Revenue Maximization of a Slice Broker in the Presence of Byzantine Faults

Muhidul Islam Khan
Dept. of Electrical Engineering and Computer Science
University of Stavanger
Stavanger, Norway
email: md.m.khan@uis.no

Gianfranco Nencioni
Dept. of Electrical Engineering and Computer Science
University of Stavanger
Stavanger, Norway
email: gianfranco.nencioni@uis.no

***Abstract*—Multi-Access Edge Computing (MEC) and network slicing are vital for advancing the Fifth Generation (5G) of cellular systems. MEC provides context awareness and reduces the latency for communication. Network slicing allows the division of a single network into multiple virtual networks so that different services can be provided. A slice broker is a business entity that buys the resources from the infrastructure providers and sells them to the tenants. A tenant sends a request for resources for different slices. In this work, we formulate the slice allocation problem to increase the revenue for the slice broker. We formulate a dynamic demand model based on the set price changes. We consider the profit maximization of a slice broker in the presence of Byzantine faults. Moreover, we propose the Comparative Gradient Elimination (CGE) method in Federated Learning (FL) for revenue maximization of the slice broker. Simulation results show that our proposed method outperforms the reference solution.**

***Index Terms*—Revenue model, Network slicing, Slice broker, Byzantine faults**

I. Introduction

The Fifth Generation (5G) cellular communication provides dynamic programming and virtualization by critical enabling technologies, e.g., Software-Defined Networking (SDN) and Network Function Virtualization (NFV), which attracted the academy and industry sectors. Network slicing is the key enabling technology for 5G, where slices are for particular applications, and network functions [1]. For making slices based on the demand of the tenants, there are some Infrastructure Providers (InPs) from where it is possible to buy the resources and make the slices. Multi-access Edge Computing (MEC) provides dynamic resource allocation for slices and reduces latency by providing distributed cloud service facilities. In the context of the new supply chain, a business entity named slice broker buys the resources from InPs and works on creating and managing the slices. Slice brokers need to consider the economic aspects for dynamic resource allocation [2].

The economic aspects of a business entity can include cost, revenue, and profit. Cost is the amount the entity needs to pay to buy the resources. The revenue is the amount when the entity receives after selling the resources. Here, the slice broker sells network slices to the tenants, and they receive an amount, which is revenue. The profit is the net benefit the entity receives calculated by revenue minus the cost paid for buying the resources.

Our paper focuses on maximizing the revenue of a slice broker by following a revenue model dynamically. We have a dynamic demand model which varies by the set of prices of the resources [3]. So, we need to set the prices to maximize revenue. We consider an adaptive method based on multi-agent-based learning in a federated architecture, which helps to learn the revenue adaptively in a multi-agent-based environment. Moreover, in this multi-agent communication system, some agents may provide inaccurate information about service availability and resource consumption and can make miscalculations about any decision. The Byzantine fault is one kind of fault that is the cause for providing this imperfect information and miscalculation about the resources [4]. We consider Byzantine faults in the system where some agents send misinformation to the server and hamper learning.

In recent years, several works have considered resource allocation based on economic aspects for 5G enabling technologies. In [5], the authors provide a solution for minimizing the cost of the slice broker. They propose a heuristic method for reducing the cost of buying resources from the InPs. However, their method is not adaptive, does not consider any dynamic pricing mechanism for demand, and no revenue model is learning for the brokers. Furthermore, this work does not assume any faults in the system. In [6], the authors propose a dynamic method in a distributed way, which helps to allocate the resources for active information flows that belong to the different slices of different characteristics. However, their method does not consider the revenue of the slice broker. Also, they do not consider any adaptive pricing method, and finally, they do not consider any faults in the system. In [6], the authors propose an optimization process for network slicing where slice customers' profit and slice providers make resource efficiency. However, there is no consideration of the broker's revenue, adaptive pricing, and no faults consideration in the system. In [7], the authors consider a framework for network slicing where slice request admission has been considered and, simultaneously, investigating the operator's profit by varying the traffic. They apply Lyapunov optimization for this problem. However, the system is not adaptive, and no faults are considered in this case. In [8], the authors propose an auction-based method for revenue calculation in slice based 5G cellular communication. However, they do not consider

any adaptive method and also do not consider revenue model learning with faults, e.g., Byzantine faults.

