network

MDPI

Article

An Uncertainty-Driven Proactive Self-Healing Model for Pervasive Applications

Maria Papathanasaki *, Panagiotis Fountas and Kostas Kolomvatsos

Department of Informatics and Telecommunications, University of Thessaly, Papasiopoulou 2-4, 35131 Lamia, Greece
* Correspondence: mpapathanasaki@uth.gr

Abstract: The ever-increasing demand for services of end-users in the Internet of Things (IoT) often causes great congestion in the nodes dedicated to serving their requests. Such nodes are usually placed at the edge of the network, becoming the intermediates between the IoT infrastructure and Cloud. Edge nodes offer many advantages when adopted to perform processing activities that are realized close to end-users, limiting the latency in the provision of responses. In this article, we attempt to solve the problem of the potential overloading of edge nodes by proposing a mechanism that always keeps free space in their queue to host high-priority processing tasks. We introduce a proactive, self-healing mechanism that utilizes the principles of Fuzzy Logic, in combination with a non-parametric statistical method that reveals the trend of nodes' loads as depicted by the incoming tasks and their capability to serve them in the minimum possible time. Through our approach, we manage to ensure the uninterrupted service of high-priority tasks, taking into consideration the demand for tasks as well. Based on this approach, we ensure the fastest possible delivery of results to the requestors while keeping the latency for serving high-priority tasks at the lowest possible levels. A set of experimental scenarios is adopted to evaluate the performance of the suggested model by presenting the corresponding numerical results.

Keywords: self-healing systems; task management; pervasive computing; Fuzzy Logic; task offloading

Citation: Papathanasaki, M.; Fountas, P.; Kolomvatsos, K. An Uncertainty-Driven Proactive Self-Healing Model for Pervasive Applications. *Network* **2022**, *2*, 568–582. https://doi.org/10.3390/network2040033

Academic Editors: Posco Fung Po Tso, Takahiro Shinagawa and Kiho Lim

Received: 16 July 2022
Accepted: 20 October 2022
Published: 25 October 2022

1. Introduction

A common challenge that the scientific community has to face today is the overloading of processing nodes, which leads to serving fewer requests, increasing the final latency and performance. Overloading happens when the resources of a processing node are depleted to the point where it cannot handle incoming requests; thus, it will not respond to them accordingly. This phenomenon can be easily met at the edge infrastructure, where numerous nodes can be 'connected' with a high number of Internet of Things (IoT) devices and the Cloud data centers. It is estimated that by the year 2030, approximately half a trillion devices will be connected to the Internet. At the same time, every Internet user is expected to use about 257 GB per month [1]. Edge nodes convey resources that can be adopted to process data reported by IoT devices, minimizing the latency in the provision of the outcomes for the requestors. Requests can have the form of the execution of simple queries upon data or the extraction of machine learning models that can be adopted for decision making either locally to peers or the Cloud back end. The edge nodes receive requests that take the form of tasks which are placed in the local queue before the local resources are allocated for their execution. Tasks, apart from the data upon which the desired processing should be executed, may have priorities: a common approach to discern immediate processing activities that are usually requested when real-time applications should be supported. In this paper, we propose a proactive self-healing model that maintains a specific number of tasks in the local queue, leaving free room for hosting high priority processing activities. Our approach focuses on the minimization of

the latency that high-priority tasks may face in overloaded nodes. The proposed model is strategically chosen to cover any type of network. Our model does not focus on the network level, but it is node oriented. We look at the individual behavior of the node, empowering it to cover the uncertainty when it receives tasks and tries to handle them. In our case, we are referring to a higher level since we are not looking at network characteristics. We care about maintaining self-healing in the dimension of QoS at a high level and in this sense, the self-healing that we do is transparent.

One can find multiple efforts in past studies that try to solve the overloading problem mainly focusing on activities performed in a network. For instance, in [2], the authors study an M/M/1 queue subject to alternative behaviors. They assume that the properties of the queue fluctuate randomly over time. The authors also consider two operational conditions which allow the modelling of queues based on two modes of arrivals with variable high–low rates. If the current environment leads to a traffic congestion, the switch to the other environment (i.e., operational condition) may yield a favorable consequence for the length of the queue, achieving a stable system. In [3], a self-healing approach is proposed in order to maintain connectivity in reconfigurable networks. In this paper, the case of the network being compromised by an attack is studied. The suggested distributed algorithm deletes the affected node and reconfigures the network structure to maintain its functionality. This approach, however, is limited only to reconfigurable networks, in contrast to the approach we propose, which can be functional no matter the type of network and the status of edge nodes. Both refs. [4,5] focus on a task offloading approach to manage the high load in mobile devices by offloading computational and heavy tasks to nearby edge nodes. This way, the presented model aims at reducing the task processing delay and minimizing the overall service time for latency-sensitive (IoT) applications. In [6], the authors introduce a framework in which the tasks with high computing requirements are offloaded from the end devices to the edge nodes. An intelligent task offloading scheme that generates the corresponding offloading decision profiles in varying scenarios is presented in [7]. This scheme combines machine learning and a learning-based offloading strategy based on historical data. In [8], the authors propose a low-complexity, greedy, heuristic algorithm to cope with the task offloading problem in a multi-cell Mobile-Edge Computing network.

In [9], the authors formulate the offloading of tasks using binary variables to determine the best task offloading choices for mobile users. A model for simultaneous resource allocation and offloading decision optimization is suggested for mobile edge computing. Another task offloading technique for mobile edge computing systems is described in [10]. The authors take a real-world scenario into consideration while optimizing the software runtime environment with task offloading. A reinforcement-learning algorithm is proposed in [11] to deal with the resource management issue in edge servers and choose the best offloading strategy for reducing system costs, such as energy usage and delay. In [12–14], the authors adopt the principles of Optimal Stopping Theory (OST) to deal with the challenge of determining when to offload data to edge servers for computing analytics tasks. At the same time, the authors take into account the overall delay caused by each server. Furthermore, in [15], the authors extend the aforementioned works, assuring the Quality of Service (QoS) and reducing the anticipated execution time for tasks. The QoS is also taken into consideration in [16]. The authors introduce a heuristic algorithm to handle the offloading decision, and assign priority to nodes that control the order in which they will perform the task offloading. Finally, in [17], a deep-reinforcement-learning algorithm is adopted for dealing with the same problem. The purpose is to perform task offloading in Vehicular Fog Computing, taking into account the priorities of tasks.

