system, the numerical results are shown in section 5 and finally the conclusion and future work is presented in section 6.

2. Related Work

Through the use of the Internet, cloud computing technology offers computer services including servers, storage, and software analysis [5]. The development of edge and fog computing offers a remedy for latency issues. With edge computing, a substantial decrease in data processing time and Internet bandwidth utilisation is achieved by processing huge volumes of processable data efficiently surrounding the source. Because edge computing is created with network connection and latency, band width limits, and other functionalities incorporated in a terminal device in mind, it may be adapted to a distributed computing architecture.

MEC is one category of edge computing that increases cloud computing capabilities and delivers them to the network edge. It is designed to alleviate major limitations of conventional cloud computing, such as problems with delays in real-time processing and transmission brought on by moving data to distant servers that are separated from consumers' devices [6]. By carrying out relevant processing activities closer to end users without transferring to the cloud data centre, it may significantly contribute to the improvement of applications and the reduction of network congestion. SDN is a conceptual architecture that makes network segmentation possible by segregating between network's control plane and data plane. The physical layer, control layer, and application layer are the three divisions of SDN architecture. SDN aims to isolate the network control function from hardware. The SDN controller has the majority of the network's control capabilities. Therefore,

programmatically managing and simplifying networks is possible for network operators and managers, rather than manually accessing and maintaining them via a range of devices for dispersed networks. It is able to easily build the virtual networks needed for cloud settings and create intricate pathways that are impossible to configure in current networks. When combined with MEC and edge computing, the SDN features can facilitate the installation and enhancement of a number of new functions. In an SDN context, a communication interface between the control plane and data plane is provided via ONF-standardized OpenFlow protocol [7]. The SDN controller may manage the fundamental elements of the network by using the OpenFlow protocol.

Many research are now being conducted on how MEC and networking functionalities like SDN and network virtualization overlap. To facilitate orchestration and administration, as well as help control mobility and quality of service (QoS) of mobile edge hosts (MEH), the authors of [8] proposed combining SDN and cloud-native virtualization schemes with the MEC. The authors introduced a MEC framework in [9] for SDN-based LTE/LTE-A networks. In [10], authors have introduces priority queue in each fog node. The tasks being offloaded from the edge servers piled into a queue in the fog layer. In order to prioritise the task assignment, authors in [11] separated the work into three categories based on the durations of the due dates for each category. To find the best order for the activities and cut down on queue waiting time, they also developed a rule-based approach for scheduling the task. An analytical queuing model was given by authors in [12] to perform work scheduling based on priority in a fog-cloud divided the jobs into two categories. Computing tasks having a higher priority and greater sensitivity to delays are referred to as class 1. Class 2, on the other hand, deals with computer jobs that are less important and less susceptible to delays. The scenario in which many mobile devices were engaged in cloud-assisted MEC was studied by authors in [13]. It was found that the cloud data centre used an M/G/c queue model, each edge server used an M/G/m non-preemptive priority queue, and the mobile devices in question used an M/G/1 non-preemptive priority queue [14-16].

3. Model Description

When it comes to large-scale crimes, terrorism, and natural catastrophes, emergency response services (ERS) are essential. Networks, whether computer-based or telecommunication-based, are essential for providing the data required to plan the reaction and relief effort in the case of an emergency. The capacity of a network infrastructure to withstand bursts of high-volume traffic, reduce end-to-end latency, and prioritize emergency services above quality-of-service (QoS) where feasible are the primary characteristics of emergency response systems (ERS). Emergency response agencies are under tremendous pressure to manage few resources while providing greater quality care for a greater number of rescues and post-crisis recovery operations. We can improve the emergency response service by the following:

- To manage the different emergency crises.
- To reduce the number of emergency crisis requests in the waiting room by implementing a perfect priority queueing model.
- To enhance the emergency service response system with limited resources.
- To quantify the system and balance the waiting time of High- priority as well as low- priority emergency requests.

Algorithm 1: Accessing Emergency Service Response system using PQ-SDNSch Framework

Input: Service Requests to emergency service response system

Output: Service Response

1. for all Service Requests want to connect to the PQ-SDNSch framework
2. if the Service Request == Normal event
3. The priority is set to low
4. if any other low-priority or high-priority event is executing
5. Enter into low-priority event queue
6. else
7. Execute the service request as per the norms of low-priority event
8. end if
9. else if the Service Request == Emergency event
10. The priority is set to High
11. if any low-priority event is executing
12. Interrupt the service of the low-priority event and resum again when there will be no high-priority events in the high-priority events Queue
13. Execute the service request as per the norms of high-priority event