One of the important aspects of resource allocation in network slicing is the presence of Byzantine faults [9]. There are some recent works based on resource allocation considering Byzantine faults. In [10], the authors propose a coordinated method for resource allocation in a multi-agent distributed network. They propose Coordinate-wise Trimmed Mean (CTM) method for aggregating the messages from the neighbors and then filtering the malicious messages by comparing them with some criteria. However, their method is not adaptive, and also, there are no specific Byzantine faults considered. In [11], the authors propose a Byzantine fault-tolerant mechanism based on comparative gradient elimination in a multi-agent environment. Their proposed method outperforms the gradient descent algorithm. However, their proposed method only focuses on some specific applications. They do not consider resource allocation in their proposal. In [3], the authors propose a cooperative reinforcement learning method for revenue model learning for a slice broker in the 5G-MEC system in the presence of adversaries. However, they only consider the security issues that arose from the attacker/intruder in the system. They do not consider any faults in the system that can mislead the broker about service or resource unavailability. We consider the same problem to solve, i.e., to maximize the broker's profit. We consider the Byzantine faults in the system and propose a comparative gradient elimination-based method in a federated architecture, which provides privacy and can also help to be fault tolerant against Byzantine faults.

To our best knowledge, our work is the first one which considers the revenue model learning for 5G-MEC in the presence of Byzantine faults.

In summary, the contributions of our paper are the following:

- We address a problem considering dynamic demand/request, which is adjusted based on set price changes. We learn the revenue model in the presence of Byzantine faults.
- We propose a solution based on comparative gradient elimination in a federated architecture. To consider this method in a 5G-MEC system is the main contribution.
- We compare our method with the reference solution. We evaluate revenues by Byzantine fault learning and without learning.

The paper is structured as follows. Section II describes the problem of maximizing the revenue in the presence of Byzantine faults. Section III introduces the proposed method to solve the presented problem. Section IV presents the results of the comparison of the proposed method with reference methods. Finally, Section V concludes the paper.

II. Problem Description

In our system, we consider three business entities, i.e., InPs, a slice broker, and a slice tenant. We also consider that each InP has one MEC system consisting of one Multi-Access Edge Orchestrator (MEO) and several MEC Hosts (MEHs). The slice broker instead manages a MEC federation composed of the various MEC systems.

We consider one tenant in our scenario. However, the problem can be generalized for multiple tenants. The tenant sends dynamic requests of resources for the slices to the slice broker. The slice broker buys the computational resources from the InPs and sells them to multiple slice tenants. Here, we assume that the slice broker has already bought resources.

We assume each MEH has one *chunk* of computational resources. The set of chunks is denoted as $\mathcal{M}$. The tenant dynamically requests to the broker for an amount of computational resources that we define as *slice demand* and denote as d^t. Here, t represents the *time interval*.

At each time interval t, the system decides the amount of chunks to be provided and the prices to sell to the tenant. The portion of the chunk to be allocated to the tenant is denoted as *subchunk*. The amount of computational resources taken from the chunk $m \in \mathcal{M}$ at time interval t is denoted as δ_m^t. The *subchunk price* (in €/vCPU), set by the system, is represented as c_m^t.

The MEO of an InP can coordinate the MEHs belonging to that particular InP. MEHs have chunks of resources that provide a subchunk for the requested resources for the slices. The whole system works as a federated architecture of federated learning. MEOs work as distributed agents, sending the gradients of losses to the server, which belong to the slice broker. Gradients of losses are calculated by the weights and calculated revenue at each time interval. Then, the server sends the updated weights to the MEOs. MEHs inform the number of subchunks to MEOs, and MEOs set the prices for that amount. MEOs calculate the revenues and gradients. MEOs send the gradients to the server.