In contrast to the aforementioned approaches, the novelty of our paper is that the proposed model has the ability to keep enough space in the local queue to host high-priority tasks and tasks that exhibit a high demand avoiding offloading decisions for these tasks. This way, we ensure that requestors will receive the final response in the minimum possible time. At the same time, when a node is detected as potentially overloaded, low priority tasks are offloaded to peer nodes. Our ultimate goal is to minimize the times that the queue

is full with obvious negative effects in the performance of the node as it is necessary in such cases to offload all the incoming tasks no matter their priority. We propose the adoption of a statistical method that reveals the trend and the probability of overloading based on the rates of incoming tasks and the rate of execution. The probability of overloading is combined with the trend of the rate of execution adopting a Fuzzy Logic system in order to handle the uncertainty in decision making. The proposed model works in a self-healing way since without external intervention, the node achieves high performance in real time, as it manages to handle the overloading situations in such a way so as to execute locally a high percentage of high-priority and high-demand tasks, providing results in the shortest possible time. More specifically, this overload avoidance provides continuous service to the aforementioned tasks, keeping latency low and QoS high. The following list reports on the contributions of our work:

- We propose a self-healing model for maximizing the number of high-priority tasks served by an edge node. We pursue keeping free space in the local queue to host those tasks in support of real-time applications;
- We ensure the smooth operation of edge nodes under heavy traffic, avoiding overloading scenarios;
- We achieve the efficient management of the incoming load by taking the appropriate decisions for the offloading actions, taking into consideration the priority of tasks combined with their demand;
- We report on an extensive evaluation process of the proposed model, revealing its pros and cons.

The rest of the paper is organized as follows. Section 2 outlines our problem and provides the main notations adopted in our model. In Section 3, we present the experimental results extracted out of the proposed mechanism. Finally, in Section 4 we conclude our paper by briefly summarizing the proposed mechanism and its benefits. We also discuss the findings and their implications, as well as our future research plans.

2. Preliminaries and Scenario Description

The ever increasing need for task servicing within the next years prompted us to develop a proactive self-healing mechanism that aims to solve the node overloading problem caused by the large amount of tasks. In the model we developed, we adopted a statistical method on the one hand and a Fuzzy System on the other. The first one facilitates the execution rate trend estimation, while the second one can more easily handle the uncertainty in decision making.

2.1. Scenario Description

We consider a set of nodes $N = \{G_1, G_2, \ldots, G_n\}$ which interact between themselves and with their environment to collect data and execute various processing activities, i.e., tasks. The collected data that every G_i locally stores are in the form of d-dimensional vectors i.e., $X^t = [x_1^t, x_2^t, \ldots, x_d^t]$ where the index t depicts the time instance when G_i receives them. This means that the recording of X^t at the assumed discrete time instances $t \in \{1, 2, 3, \ldots\}$ defines a time series dataset upon which G_i may be requested to execute various processing activities. Those requests can be initiated by users or applications, indicating the execution of tasks upon the collected data and the extraction of knowledge. Apparently, for realizing the desired processing activities, G_i adopts the local resources to perform the requested execution of tasks and store the outcomes. An additional step involves the communication with the requestor to deliver the final results.

We assume that G_i maintains an M/M/1/K queue; i.e., we consider a limited capacity of the local queue where the incoming tasks are placed before the local resources are allocated for their execution [18]. G_i serves the incoming requests for processing in a First-Come-First-Served (FCFS) order to apply 'fairness' in the allocation of resources. However, there are multiple scenarios where some tasks could have higher priority than others, especially when we consider that G_i should be able to support real-time applications. For

instance, imagine that G_i is an edge node and should respond to a query that demands information about the traffic in a specific sub-area. G_i should immediately respond to this query; otherwise, any delay may cause bottlenecks in the decision making, relying on the outcomes of the processing with clear potential negative consequences in the execution of the supported applications.

G_i does not want to be overloaded and has to offload the incoming tasks to its peers, causing additional delays in the provision of results. The research question that governs this paper is to assist G_i in maintaining the size of the queue below a pre-defined threshold θ, leaving space for hosting high-priority tasks. A monitoring process should be applied that will result in alerts when there is an increased risk of G_i being overloaded; i.e., the number of tasks in the queue is above θ. The discussed alerts will, consequently, be adopted to 'fire' a task management mechanism locally. Obviously, the scenario of having G_i overloaded is met based on the rate of receiving the incoming tasks and the rate of executing them. In dynamic environments, these two rates are continually updated, especially when we consider bursts of tasks reported in G_i. We propose the adoption of an uncertainty-based reasoning mechanism to manage the uncertainty in the detection of G_i as overloaded and the utilization of the aforementioned task management model to secure that there is always room for hosting high-priority tasks in G_i (the number of tasks in the queue should be less than θ).

A pictorial representation of the operation of an edge node in our scenario can been found in Figure 1.

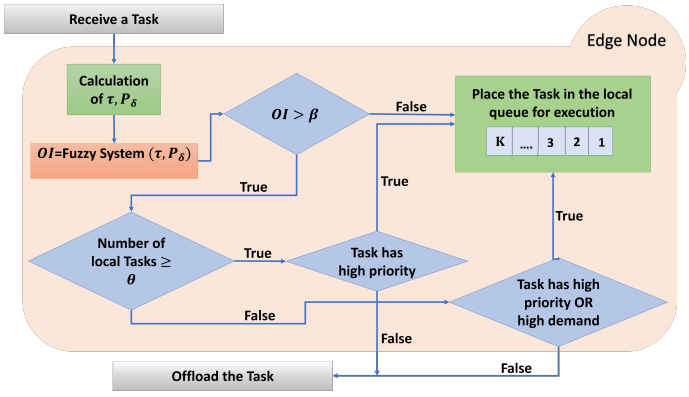

Figure 1. Proposed model adapted to an edge node.

The proposed model is easy to implement in real systems. Initially, the adoption of a component is required, which will have the role of decomposing the tasks into their characteristics. Then, the information will pass through the module we created in order to facilitate the management of the tasks, as shown in Figure 2.

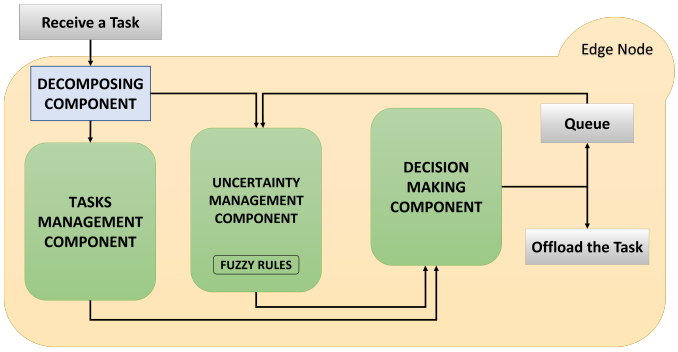

Figure 2. Components that constitute UPSHM.