14. else if any high-priority event is executing

15. Enters into high-priority event queue

16. end if

17. end if

18. end for

19. **End Algorithm**

A priority queue is maintained in the SDN controller used for solving the shortcomings of FIFO. In the controller queueing system, there are two event classes, one high- priority and the low-priority events. Suppose in each class the events follow FIFO and the events are preemptive means during the execution of a low-priority event if a high-priority event joins the system, the low-priority event interrupts service and resumes again when there is no high-priority event with the system. The high-priority event is confined to a certain number of L which includes the events waiting in the buffer as well as being processed, and the low-priority events are infinite. Let λ_h, λ_l be the arrival rates for the high and low-priority events respectively to the controller. Algorithm 1 shows the working principle of the emergency service response system using the PQ-SDNSch framework. The following components of the system's priority queue need particular consideration:

- Based on their importance or demands on the system, events are classified into two types by the SDN controller.
- The significance of emergency events surpasses that of regular occurrences. Two classes of service priority may be applied while several events are pending execution.
- When two events are prioritized, the higher- priority event denies service to the lower priority task.
- After the completion of the high-priority tasks, if the service preemption is allowed, the preempted task may continue service.

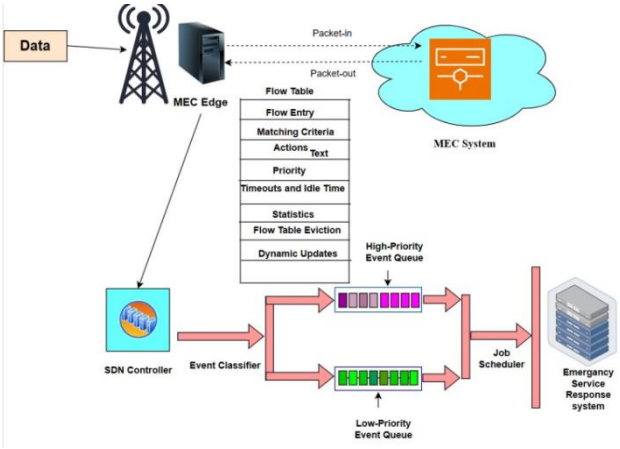

Fig. 2. The Queueing operations at the MEC edge using the SDN framework

Denote the traffic intensities by $\rho_h = \lambda_h / \mu_h$, $\rho_l = \lambda_l / \mu_l$, and let $\pi_{j,i}$ be the steady-state probability of the system is in state (j, i) where j and i are the number of high as well as low- priority events with the controller respectively. Let the recursive difference equation for $\pi j,i$ can be written as:

$$(\lambda_h + \lambda_l)\pi_{0,0} = \mu_h \pi_{1,0} + \mu_l \pi_{0,1}, \tag{1}$$

$$(\lambda_h + \lambda_l + \mu_l)\pi_{0,i} = \lambda_l \pi_{0,i-1} + \mu_h \pi_{1,i} + \mu_l \pi_{0,i+1}, \ i \geq 1, \tag{2}$$

$$(\lambda_h + \lambda_l + \mu_l)\pi_{j,0} = \lambda_h \pi_{j-1,0} + \mu_h \pi_{j+1,0}, \ 1 \leq j \leq L-1 \tag{3}$$

$$(\lambda_h + \lambda_l + \mu_h)\pi_{j,i} = \lambda_h \pi_{j-1,i} + \mu_h \pi_{j+1,i} + \lambda_l \pi_{j,i-1}, \ 1 \leq j \leq L-1, i \geq 1 \tag{4}$$

$$(\lambda_l + \mu_h)\pi_{L,0} = \lambda_h \pi_{L-1,0} \tag{5}$$

$$(\lambda_l + \mu_h)\pi_{L,i} = \lambda_h \pi_{L-1,i} + \lambda_l \pi_{L,i-1}, i \geq 1, \tag{6}$$

Considering $H_j(s)$ as the generating function (GF) for $\pi_{j,i}$

$$H_j(s) = \sum_{i=0}^{\infty} \pi_{j,i} s^i, \ j = 0,1,2...N, |s| \leq 1. \tag{7}$$

Then the above system (1)-(6) can be represented as:

$$[\lambda_h s + (\mu_l - \lambda_l s)(s-1)H_0(s) - \mu_h s H_1(s) = \mu_l (s-1)\pi_{0,0}, \tag{8}$$

$$-\lambda_h H_{j-1}(s) + [(\lambda_h - \lambda_l (s-1) + \mu_l]H_j(s) - \mu_h H_{j+1}(s) = 0, \ 1 \leq j \leq L-1 \tag{9}$$

$$-\lambda_h H_{L-1}(s) + [\mu_h - \lambda_l (s-1)]H_L(s) = 0, \tag{10}$$

These equations can be rewritten as,

$$B(s)H(s)=A(s), \quad (11)$$

where $H(s)=[H_0(s), H_1(s), ...H_L(s)]^t$ is the generating function vector for $\pi_{j,i}$.