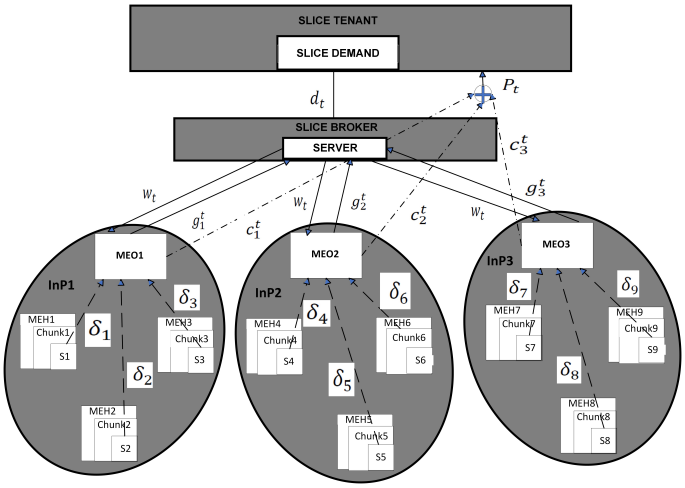

Fig. 1. 5G-MEC federation under investigation

Figure 1 represents the 5G-MEC system to be considered. The introduction of Byzantine faults will be discussed later in this paper.

We consider a practical demand model d_t, where the demand changes over time based on the price changes. Our

demand varies so that if the price increases, the demand decreases. On the other hand, when the price decreases, the demand increases. A shocking function also influences the demand function in a way that sometimes does not follow the mentioned pattern.

We calculate the demand at every time interval, t based on the weighted average price and can be computed as follows:

$$p^t = \frac{1}{d^t} \sum_{m \in \mathcal{M}} \delta_m^t \cdot c_m^t \tag{1}$$

Based on the price-demand function in [12], we compute the slice demand for next time interval $t+1$ as follows.

$$d^{t+1} = d^0 - k \cdot p^t - a \cdot s((p^t - p^{t-1})^+) + b \cdot s((p^t - p^{t-1})^-), \tag{2}$$

where

$$(p^t - p^{t-1})^+ = \begin{cases} p^t - p^{t-1}, & \text{if } p^t > p^{t-1} \\ 0, & \text{otherwise} \end{cases},$$

$$(p^t - p^{t-1})^- = \begin{cases} p^t - p^{t-1}, & \text{if } p^t < p^{t-1} \\ 0, & \text{otherwise} \end{cases},$$

and where p^t is the price at time interval t and p^{t-1} is the price at the previous time interval. The first two terms of the Equation (2) denote the linear demand model intercepting d^0 and slope k. The second two terms model the response to a price change between two intervals. Two coefficients a and b denote the sensitivity to positive and negative price changes, respectively, and s is a shock function that can be used to specify a non-linear dependency between the price change and demand. We assume $s(\cdot) = \sqrt{\cdot}$.

The problem of maximizing the revenue of the slice brokers to serve the slice demand at the time interval t can be formulated based on the Bertnard model [13]:

$$P : \max \Phi^t = \max \sum_{m \in \mathcal{M}} c_m^t \cdot \delta_m^t, \tag{3}$$

subject to

$$\begin{aligned} &C1 : \delta_m^t \leq \eta_m \qquad \forall m \in \mathcal{M}, \\ &C2 : \sum_{m \in \mathcal{M}} \delta_m^t \leq d^t \end{aligned} \tag{4}$$

where Φ^t is the defined revenue function and the objective function of the problem, $C1$ is the constraint that limits the size of the subchunk to the size of the related chunk, and $C2$ is the constraint that limits the cumulative size of all subchunks to the slice demand.