2.2. Task Management

The arrival of tasks in G_i is assumed to be governed by a Poisson process with rate λ. Based on this assumption, we can easily obtain the number of tasks $Z(t)$ that arrive during the interval $(0, t]$ that follows a Poisson distribution, described by the corresponding Probability Mass Function (PMF) [19]:

$$P[Z(t) = k] = \frac{e^{-\lambda t}(\lambda t)^k}{k!}, \quad k = 0, 1, 2, ... \tag{1}$$

The inter-arrival time instances follow an exponential distribution with the respective Probability Density Function (PDF) given by the following equation [19].

$$f_{\mathrm{I}}(x; \lambda) = \lambda e^{-\lambda x}, \quad x, \lambda > 0 \tag{2}$$

where λ represents the rate of arrivals. Under the same rationale, we consider that the time instances depicting the conclusion of the execution of tasks are governed by an exponential distribution with the following PDF [19]:

$$f_{\mathrm{S}}(x; \mu) = \mu e^{-\mu x}, \quad x, \mu > 0 \tag{3}$$

where μ depicts the service rate of G_i. In the above-discussed PDFs, we adopt an index (I or S) to depict the distribution that corresponds to the arrival or the execution of tasks, respectively. Based on the previous assumptions, we can easily observe that the average inter-arrival time and the average service time is given by the following equations [19]:

$$\Omega[\mathrm{I}] = \frac{1}{\lambda} \tag{4}$$

$$\Omega[\mathrm{S}] = \frac{1}{\mu} \tag{5}$$

Equation (6), represents the traffic intensity in an M/M/1/K system given by the ratio of the arrival rate, compared to the service rate. The following equation holds true [18,19]:

$$\rho = \frac{\lambda}{\mu} \tag{6}$$

With simple calculations, we can deliver the probability of having v tasks in G_i's local queue, as the following equation expresses [19]:

$$P_v = \begin{cases} \frac{(1-\rho)\rho^v}{1-\rho^{K+1}}, & \rho \neq 1, \quad 1 \leq v \leq K \\ \\ \frac{1}{K+1}, & \rho = 1, \quad 1 \leq v \leq K \end{cases} \tag{7}$$

If we assume that θ tasks should be at most in the local queue, we can easily calculate the probability of overloading, i.e., P_δ, as follows [19]:

$$P_\delta = \sum_{v=\theta}^{K} P_v \tag{8}$$

θ is the threshold for considering G_i as overloaded and it should start offloading the upcoming tasks to peer nodes. For instance, offloading can be performed through the adoption of various techniques such as [20–24]. With the proposed model, we are able to support G_i to maintain a free space for high-priority tasks, based on a proactive approach [25]. We have to notice that the processes upon which the offloading decisions and the selection of the appropriate peers to host the offloaded tasks take place lies beyond the scope of this paper.

2.3. Trend Estimation

Let us consider discrete time intervals where we record the inter-arrival rate of tasks and their execution rate. Obviously, if $\lambda >> \mu$, G_i is not stable, it will certainly be overloaded. However, in real setups, we can observe fluctuations in the realization of λ and μ. Hence, we decide to focus on every discrete time interval (e.g., minutes, hours, days, etc.) and deal with the rate of execution μ as a time series dataset. Our target is to check the trend of the execution rate in order to support the decision making concerning the offloading of tasks. In simple words, if we see that μ increases, we can postpone the offloading of tasks to eliminate the time required for sending tasks to peers and obtaining the final results to deliver them to the requestor. For instance, new tasks placed for execution may request simple processing activities or actions that can be based on previous results; thus, their re-use can significantly reduce the execution time. We rely on the time series $\{\mu_1, \mu_2, \ldots\}$ where μ_j is the execution rate at each discrete interval and apply a simple yet powerful technique for extracting the trend of the execution rate. This technique is Kendall's tau statistic [26,27]. Kendall's coefficient (τ) is a non-parametric statistical method that is adopted to measure the correlation and the relationship strength between two measured quantities. This coefficient is applied on the aforementioned μ time series $\{\mu_1, \mu_2, \ldots\}$. The τ value is bounded in $[-1, 1]$ and calculated with the following equation:

$$\tau = \frac{S}{\binom{n}{2}} \tag{9}$$

where n is the number of μ upon which τ is applied and S depicts the Kendall Score which is computed as follows:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sign(\mu_j - \mu_i) \tag{10}$$

$$sign(\mu_j - \mu_i) = \begin{cases} 1, & \mu_j - \mu_i > 0 \\ 0, & \mu_j - \mu_i = 0 \\ -1, & \mu_j - \mu_i < 0 \end{cases} \tag{11}$$

The closer to $+1$ the value of τ is, the more increasing the trend becomes, while the most decreasing trend is achieved when τ is very close to -1. The data have no trend through time when τ does not significantly differ from zero. In our model, we normalize τ in the unity interval ($[0, 1]$).

2.4. Uncertainty-Based Estimation of Overloading

Our model uses two parameters to detect if a node tends to become overloaded or not. The first parameter is τ, which shows the trend of μ through the considered time intervals. The second parameter is P_δ, which depicts that the status of the local queue and the number of tasks on it exceeds a pre-defined threshold θ. The realizations of τ and P_δ are used as inputs to define the antecedent part of our Mamdani Fuzzy System and set a value for the consequent part, i.e., the *Overload Indicator* (OI) bounded in the unity interval. We define a fuzzy knowledge base adopted every time a node receives a request for new processing in the form of a task. We propose a set of Fuzzy Rules (FRs) formulated as follows: "If the probability of the number of tasks exceeding θ is high and the trend of service rate is low, the OI for that node should be high". The proposed FRs have the following structure:

IF P_δ is A_1 **AND** τ is A_2 **THEN** OI is B,

where A_1, A_2 and B are the membership functions of the corresponding fuzzy sets. We characterize the value of A_1 and B by the fuzzy sets *Extreme Low*, *Low*, *Medium*, *High* and *Extreme High* and the value of A_2 through the fuzzy sets *High Negative*, *Low Negative*, *Neutral*, *Low Positive*, and *High Positive*. In our Mamdani Fuzzy System, we adopt triangular membership functions for the definition of fuzzy sets for variables P_δ, τ and OI and involve

the centroid defuzzification method for the calculation of the final crisp output value, i.e., the value of OI. We have to notice that FRs and membership functions for the proposed Mamdani Fuzzy System are defined by the experts. We consider a knowledge base with 25 rules, i.e., a rule for every combination of the two inputs.