$$A(s)=[\lambda_l(s-1)\pi_{0,0}, 0, ...0]^t \text{ and}$$

$$B(s)=[b_{j,i}] is\ a\ (N+1)X\ (L+1) \quad (12)$$

order tri-diagonal matrix and its coefficients are:

$$b_{j,i}=\begin{cases}\lambda_h s+(\mu_l-\lambda_l s)(s-1), \\ \qquad j=0, i=0, \\ -\lambda_h s, \qquad j=0, i=1, \\ \lambda_h(1-\delta_{j,L})-\lambda_l(s-1)+\mu_h, \\ \qquad i=j, \\ \qquad j=1,2,.....,L, \\ -\lambda_h, \qquad i=j-1,\ j=1,2,...,L. \\ -\mu_h, \qquad i=j-1,\ j=1,2,...,L. \\ 0, \qquad \text{otherwise}\end{cases} \quad (13)$$

$$H_i(s)=\mu_l\lambda_h^i\pi_{0,0}\times \frac{Z_{L-i}(s)}{\prod_{k=0}^{N}(\lambda_h+\lambda_l+\mu_h-\Delta_{L,k}\sqrt{\lambda_h\mu_h}-\lambda_l s)} \quad (14)$$

$$Z_p(s)=\beta Z_p(s)-\lambda_h\mu_h Z_{p-2}(s),\ 2\le p\le L. \quad (15)$$

$$Z_1(s)=\mu_h-\lambda_l(s-1), \quad (16)$$

$$Z_2(s)=\beta[\mu_h-\lambda_l(s-1)-\lambda_h\mu_h, \quad (17)$$

$$\beta=\lambda_h+\lambda_l+\mu_h-\lambda_l s, \quad (18)$$

$$\pi_{0,0}=\frac{1-\rho_h}{1-\rho_h^{N+1}}-\rho_l, \quad (19)$$

$$\rho_h=\frac{\lambda_h}{\mu_h}, \rho_l=\frac{\lambda_l}{\mu_l} and\ \Delta_{L,k}, k=1,2,...L$$

are the roots of Chebychev's polynomial of 2nd kind. The marginal distribution of the high- priority events is given by

$$\pi_{j,.}=\frac{\rho_h^j(1-\rho_h)}{1-\rho_h^{L+1}},\ j=0,1,2,....,L. \quad (20)$$

The low priority's marginal distribution is determined by

$$\pi_{.,i}=\mu_l^2\lambda_l^j\pi_{0,0}\sum_{x=0}^{N}\frac{B_{0,x}}{\beta_x^{i+2}},\ i\ge 0. \quad (21)$$

3.1. Average Wait Time for Each Priority Class

3.1.1. High Priority Class

Assume that N_h is the highest priority class's queue length.

$$E(N_h)=\sum_{j=0}^{L}\sum_{i=0}^{\infty}j\pi_{j,i}=\sum_{j=0}^{L}j\pi_{j,.}. \quad (22)$$

By applying equation (20) in eq. (22)

$$E(N_h)=\sum_{j=1}^{L}j\frac{\rho_h^j(1-\rho_h)}{(1-\rho_h^{L+1})}$$

$$=\frac{\rho_h(1-\rho_h)}{(1-\rho_h^{L+1})}\frac{d}{d\rho_h}(\frac{1-\rho_h^{L+1}}{1-\rho_h})$$

$$=\frac{\rho_h[1-(L+1)\rho_h^L+L\rho_h^{L+1}]}{(1-\rho_h)(1-\rho_h^{L+1})} \quad (23)$$

This indicates that the function of high-priority queue is independent and it follows the M/M/1/L queue.

3.1.2. Low Priority Class

Assume that N_l is the highest priority class's queue length.

$$E(N_l)=\sum_{j=0}^{L}\sum_{i=1}^{\infty}i\pi_{j,i}=\sum_{j=0}^{L}Y_j, \quad (24)$$

$$Y_j=\sum_{i=1}^{\infty}i\pi_{j,i}$$

$$(\lambda_h+\lambda_l+\mu_h)Y_0 = \mu_h Y_1+\mu_l Y_0+\mu_l\sum_{i=1}^{\infty}i\pi_{0,i+1}+\lambda_l\sum_{i=1}^{\infty}\pi_{0,i-1} \quad (25)$$