A. Environment

We propose a method based on Comparative Gradient Elimination (CGE) in Federated Leaning (FL). Traditional outlier removal techniques are based on either supervised or unsupervised learning, which is not adaptive. MEOs are the distributed agents that provide the local information, e.g., calculated gradients, to the server of the slice broker. The server updates the weights using the gradients. MEOs calculate the revenue and consider it their local data set D_m. The revenue is calculated as follows for subchunk m at the time interval t:

$$z_m^t = c_m^t \cdot \delta_m^t \tag{5}$$

The possible values of z_m^t are the data points that MEOs use to calculate the gradients of the loss together with the weight calculated by the server in the slice broker.

We define expected loss function as follows:

$$Q(w_m) = \mathop{\mathbb{E}_m}_{z_m \sim D_m} l(w_m, z_m) \qquad \forall m \in \mathcal{M} \tag{6}$$

The goal of our proposed method is to learn the optimal learning parameter $w_m \in \mathbb{R}$ that minimizes the $Q(w_m)$ in the presence of Byzantine faults.

B. Presence of Byzantine Faults

For each agent in MEO, for allocation of subchunk m, there is a collection of data points:

$$D_m^t = \{z_{m1}^t, \ldots, z_{mK}^t\}, \tag{7}$$

where K is the possible data points. For each agent, at each time interval, the actual collection of data points is as follows:

$$D_m^t = \begin{cases} z_m^t & \text{agent is non-faulty} \\ f_m^t & \text{agent is faulty} \end{cases}, \tag{8}$$

where f_m^t is a collection of faulty data points.

We consider two types of Byzantine faults as follows. However, the other types of Byzantine faults, e.g., delayed information, unresponsive nodes can also be considered for our proposed method.

*1) **Gradient reverse fault**:* Gradient reverse faults provide a significant impact on resource allocation as it causes the misallocation of resources. A gradient reverse fault acts as a Byzantine fault where nodes in the network behave in an unpredictable manner [14]. Here, the gradient reverse faulty agent sends the server a vector directly opposite to its correct stochastic gradients. If g_m^t is the calculated correct stochastic gradients, then for the faulty agents the vector will be opposite as f_m^t = -g_m^t. This fault will provide the impression of service unavailability, although the system has enough resources to offer. This fault can also mislead the system by selling resources at a lower price than the expected price.

*2) **Label-flipping fault**:* A label-flipping fault is a kind of fault where nodes in the network intentionally flip the training data's label, creating incorrect model predictions [15]. A label-flipping fault serves as a Byzantine fault where participating nodes manipulate the machine learning system for their benefit. In label-flipping faults, the agents send erroneous calculations for the stochastic gradients. This also provides wrong information in the system about the allocation of resources. The system also can have less revenue for allocating resources, whereas, in the normal situation, that could be sold at much higher prices. So, it creates a loss for the broker. Here, we calculate the label-flipping fault by f_m^t = 8 - g_m^t [11].

III. Proposed Method

To solve the problem presented in the previous section, our proposed method is based on CGE. CGE is applied in an FL architecture, where there is a collaboration between MEOs and the server in a slice broker. Each MEO selects data from data set D_m. Here, these data sets consist of gradients.

Stochastic gradients are calculated by K data points of weights and calculated revenues. It can be stated as follows:

$$g_m^t = \frac{1}{K}\sum_{j=1}^{K} \nabla l(w_m^t, z_{mj}^t) \qquad \forall m \in \mathcal{M} \tag{9}$$

MEHs send the information, e.g., amount of resources and capacity, to the MEO. MEOs also communicate with the server in the slice broker. The server calculates the weight and sends the updated weights to the MEOs. MEOs calculate the revenue and the gradients by weights and the revenues.

For each calculated weight w_m, data set, calculated revenue, z_m faces a loss, which is a loss function denoted as a real value, $l_m : (w_m, z_m) \mapsto \mathbb{R}$. The expected loss function is defined in Equation 6.

Back propagation is used to train the neural network in the learning algorithm for making it more efficient. So, the selection of amount of resources, the calculated revenues, gradients will be in a way that the proper weight will be adjusted and the loss will be minimized [16].