2.5. Offloading Strategy Based on the Overloading Indicator

Assume that the outcome of the Fuzzy System, i.e., the OI, indicates that the local queue will be overloaded and offloading actions should take place. We can consider the following cases. **Case A**: the number of tasks in the local queue is greater or equal to θ (our mechanism delays detecting the overloading status); **Case B**: the number of tasks in the local queue is less than θ (our mechanism proactively detects the overloading status). In Case A, we opt to offload all the upcoming tasks to peers and high-priority tasks will be locally placed if there is space for that. In Case B, we propose the use of an additional mechanism that opts to keep some tasks locally until the threshold θ is as follows: high-priority tasks and tasks that exhibit a high demand locally are kept in the queue only if we do not violate θ.

A study on how the demand for tasks may affect their management can be found in [28]. Hence, we can benefit from re-using previously calculated results, spending less resources and maximising the node performance. The proposed processing is depicted in Algorithm 1.

Algorithm 1 Algorithm for Task Offloading based on OI

```
while True do
    Receive a task
    OI = FuzzySystem(τ, P_δ)
    if Queue is full then
        Offload the task
    else
        if OI > β then                          ▷ β is a pre-defined threshold
            if Number of local Tasks ≥ θ then   ▷ Case A
                if Task has high priority then
                    Place the task in the local queue
                else
                    Offload the task
                end if
            else                                ▷ Case B
                if Task has high priority OR high demand then
                    Place the task in the local queue
                else
                    Offload the task
                end if
            end if
        end if
    end if
end while
```

3. Results

3.1. Setup and Performance Metrics

The experimental evaluation of the Uncertainty-driven Proactive Self-Healing Model (UP-SHM), relies on the Tasks Simulation Dataset (www.iprism.eu/assets/DatasetTasksOffloading.zip, accessed on 15 July 2022) and the comparison with a Baseline Model (BM) which is described in [6]. More specifically, the authors use a mechanism that decides whether a task needs to be offloaded when a resource constraint occurs. In our scenario, we consider the aforementioned mechanism as the BM, and we assume that the resource constraint

concerns the case in which the queue is full. Moreover, we include in the comparison an algorithm named Decision Maker (DM), which is described in [29] and has been adapted to our scenario. The goal of our experiments is to prove that UPSHM has the ability to manage the incoming tasks in such a way so as to minimize the times when the node is overloaded offloading the tasks, ignoring both their priority and their demand. The dataset mentioned above contains $E = 3564$ instances representing tasks that arrive in a node in $W = 100$ time instances. More specifically, at every instance, we record the following data for every task: the ID of the task, the required service time of the task, the arrival time, the priority value and the demand indicator. The number of tasks in each time instance follows a Poisson distribution and the service and task arrival times follow an exponential distribution. The rates of both exponential distributions are selected separately for each experiment. In addition, the priority and the demand values follow the uniform distribution. For every experiment, we randomly generate the values of λ and μ, based on the uniform distribution in the intervals $[10, 60]$ and $[10, 40]$, respectively. For comparison purposes, we define the aforementioned BM which performs an offloading action when the local queue of a node is full, regardless of the priority or the demand of tasks. In the comparison between the performance of UPSHM and BM, we pay attention to the number of tasks that the node executes locally. This quantity is expressed by the ϕ metric which depicts the service rate in a node and is defined by the following equation

$$\phi = \frac{\xi}{E} \tag{12}$$

where ξ is the number of executed tasks and E is the total number of tasks that arrive in the node. The higher the number of the tasks that are executed locally, the lower the delay in the provision of results becomes. Additionally, we examine the performance of both BM and UPSHM on the percentage of high-priority tasks that are executed locally. The following equation defines the aforementioned metric; i.e.,

$$\eta = \frac{\kappa}{L} \tag{13}$$

where κ is the number of high-priority tasks which are executed locally and L is the total number of high-priority tasks that arrive in the node. When $\eta \to 1$, it is indicated that the node executes all the high-priority tasks locally, delivering the results to the requestor in the minimum possible time. The same scenario stands for the tasks with high demand rates, where ω is the number of tasks that are executed locally. The next equation holds true:

$$\omega = \frac{\zeta}{\gamma} \tag{14}$$

where ζ is the number of high-demand tasks which are executed locally and γ is the total number of tasks with high demand. The aim behind the choice of these metrics, i.e., η, ω, is to check the ability of an edge node to execute as much as possible high-priority tasks and tasks with high demand rates locally, respectively. The ability of both UPSHM and BM to maintain free space for the high-priority tasks is described by the ψ metric as follows:

$$\psi = \frac{q}{E} \tag{15}$$

where q is the total number of times that the local queue is full. This metric allows us to check how often the load in a node is maximized. When $\psi \to 1$, it is indicated that the node offloads tasks uncontrollably, without taking into consideration the priority or the demand of tasks. The average time that the model requires to make a decision concerning whether a task will be offloaded or not is given by the following equation

$$AvT = \frac{h}{E} \tag{16}$$

where h is the time that the model spend to take all decisions for the offloading actions.

In order to set up and execute the experiments, we created a simulator written in Python. For the UPSHM, we perform simulations for the following values of thresholds; $\beta = \{0.80, 0.84, 0.88\}$ and $\theta = \{85\%, 90\%, 95\%\}$. The DM requires the initialization of a threshold (M_{low}), which depicts when the node's memory has a low level of free space. In our experiments $M_{low} = 15\%$.

3.2. Performance Assessment

Figure 3 presents the performance of the models in terms of the metrics ϕ, η, ω, and ψ, while $\theta = 85\%$ and $\beta = 0.80$. In these results, we notice that the BM and the DM execute a higher percentage of tasks than the UPSHM. However, when we focus on the percentage of high-priority tasks executed locally, we can easily come to the conclusion that the proposed model clearly outperforms the BM and the DM by executing all high-priority tasks locally. Additionally, as far as the percentage of tasks with high demand is concerned, the UPSHM exhibits better performance, executing almost all of these tasks locally. In the last metric, i.e., ψ, the ability of our model to maintain free space in the local memory for high-priority tasks is indicated, since the queue of the node did not become full during the W time instances. Moreover, the DM achieves equal performance in the same metric. In summary, from the results of the first experiment we can clearly observe the significant dominance of our model in the performance metrics that concern the execution of the high priority and high demand tasks.