From eqns. (20) and (21), we have

$$Y_1=\rho_h Y_0+\frac{\lambda_l\rho_h}{\mu_h}(\frac{1-\rho_h^{L+1}}{1-\rho_h}) \quad (26)$$

$$(\lambda_h+\lambda_l+\mu_h)Y_j=\mu_h Y_{j+1}+\lambda_l\sum_{i=1}^{\infty}\pi_{j,i-1}+\lambda_h Y_{i-1}+\lambda_l Y_i \quad (27)$$

$$Y_j = \rho_h^j Y_0 + \frac{\lambda_l}{\mu_h}\frac{\rho_h^j}{(1-\rho_h^{L+1})}[j - \sum_{x=1}^{j}\rho_h^{L+1-x}] \quad (28)$$

$$E(N_l) = \frac{1}{\rho_l}\sum_{i=0}^{\infty} i\pi_{0,i+1} = \rho_l^{-1}Y_0 - 1 \quad (29)$$

Then $Y_0 = \rho_l(1+E(N_l))$.

$$E(N_l) = \sum_{j=0}^{L} Y_i$$

$$= \sum_{j=0}^{L}[\rho_h^j Y_i + \frac{\lambda_l}{\mu_h}\frac{\rho_h^j}{(1-\rho_h^{L+1})}[j - \sum_{x=1}^{j}\rho_h^{L+1-x}]]$$

$$= \frac{(1-\rho_h)}{(1-\rho_h-\rho_l(1-\rho_h^{L+1}))} \times[\frac{\rho_l(1-\rho_h^{L+1})}{(1-\rho_h)} + \frac{\lambda_l}{\mu_h}\frac{\rho_h}{(1-\rho_h)^2(1-\rho_h^{L+1})} \times(1\text{-}\rho_h^{2L+1} - (2L+1)(1-\rho_h)\rho_h^L)]. \quad (30)$$

3.2. The Other Performance Measures

Appling Little's law, the mean sojourn time of high-priority events is

$$W_h = \frac{E(N_h)}{\lambda_h} = \frac{[1-(L+1)\rho_h^L + L\rho_h^{L+1}]}{\mu_h(1-\rho_h)(1-\rho_h^{L+1})} \quad (31)$$

The mean sojourn time of low-priority events is

$$W_l = \frac{E(N_l)}{\lambda_l} = \frac{1}{\lambda_l}[\frac{(1-\rho_h)}{(1-\rho_h-\rho_l(1-\rho_h^{L+1}))} \times[\frac{\rho_l(1-\rho_h^{L+1})}{(1-\rho_h)} + \frac{\lambda_l}{\mu_h}\frac{\rho_h}{(1-\rho_h)^2(1-\rho_h^{L+1})} \times(1\text{-}\rho_h^{2L+1} - (2L+1)(1-\rho_h)\rho_h^L)]]. \quad (32)$$

The average duration of time spent waiting for high-priority events is determined by

$$W_{q,h} = \rho_h \times W_h \quad (33)$$

The average duration of time spent waiting for low-priority events is determined by

$$W_{q,l} = \rho_l \times W_l \quad (34)$$

Table 2. Notations Used

Notation	Representation
λ_h	Rate of arrival of high-priority events
λ_l	Rate of arrival of low-priority events
μ_h	Rate at which the high-priority events are served
μ_l	Rate at which the low-priority events are served
L	Restricted number of high-priority event
ρ_h	Traffic intensity relative to high-priority event
ρ_l	Traffic intensity relative to low-priority event
i	Number of low-priority events
j	Number of high-priority events
$\pi_{j,i}$	The System's steady-state probability (j, i)
$E(N_h)$	Expected units of high-priority events in the system
$E(N_l)$	Expected units of low-priority events in the system
W_h	Mean sojourn time of high-priority events
W_l	Mean sojourn time of low-priority events
$W_{q,h}$	Average duration of stay in the high-priority event queue
$W_{q,l}$	Average duration of stay in the low-priority event queue
R_h	Revenue earned in providing service to high-priority events
R_l	Revenue earned in providing service to low-priority events
TER_h	Total expected revenue for high-priority events
TER_l	Total expected revenue for low-priority events
C_h	Cost per unit time to serve high-priority events
C_l	Cost per unit time to serve low-priority events
C_{hh}	The cost per event incurred to hold high-priority events in unit time.
C_{hl}	The cost per event incurred to hold low-priority events in unit time.
TEC_h	Total expected cost based on high-priority events
TEC_l	Total expected cost based on low-priority events
TEP_h	Total expected profit for high-priority events
TEP_l	Total expected profit for low-priority events
TEP_s	Total expected profit on the system
TOC	Total optimal cost