In the proposed method, the server in slice broker eliminates e number of gradients from n gradients at each time interval t. The estimation of weights are updated by using the average of $n-e$ stochastic gradients. The elimination of gradients happens by calculating Euclidian norms and there is an adaptive threshold for removing the e largest norms.

The server sorts the received gradients as follows:

$$||g_{m1}^t|| \leq \ldots \leq ||g_{mn-e}^t|| \leq ||g_{mn-e+1}^t|| \leq \ldots \leq ||g_{mn}^t|| \tag{10}$$

The server updates the weight with the $n-e$ gradients:

$$w_m^{t+1} = w_m^t - \theta_t \sum_{j=1}^{n-e} g_{mj}^t \tag{11}$$

Where θ_t denotes the learning rate at time interval t.

Algorithm 1 shows the step-by-step procedure about how the system works and how the training has been done. The algorithm is run for every epoch.

IV. Results and Discussions

We simulate the scenario with three MEOs, where each MEO can coordinate with the three MEHs.

Table I shows all the simulation parameters. The values for these parameters are set based on empirical studies.

We consider 150 nodes with three hidden layers, each layer consists of 50 nodes for the neural network.

Figure 2 shows the evaluation of loss over epochs where no Byzantine faults and with Byzantine faults are compared.

Algorithm 1 Proposed Method

for Each time interval, t **do**

Step S1: MEOs calculate the stochatic gradients with K data points with the initial estimate of the weight w_m^t.

$$g_m^t = \frac{1}{K}\sum_{j=1}^{K} \nabla l(w_m^t, z_{mj}^t) \qquad \forall m \in \mathcal{M}$$

Step S2: MEOs send the calculated gradients to the server of the slice broker.

Step S3: The server sort out the received gradients for eliminating the largest e Euclidian norms.

$$||g_{i1}^t|| \leq \ldots \leq ||g_{in-e}^t|| \leq ||g_{in-e+1}^t|| \leq \ldots \leq ||g_{in}^t||$$

Step S4: The server update the weight with the $n-e$ gradients.

$$w_m^{t+1} = w_m^t - \theta_t \sum_{j=1}^{n-e} g_{mj}^t$$

Step S5: The server of the slice broker broadcasts the estimate of weight w_m^{t+1} to all MEOs. Each non-faulty agent i sends to the server a stochastic gradient of the global expected loss function $Q_m(w)$.

end for

TABLE I
Simulation Parameters and Their Values.

Parameter	*Symbol*	*Value*
Available chunks	$\|\mathcal{M}\|$	9
Number of agents	$\|\mathcal{N}\|$	3
Learning rate	θ_t	0.5
Time interval	T	1000
Epoch	E	20
Size of the chunks	η_m	5000 vCPU
Intercept	d_0	5000 vCPU
Slope	k	20 vCPU/€
Response coefficient for price increase	a	300 vCPU/$€^{1/2}$
Response coefficient for price decrease	b	100 vCPU/$€^{1/2}$

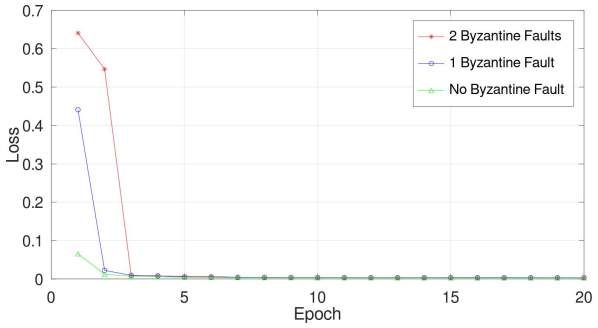

Fig. 2. Evaluation of loss with no faults and with faults

We can observe that with no Byzantine faults, the amount of loss is significantly less. But on the other hand, the loss also increases with the increase of Byzantine faults. Here, we can see that with only one Byzantine fault, loss started with a higher value and then decreases over the following epochs. Finally, the learning algorithm is converged.