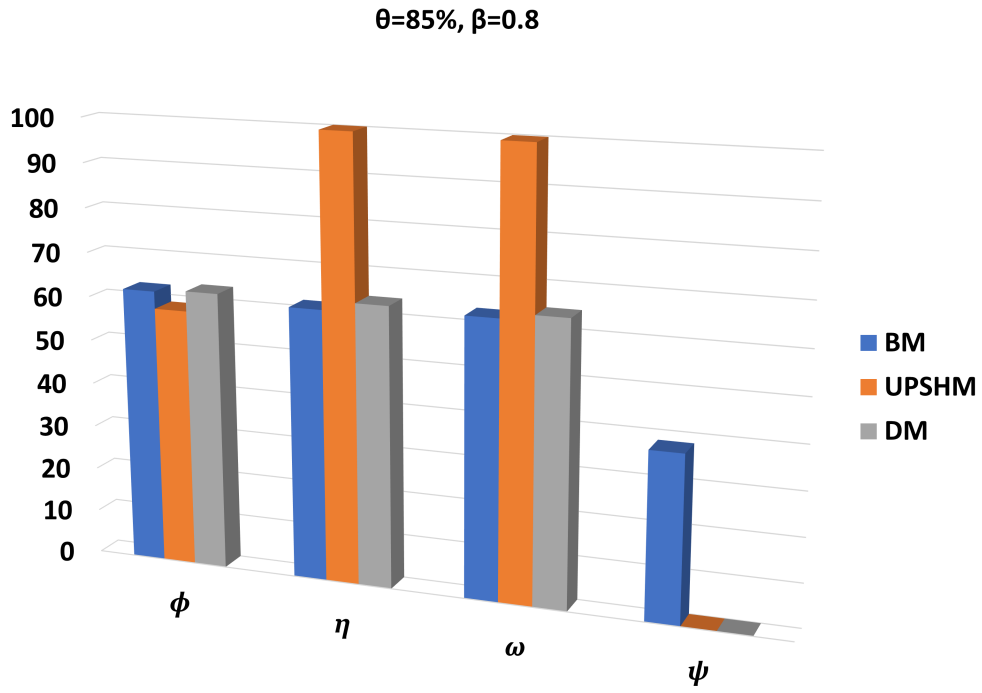

Figure 3. Comparison of results for all performance metrics, for $\theta = 85\%$ and $\beta = 0.8$.

The time comparison between the models for $\theta = 85\%$ and $\beta = 0.8$ is presented in Figure 4. As we can easily observe, the UPSHM requires more time than BM and DM to make an offloading decision.

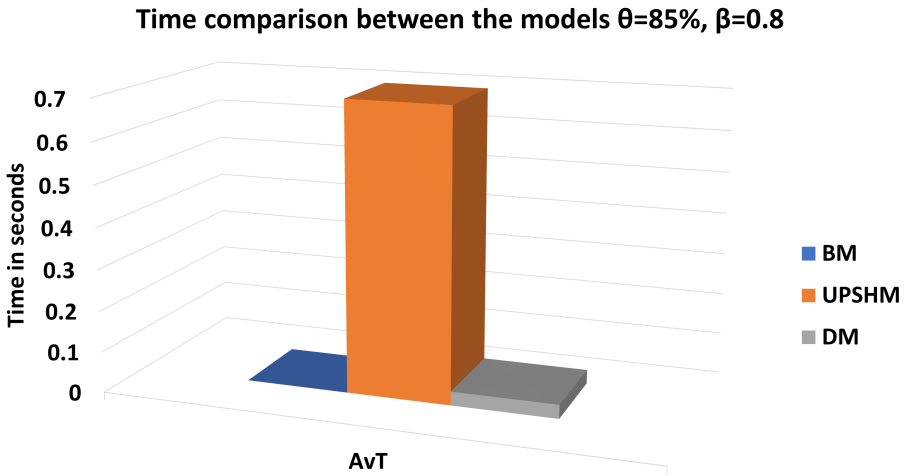

Figure 4. Time comparison between the models for $\theta = 85\%$ and $\beta = 0.8$.

In the second experimental scenario, we set $\beta = 0.88$ and keep $\theta = 85\%$. Figure 5 depicts the performance of BM, UPSHM and DM for metrics ϕ, η, ω and ψ. We notice that the BM executes slightly lower percentages of received tasks than the UPSHM and DM. Nevertheless, when we examine the percentage of high-priority tasks executed locally, we can easily perceive that the difference in the performance among the BM, DM and UPSHM is noticeably higher. In the same figure, we can also see the behavior of the models in the execution of tasks with high demand and also the frequency of the local queue having no space for all types of tasks. We can easily come to the conclusion that the UPSHM has the best performance for the ω metric by executing more tasks with high demand in contrast to the BM and DM. Additionally, concerning the ψ metric, we notice that both the UPSHM and the BM exhaust the capacity of the local queue several times in contrast to DM. This means that in this specific experimental scenario, using either BM or UPSHM for the management of the incoming tasks, the node is forced to offload the incoming tasks without considering their priority or their demand. Even in this case, though, the UPSHM outperforms the BM with regard to the ψ metric, as the number of times that the local queue is full is significantly less than the BM. Nevertheless, the DM achieves the best performance in this metric.

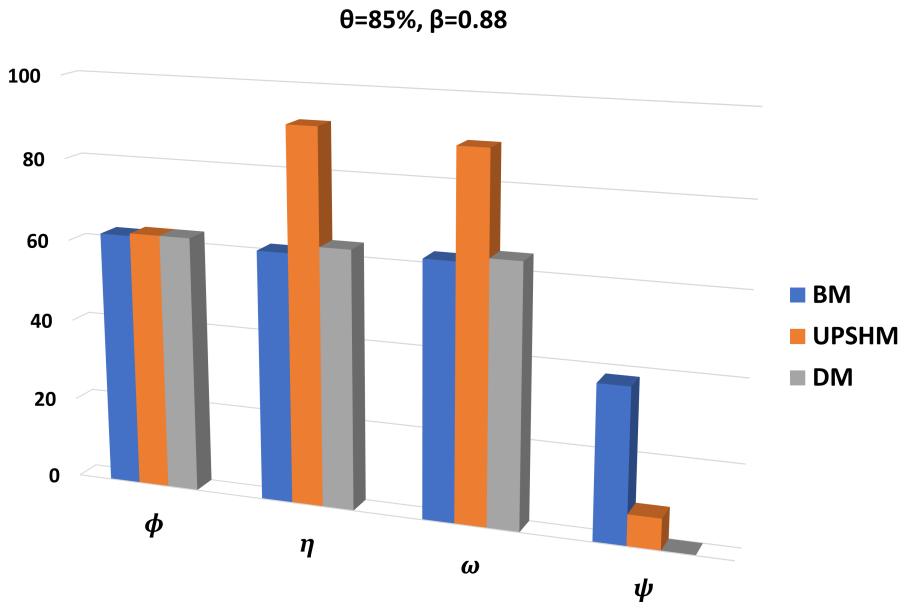

Figure 5. Comparison of results for all performance metrics, for $\theta = 85\%$ and $\beta = 0.88$.

In Figure 6 we see the comparison between the models based on the AvT; similar to the previous scenario, the UPSHM needs more time to make the offloading decision in contrast to the other models.