4. Profit Analysis of the System

4.1. Profit Analysis of High-Priority Events

Let R_h be the revenue earned in providing service to high priority events, then the total expected revenue (TER_h) is

$$TER_h = R_h E(N_h) \tag{33}$$

The total expected cost (TECh) based on the high-priority events to the controller is

$$TEC_h = C_h \mu_h + C_{hh} E(N_h) \tag{34}$$

The total expected profit (TEPh) for high-priority events is

$$\begin{aligned} TEP_h &= TER_h - TEC_h \\ &= R_h E(N_h) - (C_h \mu_h + C_{hc} E(N_h)) \\ &= (R_h - C_{hh}) E(N_h) - C_h \mu_h \\ &= (R_h - C_{hh}) \frac{\alpha}{\beta} \end{aligned} \tag{35}$$

where

$$\alpha = \rho_h [1 - (L+1)\rho_h^L + L\rho_h^{L+1}]$$

$$\beta = (1-\rho_h)(1-\rho_h^{L+1})$$

4.2. Profit Analysis of Low-Priority Events

Let R_l be the revenue earned in providing service to low-priority events, then the total expected revenue (TERl) is

$$TER_l = R_l E(N_l) \tag{36}$$

The total expected cost (TEC) based on the low-priority events to the controller is

$$TEC_l = C_l \mu_l + C_{hl} E(N_l) \tag{37}$$

The total expected profit (TEPl) for low-priority events is

$$\begin{aligned} TEP_l &= TER_l - TEC_l \\ &= R_l E(N_l) - (C_l \mu_l + C_{hl} E(N_l)) \\ &= (R_l - C_{hl}) E(N_l) - C_l \mu_l \\ &= (R_l - C_{hl}) \left(\frac{\theta}{\psi} + \frac{\lambda_l}{\mu_h} \frac{\phi}{\beta\psi}\right) - C_l \mu_l \end{aligned} \tag{38}$$

where

$$\beta = (1-\rho_h)(1-\rho_h^{L+1}), \qquad \theta = \rho_l (1-\rho_h^{L+1}),$$

$$\psi = 1 - \rho_h - \rho_l (1-\rho_h^{L+1}),$$

$$\phi = \rho_h - \rho_h^{2L+1} - (2L+1)(\rho_h^{L+1} - \rho_h^{L+2})$$

Total Expected Profit (TEPs) of the System

$$\begin{aligned} TEP_s &= TEP_h + TEP_l \\ &= (R_h - C_{hh}) \frac{\alpha}{\beta} + (R_l - C_{hl}) \left(\frac{\theta}{\psi} + \frac{\lambda_l}{\mu_h} \frac{\phi}{\beta\psi}\right) - C_l \mu_l \end{aligned} \tag{39}$$

The algorithm for finding the profit analysis of the system is shown in Algorithm 2.

Algorithm 2 Finding the Profit for events of Priorities

Input: Rh, Rl, E(Nh), C_h,C_l μ_h, C_{hh}, L,

Output : TERh, TECh, TEP_h, TER_l, TEC_l, TEP_l, TEPs

1. Initialize:

2. $\rho_h = \frac{\lambda_h}{\mu_h} < 1,\ \rho_l = \frac{\lambda_l}{\mu_l} < 1.$

3. C_h= Cost per service per unit time associated with high- priority events

4. C_l= Cost per service per unit time associated with low-priority events

5. Compute:

6. $TER_h = R_h E(N_h)$

7. $TEC_h = C_h \mu_h + C_{hh} E(N_h)$

8. $TEP_h = TER_h - TEC_h$

9. $TER_l = R_l E(N_l)$

10. $TEC_l = C_l \mu_l + C_{hl} E(N_l)$

11. $TEP_l = TER_l - TEC_l$

12. $TEP_s = TEP_h + TEP_l$

13. Exit

5. Numerical Examples

The numerical computations were done using Python 3.11.5. Table 3 gives the idle probability for the high-priority events for various traffic intensities ρh and ρl. As expected, as the low-priority events are increasing, the idle probability for the high-priority events is decreasing.

Table 3. Idle probability values for varying intensities in the high- priority class ρ_h and ρ_l

i	ρ_h=0.65, ρ_l=0.30	ρ_h=0.70, ρ_l=0.25
0	0.06837	0.05838
1	0.04638	0.01282
2	0.03536	0.00230
3	0.02737	0.00373
4	0.03738	0.00276
5	0.01292	0.00162
6	0.01920	0.00128
7	0.01728	0.00092
8	0.01518	0.00016
9	0.01478	0.00057
10	0.01289	0.00041

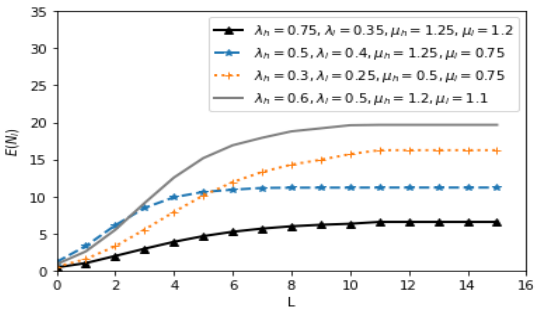

Fig.3. Mean queue length values for low- priority classes for various values of L.