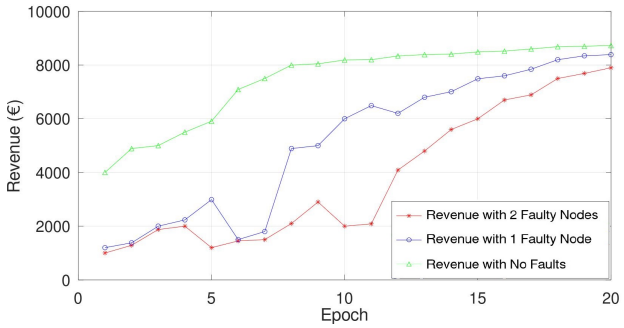

Fig. 3. Evaluation of revenues with no faults and with faults

Figure 3 shows the revenue over the epochs for no faults and with Byzantine faults. We perform a simulation considering the presence of one Byzantine node and then two Byzantine nodes. We can observe that the revenue is increased over the epochs with no Byzantine faults. It starts initially with a mediocre value and then increases. This is because, over the epochs, learning becomes more efficient. Our proposed method helps to maximize revenue by eliminating the gradients in federated learning. With one Byzantine faulty node, we can observe that the initial revenue is lower than without any faults. Then we can see an increment over the epochs and after that a sharp decrement for the faults and finally over the period, the method becomes Byzantine fault tolerant. The same trend can be observed for the two Byzantine faults but with lower revenue at every epoch. We can observe that we can get the lowest revenue for the 2 Byzantine faulty agents. For the Byzantine fault-tolerant property, we can observe that even with faulty nodes, it overcomes the loss and come to a considerable profit level.

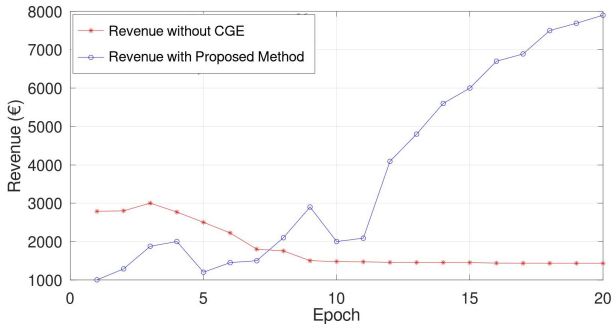

Fig. 4. Evaluation of revenues without CGE and with our proposed method

Figure 4 portrays the evaluation of revenues without CGE and with our proposed method. We consider here two Byzantine faulty nodes among three nodes. Without CGE, there is no gradient elimination in the system. Without gradient elimination, the revenue goes down over the epoch. We can observe that there is a slight decrease over the epoch and got converged. This is because if we consider all the gradients, then with two Byzantine faulty nodes, the gradients mislead the system about the resources, which is the cause for the lower revenue. On the other hand, if we eliminate the gradients based on our proposed method, it gradually becomes Byzantine fault-tolerant, and the revenue rises slowly over the epoch for learning.

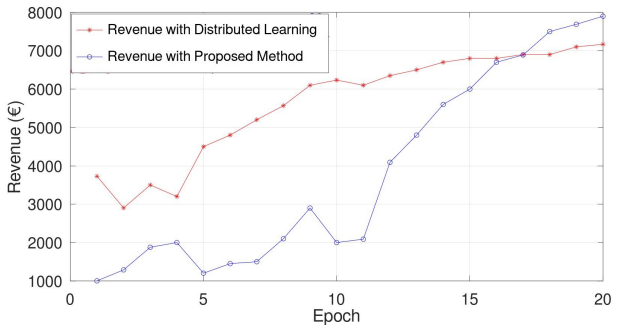

Fig. 5. Evaluation of revenues for distributed learning and our proposed method

Figure 5 compares revenues based on distributed learning and our proposed method. Here, distributed learning is classical Deep Q Learning (DQL). We consider two nodes faulty here for both cases. By applying distributed learning over the epoch, we can observe that there are increases and few decreases. It is for the exploration and exploitation of learning algorithms. We can see that in the initial epochs distributed one outperforms the proposed method in terms of revenue. However, due to the lack of a fault tracking system in the distributed one, after 15 epochs, it starts to go down comparing with the proposed method. Whereas, in our proposed method, for two faulty Byzantine nodes, initially the revenue was not good enough, but when over the epoch, the algorithm becomes Byzantine fault tolerant, the revenue starts rising, and finally we can observe that at the last few epochs, our proposed method started outperforming the distributed one.