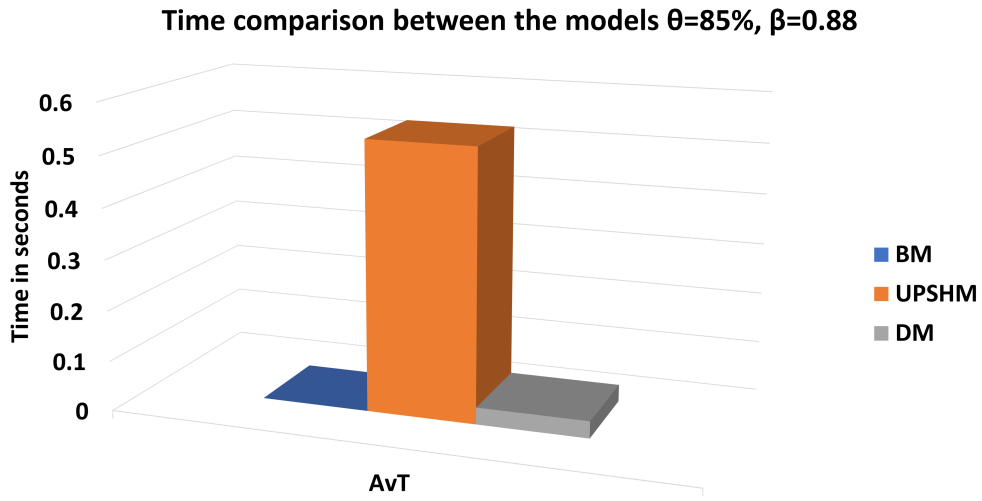

Figure 6. Time comparison of results for BM, UPSHM, DM.

From the previous two experimental scenarios, we observe that the UPSHM charges the node with more computational load than the BM and DM. However, we can accept this extra computational load because the node achieves clearly better performance in the metrics ϕ, η, ω and ψ. The performance of the individual node we are studying is not affected by the rest of the nodes in the network since the communication of the node with other nodes is not required to apply UPSHM.

In Figure 7, we present a comparison of the performance of the UPSHM for different values of β and $\theta = 85\%$. We can observe by the values of the ϕ metric that the higher the β is, the more tasks are executed locally. Concerning the number of tasks with high priority or high demand, η and ω depict that they are affected in the opposite direction to ϕ, as we notice a slight decrement in their values. When we focus on the number of times that the local queue is full, we perceive that it is increased when we increase β, as the ψ metric shows.

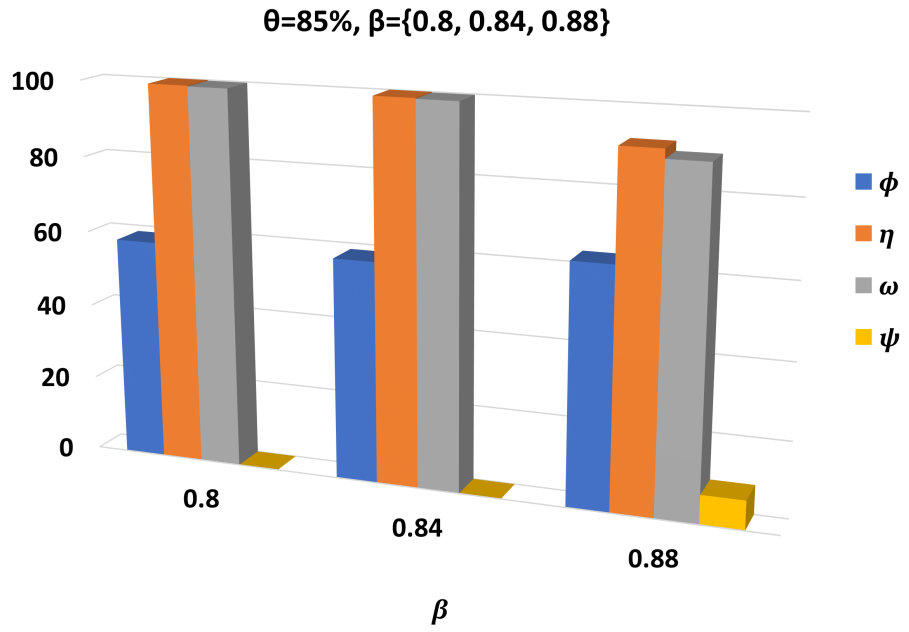

Figure 7. Comparison of UPSHM performances for all metrics for different values of β threshold.

In Figure 8, we present the time performance of UPSHM for $\theta = 85\%$ and $\beta = \{0.8, 0.84, 0.88\}$. As we can see, AvT decreases as the value of β increases. AvT in the worst case is equal to 0.7 seconds and in the best case it is approximately equal to 0.53 seconds.

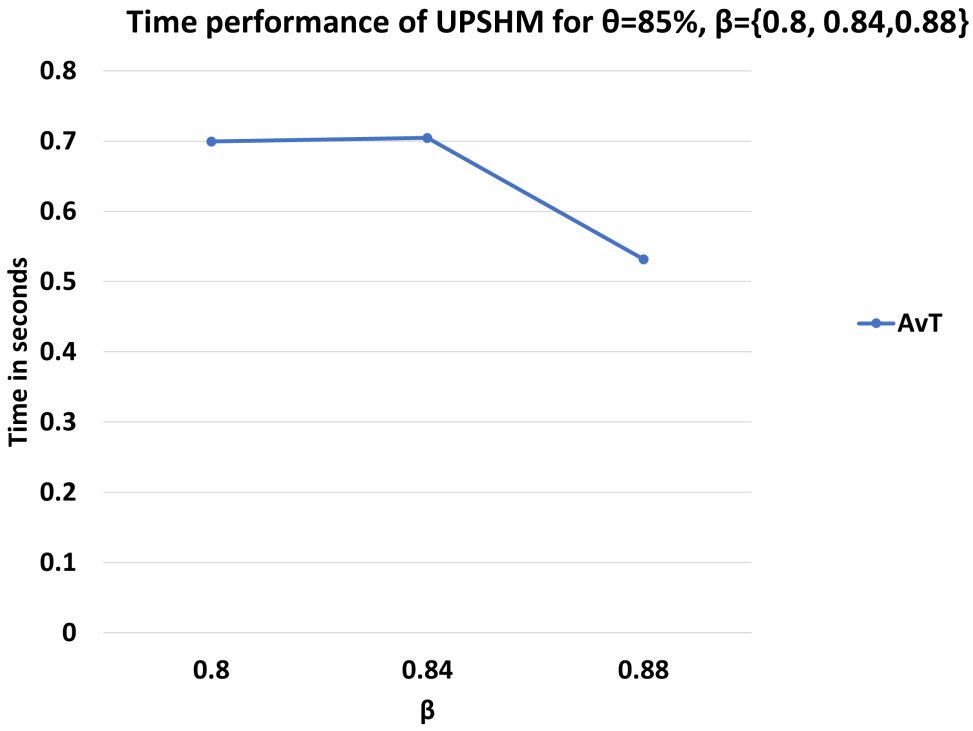

Figure 8. AvT of UPSHM for different values of β.