Figures 3 and 4 show the mean queue length of low-priority events and high priority events, respectively for various values of the restricted number of high-priority events (L). Table 4 gives the performance matrices of two priority class events for $\mu_h = \mu_l$. λ_h is varied from 1 to 4 and the other parameters are taken as λ_l =5.0, μ_h= μ_l=11.0. Table 5 gives the performance matrices of two priority class events for $\mu_h \neq \mu_l$. λ_h is varied from 1 to 4 and the other parameters are taken as λ_l =5.0, μ_h=11.0 and μl=12.0. It has been found that the increase in λ_h, increases the other performance indices. Comparing high-priority and low-priority, one can see that the mean number of events in the queue (system) and for the high-priority events, there is a reduction in the average waiting time in the queue (system).

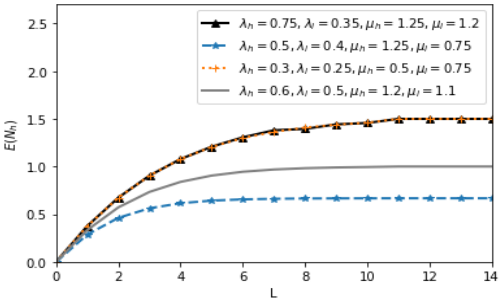

Fig.4. Mean queue length of high-priority events for Vs L.

From tables 6 to 9 one can get the following observations:

1. The system's total optimum cost (TOC) rises in tandem with the service costs related to high-priority events (C_h).

2. A rise in the total optimum cost (TOC) is the result of an increase in service costs related to low-priority events occurrences (C_l).

3. The total expected profit of high-priority events (TEP_h) and low-priority events (TEP_l) rises when revenue per unit event of high priority (TER_h) as well as low-priority events (TER_l) rises.

4. When holding costs related to high-priority events (C_{hh}) and low-priority events (C_{hl}) events rise, the system's total anticipated profit (TEPs) shows a little decrease.

5. Additionally, it shows that the total expected profit of high-priority events (TEP_h), and low-priority events (TEP_l) declines when the cost per unit event of both priority events (C_h, C_l) rises.

6. The total optimum cost (TOC) of the system exhibits a small decrease as holding costs related to high-priority events (C_{hh}) rise.

7. As holding costs for low-priority class events (C_{hl}) rise, there is a small decrease in the system's total optimum cost (TOC).

8. Additionally, we saw that the total expected profit of the system (TEPs) increased along with improvements in revenue per unit event for both priority groups (TER_h, TER_l).

Table 4. Performance matrices of two priority events with equal serice rate ($\mu_h = \mu_l$)

λ_l =5.0, $\mu_h = \mu_l$=11.0						
	λ_h=1			λ_h =2		
	Overall	**High-priority Events**	**Low- priority Events**	**Overall**	**High-priority Events**	**Low-priority Events**
E(N_h)	1.222222	0.100917	1.121305	1.790698	0.22449	1.566208
E(N_l)	0.672222	0.009251	0.662971	1.149031	0.041156	1.107875
W_h	0.203704	0.100917	0.224261	0.255814	0.112245	0.313242
W_l	0.112037	0.009251	0.132594	0.164147	0.020578	0.221575
	λ_h=3			λ_h =4		
	Overall	**High-priority Events**	**Low- priority Events**	**Overall**	**High-priority Events**	**Low-priority Events**
E(N_h)	2.75	0.37931	2.37069	4.714286	0.578947	4.135338
E(N_l)	2.016667	0.10431	1.912356	3.889286	0.212281	3.677005
W_h	0.34375	0.126437	0.474138	0.52381	0.144737	0.827068
W_l	0.252083	0.03477	0.382471	0.432143	0.05307	0.735401

Table 5. Performance matrices of two priority events with distinct service rates ($\mu_h \neq \mu_l$)