V. Conclusions

In a slice allocation problem, the business entity slice broker focuses on revenue maximization by adaptively allocating the resources based on the demand. We consider the demand that changes dynamically based on set price changes. We propose a CGE-based FL method for slice allocation. Our simulation results show that our proposed method outperforms in the case of the presence of Byzantine faults. Our proposed method minimizes the loss and maximizes the revenue. We also compared our proposed method with the method without CGE, where our method outperforms in terms of revenue. We also compared our method with distributed reinforcement learning, where we can observe that our method outperforms revenue.

ACKNOWLEDGMENT

This work was funded by Norwegian Research Council through the 5G-MODaNeI project (no. 308909).

REFERENCES

[1] B. Ojaghi, F. Adelantado, and C. Verikoukis, "On the benefits of vdu standardization in softwarized ng-ran: Enabling technologies, challenges, and opportunities," *IEEE Communications Magazine*, 2023.

[2] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5g network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[3] M. I. Khan and G. Nencioni, "Revenue-model learning for a slice broker in the presence of adversaries," in *IEEE International Conference on Advanced Networks and Telecommunications Systems*. IEEE, 2023.

[4] M. A. AlZain, B. Soh, and E. Pardede, "A byzantine fault tolerance model for a multi-cloud computing," in *2013 IEEE 16Th International Conference On Computational Science And Engineering*. IEEE, 2013, pp. 130–137.

[5] A. Gohar and G. Nencioni, "Minimizing the cost of 5g network slice broker," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2021, pp. 1–6.

[6] F. Mason, G. Nencioni, and A. Zanella, "Using distributed reinforcement learning for resource orchestration in a network slicing scenario," *IEEE/ACM Transactions on Networking*, 2022.

[7] J. Feng, Q. Pei, F. R. Yu, X. Chu, J. Du, and L. Zhu, "Dynamic network slicing and resource allocation in mobile edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7863–7878, 2020.

[8] N. Tadayon and S. Aissa, "Radio resource allocation and pricing: Auction-based design and applications," *IEEE Transactions on Signal Processing*, vol. 66, no. 20, pp. 5240–5254, 2018.

[9] H. Xu, P. V. Klaine, O. Onireti, B. Cao, M. Imran, and L. Zhang, "Blockchain-enabled resource management and sharing for 6g communications," *Digital Communications and Networks*, vol. 6, no. 3, pp. 261–269, 2020.

[10] R. Wang, Y. Liu, and Q. Ling, "Byzantine-resilient decentralized resource allocation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 5293–5297.

[11] N. Gupta, S. Liu, and N. Vaidya, "Byzantine fault-tolerant distributed machine learning with norm-based comparative gradient elimination," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2021, pp. 175–181.

[12] S. K. Vishnoi, T. Bagga, A. Sharma, and S. N. Wani, "Artificial intelligence enabled marketing solutions: A review," *Indian Journal of Economics & Business*, vol. 17, no. 4, pp. 167–177, 2018.

[13] W. W. Sharkey and D. S. Sibley, "A bertrand model of pricing and entry," *Economics Letters*, vol. 41, no. 2, pp. 199–206, 1993.

[14] Z. Xiang, D. Malkhi, K. Nayak, and L. Ren, "Strengthened fault tolerance in byzantine fault tolerant replication," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 205–215.

[15] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6893–6901.

[16] H. Feng, T. Pang, C. Du, W. Chen, S. Yan, and M. Lin, "Does federated learning really need backpropagation?" *arXiv preprint arXiv:2301.12195*, 2023.