In Figure 9, a comparison of the performance of the UPSHM for different values of θ and $\beta = 0.8$ is presented. As previously, the values of ϕ and ψ metrics increase as θ increases. In contrast, the values of metrics η and ω are affected in an adverse direction.

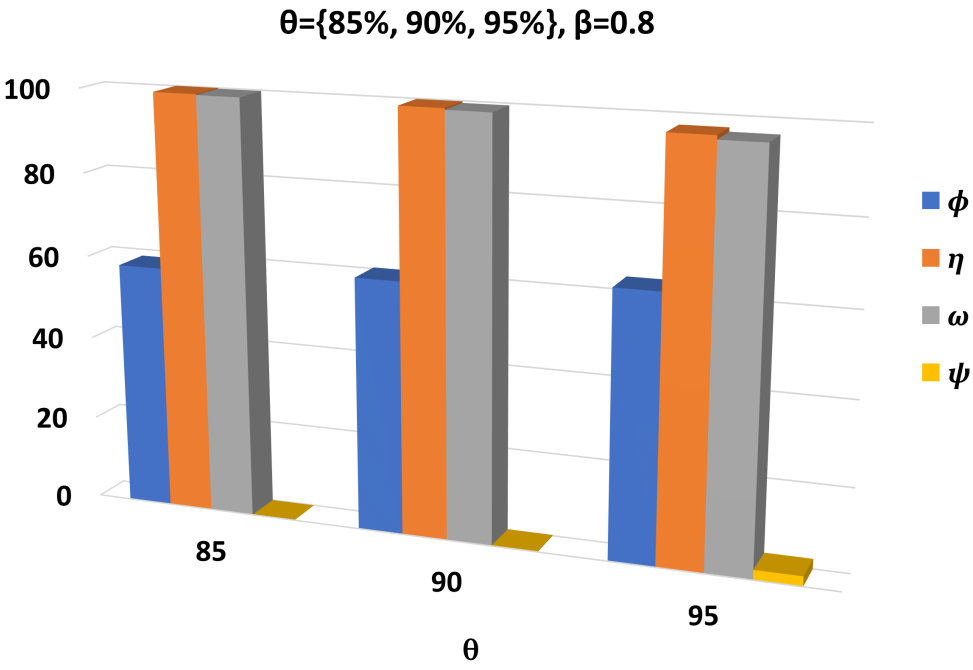

Figure 9. Comparison of UPSHM performances for all metrics for different values of the θ threshold.

In Figure 10, when we focus on the effect of θ on the time performance of the UPSHM, we can see that AvT decreases as θ increases. As previously, in the worst case, AvT is equal to 0.7 s, while in the best case the average time for decision offloading is slightly higher than 0.6 s.

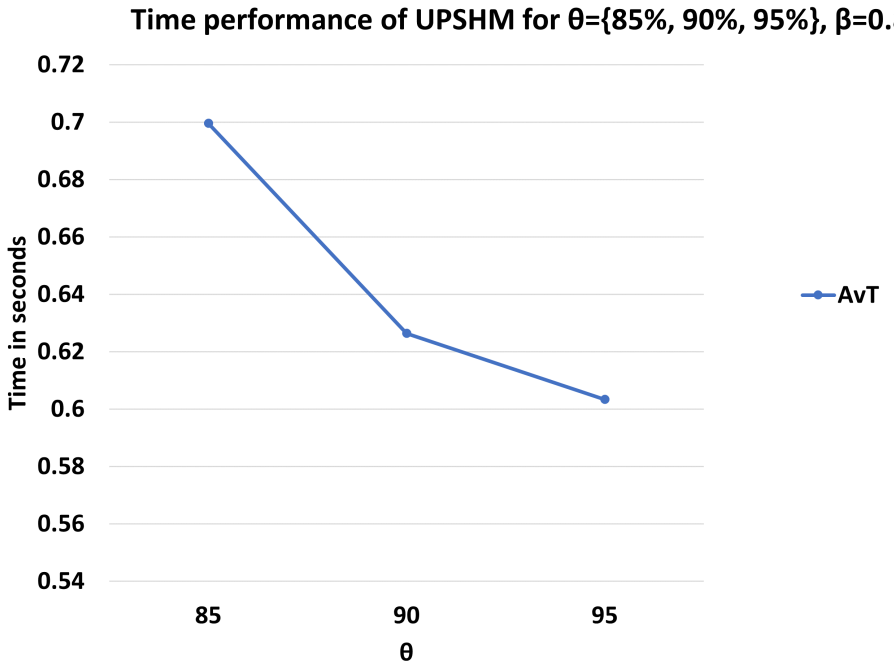

Figure 10. Average Time for decision offloading of UPSHM for different values of θ.

Observing Figures 7 and 9, we can draw conclusions concerning the effect of θ and β in the performance of the UPSHM. We see that ϕ has almost the same performance in both figures. When we focus on η, ω and ψ, we notice that both thresholds affect the performance of the UPSHM in a negative manner. However, an increment in β has greater negative effects than the increment of θ. Additionally, from Figures 8 and 10 we notice that the less time UPSHM spends to make the decision, the greater the values of the ψ and ϕ metric become. Simultaneously, the values of η and ω increase.

4. Discussion

A significant research subject is the prevention of processing nodes overloading, which leads to performance degradation and the need for task offloading to peer nodes. Some of these offloaded tasks might be in high demand or high priority; thus, efficient mechanisms that allow serving locally more of these tasks should be created. In this paper, we approach this problem by developing a self-healing mechanism that makes a node capable of handling a large volume of tasks without being critically overloaded. This way, we succeed in serving a high percentage of tasks with high priority and demand since we always try to keep space for them in the local queue based on a proactive approach. In the proposed model, we adopt an uncertainty-based reasoning mechanism to manage the uncertainty in the detection of the node as overloaded and the utilization of a monitoring process to secure that there is always enough free space for hosting high-priority and high-demand tasks. We evaluate the performance of our model through various experiments and compare it with a baseline method. Our experimental evaluation demonstrates that the suggested approach can effectively help to achieve the desired outcomes, which are supported by numerical results. Our future research directions involve the definition and adoption of a more complex self-healing methodology for making a decision about when a node is considered as overloaded and when an offload action should take place. Furthermore, an additional future direction is the study of the mobility of users, in combination with the tasks that they render as high demand.