λ_l =5.0, μ_h= 11.0, μ_l=12.0						
	λ_h=1			λ_h =2		
	Overall	**High-priority Events**	**Low- priority Events**	**Overall**	**High-priority Events**	**Low- priority Events**
E(N_h)	0.165636	0.100917	0.064719	0.305812	0.224489	0.081322
E(N_l)	0.02397	0.009251	0.014719	0.072478	0.041156	0.031322
W_h	0.103523	0.100917	0.107865	0.117619	0.112245	0.135537
W_l	0.014981	0.009251	0.024532	0.027876	0.020578	0.052203
	λ_h=3			λ_h =4		
	Overall	**High-priority Events**	**Low- priority Events**	**Overall**	**High-priority Events**	**Low- priority Events**
E(N_h)	0.484291	0.37931	0.104981	0.719247	0.578947	0.140301
E(N_l)	0.159291	0.10431	0.054981	0.302581	0.21228	0.090301
W_h	0.134525	0.126436	0.174968	0.156358	0.144737	0.233834
W_l	0.044247	0.03477	0.091635	0.065778	0.05307	0.150501

Table 6. Measures of system performance under ideal operating circumstances for a range of values of (c_h, c_l, c_{hh} and c_{hl}) and fixed (λ_h=7.0, λ_l=9.0 and L=100).

C_h	C_l	C_{hh}	C_{hl}	μ_h^*	μ_l^*	TOC (μ_h^*, μ_l^*)
16	14	21	26	12.3	14.2	299.4
18	14	21	26	12.3	15.3	334.8
20	14	21	26	12.3	16.4	370.3
22	14	21	26	12.3	17.5	405.6
24	14	21	26	12.4	18.6	442.5
16	16	21	26	12.3	13.9	321.4
16	18	21	26	12.4	13.7	342.3
16	20	21	26	12.4	13.5	362.3

16	22	21	26	12.5	13.3	381.4
16	14	23	26	12.5	13.3	292.6
16	14	25	26	12.2	12.5	286.3
16	14	27	26	12.2	11.7	279.3
16	14	29	26	12.2	10.8	272.9
16	14	21	28	12.2	14.3	294.3
16	14	21	30	12.2	14.4	290.4
16	14	21	32	12.2	14.5	286.4
16	14	21	34	12.2	14.5	281.0
16	14	21	36	12.1	14.6	277.2

Table 7. Total expected profit for high priority events for μ_h=7.0, λ_l=5.0 and L=100

C_{hh}	C_h	R_h	TEP_h
16	26	210	17865.7
18	26	210	17668.6
20	26	210	17423.4
22	26	210	17278.1
16	28	210	17850.9
16	30	210	17835.8
16	32	210	17822.7
16	34	210	17808.4
16	26	260	22739.3
16	26	310	27614.0
16	26	360	32489.3
16	26	460	37363.4
16	26	510	42239.1

Table 8. Total expected profit for low priority events for μ_h=5.0, μ_l=4.0, λ_h=10.0, λ_l=12.0 and L=100

C_{hh}	C_h	R_l	TEP_l
16	11	720	1597.40
18	11	720	1589.55
20	11	720	1581.30
22	11	720	1573.70
24	11	720	1565.74
16	13	720	1592.45
16	15	720	1587.45
16	17	720	1582.65
16	19	720	1577.65
16	11	770	1717.55
16	11	820	1837.35
16	11	870	1957.54

Table 9. Total expected profit of the system for μ_h=5.0, μ_l=4.0, λh=10.0, λ_l=12.0 and L=100.

C_h	C_l	C_{hh}	C_{hl}	R_h	R_l	TEP_h	TEP_l	TEP_s
26	16	16	11	720	1020	580	2336	2896
29	16	16	11	720	1020	565	2336	2881
32	16	16	11	720	1020	550	2336	2866
35	16	16	11	720	1020	535	2336	2851

38	16	16	11	720	1020	520	2336	2836
26	19	16	11	720	1020	580	2324	2886
26	22	16	11	720	1020	580	2312	2872
26	25	16	11	720	1020	580	2300	2860
26	28	16	11	720	1020	580	2288	2848
26	16	19	11	720	1020	577	2336	2893
26	16	22	11	720	1020	574	2336	2890
26	16	25	11	720	1020	571	2336	2887
26	16	28	11	720	1020	568	2336	2884
26	16	16	14	720	1020	580	2328	2888
26	16	16	17	720	1020	580	2321	2881
26	16	16	20	720	1020	580	2314	2874
26	16	16	22	720	1020	580	2307	2867
26	16	16	11	770	1020	630	2336	2946
26	16	16	11	820	1020	680	2336	2996
26	16	16	11	870	1020	730	2336	3046
26	16	16	11	720	1070	580	2456	3016
26	16	16	11	720	1120	580	2576	3136
26	16	16	11	720	1170	580	2696	3256

6. Conclusion

By eliminating transmission delays that may arise from sending data through many processes already in place and by providing the MEC edge with a computational role, MEC can get data to destinations faster in an Internet of Things scenario. It's a technology that will be useful in industrial settings in the future since it can optimise and transfer a lot of data.