Author Contributions: Conceptualization, M.P., P.F. and K.K.; methodology, M.P., P.F. and K.K.; software, P.F.; validation, P.F.; formal analysis, M.P., P.F. and K.K.; investigation, M.P. and P.F.; data curation, P.F.; writing—original draft preparation, M.P., P.F. and K.K.; writing—review and editing, K.K.; visualization, M.P., P.F. and K.K.; supervision, K.K.; project administration, K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset supporting reported results can be found in http://iprism.eu/index.php/datasets-ppts/, accessed on 15 July 2022. For more information about the dataset, contact with the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AvT	Average Time for Offloading Decision
BM	Baseline Model
FCFS	First-Come-First-Served
FL	Fuzzy Logic
FRs	Fuzzy Rules
OI	Overlad Indicator
PDF	Probability Density Function
PMF	Probability Mass Function
QoS	Quality of Service
UPSHM	Uncertainty-driven Proactive Self-Healing Model

References

1. Yastrebova, A.; Kirichek, R.; Koucheryavy, Y.; Borodin, A.; Koucheryavy, A. Future networks 2030: Architecture & requirements. In Proceedings of the 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Moscow, Russia, 5–9 November 2018; pp. 1–8.
2. Di Crescenzo, A.; Giorno, V.; Kumar, B.K.; Nobile, A.G. M/M/1 queue in two alternating environments and its heavy traffic approximation. *J. Math. Anal. Appl.* **2018**, *465*, 973–1001. [CrossRef]
3. Saia, J.; Trehan, A. Picking up the pieces: Self-healing in reconfigurable networks. In Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, 14–18 April 2008; pp. 1–12.
4. Tang, M.; Wong, V.W. Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Trans. Mob. Comput.* **2020**, *21*, 1985–1997. [CrossRef]
5. Almutairi, J.; Aldossary, M. A novel approach for IoT tasks offloading in edge-cloud environments. *J. Cloud Comput.* **2021**, *10*, 1–19. [CrossRef]
6. Naouri, A.; Wu, H.; Nouri, N.A.; Dhelim, S.; Ning, H. A novel framework for mobile-edge computing by optimizing task offloading. *IEEE Internet Things J.* **2021**, *8*, 13065–13076. [CrossRef]
7. Guo, H.; Liu, J.; Lv, J. Toward intelligent task offloading at the edge. *IEEE Netw.* **2019**, *34*, 128–134. [CrossRef]
8. Tran, T.X.; Pompili, D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Trans. Veh. Technol.* **2018**, *68*, 856–868. [CrossRef]
9. Vu, T.T.; Van Huynh, N.; Hoang, D.T.; Nguyen, D.N.; Dutkiewicz, E. Offloading energy efficiency with delay constraint for cooperative mobile edge computing networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
10. Ni, W.; Tian, H.; Lyu, X.; Fan, S. Service-dependent task offloading for multiuser mobile edge computing system. *Electron. Lett.* **2019**, *55*, 839–841. [CrossRef]
11. Alfakih, T.; Hassan, M.M.; Gumaei, A.; Savaglio, C.; Fortino, G. Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA. *IEEE Access* **2020**, *8*, 54074–54084. [CrossRef]
12. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. On the optimality of task offloading in mobile edge computing environments. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
13. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. Time-optimized task offloading decision making in mobile edge computing. In Proceedings of the 2019 Wireless Days (WD), Manchester, UK, 24–26 April 2019; pp. 1–8.

14. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. Optimized Contextual Data Offloading in Mobile Edge Computing. In Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 17–21 May 2021; pp. 473–479.
15. Alghamdi, I.; Anagnostopoulos, C.; Pezaros, D.P. Data quality-aware task offloading in mobile edge computing: An optimal stopping theory approach. *Future Gener. Comput. Syst.* **2021**, *117*, 462–479. [CrossRef]
16. Kan, T.Y.; Chiang, Y.; Wei, H.Y. Task offloading and resource allocation in mobile-edge computing system. In Proceedings of the 2018 27th wireless and optical communication conference (WOCC), Hualien, Taiwan, 30 April–1 May 2018; pp. 1–4.
17. Shi, J.; Du, J.; Wang, J.; Wang, J.; Yuan, J. Priority-aware task offloading in Vehicular Fog Computing based on deep reinforcement learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 16067–16081. [CrossRef]
18. Sztrik, J. *Basic Queueing Theory*; University of Debrecen, Faculty of Informatics: Debrecen, Hungary, 2012; Volume 193, pp. 60–67.
19. Bhat, U.N. *An Introduction to Queueing Theory: Modeling and Analysis in Applications*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 36.
20. Kolomvatsos, K.; Anagnostopoulos, C. A Proactive Statistical Model Supporting Services and Tasks Management in Pervasive Applications. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 3020–3031. [CrossRef]
21. Soula, M.; Karanika, A.; Kolomvatsos, K.; Anagnostopoulos, C.; Stamoulis, G. Intelligent tasks allocation at the edge based on machine learning and bio-inspired algorithms. *Evol. Syst.* **2022**, *13*, 221–242. [CrossRef]
22. Kolomvatsos, K. Data-Driven Type-2 Fuzzy Sets for Tasks Management at the Edge. *IEEE Trans. Emerg. Top. Comput. Intell.* **2021**, *6*, 377–386. [CrossRef]
23. Kolomvatsos, K. Proactive tasks management for pervasive computing applications. *J. Netw. Comput. Appl.* **2021**, *176*, 102948. [CrossRef]
24. Kolomvatsos, K.; Anagnostopoulos, C. Multi-criteria optimal task allocation at the edge. *Future Gener. Comput. Syst.* **2019**, *93*, 358–372. [CrossRef]
25. Fountas, P.; Kolomvatsos, K.; Anagnostopoulos, C. A Deep Learning Model for Data Synopses Management in Pervasive Computing Applications. In *Intelligent Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 619–636.
26. Muñoz-Pichardo, J.M.; Lozano-Aguilera, E.D.; Pascual-Acosta, A.; Muñoz-Reyes, A.M. Multiple Ordinal Correlation Based on Kendall's Tau Measure: A Proposal. *Mathematics* **2021**, *9*, 1616. [CrossRef]
27. Brossart, D.F.; Laird, V.C.; Armstrong, T.W. Interpreting Kendall's Tau and Tau-U for single-case experimental designs. *Cogent Psychol.* **2018**, *5*, 1518687. [CrossRef]
28. Karanika, A.; Oikonomou, P.; Kolomvatsos, K.; Loukopoulos, T. A demand-driven, proactive tasks management model at the edge. In Proceedings of the 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Glasgow, UK, 19–24 July 2020; pp. 1–8.
29. Wijayasekara, V.A.; Vekneswaran, P. Decision Making Engine for Task Offloading in On-device Inference Based Mobile Applications. In Proceedings of the 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Toronto, ON, Canada, 21–24 April 2021; pp. 1–6.