In this paper, we proposed a PQ-SDNSch, a SDN-based event scheduling technique for mobile edge computing settings to provide emergency data. We attempted to integrate SDN into the MEC edge and suggested a novel scheduling technique that makes use of two different types of MEC edge queues. The simulation results indicate that the suggested approach performs better for emergency packet processing than FCFS scheduling; nevertheless, for typical data, when utilisation rises, the delay time may grow more than FCFS's. The experimental results as well as the performance matrices, evaluation, and analysis of the given framework have been explained using the appropriate tables and diagrams.

Acknowledgement

The authors gratefully given credit to the Research Lab of the School of Computer Applications, KIIT Deemed to be University, Bhubaneswar and Faculty of Emerging Technologies, Sri Sri University, Cuttack for providing computational resources.

Author contributions

Rabinarayan Satpathy, Sudhansu Shekhar Patra: Conceptualization, Methodology, Software, Field study **Bibhuti Bhusan Dash:** Data curation, Writing-Original draft preparation, Software, Validation., Field study **Name3 Sudhansu Shekhar Patra, Bibhuti Bhusan Dash:** Visualization, Investigation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] G. Li, Y. Yao, J. Wu, X. Liu, X. Sheng, and Q. Lin, "A new load balancing strategy by task allocation in edge computing based on intermediary nodes," EURASIP Journal on Wireless Communications and Networking, vol. 2020(1), pp. 1-10, 2020.

[2] ETSI, "Network functions virtualisation (NFV); architectural framework v1.2," ETSI, Sophia Antipolis, France, White Paper, Dec. 2014.

[3] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of network function virtualization," Computer Networks, 133, pp. 212-262, 2018.

[4] S. Rout, K. S. Sahoo, S. S. Patra, B.Sahoo & D. Puthal, "Energy efficiency in software defined networking: A survey," SN Computer Science, 2(4), 308, 2021.

[5] Z. Lv and W. Xiu, "Interaction of edge-cloud

computing based on SDN and NFV for next generation IoT," IEEE Internet of Things Journal, 7(7), pp. 5706-5712, 2019.

[6] B. B. Dash, S. S. Patra, R. Satpathy and B. Dash, "Improvement of SDN-based Task Offloading using Golden Jackal Optimization in Fog Center," World Conference on Communication & Computing (WCONF), pp. 1-6, 2023.

[7] D. Carrascal, E. Rojas, J.M. Arco, D. Lopez-Pajares, J. Alvarez-Horcajo, & J.A. Carral, "A Comprehensive Survey of In-Band Control in SDN: Challenges and Opportunities," Electronics, 12(6), 1265, 2023.

[8] S. Rout, S. S. Patra, B. Sahoo and A. K. Jena, "Load balancing in SDN using effective traffic engineering method," International Conference on Signal Processing and Communication (ICSPC), pp. 452-456, 2017.

[9] B. B. Dash, R. Satapathy and S. S. Patra, "Energy Efficient SDN-assisted Routing Scheme in Cloud Data Center," 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN), pp. 1-5, 2023.

[10] S. S. Patra, "Energy-efficient task consolidation for cloud data center", International Journal of Cloud Applications and Computing (IJCAC), 8(1), pp. 117-142, 2018.

[11] V. Goswami, S. S. Patra and G. B. Mund, "Performance analysis of cloud with queue-dependent virtual machines," International Conference on Recent Advances in Information Technology (RAIT), pp. 357-362, 2012.

[12] V. Goswami, S. S. Patra and G. B. Mund, "Performance analysis of cloud computing centers for bulk services," International Journal of Cloud Applications and Computing (IJCAC), 2(4), pp. 53-65, 2012.

[13] B. B. Dash, R. Satapathy and S. S. Patra, "SDN-Assisted Routing Scheme in Cloud Data Center using Queueing Vacation Policy," International Conference on Edge Computing and Applications (ICECAA), , pp. 1-6, 2023.

[14] B. B. Dash, U. C. De, M. R. Mishra, R. Satapathy, S. Behera, N. Panda, S. S. Patra, "Leasing in IaaS Cloud Using Queuing Model," International Conference on Expert Clouds and Applications, pp. 159-167, 2022.

[15] B.B. Dash, R.N. Satpathy, & S.S. Patra, "Efficient SDN-based Task offloading in fog-assisted cloud environment," EAI Endorsed Transactions on Internet of Things, 10,pp. 1-7, 2014

[16] S. S. Patra, R. Govindaraj, S. Chowdhury, M. A. Shah, R. Patro and S. Rout, "Energy Efficient End Device Aware Solution Through SDN in Edge-Cloud Platform," in IEEE Access, vol. 10, pp. 115192-115204, 2022.